Image Restoration Using a Modified Hopfield Network

Joon K. Paik, *Member, IEEE,* and Aggelos K. Katsaggelos, *Member, IEEE*

Abstract—**In this paper a modified Hopfield neural network model for regularized image restoration is presented. The proposed network allows negative autoconnections for each neuron. A set of algorithms using the proposed neural network model is presented, with various updating modes: i) sequential updates, ii) n-simultaneous updates, and iii) partially asynchronous updates. The sequential algorithm is shown to converge to a local minimum of the energy function after a finite number of iterations. This local minimum is defined in terms of a unit distance neighborhood in the discrete intensity value space. Since an algorithm which updates all n neurons simultaneously is not guaranteed to converge, a modified algorithm is presented, which is called a greedy algorithm. Although the greedy algorithm is not guaranteed to converge to a local minimum, the l_1 norm of the residual at a fixed point is bounded. It is also shown that the upper bound on the l_1 norm of the residual can be made arbitrarily small by using an appropriate step size. Finally, a partially asynchronous algorithm is presented, which allows a neuron to have a bounded time delay to communicate with other neurons. Such an algorithm can eliminate the synchronization overhead of synchronous algorithms. It is shown that the l_1 norm of the residual at the fixed point of this algorithm increases as the upper bound on the delay increases. Experimental results are shown testing and comparing the proposed algorithms.**

I. Introduction

MOTIVATED by the biological neural network in which the processing power lies in a large number of neurons linked with synaptic weights, artificial neural network models, or simply "neural nets," attempt to achieve good performance via dense interconnection of simple computational elements. Neural net models have great potential in areas where many hypotheses are pursued in parallel, high computation rates are required, and the current best systems are far from equaling human performance [8], [16]. In addition to speech and image recognition which are the well-known application areas of neural nets, restoration of a high quality image from a degraded recording is also a good application area of neural nets [21], [22].

The problem of restoring noisy-blurred images is very important for a large number of applications [1]. In many practical situations, the image degradation can be adequately modeled by a linear blur (motion, defocussing, atmospheric turbulence) and an additive white Gaussian process. Then the degradation model is given by

$$z = Dx + \eta \tag{1}$$

where x, z, and η represent, respectively, the lexicographically ordered original and degraded images and the additive noise. The matrix D represents the linear spatially invariant or spatially varying distortion; it has as elements samples of the point spread function (PSF) of the imaging system. D can have a special structure depending on the properties of the PSF. For the convolutional case, for example, D is a block Toeplitz matrix, which can be approximated by a block circulant matrix. In the following we assume that the size of the images is $M \times N$, resulting in x and z being $MN \times 1$ vectors and D an $MN \times MN$ matrix, where we set $n = MN$.

With this model, the purpose of digital image restoration is to operate on the degraded image z to obtain an improved image that is as close to the original image x as possible, subject to a suitable optimality criterion. A number of algorithms providing a solution to the image restoration problem have appeared in the literature [1], [15]. In this paper the set theoretic regularization approach presented in [13], [14] is used. A restored image is obtained by the following optimization problem:

$$\text{minimize } f(x) = \tfrac{1}{2}x^T Tx - b^T x \tag{2}$$

$$\text{subject to } 0 \le x_i \le 255, \quad i = 1, \cdots, n \tag{3}$$

where x_i denotes the ith element of the vector x, $b = D^T z$ and T is a symmetric, positive semi-definite matrix equal to

$$T = D^T D + \lambda C^T C. \tag{4}$$

In (4), C is a high-pass filter and λ, the regularization parameter, controls the tradeoff between deconvolution and noise smoothing. According to the set theoretic approach followed here, λ is determined in advance, based on the assumed knowledge about the original image [12]–[14]. The same form of the restored image is obtained by the constrained least squares approach [11], with the main difference being that λ is unknown and needs to be deter-

Manuscript received July 26, 1990; revised January 29, 1991. This work was supported in part by the National Science Foundation under Grant MIP-8614217.

J. K. Paik was with the Department of Electrical Engineering and Computer Science, McCormick School of Engineering and Applied Science, The Technological Institute, Northwestern University, Evanston, IL. He is now with the Video Design Department, Sampung Electronics, Buchan, Korea.

A. K. Katsaggelos is with the Department of Electrical Engineering and Computer Science, McCormick School of Engineering and Applied Science, The Technological Institute, Northwestern University, Evanston, IL 60208-3118.

IEEE Log Number 9104331.

mined from the data in an iterative fashion. It should be noted that, if we denote by t_{ij} the elements of the matrix T, it holds in general that

$$t_{ii} = \tilde{d}_{ii} + \lambda \tilde{c}_{ii} > 0, \qquad i = 1, \cdots, n \tag{5}$$

since all $\tilde{d}_{ii}$, $\tilde{c}_{ii}$, and λ are non-negative, where $\tilde{d}_{ii}$ and $\tilde{c}_{ii}$ are the diagonal elements of the matrices D^TD and C^TC, respectively.

Among the various neural networks, Hopfield's model [9] is suitable for solving optimization problems, such as the restoration problem [10], [21], [22]. This network consists of n interconnected nonlinear devices (neurons). Interconnection weights w_{ij}, $i = 1, \cdots, n$, $j = 1, \cdots, n$ are associated with neurons i and j and a bias or threshold term b_i is attached to each neuron. The state of the network at time t is denoted by $v(t) = (v_1(t), v_2(t), \cdots, v_n(t))^T$, where $v_i(t)$ is the state of neuron i. The state of a neuron at time $(t + 1)$ is computed by

$$v_i(t + 1) = G\left(\sum_{j=1}^{n} w_{ij} v_i(t) + b_i\right) \tag{6}$$

where

$$G(u) = \begin{cases} 1, & u \geq 0 \\ 0, & u < 0. \end{cases} \tag{7}$$

Clearly, each neuron is a linear threshold element with the states of the other neurons being its inputs and b_i being the threshold. The convergence properties of Hopfield's network depend on the structure of W (the matrix with elements w_{ij}) and the updating mode. An important property of the Hopfield model is that if it operates in a sequential mode and W is symmetric with non-negative diagonal elements, then the energy function

$$\begin{aligned} E_{hs}(t) &= -\tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} v_i(t)\, v_j(t) - \sum_{i=1}^{n} b_i v_i(t) \\ &= -\tfrac{1}{2} v^T(t)\, Wv(t) - b^T v(t) \end{aligned} \tag{8}$$

is nonincreasing [6], [9]. Furthermore, the network always converges to a fixed point (to be defined in the next section).

Comparing (2) and (8) it is clear that the function $f(x)$ to be minimized for the restoration problem equals E_{hs} for $W = -T$ and $x = v$ (consider for the time being a binary image). However, since $w_{ij} = -t_{ij} < 0$, according to (5), the convergence of Hopfield's network, when applied to the solution of the restoration problem, is not guaranteed. Zhou *et al.* [22] proposed the use of Hopfield's network for solving the restoration problem as formulated above. In order to ensure convergence of the network, the reduction of the energy function $f(x)$ was checked at the update of each neuron. Such an approach causes two major problems: time delay caused directly by the checking step and difficulty in both parallelizing the algorithm and analyzing its convergence. Therefore, in this paper, we propose a modified Hopfield network model for solving the restoration problem, which introduces negligible extra computation over the original network. Its properties and convergence analysis are presented.

The fully parallel (n-simultaneous updates) and partially asynchronous updating modes are also considered for the modified network. The convergence of the original Hopfield network in a fully parallel mode has been studied in the literature [3], [4], [6], [7]. The main result, derived by Goles *et al.* [6], [7], is that if W is symmetric (no constraints on its diagonal elements) and the original network operates in a fully parallel mode, then the energy function

$$\begin{aligned} E_{hp}(t) = &-\tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} v_i(t)\, v_j(t-1) \\ &- \tfrac{1}{2} \sum_{i=1}^{n} b_i (v_i(t) + v_i(t-1)) \end{aligned} \tag{9}$$

is nonincreasing. Furthermore, the network will always converge to a fixed point or a cycle of length two (meaning that there may be two distinct states v^* and v^{**} and the updating results in $v^*v^{**}v^*v^{**} \cdots$, repeating forever). The known convergence properties of the Hopfield model were also rederived by Bruck *et al.* [3], [4] in a simple way, without the use of an energy function, by showing the equivalence between finding the maximum value of the energy function and a minimum cut in an undirected graph.

We observe that the energy functions in (8) and (9) differ. As reported by Bruck and Goodman [3] in a fully parallel mode E_{hs} is not necessarily nonincreasing. A sufficient condition for E_{hs} to be nonincreasing in this case is that W is non-negative definite over the range $(-1, 0, 1)$. Clearly, for the restoration problem under consideration, the minimization of E_{hs} (or equivalently of $f(x)$), and not of E_{hp}, is sought. Since $W = -T$ in this case is nonpositive definite, the nonincrease of E_{hs} is not guaranteed. Therefore, we propose the use of the modified network in fully parallel and partially asynchronous modes, and analyze their convergence.

In using the Hopfield model for the restoration problem, a scheme for representing the image gray levels in terms of the binary state variables has to be chosen. The binary, simple sum and group-and-weight schemes are discussed by Takeda and Goodman [20] in the context of the Hopfield model. In the implementation of Zhou *et al.*, [22] the image gray levels are represented by the simple sum of the binary neuron state variables. Although this is a fault-tolerant implementation it results in tremendous storage and a large number of interconnections. In reducing the computational complexity, sequential accumulation of excited neurons was suggested. Such an approach can eliminate interconnections among redundant neurons, but the following problems still remain: i) most of the redundant neurons are not changed during the complete iteration step, and ii) only a sequential update of redundant neurons for each variable is possible, in other words, the update of redundant neurons for each variable cannot be parallelized. On the other hand, the binary represen-

tation scheme for the image gray levels has different problems, such as: i) lack of robustness, that is, an error, for example, in the most significant bit results in a large restoration error, and ii) all neurons must be checked for each variable no matter how small the change in the variable is. We discuss an implementation whereby a nonelementary neuron is required.

The paper is organized as follows: in Section II a modified Hopfield network and the corresponding updating rule for obtaining a minimum of $f(x)$ in (2), are proposed. In Sections III–V algorithms with various updating modes are proposed and analyzed, based on the updating rule of Section II. Finally, experimental results are presented in Section VI and Section VII concludes the paper.

II. Description of the Modified Hopfield Network

In this section we first briefly review the implementation of the restoration algorithm (2)–(4) using Hopfield's neural network [9], [10]. The block diagram of the network is shown in Fig. 1. The symbol s_i denotes a switch; the order by which the switches are operated determines the updating mode (i.e., sequential, asynchronous, etc). The interconnection weights are denoted by $-t_{ij}$ to reinforce the point that they are equal to w_{ij} in (8) in the original network. Each block in Fig. 1 can be replaced by the network of Fig. 2(a) or (b), if a simple sum or a binary sum representation schemes are used, respectively [20]. Zhou *et al.* [22] proposed the use of the network of Fig. 1 for solving the restoration problem of (2)–(4) with a simple sum number representation scheme and sequential updates. Let the subscript i denote the ith element of a vector, or the value of a variable when the ith element of a vector is used for its evaluation. Then their algorithm has the following form:

Algorithm 1 (Simple sum)

For $i = 1, \cdots, n$ do

For $k = 0, \cdots, 255$ do

$$u_i = b_i - \sum_{j=1}^{n} t_{ij} x_j \tag{10}$$

$$\Delta x_i = d(u_i) = \begin{cases} -1, & u_i < 0 \\ 0, & u_i = 0 \\ 1, & u_i > 0 \end{cases} \tag{11}$$

$$\Delta f = \tfrac{1}{2} t_{ii} (\Delta x_i)^2 - u_i \, \Delta x_i \tag{12}$$

$$\text{If } \Delta f < 0, \quad \text{then } v_{i,k} = v_{i,k} + \Delta x_i$$

$$x_i = \sum_{k=0}^{255} v_{i,k} \tag{13}$$

□

where $v_{i,k}$ is the state of the neuron (i, k) at the kth location of the gray level range of the ith pixel, in the lexicographically ordered image, and Δf represents the change in the energy function $f(x)$ in (2), at each iteration step. The check of the sign of Δf is necessary, since as was mentioned in the introduction, energy reduction is not guaranteed because $-t_{ii} = w_{ii} < 0$. The algorithm uses the available degraded image as an initial value, that is, $x_i = b_i$ and $v_{ik} = b_{ik}$, the value of the kth bit in the simple sum representation of b_i. The algorithm with a binary sum representation scheme can be easily obtained from Algorithm 1, by allowing k to vary from 0 to 7 and substituting (13) by

$$x_i = \sum_{k=0}^{7} 2^k v_{i,k}. \tag{14}$$

We now propose a modified Hopfield network for image restoration [18], [19], which improves upon the implementation of Fig. 1. The block diagram of the proposed modified Hopfield model is shown in Fig. 3. The corresponding updating rule (henceforth, called UR 1) is given by the following equations.

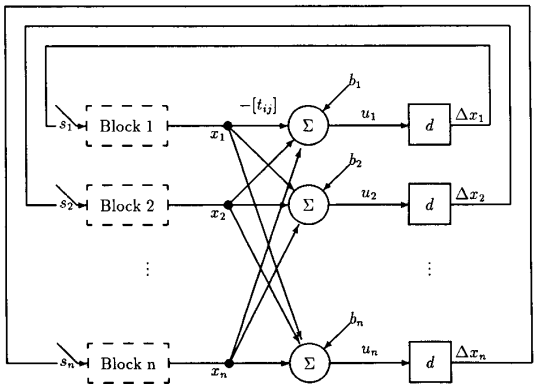


Fig. 1. Block diagram of the original Hopfield model applied to image restoration.

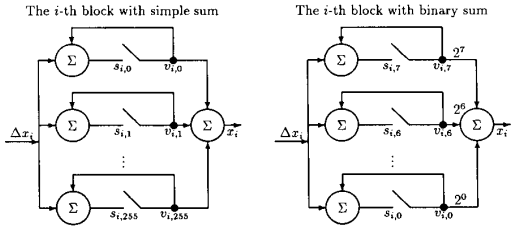


Fig. 2. Block diagram of two different number representation schemes corresponding to block i in the Hopfield network model. (a) Simple sum scheme. (b) Binary sum scheme.

A. Updating Rule 1

$$x_i(t+1) = g(x_i(t) + \Delta x_i), \quad i = 1, \cdots, n \tag{15}$$

where

$$g(v) = \begin{cases} 0, & v < 0 \\ v, & 0 \le v \le 255 \\ 255, & v > 255 \end{cases} \tag{16}$$

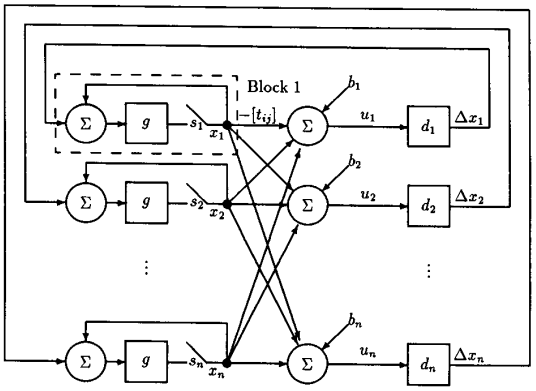


Fig. 3. Block diagram of the modified Hopfield network model applied to image restoration.

$$\Delta x_i = d_i(u_i) = \begin{cases} -1, & u_i < -\theta_i \\ 0, & -\theta_i \le u_i \le \theta_i, \\ 1, & u_i > \theta_i \end{cases}$$

$$\text{where } \theta_i = \tfrac{1}{2} t_{ii} > 0 \tag{17}$$

$$u_i = b_i - \sum_{j=1}^{n} t_{ij} x_j(t) = e_i^T(b - Tx(t)) \tag{18}$$

□

and e_i represents the ith unit vector. The available degraded image is used as initial condition for x, as with Algorithm 1. The main difference between the previous updating rule and that used in Algorithm 1 is in (17). That is, if $t_{ii} = 0$, which results in $\theta_i = 0$, the updating rule of (11) is obtained. As will be shown in the next section, the implication of (17) is that when sequential updates are assumed, the proposed updating rule guarantees energy reduction with negative autoconnections, thus eliminating the check of the sign of Δf at the update of each neuron. A similar result is demonstrated by Yeh *et al.* [21].

For UR 1, (13) or (14) can be used for the representation of the image gray levels. On the other hand, we view x_i as the state of a nonelementary neuron which takes discrete values between 0 and 255, instead of binary values. This consideration is possible because the interconnections are determined in terms of pixel locations and not gray level values, as can be seen from Fig. 3 (see also [22]). This nonelementary neuron uses eight bits, and due to its assumed "internal hardware" it can increase or decrease its value by 1 or keep the same value, in one step. Therefore, on one hand it avoids the excessive storage requirements of the simple sum representation scheme, while on the other hand, it avoids the checking steps for each update of the variable x_i with the binary sum scheme. In the following the implementation in Fig. 3 with UR 1 will be used and analyzed. This implementation issue is of secondary importance, since equivalent results can be obtained by replacing the ith block in Fig. 3 by the network of Fig. 2(a) or (b).

III. A Sequential Algorithm

In this section we present and analyze an algorithm which sequentially updates each pixel value according to UR 1. For the analysis to be followed let $l(t)$ denote a partition of the set $\{1, \cdots, n\}$. Then the algorithm has the following form (for the definition of the various quantities see (15)–(18)).

Algorithm 2 (A sequential algorithm)

0) $x(0) = D^T z$; $t := 0$ and $i := 1$.
1) Check termination.
2) Choose $l(t) = \{i\}$.
3) temp $= g(x(t) + \Delta x_i e_i)$ where Δx_i is given by UR 1.
4) If temp $\neq x(t)$ then $x(t+1) :=$ temp and $t := t + 1$.
5) $i := i + 1$ (if $i > n$, $i = i - n$) and go to step 1.

□

In step 3) of the above algorithm the function $g(\cdot)$ is used with a vector as an input. In this case $g(x) = (g(x_1), \cdots, g(x_n))^T$, where $g(x_i)$ is defined by (16). The introduction of the set $l(t)$ may seem redundant at this point, although it will be useful when other modes of updating are considered.

A. Convergence Analysis

We now show that algorithm (2) converges to a fixed point, which is a local minimum of $f(x)$, after a finite number of iterations. The converse is also shown to be true, that is, a local minimum of $f(x)$ is a fixed point of the proposed sequential algorithm. In order to show these results the following definitions are required.

Definition 1 (Solution Space): Let us define the integer subset W of the set of integers I by

$$W = \{0, 1, 2, \cdots, 255\} \subset I. \tag{19}$$

□

If $x(0) \in W^n = W \times W \times \cdots \times W$, any solution obtained according to UR 1 satisfies

$$x(t) \in W^n, \quad \text{for } t = 0, 1, \cdots. \tag{20}$$

Definition 2 (Fixed Point): $x(t) \in W^n$, $t = 0, 1, \cdots$, is a fixed point of an iterative algorithm on W^n if and only if

$$x(t+1) = x(t). \tag{21}$$

□

Definition 2 also means that $x(t)$ is a fixed point if and only if

$$x_i(t+1) = x_i(t), \quad \forall i. \tag{22}$$

Using the above definitions we obtain the following corollary.

Corollary 1: $x(t) \in W^n$, $t = 0, 1, \cdots$, is a fixed point of UR 1 if and only if each $x_i(t)$ satisfies one of the following three conditions.

i) $\Delta x_i = 0$ ($|u_i| \leq (1/2)t_{ii}$).
ii) $\Delta x_i = -1$ ($u_i < -(1/2)t_{ii}$) and $x_i(t) + \Delta x_i < 0$.
iii) $\Delta x_i = 1$ ($u_i > (1/2)t_{ii}$) and $x_i(t) + \Delta x_i > 255$. □

Note that Definitions 1 and 2 are independent of UR 1, but Corollary 1 provides the three cases which result in a fixed point of UR 1, in terms of u_i and bound constraints.

Definition 3 (Local Minimum): $x \in W^n$ is a local minimum of f: $W^n \mapsto \Re$ in (15) if and only if

$$f(y) - f(x) \geq 0 \tag{23}$$

where

$$y = g(x + \Delta x_i e_i), \qquad \forall i \text{ and } \Delta x_i \in \{-1, 1\}. \tag{24}$$

□

With the above definitions, the following theorems describe the convergence properties of Algorithm 2. The proofs of all theorems to be presented in this paper are shown in Appendix I.

Theorem 1 (Energy Reduction Property): Let x be any element in W^n and

$$y = g(x + \Delta x_i e_i), \qquad \text{for any } i \text{ and } \Delta x_i \in \{-1, 0, 1\} \tag{25}$$

where Δx_i is given by UR 1. Then it is guaranteed that

$$\Delta f = f(y) - f(x) = \tfrac{1}{2}t_{ii}a_i^2 - u_i a_i \leq 0 \tag{26}$$

where

$$a_i = (y - x)^T e_i. \tag{27}$$

The equality holds if and only if $y = x$. □

Based on Theorem 1 the following theorem is shown.

Theorem 2 (Fixed Point/Local Minimum): If $x \in W^n$ is a local minimum of f according to Definition 3, it is a fixed point of UR 1, and vice versa. □

Due to Theorem 2 the algorithm terminates if and only if $x(t)$ reaches a local minimum of f. Furthermore, this local minimum is reached after a finite number of iterations, according to the following theorem.

Theorem 3: Algorithm 2 converges to a fixed point after a finite number of iterations. □

Since Algorithm 2 allows only for integer steps, the next important question is how "good" is the solution obtained? In other words, what is the distance of the local minimum obtained by the algorithm, from the local minimum of $f(x)$ in (2), when infinite precision arithmetic is available? An estimate of this distance is given by the following theorem.

Theorem 4: Consider a fixed point x^* of Algorithm 2. If we assume that no bound constraints are activated at x^*, then

$$\|u^*\|_1 \leq \tfrac{1}{2} \text{ trace } (T) \tag{28}$$

where trace $(T) = \Sigma_{i=1}^{n} t_{ii}$. □

It is interesting to note that the upper bound on the l_1 norm of the residual depends on the amount of noise on the data and the severity of the distortion, for a given n. In other words, let us consider the case when the degradation is space invariant and C in (4) also represents a space-invariant filter. Then, $t_{ii} = \tilde{d}_{ii} + \lambda \tilde{c}_{ii}$, for all i, and trace $(T) = n \cdot t_{ii}$, where $\tilde{d}_{ii}$ and $\tilde{c}_{ii}$ are the diagonal positive elements of the matrices D^TD and C^TC, respectively. Clearly, for noise-free data ($\lambda = 0$) the resulting upper bound is smaller than the bound obtained by considering the same D but a certain amount of noise ($\lambda > 0$). This result confirms our intuition. On the other hand, let us consider the noise-free data case ($\lambda = 0$) with two types of degradation—a severe one (t_{ii}^s) and a mild one (t_{ii}^m). In this case, $t_{ii}^s \leq t_{ii}^m$, due to the fact that the impulse response of the degradation system is normalized. This result is somewhat counter-intuitive, since a severely blurred image may be restored more accurately than a mildly blurred one by Algorithm 2.

B. A Decoupled Parallel Algorithm

Due to the fact that the support of the impulse response of the degradation system is finite, matrix T in (4) is banded. Then the following decoupled parallel algorithm is proposed [17], which is a direct modification of Algorithm 2.

Algorithm 3

0) $x(0) = D^Tz$; $t := 0$.
1) Check termination.
2) Choose $l(t)$ such that $t_{ij} = 0$, for $i \neq j$ and $i, j \in l(t)$.
3) temp $= g(x(t) + \Sigma_{i \in l(t)} \Delta x_i e_i)$ where Δx_i is given by UR 1.
4) If temp $\neq x(t)$ then $x(t + 1) :=$ temp and $t := t + 1$; go to step 1). □

Algorithm 3 has the same properties as Algorithm 2. The basic difference is that the computation time is now reduced by a factor equal to the number of elements in $l(t)$. The main result is summarized by the following theorem.

Theorem 5: Algorithm 3 leads $x(t) \in W^n$ to a fixed point after a finite number of iterations.

IV. Algorithms with n-Simultaneous Updates

In the previous section we have shown that Algorithm 2 converges to a fixed point (which is a local minimum of f) after a finite number of iterations without checking the reduction of the energy at the update of each neuron. In spite of this major improvement over the algorithm using the original Hopfield network of Zhou *et al.* [22], Algorithm 2 still has the inefficiency of *sequential updates*. In running a sequential algorithm, all neurons, except from the one that is currently updated, are idle, waiting for their turns. Therefore, the behavior of the algorithm with n-simultaneous updates as described by UR 1 is analyzed next.

A. The Algorithm with n-Simultaneous Updates

The algorithm with n-simultaneous updates according to UR 1 has the following form.

Algorithm 4

0) $x(0) = D^T z$; $t := 0$.
1) Check termination.
2) Choose $l(t) = \{1, 2, \cdots, n\}$.
3) temp $= g(x(t) + \Sigma_{i=1}^{n} \Delta x_i e_i)$ where Δx_i is given by UR 1.
4) If temp $\neq x(t)$ then $x(t+1) =$ temp and $t = t + 1$.
5) Go to step 1. □

Algorithm 4 uses a unit length step; thus we cannot expect the solution to be arbitrarily close to the minimum of the functional $f(x)$ in (2). The following theorem summarizes the behavior of Algorithm 4.

Theorem 6: According to Algorithm 4 if

$$|u_i| > \tfrac{1}{2}\|T\|_2, \qquad \forall i \text{ for which } \Delta x_i \neq 0 \tag{29}$$

then f decreases; otherwise if

$$|u_i| \leq \tfrac{1}{2}\|T\|_2, \qquad \text{for some } i \tag{30}$$

then f may increase. □

According to the above theorem, when $x(t)$ is far from the solution, which means that the l_1 norm of the residual is large, the algorithm decreases f, and therefore, $x(t)$ moves closer to the solution. On the other hand, if $x(t)$ is sufficiently close to the solution, the algorithm is not guaranteed to decrease f any more. Therefore, there is the need for an approach which will "force" the algorithm to reach a fixed point in the later case. Such an approach is described in the next section.

B. A Greedy Algorithm

Once $x(t)$ satisfies (30) in Theorem 6, in order to force the algorithm to terminate, a greedy algorithm can be used. Such an algorithm has the following form.

Algorithm 5: After step 4) in Algorithm 4, check the sign of the change in f, that is:

$$\Delta f = f(x(t+1)) - f(x(t)). \tag{31}$$

If it becomes positive, increase θ_i in (17) by

$$\theta_i = \delta\theta_i, \qquad \delta > 1. \tag{32}$$

Otherwise θ_i is unchanged and proceed with Algorithm 4. □

Theorem 7: Algorithm 5 converges to a fixed point. □

The fixed point, however, is not guaranteed to be a local minimum of f, since we have arbitrarily increased θ_i. On the other hand, the l_1 norm of the residual at that point is bounded, according to the following theorem.

Theorem 8: Consider a fixed point x^* of Algorithm 5. If we assume that no bound constraints are activated at x^*, there exists an upper bound of $\|u^*\|_1$, such as

$$\|u^*\|_1 \leq U_{\text{greedy}} \tag{33}$$

where $U_{\text{greedy}} = (n/2)\delta^q t_{\max}$, q represents the number of times θ_i is increased and $t_{\max} = \max_i \{t_{ii}\}$. □

Clearly, if we set $\delta = (\|T\|_2/t_{\max})^{(1/q)}$, we obtain

$$U_{\text{greedy}} = \frac{n}{2}\|T\|_2. \tag{34}$$

In this case, for $\|T\|_2 = 1$ (which is the case when D and C represent, respectively, low-pass and high-pass shift-invariant filters [12], [13]), we obtain $U_{\text{greedy}} = n/2$. This bound is $1/t_{ii}$ times greater than the bound obtained when Algorithm 2 is used, according to (28). As expected, the approximation error most probably is larger with n-simultaneous that with sequential updates.

In running Algorithm 5, the computational load increases significantly, since it is necessary to evaluate f or Δf at each iteration step. One way to reduce this extra computation is to check the sign of the following quantity:

$$\sum_{i \in l(t)'} (|u_i| - \|T\|_2) \tag{35}$$

and increase θ_i if (35) becomes negative, where $l(t)'$ is the set of processors with nonzero updates. Since u_i has already been computed and $\|T\|_2$ is a given constant, the computation of (35) requires only $2n$ summations at the most. A positive Δf always results in a negative value of (35), but the inverse is not true.

C. A Modified Hopfield Network with Higher Resolution

Although the modified Hopfield network and the corresponding updating rule (UR 1) are using unit nonzero step size, i.e., $|\Delta x_i| = 1$, a smaller step size can be used for achieving higher resolution. In this case the algorithm converges to a fixed point which is closer to the local minimum point obtained by infinite precision arithmetic, in terms of the l_1 norm of the residual. One such approach for achieving higher resolution is to use additional bits for each modified neuron. That is, if we are using p bits per neuron ($p \geq 8$), the following updating rule replaces UR 1.

1) Updating Rule 2:

$$x_i(t+1) = g(x_i(t) + \Delta x_i) \tag{36}$$

where

$$\Delta x_i = \begin{cases} -2^{(8-p)}, & u_i < -2^{(7-p)} t_{ii} \\ 0, & |u_i| \leq 2^{(7-p)} t_{ii} \\ 2^{(8-p)}, & u_i > 2^{(7-p)} t_{ii} \end{cases} \tag{37}$$

$$u_i = b_i - \sum_{j=1}^{n} t_{ij} x_j(t) \tag{38}$$

and

$$g(v) = \begin{cases} 0, & v < 0 \\ v, & 0 \le v \le 255 \\ 255, & v > 255. \end{cases} \tag{39}$$

□

Based on this modified updating rule, tighter bounds on the residual are obtained for both algorithms with sequential and n-simultaneous updates. Such a bound is given by the following theorem for the latter case.

Theorem 9: Consider a fixed point x^* of Algorithm 5. If we assume that no bound constraints are activated at x^*, there is an upper bound of $\|u^*\|_1$, such that

$$\|u^*\|_1 \le U_{\text{greedy}.p} = 2^{(7-p)}n\|T\|_2. \tag{40}$$

□

According to the above theorem, if we use 9 b per neuron, the upper bound of $\|u^*\|_1$ is halved. If the original Hopfield network with a simple sum number representation scheme were used, then 256 additional neurons would have been required in this case.

V. A Partially Asynchronous Algorithm

Although Algorithm 5 allows for n-simultaneous updates, it still faces synchronization penalties. For example, if the communication or updating speed of each neuron is not the same, all neurons should wait until the slowest neuron completes its updating and communication. In order to avoid synchronization penalties, asynchronous algorithms are used, which allow for time delays in accessing other neurons.

Asynchronous algorithms are divided into two classes: *totally asynchronous* or *chaotic*, and *partially asynchronous* algorithms [2]. Totally asynchronous algorithms allow arbitrarily large time delays in accessing each neuron. On the other hand, partially asynchronous algorithms place an upper bound on this delay. In this section we deal with partially asynchronous algorithms only. The totally asynchronous case is simply mentioned as a limiting case, by arbitrarily increasing the upper bound on the delay.

Let us first define the time delay for each neuron in accessing another neuron.

Definition 4: We define by $x^i(t) \in W^n$ the available pixel values for updating $x_i(t)$ at time t, that is:

$$x^i(t) = (x_1(\tau_1^i(t)), x_2(\tau_2^i(t)), \cdots, x_n(\tau_n^i(t))) \tag{41}$$

where $0 \le \tau_j^i(t) \le t$, for all j and $i \in l(t)$. □

Then the time delay for neuron i to access neuron j is given by

$$t - \tau_j^i(t). \tag{42}$$

The following assumption describes the required restrictions on the delay for the partial asynchronism under consideration.

Assumption 1:

i) A neuron uses its current value, i.e.:

$$\tau_i^i = t, \qquad \forall i \in l(t). \tag{43}$$

ii) The time delay in accessing any pixel is bounded, that is:

$$\max\{0, t - B + 1\} \le \tau_j^i(t) \le t \tag{44}$$

where B is a prespecified constant. □

The algorithm described by UR 1 with the partially asynchronous updates is not guaranteed to converge, since Algorithm 4, which is a special case of a partially asynchronous algorithms, does not converge. On the other hand, it is unrealistic to use Algorithm 5 in the asynchronous mode, since it is not easy to compute Δf at each iteration. Therefore, the following algorithm is proposed.

Algorithm 6:

0) $x(0) = D^T z$; $t := 0$.
1) Check termination.
2) Assume that $l(t) \subseteq \{1, 2, \cdots, n\}$ is given.
3) temp $= g\{x(t) + \Sigma_{i \in l(t)} \Delta x_i e_i\}$ where

$$\Delta x_i = \begin{cases} d(u_i), & i \in l(t) \\ 0, & \text{otherwise} \end{cases} \tag{45}$$

$$u = b - Tx^i(t) \tag{46}$$

and

$$d(u_i) = \begin{cases} -1, & u_i < -\theta \\ 0, & -\theta \le u_i \le \theta, \\ 1, & u_i > \theta \end{cases}$$

$$\text{where } \theta = \tfrac{1}{2}\|T\|_2(2B - 1). \tag{47}$$

4) If temp $\neq x(t)$ then $x(t+1) =$ temp and $t = t + 1$.
5) Go to step 1. □

The following two theorems guarantee the convergence of the above algorithm and also offer a bound on the residual error.

Theorem 10: Algorithm 6 converges to a fixed point. □

Theorem 11: Consider a fixed point x^* of Algorithm 6. If we assume that no bound constraints are activated at x^*, there exists an upper bound of $\|u^*\|_1$, such as

$$\|u^*\|_1 \le U_{\text{asynch}} \tag{48}$$

where

$$U_{\text{asynch}} = \frac{n}{2}\|T\|_2(2B - 1). \tag{49}$$

□

From the above theorem it is clear that if $B = 1$ is used, Algorithm 6 behaves like a synchronous algorithm. That

is, in this case we obtain

$$\|u^*\|_1 \leq \frac{n}{2}\|T\|_2 \tag{50}$$

which is the result in (34). On the other hand as B increases, the upper bound U_{asynch} increases proportionally. Therefore, the totally asynchronous version of Algorithm 6 ($B = \infty$) becomes impractical.

VI. Experimental Results

In this section simulation results obtained with the proposed algorithms are presented. In a simulation environment where the original signal is available, the improvement in signal-to-noise ratio (SNR), denoted by Δ_{SNR}, is used as an objective measure of the quality of the restored signal. It is defined in decibels by

$$\Delta_{\text{SNR}} = 10 \log_{10} \frac{\|z - x\|^2}{\|x(t) - x\|^2} \tag{51}$$

where z, x, and $x(t)$ represent, respectively, the distorted, original, and restored signal at the tth iteration step.

Experiment 1: In demonstrating the behavior of the algorithms with n-simultaneous updates, an image line is used. It is blurred by 1×9 motion without noise, and restored by Algorithms 4 and 5. In this case $n = 256$ and $\|T\|_2 = 1$.

The change in energy Δf at each iteration is plotted in Fig. 4, and the corresponding threshold values θ_i are plotted in Fig. 5, when Algorithm 5 is run using 8, 9, and 10 neurons per pixel. When 8 b per neuron are used, for example, Algorithms 4 and 5 run identically, until Δf reaches its first positive value at the 49th iteration, as shown in Fig. 4 (curve (a)). Up to this iteration, condition (29) holds. As the iteration continues, Δf becomes positive three times at the 65th, 75th, and 77th iterations, where the threshold θ_i is increased twice, as shown in Fig. 5 (curve (a)).

When both Algorithms 4 and 5 are run, $\|u\|_1$ and Δ_{SNR} at each iteration are plotted in Figs. 6 and 7, respectively. In running Algorithm 4 with 8 b per neuron $\|u\|_1$ does not become larger than $(n/2)\|T\|_2 = 128$, as shown in Fig. 6. Similarly, in the cases of 9 and 10 b, $\|u\|_1$ stays below $(n/2 \cdot 2)\|T\|_2 = 64$ and $(n/2 \cdot 4)\|T\|_2 = 32$, as shown in Fig. 6 (curves (e) and (f), respectively). When we consider Algorithm 5 for the same cases, $\|u\|_1$ decreases right after Δf obtains positive values and the iteration terminates with much smaller error as shown in Figs. 6 (curves (a)–(c)). The corresponding values of $\|u\|_1$, at termination are 41.0, 16.4, and 8.9 for Algorithm 5, and 125.5, 59.7, and 31.3 for Algorithm 4. Algorithm 5 results in higher Δ_{SNR} than Algorithm 4, as shown in Fig. 7. The results in Figs. 6 and 7 clearly agree with Theorems 6 and 7.

Experiment 2: In demonstrating the performance of the partially asynchronous algorithm an image line is blurred by 1×9 motion without noise, and restored by Algorithm 6, using $B = 2$, 3, and 4. In this case $n = 256$ and

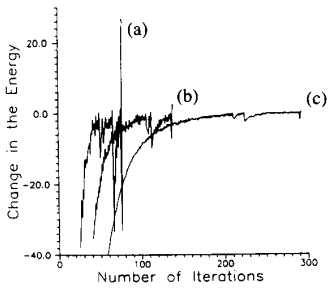


Fig. 4. Change in the energy at each iteration produced by Algorithm 5 using (a) 8 neurons, (b) 9 neurons, and (c) 10 neurons per pixel.

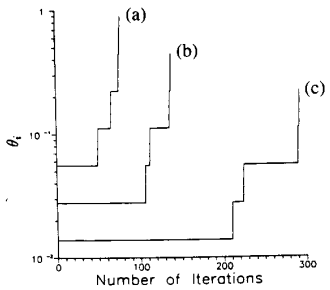


Fig. 5. Change of the threshold θ_i at each iteration in running Algorithm 5 using (a) 8 neurons, (b) 9 neurons, and (c) 10 neurons per pixel.

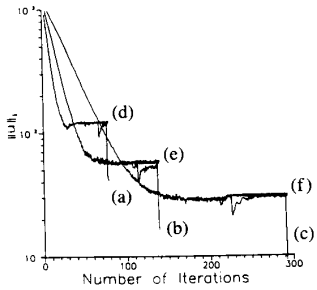


Fig. 6. $\|u\|_1 = \|b - Tx\|_1$ in running Algorithm 5 using (a) 8 neurons, (b) 9 neurons, and (c) 10 neurons, and in running Algorithm 4 using (d) 8 neurons, (e) 9 neurons, and (f) 10 neurons per pixel.

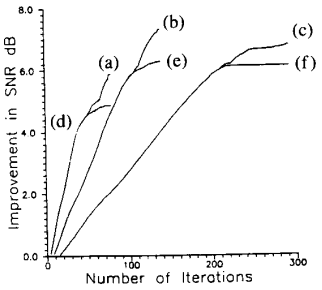


Fig. 7. Improvement in SNR in running Algorithm 5 using (a) 8 neurons, (b) 9 neurons, and (c) 10 neurons, and in running Algorithm 4 using (d) 8 neurons, (e) 9 neurons, and (f) 10 neurons per pixel.

$\|T\|_2 = 1$. For each B, the upper bound on the l_1 norm of the residual computed by (49), the number of iterations required, the l_1 norm of the residual at the fixed point,

TABLE I
RESTORATION RESULTS OBTAINED BY THE PARTIALLY ASYNCHRONOUS ALGORITHM

B	U_{asynch}	Iteration	$\|u^*\|_1$	Δ_{SNR}
2	384	47	101.9	4.2638
3	640	49	261.1	3.8234
4	896	48	356.3	3.6091

and Δ_{SNR} are summarized in Table I. As expected, in agreement with Theorem 11, as the upper bound on the delay B increases, Δ_{SNR} decreases and $\|u^*\|_1$ increases.

Experiment 3: In this experiment, various algorithms for the restoration of two images degraded by 1×9 motion blur without noise and with 20-dB SNR additive Gaussian noise, are compared. More specifically, the following algorithms have been tested.

Test 1. The algorithm proposed by Zhou *et al.* [22].

Test 2. Algorithm 2 with sequential updates.

Test 3. Algorithm 3 with 8 decoupled parallel updates.

Test 4. Algorithm 5 with n-simultaneous updates and $\delta = 2$.

Test 5. The same algorithm of test 4, except that 1 and 10% of randomly chosen neurons are disabled at each iteration, for the noiseless and the 20-dB image, respectively.

At each iteration, the following quantities are computed:

- the change in the energy function Δf;
- the norm of the change in the iterate x, that is:

$$\|x(t) - x(t-1)\|_2 \tag{52}$$

which also equals the number of neurons changed at the tth iteration:

- the improvement in SNR, Δ_{SNR}.

The iteration terminates when no neurons are changed. The original lady image and the distorted image without noise are shown in Fig. 8(a) and (b), respectively. The test results are summarized in Table II and the corresponding restored images are shown in Figs. 9 and 10.

Tests 1 and 2 give identical restoration results, although the algorithm in test 1 needs to compute the change in the energy function at the update of each neuron while the algorithm in test 2 does not. Faster convergence is achieved for the image with 20-dB additive Gaussian noise, since the interconnection matrix T, given in (4), becomes better conditioned with a nonzero λ.

Test 3 gives the same restoration results with experiments 1 and 2, but the computation time is reduced by a factor of 8, which equals the number of processors used in parallel. In general, the computation time may be reduced by a factor of m, neglecting the communication overhead, where m is the number of processors in the multiprocessor computer and $2 \leq m \leq \lfloor n/(L_1 \times L_2) \rfloor$, where $L_1 \times L_2$ is the support of the filter represented by T and $\lfloor\ \rfloor$ denotes the integer part of the resulting division. In our example, the practical range of m is $2 \leq m \leq 7281$ since $n = 65\,636$, $L_1 = 9$, and $L_2 = 1$.

(a)

(b)

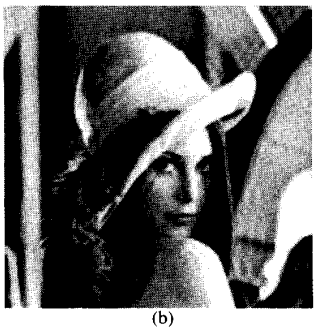
Fig. 8. (a) Original image. (b) Image distorted by 1×9 motion without noise.

TABLE II
COMPARISON OF THE PERFORMANCE OF THE VARIOUS HOPFIELD NETWORK-BASED ALGORITHMS

	No Noise, $\lambda = 0$		SNR = 20 dB, $\lambda = 0.5$	
Tests	No. of Iter.	Δ_{SNR}	No. of Iter.	Δ_{SNR}
1	187	8.66	56	2.38
2	187	8.66	56	2.38
3	187	8.66	56	2.38
4	104	5.90	68	2.51
5	206	5.86	108	2.35

Since Algorithm 4 does not converge, Algorithm 5 with n-simultaneous updates was used. For the image without noise, test 4 converges faster than tests 1–3, but the objective quality, based on Δ_{SNR}, is not as good. For the image with 20-dB noise, test 4 requires a few more iterations resulting in a higher Δ_{SNR}. The explanation of this latter result is that the point the algorithm was forced to converge to is closer to the original image than the regularized solution obtained by tests 1–3. This is in agreement with the behavior of iterative regularized image restoration algorithms, according to which Δ_{SNR} increases and then decreases before convergence [13], [14]. Test 5 is shown to converge to similar restoration results but it takes approximately twice as many iterations as test 4. Finally, in restoring the blurred image with no noise,


(a)


(b)


(c)

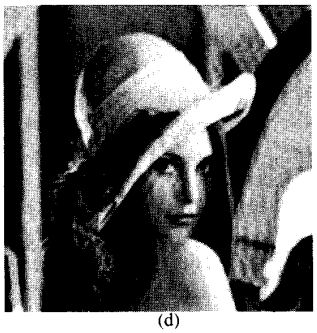
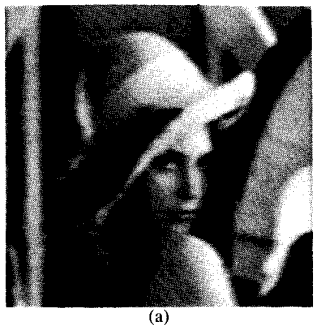
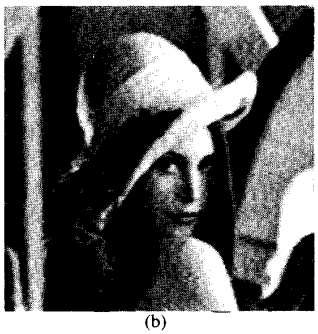
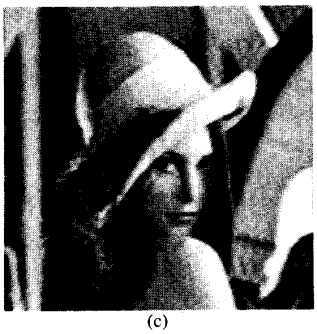

(d)

Fig. 9. Images restored for 1 × 9 motion without noise. (a) Image restored by test 3. (b) Image restored by test 4. (c) Image restored by test 5. (d) Image restored by test 4 with two times higher resolution.


(a)


(b)


(c)

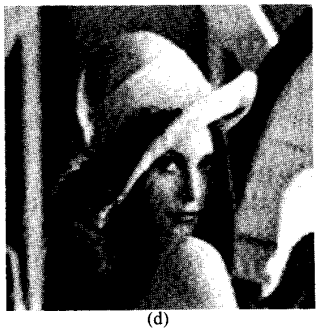

(d)

Fig. 10. Images restored for 1 × 9 motion with 20-dB noise. (a) Image distorted by 1 × 9 motion with 20-dB noise. (b) Image restored by test 3. (c) Image restored by test 4. (d) Image restored by test 5.

when the algorithm of test 4 is used with 9 b/pixel, a higher improvement results (Δ_{SNR} = 6.32 dB), at the expense of more iterations (199 iterations), as expected. The restored image is shown in Fig. 9(d). As a general observation, it is well known that the objective criterion of merit Δ_{SNR} does not necessarily agree with the visual quality of

a restored image. Thus in comparing, for example, the restored images of Fig. 9, images (b) and (c) may have better visual quality (at least portions of the image) than images (a) and (d), due to the severe ringing effects apparent in the latter two images.

VII. Conclusions

In this paper we have proposed a modified Hopfield network model for image restoration. This model is used in developing algorithms with sequential, decoupled, n-simultaneous, and partially asynchronous modes of updates. In all cases the convergence and residual error analyses are presented. Depending on the implementation environment and the available resources one of the proposed algorithms should be chosen.

The algorithm with sequential updates improves upon the algorithm proposed by Zhou *et al.* [22], since it does not require the checking of the energy reduction at the update of each neuron. Faster computational times are achieved by considering algorithms with n-simultaneous updates, at the expense of accuracy of the solution. In turn, we have presented a modified updating rule which allows for trading accuracy with cost of implementation, as specified by the number of bits per pixel. According to this rule, for example, a greedy algorithm with n-simultaneous updates results in a residual error which can be made arbitrarily small. Since asynchronous implementations are very desirable because they avoid synchronization penalties, we have also proposed a partially asynchronous algorithm which allows a neuron to have a bounded delay in accessing other neurons. It has been shown that the norm of the residual error at the fixed point increases proportionally to the delay allowed. Asynchronous algorithms are also very attractive since optical asynchronous implementations have been reported in the literature [5].

Appendix

A. Proof of Theorem 1

Clearly the following cases occur:

i) $u_i > (1/2)t_{ii}$: $\Delta x_i = 1$, which results in either:
  (a) $x_i + \Delta x_i \le 255$, $a_i = 1$, and $\Delta f < 0$, or
  (b) $x_i + \Delta x_i > 255$, $a_i = 0$, and $\Delta f = 0$.

ii) $u_i < -(1/2)t_{ii}$: $\Delta x_i = -1$, which results in either
  (a) $x_i + \Delta x_i \ge 0$, $a_i = -1$, and $\Delta f < 0$, or
  (b) $x_i + \Delta x_i < 0$, $a_i = 0$, and $\Delta f = 0$.

iii) $|u_i| \le (1/2)t_{ii}$: $\Delta x_i = 0$, which results in $a_i = 0$ and $\Delta f = 0$. From i) to iii), it can be seen that for any u_i

$$f(y) - f(x) \le 0 \tag{A1}$$

and the equality holds if and only if $y = x$. (Q.E.D.)

B. Proof of Theorem 2

If we assume that x is not a fixed point of UR 1, there exists at least one $y \in W^n$, $y \ne x$, which satisfies

$$f(y) - f(x) < 0 \tag{A2}$$

where

$$y = g(x + \Delta x_i e_i), \quad \text{for some } i \text{ and } \Delta x_i \in \{-1, 1\} \tag{A3}$$

and Δx_i is given by UR 1. This contradicts the assumption that x is a local minimum of f.

In proving the reverse, let x be a fixed point, $y \in W^n$ be any neighbor of x, such as

$$y = g(x + \Delta x_i e_i), \quad \text{for any } i \text{ and } \Delta x_i = \{-1, 1\} \tag{A4}$$

and $a_i = (y - x)^T e_i \in \{-1, 0, 1\}$. Then the change in $f(x)$ by a_i, is given by

$$\Delta f = f(y) - f(x) = \tfrac{1}{2} t_{ii} a_i^2 - u_i a_i. \tag{A5}$$

We can have the following three cases in terms of u_i:

i) $|u_i| \le (1/2)t_{ii}$, which results in $a_i = 0$ and $\Delta f = 0$;
ii) $u_i > (1/2)t_{ii}$, which results in either:
  (a) $a_i = 1$ and $\Delta f > 0$, or
  (b) $a_i = 0$ and $\Delta f = 0$.

In this case $a_i = -1$ is prohibited according to Corollary 1) ii).

iii) $u_i < -(1/2)t_{ii}$, which results in either:
  (a) $a_i = -1$ and $\Delta f > 0$, or
  (b) $a_i = 0$ and $\Delta f = 0$.

In this case $a_i = 1$ is prohibited according to Corollary 1 iii).

From i) to iii), it always holds that

$$f(y) - f(x) \ge 0. \tag{A6}$$

Therefore, x is a local minimum of f. (Q.E.D.)

C. Proof of Theorem 3

A nonzero step in Algorithm 2 is given by

$$s(t) = x(t+1) - x(t) = \Delta x_i e_i, \quad \text{for } \Delta x_i \in \{-1, 1\}$$

since $x(t+1) \ne x(t)$ due to step 4) in Algorithm 2. In other words, $s(t)$ is defined only when both

$$|u_i| > \tfrac{1}{2} t_{ii} \quad \text{and } 0 \le x_i(t) + \Delta x_i e_i \le 255$$

are satisfied.

For a nonzero $s(t)$, we have that

$$\begin{aligned} f(x(t+1)) &= f(x(t) + s(t)) \\ &= f(x(t)) + s(t)^T(Tx - b) + \frac{1}{2} s(t)^T T s(t) \\ &= f(x(t)) + \Delta x_i e_i^T(Tx - b) + \frac{1}{2} \Delta x_i^2 t_{ii} \\ &= f(x(t)) + \Delta x_i(-u_i) + \frac{1}{2} \Delta x_i^2 t_{ii} \\ &= f(x(t)) - \Delta x_i^2 \left(\frac{u_i}{\Delta x_i} - \frac{1}{2} t_{ii} \right) \\ &= f(x(t)) - \alpha(t), \end{aligned} \tag{A7}$$

where

$$\alpha(t) = |u_i| - \tfrac{1}{2} t_{ii} > 0, \qquad \forall t = 0, 1, \cdots. \tag{A8}$$

Since f is bounded below

$$M \le f(x(t+1)) = f(x(t)) - \alpha(t) = f(x(0)) - \sum_{\tau=0}^{t} \alpha(\tau). \tag{A9}$$

If we assume that there exists infinitely many nonzero $s(t)$'s, we have

$$\sum_{t=0}^{\infty} \alpha(t) \leq f(x(0)) - M < \infty \tag{A10}$$

which implies $\lim_{t\to\infty} \alpha(t) = 0$. This contradicts (A8), because the number of values that $\alpha(t)$ can obtain is finite, since the number of values of $x(t)$ is finite. Therefore, after a finite number of iterations, all $s(t)$'s will be equal to zero. In other words, Algorithm 2 converges to a fixed point after a finite number of iterations. (Q.E.D.)

D. Proof of Theorem 4

The l_1 norm of u^* is given by

$$\|u^*\|_1 = \sum_{i=1}^{n} |u_i^*|. \tag{A11}$$

Since no bound constraints are activated:

$$|u_i^*| \leq \tfrac{1}{2} t_{ii}, \qquad \forall i \tag{A12}$$

according to Corollary 1(i). By substituting (A12) into (A11), we have

$$\|u^*\|_1 \leq \tfrac{1}{2} \sum_{i=1}^{n} t_{ii} = \tfrac{1}{2} \operatorname{trace}(T). \tag{A13}$$

(Q.E.D.)

E. Proof of Theorem 5

A nonzero step in Algorithm 3 is given by

$$s(t) = x(t+1) - x(t) = \sum_{i \in l(t)} \Delta x_i e_i \tag{A14}$$

where at least one Δx_i, $i \in l(t)$, is nonzero and $0 \leq x_i(t) + \Delta x_i e_i \leq 255$. Let us define the nonempty index set $l(t)'$ by

$$l(t)' = \{i | \Delta x_i \neq 0, \quad 0 \leq x_i(t) + \Delta x_i e_i \leq 255, \text{ and } i \in l(t)\} \neq 0. \tag{A15}$$

For a nonzero $s(t)$, we have that

$$\begin{aligned} f(x(t+1)) &= f(x(t) + s(t)) \\ &= f(x(t)) + s(t)^T(Tx - b) + \tfrac{1}{2} s(t)^T T s(t) \\ &= f(x(t)) + \sum_{i \in l(t)'} \Delta x_i e_i^T (Tx - b) \\ &\quad + \tfrac{1}{2} \left(\sum_{i \in l(t)'} \Delta x_i e_i \right)^T T \left(\sum_{j \in l(t)'} \Delta x_j e_j \right) \\ &= f(x(t)) + \sum_{i \in l(t)'} \Delta x_i (-u_i) + \tfrac{1}{2} \sum_{i \in l(t)'} \Delta x_i^2 t_{ii} \\ &= f(x(t)) - \sum_{i \in l(t)'} (u_i \Delta x_i - \tfrac{1}{2} t_{ii} \Delta x_i^2) \\ &= f(x(t)) - \beta(t) \end{aligned} \tag{A16}$$

where

$$\beta(t) = \sum_{i \in l(t)'} (u_i \Delta x_i - \tfrac{1}{2} t_{ii} \Delta x_i^2) > 0, \quad \forall t = 0, 1, \cdots.$$

Similarly to Theorem 3, if we assume that there exist infinitely many $s(t)$'s, we have

$$\sum_{\tau=0}^{\infty} \beta(\tau) \leq f(x(0)) - M < \infty$$

which implies that $\lim_{t\to\infty} \beta(t) = 0$, which contradicts (7). This means that only a finite number of nonzero $s(t)$'s exist. In other words, Algorithm 3 converges to a fixed point after a finite number of iterations. (Q.E.D.)

F. Proof of Theorem 6

A nonzero step in Algorithm 4 is given by

$$s(t) = x(t+1) - x(t) = \sum_{i=1}^{n} \Delta x_i e_i \tag{A17}$$

where at least one of the Δx_i, $i \in \{1, \cdots, n\}$ is nonzero and $0 \leq x_i(t) + \Delta x_i \leq 255$. We also define the nonempty index set, such as

$$l(t)' = \{i | \Delta x_i \neq 0, \quad 0 \leq x_i(t+1) + \Delta x_i \leq 255, \text{ and } i \in l(t)\} \neq 0. \tag{A18}$$

For a nonzero step $s(t)$ for some t, we have

$$\begin{aligned} f(x(t+1)) &= f(x(t) + s(t)) \\ &= f(x(t+1)) + s(t)^T(Tx - b) \\ &\quad + \frac{1}{2} s(t)^T T s(t) \\ &= f(x(t+1)) + \sum_{i \in l(t)'} \Delta x_i e_i^T (Tx - b) \\ &\quad + \frac{1}{2} (\sqrt{m} y)^T T (\sqrt{m} y) \\ &= f(x(t+1)) - \sum_{i \in l(t)'} |u_i| + \frac{1}{2} m y^T T y \\ &\leq f(x(t)) - \sum_{i \in l(t)'} |u_i| + \frac{m}{2} \|T\|_2 \\ &= f(x(t)) - \sum_{i \in l(t)'} \left(|u_i| - \frac{1}{2} \|T\|_2 \right) \end{aligned} \tag{A19}$$

where m is the number of nonzero elements in $s(t)$ and $y = (1/\sqrt{m}) s(t)$. If (29) is satisfied, then according to (A19) we have

$$f(x(t+1)) - f(x(t)) \leq - \sum_{i \in l(t)'} (|u_i| - \tfrac{1}{2} \|T\|_2) < 0. \tag{A20}$$

On the other hand if (30) is satisfied, the function reduction is not guaranteed, and the algorithm may increase f. (Q.E.D.)

G. Proof of Theorem 7

Consider the definition of $l(t)'$ in (A18). As shown in (A19) in Theorem 6, we have

$$f(x(t+1)) \leq f(x(t)) - w(t)$$

where

$$w(t) = \sum_{i \in l(t)'} (|u_i| - \tfrac{1}{2} \|T\|_2)$$

which is always positive when (29) in Theorem 6 is satisfied. On the other hand when (30) is satisfied, after increasing θ_i by q times, we have a θ_i' such that

$$\theta_i' = \delta^q \theta_i = \tfrac{1}{2}\delta^q t_{ii} \tag{A21}$$

where

$$\delta^q t_{ii} \geq \|T\|_2 \quad \text{and } \delta^{(q-1)} t_{ii} < \|T\|_2 \tag{A22}$$

so that the qth increase of θ_i is required, in order to guarantee energy reduction since now from $|u_i| > (1/2)\delta^q t_{ii}$ it is concluded that $|u_i| > (1/2)\|T\|_2$. Then

$$\begin{aligned} w(t) &= \sum_{i \in l(t)'} (|u_i| - \tfrac{1}{2}\|T\|_2) \\ &> \tfrac{1}{2} \sum_{i \in l(t)'} (\delta^q t_{ii} - \|T\|_2) \geq 0 \end{aligned} \tag{A23}$$

since $|u_i| > (1/2)\delta^q t_{ii}$ for $i \in l(t)'$. Since f is bounded below it holds that

$$\begin{aligned} M \leq f(x(t+1)) &\leq f(x(t)) - w(t) \\ &= f_{(q-1)} - \sum_{\tau = t_{(q-1)}+1}^{t} w(\tau) \end{aligned} \tag{A24}$$

where

$$\begin{aligned} f_{(q-1)} = f(x(0)) &- \sum_{\tau=0}^{t_1} w_1(\tau) - \sum_{\tau=t_1+1}^{t_2} w_2(\tau) \\ &- \cdots - \sum_{\tau = t_{(q-2)}+1}^{t_{(q-1)}} w_{(q-1)}(\tau). \end{aligned} \tag{A25}$$

If we assume that $l(t)'$ is nonempty as $t \to \infty$:

$$\sum_{t = t_{(q-1)}}^{\infty} w(t) \leq f_{(q-1)} - M < \infty \tag{A26}$$

which implies that $\lim_{t \to \infty} w(t) = 0$ and contradicts (A23). This means that only a finite number of nonzero $s(t)$'s exist, resulting in the convergence of Algorithm 5 after a finite number of iterations. (Q.E.D.)

H. Proof of Theorem 8

According to Theorem 7, since no bound constraints are activated, a fixed point satisfies

$$|u_i^*| \leq \tfrac{1}{2}\delta^q t_{ii}, \qquad \forall i \tag{A27}$$

which yields

$$\|u^*\|_1 \leq \frac{n}{2}\delta^q t_{\max}. \tag{A28}$$

Clearly, by choosing δ according to

$$\delta = \left(\frac{\|T\|_2}{t_{\max}}\right)^{1/q} \tag{A29}$$

we obtain $\delta^q t_{\max} = \|T\|_2$. (Q.E.D)

I. Proof of Theorem 9

Since no bound constraints are activated, u_i^* should satisfy

$$|u_i^*| \leq \tfrac{1}{2}2^{(8-p)}\delta^q t_{\max}, \qquad \forall i. \tag{A30}$$

If $\delta^q t_{\max} = \|T\|_2$, we have that

$$\|u^*\|_1 \leq 2^{(7-p)} n \|T\|_2. \tag{A31}$$

(Q.E.D)

J. Proof of Theorem 10

A nonzero step in Algorithm 6 is given by

$$s(t) = x(t+1) - x(t) = \sum_{i \in l(t)} \Delta x_i e_i \tag{A32}$$

where at least one of the Δx_i, $i \in \{1, \cdots, n\}$ is nonzero and $0 \leq x_i(t) + \Delta x_i \leq 255$. We also define the nonempty index set, such as

$$\begin{aligned} l(t)' = \{i | \Delta x_i \neq 0, \quad & 0 \leq x_i(t+1) \\ & + \Delta x_i \leq 255, \text{ and } i \in l(t)\} \neq 0. \end{aligned} \tag{A33}$$

We now need to obtain an expression for $x(t) - x^i(t)$ in terms of the $s(t)$'s. We do that by considering an example when the update of x_i, $(i = 2)$ is considered, at time t. Let us assume that we have

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} \quad \text{and } x^i(t) = \begin{bmatrix} x_1(t-B+1) \\ x_2(t) \\ x_3(t-1) \\ x_4(t-2) \end{bmatrix}. \tag{A34}$$

Then the difference between $x(t)$ and $x^i(t)$ is given by

$$\begin{aligned} x(t) - x^i(t) &= \begin{bmatrix} x_1(t) - x_1(t-B+1) \\ 0 \\ x_3(t) - x_3(t-1) \\ x_4(t) - x_4(t-2) \end{bmatrix} \\ &= \begin{bmatrix} x_1(t) - x_1(t-1) \\ 0 \\ x_3(t) - x_3(t-1) \\ x_4(t) - x_4(t-1) \end{bmatrix} + \begin{bmatrix} x_1(t-1) - x_1(t-2) \\ 0 \\ 0 \\ x_4(t-1) - x_4(t-2) \end{bmatrix} + \cdots + \begin{bmatrix} x_1(t-B+2) - x_1(t-B+1) \\ 0 \\ 0 \\ 0 \end{bmatrix}. \end{aligned} \tag{A35}$$

Let us define by $I(\tau)$, $\tau = t-1, t-2, \cdots, t-B+1$, the index set of nonzero elements in $x(t) - x^i(t)$ at time τ. In this example, we have

$$I(t-1) = \{1, 3, 4\}, I(t-2) = \{1, 4\} \quad \text{(A36)}$$

and

$$I(t-3) = \cdots = I(t-B+1) = \{1\} \quad \text{(A37)}$$

If we also define $J(\tau) = I(\tau) \cap l(\tau)'$, $\tau = t-1, t-2, \cdots, t-B+1$, then we further have

$$\begin{aligned} x(t) - x^i(t) &= \sum_{j \in J(t-1)} s_j(t-1)e_j + \sum_{j \in J(t-2)} s_j(t-2)e_j \\ &\quad + \cdots + \sum_{j \in J(t-B+1)} s_j(t-B+1)e_j \\ &= \sum_{\tau=t-B+1}^{t-1} \sum_{j \in J(\tau)} s_j(\tau)e_j \end{aligned} \quad \text{(A38)}$$

thus

$$\begin{aligned} \|x(t) - x^i(t)\| &= \left\| \sum_{\tau=t-B+1}^{t-1} \sum_{j \in J(\tau)} s_j(\tau)e_j \right\| \\ &\leq \sum_{\tau=t-B+1}^{t-1} \left\| \sum_{j \in J(\tau)} s_j(\tau)e_j \right\|. \end{aligned} \quad \text{(A39)}$$

Let us define $v = b - Tx(t)$. Then for a nonzero $s(t)$ we have that

$$\begin{aligned} f(x(t+1)) &= f(x(t) + s(t)) \\ &= f(x(t)) - s(t)^T v + \tfrac{1}{2}s(t)^T T s(t) \\ &= f(x(t)) - s(t)^T u + s(t)^T(u - v) \\ &\quad + \tfrac{1}{2}s(t)^T T s(t) \\ &\leq f(x(t)) - \sum_{i \in l(t)'} (|u_i| - \tfrac{1}{2}\|T\|_2) \\ &\quad + s(t)^T(u - v) \end{aligned} \quad \text{(A40)}$$

where u is defined in (46). It is noted here that if $B = 1$, then $s(t)^T(u - v) = 0$ and the synchronous Algorithm 4 results. For $B \geq 2$ the bound of the third term in (A40) is given by

$$\begin{aligned} s(t)^T(u - v) &\leq \|s(t)\|_2 \|u - v\|_2 \\ &= \sqrt{m(t)}\|u - v\|_2 \\ &\leq \sqrt{m(t)}\|T\|_2 \|x^i(t) - x(t)\|_2 \end{aligned} \quad \text{(A41)}$$

where $m(t)$ is the number of elements in the set $l(t)'$.

By combining (A41) and (A39) we obtain

$$\begin{aligned} s(t)^T(u - v) &\leq \sqrt{m(t)}\|T\|_2 \sum_{\tau=t-B+1}^{t-1} \left\| \sum_{j \in J(\tau)} s_j(\tau)e_j \right\|_2 \\ &= \sqrt{m(t)}\|T\|_2 \sum_{\tau=t-B+1}^{t-1} \sqrt{m'(t)} \end{aligned} \quad \text{(A42)}$$

where $m'(t)$ equals the number of elements in the set $J(\tau)$. Let us also denote by $m''(t)$ the number of elements in the set $I(\tau)$. Then it holds true that

$$m''(t-1) \geq m''(t-2) \geq \cdots \geq m''(t-B+1). \quad \text{(A43)}$$

The following relation also holds true

$$m'(\tau) \leq \min \{m(\tau), m''(\tau)\}. \quad \text{(A44)}$$

A key observation now is that the value of $m'(t)$ is determined primarily by the value of $m''(t)$, according to the above equation, since $m''(\tau)$ refers to one vector elements, while $m(\tau)$ refers to the complete vector. Therefore, it is reasonable to assume that

$$m'(t-1) \geq m'(t-2) \geq \cdots \geq m'(t-B+1). \quad \text{(A45)}$$

By combining (A42) and (A45) we further obtain

$$s(t)^T(u - v) \leq \sqrt{m(t)}\|T\|_2(B-1)\sqrt{m'(t-1)} \quad \text{(A46)}$$

or

$$s^T(u - v) \leq m(t)(B-1)\|T\|_2. \quad \text{(A47)}$$

By combining (A40) and (A47) we have that

$$\begin{aligned} f(x(t+1)) &\leq f(x(t)) \\ &\quad - \sum_{i \in l(t)'} \{|u_i| - \tfrac{1}{2}\|T\|_2 - \|T\|_2(B-1)\} \\ &= f(x(t)) - w(t) \end{aligned} \quad \text{(A48)}$$

and consequently,

$$f(x(t+1)) \leq f(x(0)) - \sum_{\tau=0}^{t} w(\tau) \quad \text{(A49)}$$

where

$$w(t) = \sum_{i \in l(t)'} \{|u_i| - \tfrac{1}{2}\|T\|_2(2B-1)\} > 0. \quad \text{(A50)}$$

If we assume that there exist infinitely many nonzero $s(t)$'s, we have that

$$\sum_{t=0}^{\infty} w(t) \leq f(x(0)) - M < \infty \quad \text{(A51)}$$

which implies that $\lim_{t \to \infty} w(t) = 0$ and contradicts (A50). This means that only a finite number of nonzero $s(t)$'s exist. Therefore, Algorithm 6 converges to a fixed point after a finite number of iterations. (Q.E.D.)

K. Proof of Theorem 11

Since no bound constraints are activated, we have that

$$|u_i^*| \leq \tfrac{1}{2}\|T\|_2(2B-1), \quad \forall i \quad \text{(A52)}$$

which yields

$$\|u^*\|_1 \leq \frac{n}{2} \|T\|_2(2B - 1). \quad \text{(A53)}$$

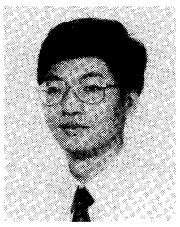
(Q.E.D.)

ACKNOWLEDGMENT

The authors thank the anonymous reviewer for the careful reading of their manuscript and for pointing out related work in the literature.

REFERENCES

[1] H. C. Andrews and B. R. Hunt, *Digital Image Restoration*. Englewood Cliffs, NJ: Prentice-Hall, 1977.
[2] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation; Numerical Methods*. Englewood Cliffs, NJ: Prentice Hall, 1989.
[3] J. Bruck and J. W. Goodman, "A generalized convergence theorem for neural networks," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1089-1092, Sept. 1988.
[4] J. Bruck, "On the convergence properties of the Hopfield model," *Proc. IEEE*, vol. 78, pp. 1579-1585, Oct. 1990.
[5] N. H. Farhat, D. Psaltis, and E. Paek, "Optical implementation of the Hopfield model," *Appl. Opt.*, vol. 24, no. 10, pp. 1469-1475, May 1985.
[6] E. Goles-Chacc, F. Fogelman-Soulie, and D. Pellegrin, "Decreasing energy functions as a tool for studying threshold networks," *Disc. Appl. Math.*, vol. 12, pp. 261-277, 1985.
[7] E. Goles and S. Martinez, *Neural and Automata Networks: Dynamical Behavior and Applications*. Higham, MA: Kluwer Academic, 1990.
[8] L. J. Griffiths, guest ed., "Special section on neural networks," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1107-1191, July 1988.
[9] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. Nat. Acad. Sci. USA*, vol. 79, Apr. 1982, pp. 2554-2558.
[10] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141-152, 1985.
[11] B. R. Hunt, "The application of constrained least squares estimation to image restoration by digital computer," *IEEE Trans. Comput.*, vol. C-22, pp. 805-812, Sept. 1973.
[12] A. K. Katsaggelos, J. Biemond, R. M. Mersereau, and R. W. Schafer, "A general formulation of constrained iterative restoration algorithms," in *Proc. 1985 Int. Conf. Acoust., Speech, Signal Processing*, Tampa FL, Mar. 1985, pp. 700-703.
[13] A. K. Katsaggelos, "Iterative image restoration algorithms," *Opt. Eng.*, vol. 28, no. 7, pp. 735-748, July 1989.
[14] A. K. Katsaggelos, J. Biemond, R. W. Schafer, and R. M. Mersereau, "A regularized iterative image restoration algorithm," *IEEE Trans. Signal Processing*, vol. 39, pp. 914-929, Apr. 1991.
[15] A. K. Katsaggelos, ed., *Digital Image Restoration*. Berlin, Germany: Springer Verlag, 1991.
[16] R. P. Lippmann, "An introductin to computing with neural nets," *IEEE ASSP Mag.*, pp. 4-22, Apr. 1987.
[17] J. K. Paik and A. K. Katsaggelos, "Parallel iterative image restoration algorithms," in *Proc. 32nd Midwest Symp. on Circuits and Systems*, Urbana, IL, Aug. 1989, pp. 63-66.
[18] ——, "Image restoration using the Hopfield network with nonzero autoconnections," in *Proc. 1990 Int. Conf. Acoust., Speech, Signal Processing*. Albuquerque, NM, Apr. 1990, pp. 1909-1912.
[19] J. K. Paik, "Image restoration and edge detection using neural networks," Ph.D. dissertation, Dep. Elec. Eng., Computer Sci., Northwestern Univ., June 1990.
[20] M. Takeda and J. W. Goodman, "Neural networks for computation: Number representations and programming complexity," *Appl. Opt.*, vol. 25, no. 18, pp. 3033-3046, Sept. 1986.
[21] S. J. Yeh, H. Stark and M. I. Sezan, "Hopfield-type neural networks" in *Digital Image Restoration*, A. K. Katsaggelos, ed. Berlin, Germany: Springer-Verlag, vol. 23, ch. 3, 1991.
[22] Y. T. Zhou, R. Chellappa, and B. K. Jenkins, "Image restoration using a neural network," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-36, pp. 1141-1151, July 1988.



Joon K. Paik (S'88-M'90) was born in Seoul, Korea, on July 10, 1960. He received the B.S. degree in control and instrumentation engineering from Seoul National University, Seoul, Korea, in 1984. He also received the M.S. and Ph.D. degrees in electrical engineering and computer science from Northwestern University, Evanston, IL, in 1987 and 1990, respectively.

In 1984, he worked as a researcher in the Central Research Laboratory in Goldstar Instrumentation Co. Ltd. in Osan, Korea. From 1988 to 1989, he was a Research Assistant at the Department of Electrical Engineering and Computer Science at Northwestern University. Since 1990, he has been a Manager of the Audio/Video systems technology team in Bucheon Research and Development Center, Semiconductor Business, Samsung Electronics, Bucheon, Korea. His current research interests include signal and image processing, motion detection systems in digital video cameras, parallel implementation of large scale optimization problems, and artificial neural network implementations of image processing algorithms.

Aggelos K. Katsaggelos (S'80-M'85) was born in Arnea, Greece, on April 17, 1956. He received the Diploma degree in electrical and mechanical engineering from the Aristotelian University of Thessaloniki, Thessaloniki, Greece, in 1979. He received the M.S. and Ph.D. degrees both in electrical engineering from the Georgia Institute of Technology, Atlanta, Georgia, in 1981 and 1985, respectively.

From 1980 to 1985, he was a Research Assistant at the Digital Signal Processing Laboratory of the Electrical Engineering School at Georgia Tech. He is currently an Assistant Professor in the Department of Electrical Engineering and Computer Science at Northwestern University, Evanston, IL. During the 1986-1987 academic year he was an Assistant Professor at Polytechnic University, Department of Electrical Engineering and Computer Science, Brooklyn, NY. His current research interests include signal and image processing, processing of moving images, computational vision and parallel implementations of signal processing algorithms.

Dr. Katsaggelos is an Ameritech Fellow and a member of the Associate Staff, Department of Medicine, at Evanston Hospital. He is also a member of SPIE, the Steering Committee of the IEEE TRANSACTIONS ON MEDICAL IMAGING, the IEEE-CAS Technical Committee on Visual Signal Processing and Communications, the Technical Chamber of Commerce of Greece and Sigma Xi. He is an Associate editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING and also the editor of the book *Digital Image Restoration* (Springer-Verlag, 1991).