Fast Block-Based Image Restoration Employing the Improved Best Neighborhood Matching Approach

Wen Li, David Zhang, *Senior Member, IEEE*, Zhiyong Liu, and Xiangzhen Qiao

***Abstract*—The best neighborhood matching (BNM) algorithm is an efficient approach for image restoration. However, its high computation overhead imposes an obstacle to its application. In this paper, a fast image restoration approach named jump and look around BNM (JLBNM) is proposed to reduce computation overhead of the BNM. The main idea of JLBNM is to employ two kinds of search mechanisms so that the whole search process can be sped up. Some optimization techniques for the restoration algorithm JLBNM are also developed, including adaptive threshold in the matching stage, the terminal threshold in the searching stage, and the application of an appropriate matching function in both the matching and recovering stages. Theoretical analysis and experiment results have shown that JLBNM not only can provide high quality for image restoration but also has low computation overhead.**

***Index Terms*—Best neighborhood matching (BNM) algorithm, block-based coding image, computation complexity, image restoration, transmission error.**

I. Introduction

BLOCK-BASED techniques are adopted by most existing standards for image and video compression, such as JPEG, H.261, and MPEG [2]–[4]. Variable length coding is usually used in a block-based coding system. As a result, the encoded bit stream is vulnerable to transmission error. Loss of a single bit often results in the loss of a whole block, and may even cause consecutive block losses.

Several techniques have been proposed to minimize the communication errors, including error resilience encoding [10], feedback channels (FBCs) [11] and error concealment [1], [14], [5]–[8]. However, communication errors still exist after error resilience coding, while FBC may introduce additional delay. The error concealment technique belongs to postprocessing, where the errors are first located, and then are masked to create subjectively acceptable images. The main idea of an error concealment technique is to extract some information from the image and apply the information obtained to restore the damaged blocks.

Error concealment algorithms can be categorized into two groups according to their applications: One for image restoration and the other for video sequence restoration. Error concealment algorithms for images may not need to be completed in real-time. Restored images require high quality. Image restoration algorithms may be used for restoring the I frames of video sequences. This depends on whether the algorithms can finish in real-time. In addition to image restoration algorithms for I frames, restoration algorithms for B frames and P frames also need to be developed when they are used for restoring video sequences. Therefore, the benefits of the research for image restoration can be extended from images to video sequences.

In image restoration algorithms, the spatial correlation property of natural images is used for restoring damaged blocks. Error concealment by spatial correlation can be achieved by interpolating the missing dc and the lowest frequency ac from those belonging to the surrounding blocks [5]–[8]. Error concealment can also be achieved by block-wise similarity within nature images [1], [14]. For a video sequence, both spatial and temporal correlation properties are employed in the restoration process.

With regard to the information used for restoring damaged blocks, error concealment algorithms by block-wise similarity can be categorized into two frameworks: The local correlation framework [5]–[8] and the long-range correlation framework [1], [14]. In the local correlation framework, only neighboring pixels are used as the source to generate information about damaged blocks. This kind of algorithms relies on some predefined constraints on the spectra or structures of the damaged blocks. The reconstructed pixels should be smoothly connected with adjacent regions either in the spatial or in the transformed domain. Some detailed information about the images may be lost after this kind of image restoration.

The long-range correlation framework [1], [14], on the other hand, is based on the idea of the existence of abundant long-range correlation within natural images. A vision system consistsing of our eyes and brains can sufficiently utilize such kind of information redundancy to implement the functions of image restoration. In other words, long-range correlation image restoration is able to utilize not only the information of neighboring pixels, but also correlation information among remote regions in the image. The best neighborhood matching (BNM) algorithm [1] belongs to this framework. It shows that BNM performs substantially better than the local correlation-based

Manuscript received August 2, 2001; revised February 21, 2003 and July 23, 2004. This paper was recommended by Associate Editor D. Brown. This work was supported by the National Natural Science Foundation of China Project under Grant 60 303 033, the Natural Sciences Foundation of Inner Mongolia Autonomous Region under Grant 200 308 020 212, the central fund from the Hong Kong Polytechnic University, the University Research Project of Inner Mongolia under Grant NJ03120, and a Doctoral Grant of Inner Mongolia.

W. Li is with the Department of Computer Science, Inner Mongolia University, Huhehaote 010021, China, and also with the Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong, and the National Research Center for Intelligent Computing System, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China.

D. Zhang is with the Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: csdzhang@comp.polyu.edu.hk).

Z. Liu and X. Qiao are with the National Research Center for Intelligent Computing System, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China.

Digital Object Identifier 10.1109/TSMCA.2005.850605

image restoration algorithms with respect to the quality of the restored images.

However, since BNM requires excessive computing time there is an obstacle to its real application—A problem that also exists in other image restoration algorithms. In this paper, we propose an improved BNM approach, namely jump and look around BNM (JLBNM), to provide fast block-based image restoration. JLBNM uses the intelligence of the human vision system when searching for objects within certain surroundings. When one quickly searches for the best similar part for replacing the damaged part, a jump and look around method may be applied. In other words, a search for some approximately similar part is conducted followed by a more carefully search of nearby information. The above two procedures may be applied alternately until the best similar part is found.

Some optimization techniques for JLBNM are developed to provide fast restoration. These techniques include adaptive threshold application in the matching stage, and the terminal threshold used in the searching stage. Also, appropriate matching functions are investigated and used in both the matching and recovering stages.

The rest of this paper is organized as follows. In Section II, we briefly describe the BNM algorithm, focusing on its limitations. The jump and look around BNM approach and some optimization techniques are developed in Sections III and IV. Then, computation overhead of the JLBNM is analyzed in Section V. Experiment results are given in Section VI for a comparison between BNM and JLBNM with respect to their processing time and quality of the restored images. Some concluding remarks are given in Section VII.

II. Overview of the BNM

A. Notations and Definitions

A digital gray scale image X with M × M pixels, can be denoted as a vector $X = \{x_{i,j} | 0 \leq i < M, 0 \leq j < M\}$. Let x_{ij} and $x_{ij}^{(\text{new})}$ be the pixel values at a position, (i,j), in a damaged image and a restored image, respectively. Using some error detection algorithms, such as those described in [9], we can tell whether a pixel is damaged. The locations of damaged blocks may also be detected at the decoder [6]. So, it is easy to associate each pixel with a binary flag $f_{i,j}$, indicating whether a block or a pixel is lost. When $f_{i,j} = 1, x_{ij}$ is missing; when $f_{i,j} = 0$, it is good. The goal of block-based image restoration is to restore all the damaged blocks. In BNM [1], the restoration procedure of each damaged block consists of the following five steps (refer to Fig. 1).

1) Extract a lost block of $K \times K$ *with its neighborhood* from the damaged image as a local window l of $N \times N$ pixels (see Fig. 1). The flag value of each pixel in the window l is denoted as $f_{p+g,q+h}^{l}$, where $0 \leq g \leq N, 0 \leq h \leq N$, and (p,q) represents the top-left corner of l. A searching range S is also extracted.
2) Find a remote window, r, in the searching range, which is of the same shape and the same size as the local window, l. The flag value of each pixel in the window r is denoted as $f_{a+g,b+h}^{r}$, where $0 \leq g \leq N, 0 \leq h \leq N$, and (a,b) represents the top-left corner of r. When every pixel in the window r is good the remote window is regarded as a candidate window.
3) Search for the best matching candidate window in the searching range by trying to match the good pixels in the local window and candidate windows. The matching method is determined by a one-dimensional luminance transformation function v (see [1]). The matching result can be evaluated by the mean squared error (MSE_M) between the transformed r and l, i.e.,

$$\text{MSE}_M = \frac{1}{n_M} \sum_{g=0}^{N-1} \sum_{h=0}^{N-1} [1 - f_{\text{p+g,q+h}}^{l}] \cdot [x_{p+g,q+h} - v(x_{a+g,b+h})]^2 \quad (1)$$

In practice, we can use the condition of the minimal MSE_M to calculate the parameters in the luminance transformation, v.
4) Find the candidate window with the minimal MSE_M, i.e., the best candidate window.
5) Recover damaged pixels in the local window using the corresponding part of the best candidate window (transformed using the function v), keeping the good part of the local window unchanged, as follows:

$$\begin{cases} x_{p+g,q+h}^{(\text{new})} = v(x_{a+g,b+h}), & \text{if } f_{p+g,q+h}^{l} = 1 \\ x_{p+g,q+h}^{(\text{new})} = x_{p+g,q+h}, & \text{if } f_{p+g,q+h}^{l} = 0. \end{cases} \quad (2)$$

B. Limitations of BNM

The search scheme in BNM is the most time-consuming part in the restoration process. For a damaged block, the search route starts from the top-left corner of the searching range, scanning from left to right with one pixel in each step toward the top-right corner, then moves back to the very left of the searching range for the next row. This procedure continues until the bottom-right corner is reached.

In the search procedure, a remote window is generated in each step of one-pixel. All the candidate windows in the search range are processed. The candidate window that best matches the local window will not be missed in this search procedure. However, such a search procedure is exhaustive and heavily time consuming.

The search algorithm of BNM might look similar to the motion estimation of the video compression algorithm. In fact, they have great difference. In motion estimation, macro-block in the current frame or picture can be modeled as the translation of macro-block in a reference picture at some previous or future time. The macro-blocks are all intact. However, in BNM, the damaged pixels in the local window are not considered in the matching process, and the damaged block is just located in the center of the local window. Thus, block-wise similarity is primarily based on the similarity of the window border. The damaged blocks are reconstructed with similar blocks. Therefore, the principle of BNM search is different from that of motion estimation.

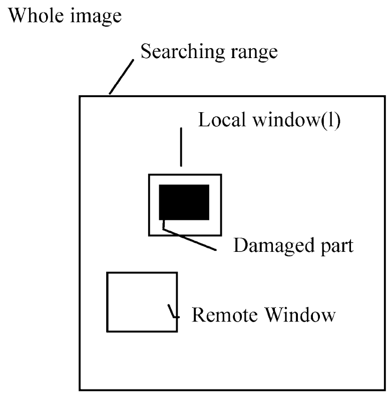


Fig. 1. BNM demonstration graph with a damaged part, a local window, a remote window, and a searching range.

III. JLBNM Algorithm

A. Search Strategies in JLBNM

In the JLBNM algorithm, there are two search styles. One is called "jump search" and the other is "look around search." For a certain damaged block, the first remote window is formed at the upper-left corner of the searching range. If the remote window is a perfect one, it becomes a candidate window and the MSE_M is calculated. If the remote window has some damaged pixels, MSE_M will not be calculated. Instead, the remote window will be skipped, and the search will go on forming another remote window.

The search procedure begins with "jump search." In jump search, the next remote window is formed always in $J_s(J_s > 1)$ pixels away from the previous one, i.e., after J_s pixels in column or below J_s pixels in row from the previous remote window. The search goes on until a "best" (best in all compared candidate windows so far) candidate window is obtained.

We then perform the "look around search," in which a small searching range will be formed by extending the edge of the best candidate window from top, bottom, left and right, respectively. Each direction is extended with $J_s - 1$ pixels. So the small searching range, called S', consists of $((J_s - 1) \times 2 + N) \times ((J_s - 1) \times 2 + N)$ pixels. In the small searching range, remote windows are formed in L_s pixels apart, where L_s is carefully decided and $L_s < J_s$. This means that the small searching range is looked at more carefully than in the jump search process.

After look around search is completed for the small searching range, another jump search will begin. The above search processes are executed alternately until the whole searching range is finished and the best similar block is found in the range. The following is a formal description of the search process in JLBNM.

```
For $(x = S\cdot bx; x \le S\cdot ex; x+ = J_s)$/ ∗ $S\cdot bx$ and $S\cdot ex$ are begin and end points of $S$ in row ∗/
  For $(y = S\cdot by; y \le S\cdot ey; y+ = J_s)$/ ∗ $S\cdot by$ and $S\cdot ey$ are begin and end points of $S$ in column ∗/
    {Generate a remote window, $r_{(x,y)}$;
      If ($r_{(x,y)}$ is a candidate window)
        {Calculate the $\mathrm{MSE}_M$ for the local window, $l$, and $r_{(x,y)}$;
          If ($r_{(x,y)}$ is the best one in all compared candidate windows)
            {Update the best matching $\mathrm{MSE}_M$ with that obtained for $r_{(x,y)}$ and $l$, and choose the
            candidate window $r_{(x,y)}$ as the best matching window;
          /∗ now a small searching range will be generated, and look around search will begin∗/
            For $(m = x - J_s + 1; m \le x + J_s + N - 1; m+ = L_s)$
              For $(n = y - J_s + 1; n \le y + J_s + N - 1; n+ = L_s)$
                If ($r_{(m,n)}$ is a candidate window)
                  {Calculate the $\mathrm{MSE}_M$ for $l$ and $r_{(m,n)}$;
                  If ($r_{(m,n)}$ is the best one in all compared candidate windows)
                  {Update the best matching $\mathrm{MSE}_M$ with that obtained for $r_{(m,n)}$ and $l$, and choose
                      $r_{(m,,n)}$ as the best matching window;
                  }
                  }/∗ look around search ends and jump search will begin ∗/
            }/∗ end of If ($r_{(x,y)}$ is the best one in all compared candidate windows) ∗/
        }/∗ If ($r_{(x,y)}$ is a candidate window) ∗/
    }/∗ end of jump and look around search ∗/
```

B. Continuity of Images

The information in any image is correlated in a certain way, especially within a limited local searching range. If a candidate window has a great difference with the local window, the remote windows it will have little chance to get a better match. Due to this property, the search procedure can be done in the jump style. When a good match between a candidate window and a local window is obtained, the candidate windows nearby may have a better chance to getting a superior match. At this time, a careful search should be done among these candidate windows for a better matching window. This is why we extend it to a small searching range S', and search more carefully by using a smaller step length in the small searching range. Note that the small searching range S' covers all the nearby area defined with J_s.

C. Deciding the Parameters: J_s and L_s

In JLBNM, choosing the correct J_s and L_s is a key issue for the quality of the restored image and the speed of the algorithm. In fact, natural images can be classified into coarse ones and fine ones. Coarse images, such as the sea, a desert, a cornfield, and so on, come with little detail. In these types of images, we can choose large J_s and L_s. For fine images with more details, we should select small J_s and L_s. So they vary for different images. When J_s is very large, small range search can also be done in the "jump and look around" style. For example, if we choose $J_s = 16$, in the small search range, we can choose $L_s = 4$. Then, in the small searching range, we let $J'_s = L_s = 4$, and $L'_s = 1$. Therefore, the jump and look around search method can be used recursively according to image types. Two adjacent remote windows may have some, many, or no overlap; they may also come with some gaps (see Fig. 2).

Since most images contain detailed information and their restorations are computation intensive, we will put our emphasis on images with details. If the images have detailed information, two adjacent remote windows should not have gap, or else the possibility of missing the best candidate window will be too large. So J_s should not be larger than N. In order to get good image quality, the step length L_s in the small searching range is one-pixel. The value of J_s will be determined in Section V.

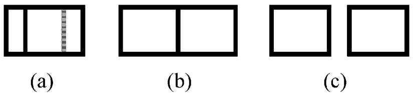


Fig. 2. Space relation of two adjacent remote windows. (a) With overlap. (b) Without overlap or gap. (c) With gap.

IV. Optimization for JLBNM

A. Adaptive Threshold for Matching Procedure

From our experience, we find that the MSE_M of many candidate windows for a local window is much larger than that of the best candidate window selected. So, we developed an idea to divide the match procedure into two stages. First, we only use part of the whole matching pixels in calculating MSE_M. We call the MSE_M of partial matching pixels PMSE_M. If it is over a threshold W_t, which indicates that the current candidate window will not be the best matching window, we discard it and begin to find the next candidate window. If it is lower than the threshold we continue calculating the MSE_M using the other matching pixels until we complete the MSE_M calculation for all the matching pixels in the whole window. Then, we decide whether it is the best window in all the compared candidate windows. As a result, the total matching pixels in the searching procedure will be reduced.

The number of matching pixels involved in the computation for MSE_M also has effects on memory access rate. If fewer matching pixels are used in the search procedure, a lower memory bandwidth will be needed.

There are two important parameters to be fixed: The number of pixels for PMSE_M in the matching procedure, and the threshold, W_t. Since we do not want to lose image restoration quality, we choose W_t being equal to the smallest MSE_M obtained in all the compared candidate windows so far. The smallest MSE_M changes dynamically in the matching stage, so W_t will also change with the smallest MSE_M. When we choose W_t to be equal to the smallest MSE_M, it is observed that the number of partial matching pixels is about $2/3$ the number of the whole matching pixels. At this point, not only can we reduce processing time, but also we can keep the restored image quality. Of course, when the small W_t is used, the fewer matching pixels will be involved in. It is evident that this can reduce the processing time further, but there are some risks to the quality of the restored images.

B. Terminal Threshold

In most cases, the best matching candidate window is not very far away from a damaged block. Fig. 3 shows the statistics on the distances of the best matching windows. From the statistics we know the best matching candidate windows are usually located 10–34 pixels away from the damaged blocks. As a result, we find that the best matching candidate window is in the middle of the searching procedure, not at the end of the searching procedure. Therefore, with an appropriate terminal threshold (T_M), i.e., an acceptable MSE_M, we can terminate the search process earlier and more processing time can be saved. An appropriate T_M also means that image restoration quality will not be lost. Experiments show that $T_M = 1\,500$ is an appropriate terminal threshold for restoration.

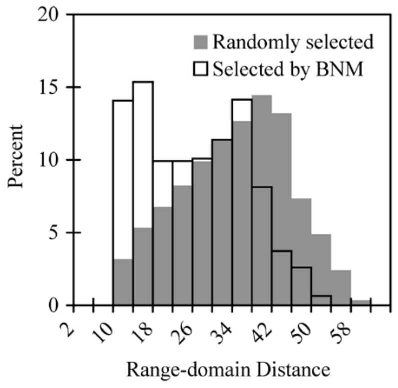


Fig. 3. Statistics on the distance between best matching windows and local windows on randomly selected method and BNM, respectively.

TABLE I
Nonoverlap Rate of Q_d for "*Barb*" and "*Lena*" With Block Loss Rates Ranging From 2.5%–15.0%

	The non-overlap rate of Q_d for different block loss rates					
Image	2.5%	5%	7.5%	10.0%	12.5%	15.0%
Lena	0.938	0.806	0.771	0.739	0.714	0.701
Barb	0.975	0.812	0.769	0.737	0.714	0.701

C. Matching Functions

In both BNM and JLBNM, the matching function v is a very important parameter that has great influence on both processing time and the quality of the restored images. From our experiments, it is shown that the direct matching function is more practical and the first-order matching function can achieve a better image restoration quality. However, the computation complexity of the first-order matching function is three times larger than that of the direct matching. It is also shown that higher order matching functions could give smaller MSE_M, but may lead to absurd recovery. Therefore, the direct matching function is used in the matching procedure, while the first-order matching function is used in the recovering procedure. By using these matching functions, the computation overhead is low while good image restoration quality can be achieved.

V. Performance Analysis

Since the basic computation overhead in both BNM and JLBNM is spent on the computation of MSE_M for local windows and candidate windows, the number of candidate windows is a key factor for computation overhead (when the matching function is decided). Also, the fewer the candidate windows there are, the smaller the memory bandwidth is required, since the MSE calculation for each candidate window requires a large amount of memory access. We now analyze the computation overhead with regard to the number of candidate windows.

A. Computation Overhead of BNM

Based on the parameters given in Section II, the size of a damaged image is $M \times M$, the size of local windows and remote windows is $N\times N$, and the searching range is $(S+N)\times(S+N)$. For a damaged block with $K \times K$ pixels, the number of remote windows, which are generated in the search process, is S^2. Since

TABLE II
AVERAGE NUMBER OF THE CANDIDATE REMOTE WINDOWS "N_A," PROCESSING TIME "T_m" (SECOND) AND PSNR BY CHOOSING THE DIFFERENT J_s FOR "*BARB*" WITH 10% BLOCK LOSS RATE

	Performance for image "*Barb*" with different jump step: J_s								
	1	2	3	4	5	6	7	8	9
N_A	3391	1200	717	593	560	545	546	538	592
T_m(s)	23.42	8.97	5.03	4.39	3.97	3.98	4.32	4.05	4.48
PSNR	35.29	35.44	35.12	35.00	35.03	34.78	34.38	34.04	34.10

TABLE III
AVERAGE NUMBER OF THE CANDIDATE REMOTE WINDOWS "N_A," PROCESSING TIME "T_m" (SECOND) AND PSNR BY CHOOSING THE DIFFERENT J_s FOR "*LENA*" WITH 10% BLOCK LOSS RATE

	Performance for image "*Lena*" with different jump step: J_s								
	1	2	3	4	5	6	7	8	9
N_A	3382	1208	728	615	561	552	544	570	593
T_m(s)	24.55	10.95	5.03	3.90	4.05	4.01	4.10	4.50	4.04
PSNR	34.95	34.90	33.97	33.79	33.74	33.95	33.28	33.30	33.78

TABLE IV
COMPARISON OF THE PERFORMANCE BETWEEN BNM AND JLBNM FOR "*BARB*," WHERE N_B AND N_J—THE AVERAGE NUMBERS OF THE CANDIDATE REMOTE WINDOWS BY BNM AND JLBNM; T_B AND T_J-THE PROCESSING TIME BY BNM AND JLBNM; T_{JA}—JLBNM WITH ADAPTIVE THRESHOLD OPTIMIZATION; T_{JAT}—JLBNM WITH ADAPTIVE AND TERMINAL THRESHOLD OPTIMIZATIONS; P_B—PSNR BY BNM; AND P_{JAT}—PSNR BY JLBNM WITH ADAPTIVE AND TERMINAL THRESHOLD OPTIMIZATIONS

	Performance Ratios for image "*Bard*" with the different Block Loss Rates (J_s=4)					
	2.5%	5.0%	7.5%	10.0%	12.5%	15.0%
N_J/N_B	15.7%	16.3%	16.2%	17.4%	18.6%	19.3%
T_B/T_J	20.0%	16.2%	16.7%	16.5%	16.5%	17.0%
T_B/T_{JA}	13.4%	15.3%	14.8%	14.1	15.1	15.5%
T_B/T_{JAT}	10.8%	8.0%	7.3%	7.9%	10	9.2%
P_B/P_{JAT}	95.2%	98.9%	97.6%	99.2%	98.1%	97.9%

some remote windows may contain some bad pixels, the number of candidate windows, which are generated in the matching process, may be less than the number of remote windows. Let Q_d be a region with $(N-1+K+N-1)\times(N-1+K+N-1)$ pixels, and there is a damaged block of $K \times K$ in the center of Q_d. $(N+K-2)\times(N+K-2)$ remote windows in the region contain the bad pixels of the damaged block. When some damaged blocks are not apart far enough, the regions Q_d will have some overlap. Suppose that block loss rate is δ, and average overlap rate of Q_d caused by all damaged blocks is $(1-\eta)$. The number of remote windows N_d, which contain the bad pixels in a searching range, is expressed by

$$N_d = \frac{(S+N)^2}{K^2} * \delta * (N+K-2)^2 * \eta. \tag{3}$$

The number of candidate remote windows N_c in a searching range is

$$N_c = S^2 - N_d. \tag{4}$$

The number of damaged blocks in the whole image is

$$S_d = \frac{M*M}{K^2} * \delta. \tag{5}$$

Therefore, the computation overhead of BNM is defined with

$$F(\delta,\eta) = S_d * N_c. \tag{6}$$

The overlap rate of Q_d has some relation with the block loss rate. Based on our experiments, Table I lists the nonoverlap rate in damaged images "*Barb*" and "*Lena*" with the block loss rate ranging from 2.5% to 15%. From this table, it can be seen that the lower the block loss rate is, the higher the nonoverlap rate is.

B. Computation Overhead of JLBNM

In JLBNM, for a damaged block, the number of remote windows N_{jr} that are generated in the jump search process is S^2/J_s^2, which is $1/J_s^2$ of that in BNM. Suppose that the block loss rate is δ, and overlap rate of Q_d of all the damaged blocks is $(1-\eta)$, the same as before. The number of remote windows with damaged pixels in jump search process is

$$N_{jd} = \frac{(S+N)^2}{K^2} * \frac{\delta}{J_s^2} * \left(\frac{N+K-2}{J_s}\right)^2 * \eta. \tag{7}$$

The number of candidate windows N_{lc} in a small searching range with $(N+2\times J_s-2+N)\times(N+2\times J_s-2+N)$ pixels can be calculated by

$$N_{lc} = (N+2J_s-2)^2 - \frac{(N+2J_s-2+N)^2}{K^2} * \delta * (N+K-2)^2 * \eta. \tag{8}$$

Suppose that the rate of the best candidate window coming out in a searching range in the jump search process is γ. Then, we can obtain the computational overhead of JLBNM

$$F(\delta,\eta,\gamma,J_s) = \frac{M^2}{K^2} * \delta * (N_{jr} - N_{jd} + \gamma * N_{jr} * N_{lc}). \tag{9}$$

If only the remote windows generated by the jump search process are considered, the number of remote windows N_{jr} will be reduced to $1/J_s^2$ of that in BNM. As a result, the number of candidate windows $(N_{jr} - N_{jd})$ will also be reduced accordingly. However, the small search range generated for the look around search process will become larger as J_s increases. The number of remote windows N_{lc} in the small searching range will increase as the small range become larger (the step length of the look around search is a constant 1 pixel). When jump search and look around search are considered, computation overhead $F(\delta, \eta, \gamma, J_s)$ will be reduced when J_s increases but smaller than a threshold J_{MAX}. However, when J_s is greater than J_{MAX} and continues increasing, $F(\delta, \eta, \gamma, J_s)$ *will stop decreasing and begin to increase*. It is obvious that when J_s is equal to one, JLBNM will become the original BNM.

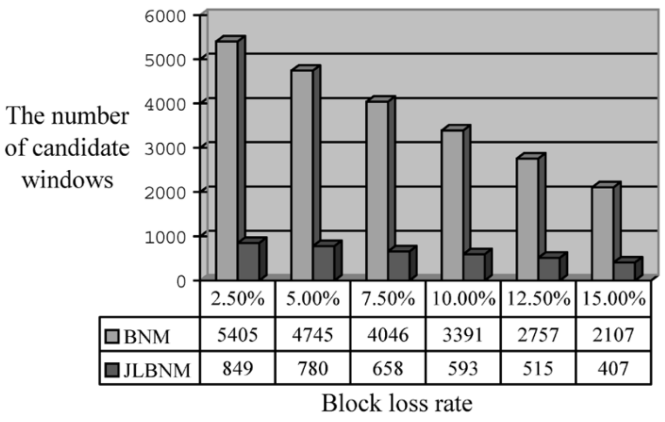
Another important factor in choosing J_s is that the quality of the restored images will decrease as J_s increases. So, we must choose an appropriate J_s to keep the quality of the restored image high and to greatly reduce the computation overhead at the same time.

Tables II and III show results of the average number of candidate windows for a damaged block and PSNR with different J_s for images "*Barb*" and "*Lena*", respectively. These images have complex information in detail, such as sharp edge, texture, and stripe area. The block lost rate is 10%. From these tables we can see that the average number of candidate windows "N_A" and processing time "T_m" reduced rapidly when J_s increases but is smaller than 4, while the PSNR is almost as high as that of BNM., However, when J_s becomes larger than 4 and continues to become greater, "N_A" will stop decreasing and begin to increase (with some PSNR loss). Therefore, according to our experiment results the best choice of J_s is 4 pixels, where the computation overhead of JLBNM is low while PSNR can be kept high.

VI. Experiment Results

Asynchronous transfer mode (ATM) is expected to be the target protocol for broadband integrated services digital network (B-ISDN). In ATM network, data is segmented into fixed length blocks before they are transmitted. They are then inserted into what is called an ATM cell. A cell is 53 bytes in length and consists of an information field carrying user data, and a header containing information used for routing and error detection. Issues such as packing encoded image or video sequences and the influence of packing defects on picture quality have been examined in [12] and [13]. In an ATM network, block losses are mostly due to network congestion. For still image and I frames of video sequences, damaged blocks are distributed randomly. For P frames and B frames, several damaged blocks are always connected into a line. Damage rates range from 0.2% to 5%.

In our experiments the size of both local windows and remote windows is set to be 10 × 10 pixels (or 18 × 18 pixels), because block size is 8 × 8 (or 16 × 16) pixels in JPEG compress standard (or MPEG compress standard). A lost block with 8 × 8 or 16 × 16 pixels is at the center of a local window. In this way, there is a one-pixel wide boundary around the lost block in a local window. Based on the statistics between the best matching windows and local windows in [1], we chose an 80 × 80 square region as a searching range. In most cases, a local window is located at the center of the searching range. When a local window



	2.50%	5.00%	7.50%	10.00%	12.50%	15.00%
▨ BNM	5405	4745	4046	3391	2757	2107
■ JLBNM	849	780	658	593	515	407

Fig. 4. Comparison of the number of candidate windows by BNM and JLBNM.

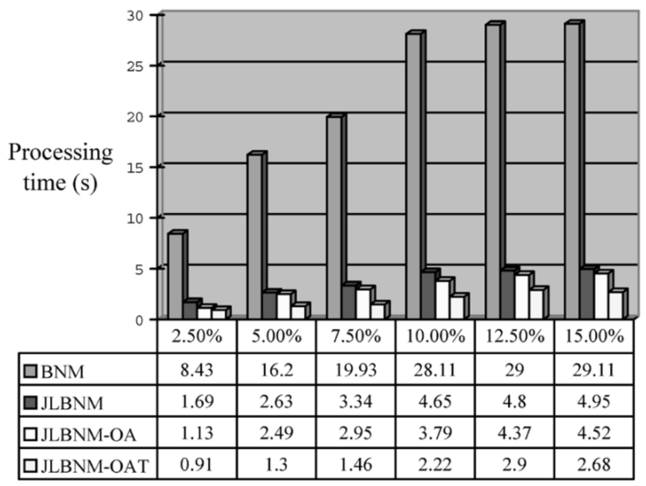
is close to the image boundary, the 80×80 searching range may exceed the image boundary. In such a case, we move the searching range into the image until its boundary overlaps the image boundary. For the matching function v, as mentioned in Section IV, the direct matching function is used in the matching procedure, and a first-order matching function is used in the recovering procedure.

Experiments have been carried out with 8b/pixel gray-level images. Two cases of damaged images are generated by randomly discarding some 8 × 8 or 16 × 16 blocks from some original images. In the first case, lost blocks are restricted to be isolated; in the second case, there exist some adjacent missing blocks connected into big lost regions. For the second case we include images with contiguous damaged blocks, several damaged blocks connected together, and some whole lines damaged.

Experiment results are presented in four groups. In the first group, the average number of candidate windows for a damaged block in the original BNM and JLBNM is shown. In the second group, the processing time for restoring the whole image by the original BNM and JLBNM is given. The PSNR obtained by the original BNM and JLBNM is presented in the third group. In the forth group the real images restored by JLBNM and BNM are displayed in order to compare the subjective quality of restored images.

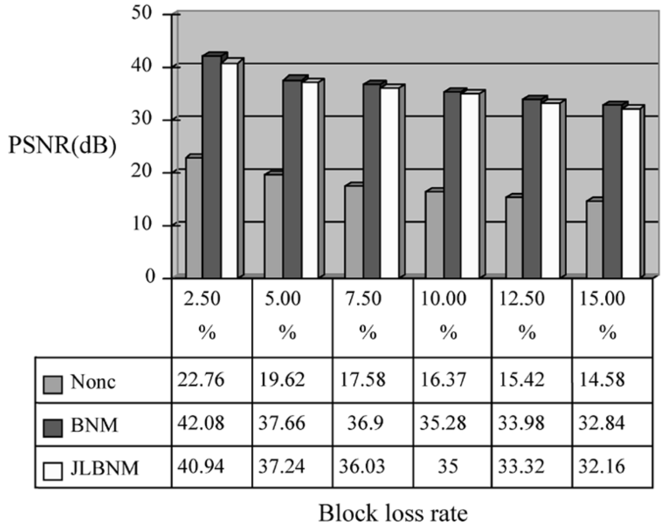
Group 1: Comparison of the Number of Candidate Windows: Fig. 4 shows the average number of candidate windows for damaged blocks by the original BNM and JLBNM. Hereafter, we call them N_B and N_J, respectively. The block loss rates are arranged from 2.5% to 15%. From Fig. 4 we can see that N_J is much smaller than N_B. The ratio of N_J to N_B, "$\mathrm{N_J/N_B}$," is listed in Table IV (from 15% to 19%). For example, when the loss rate is 10%, N_B is 3 391 and N_J is 593. The radio is $593/3391 = 17.4\%$. The processing time will be reduced greatly by applying the JLBNM algorithm since the number of candidate windows is reduced.

Group 2: Comparison of Restoration Time: The processing time for restoring an image using the original BNM, JLBNM, and JLBNM with adaptive threshold (JLBNM-OA, JLBNM-OAT) is shown in Fig. 5. The experiments were carried out on a 233 MHz Pentium PC with 64 MB memory. The processing time by JLBNM is about 17% of that by the original BNM. As



	2.50%	5.00%	7.50%	10.00%	12.50%	15.00%
▨ BNM	8.43	16.2	19.93	28.11	29	29.11
■ JLBNM	1.69	2.63	3.34	4.65	4.8	4.95
□ JLBNM-OA	1.13	2.49	2.95	3.79	4.37	4.52
□ JLBNM-OAT	0.91	1.3	1.46	2.22	2.9	2.68

Fig. 5. Comparison of the processing time by BNM, JLBNM, JLBNM with adaptive threshold, which is represented by “JLBNM-OA,” JLBNM with adaptive threshold and terminal threshold, which is represented by “JLBNM-OAT.”



	2.50 %	5.00 %	7.50 %	10.00 %	12.50 %	15.00 %
▨ Nonc	22.76	19.62	17.58	16.37	15.42	14.58
■ BNM	42.08	37.66	36.9	35.28	33.98	32.84
□ JLBNM	40.94	37.24	36.03	35	33.32	32.16

Fig. 6. Comparison of the PSNR achieved by BNM, JLBNM-OAT. The PSNR without any error concealment, which is represented by “Nonc,” is also shown.

an example, the processing time is 8.43 (second) by the original BNM, and 1.69 (second) by JLBNM for *“Barb”* with a loss rate of 2.5%. For the same loss rate of *“Barb”*, it only takes 1.13 (s) by JLBNM-OA. The radio of the processing time by JLBNM-OA to the original BNM is $1.13/8.43 = 13.4\%$. When the terminal threshold is used, the processing time can be reduced to 10.0% of that by BNM. The ratios of the processing time by the different algorithms are listed in Table IV.

Group 3: Comparison of Objective Image Quality: Fig. 6 shows the objective image quality PSNR by the original BNM and JLBNM for “*Barb*” with the block loss rates ranging from 2.5% to 15%. For easy comparison, the PSNR without any error concealment, marked by “Nonc,” is also shown. The original BNM and JLBNM can improve PSNR by about 18–20 dB. The PSNR by JLBNM is about 1%–3% less than that by the original BNM. In Table IV, the line P_B/P_{JAT} shows the PSNR ratio of JLBNM to BNM. For example, for *“Barb”* with the block loss rate of 10%, the PSNR by JLBNM is 35.00, and 35.28 by the original BNM. The radio is $35.00/35.28 = 99.2\%$. Notice that the PSNR by JLBNM is lower than that of the original BNM, it is still higher than 31.2 dB of Sun’s scheme [6], 32 dB of HCIE scheme [4] and 34.5 dB of Lee’s fuzzy logic scheme [4]. Therefore, the highest image quality can still be achieved by using JLBNM in comparison with all existing image restoration algorithms except the original BNM. Tables V and VI show both the processing time and PSNR using the optimized JLBNM and the original BNM algorithm for the standard images *“Lena”* and *“Baboon”*. From the tables, we can see that the processing time

TABLE V
PROCESSING TIME, PSNR AND CORRESPONDING RADIOS FOR “*LENA*” WITH BLOCK LOSS RATES RANGING FROM 2.5%–15.0% BY BNM AND JLBNM WITH ALL OPTIMIZATION. FOR EASY COMPARISON, “P_N,” THE PSNR WITHOUT ANY CONCEALMENT IS ALSO PROVIDED

	Performance for image “*Lena*” with different Block Loss Rates (*J_s=4)*					
	2.5%	5.0%	7.5%	10.0%	12.5%	15.0%
T_B	8.43	15.24	19.27	23.55	27.43	28.89
T_J	1.10	1.25	2.14	2.66	2.91	3.00
P_N	21.46	18.42	16.71	15.60	14.52	13.65
P_B	36.43	39.77	37.43	34.95	33.75	32.55
P_J	36.40	40.02	37.38	33.92	33.59	32.23
T_J/T_B	13.0%	14.5%	11.1%	11.3%	10.6%	10.4%
P_J/P_B	99.9%	99.3%	99.8%	97.1%	99.5%	99.0%

TABLE VI
PROCESSING TIME, PSNR AND CORRESPONDING RADIOS FOR “*BABOON*” WITH BLOCK LOSS RATES RANGING FROM 2.5%–15.0% BY BNM AND JLBNM WITH ALL OPTIMIZATION

	Performance for image “Baboon” with different Block Loss Rates (*J_s=4)*					
	2.5%	5.0%	7.5%	10.0%	12.5%	15.0%
T_B	8.38	14.95	19.70	24.10	26.73	28.97
T_J	1.06	2.29	2.43	3.11	3.30	3.30
P_N	22.45	19.43	17.60	16.44	15.58	14.72
P_B	36.46	33.04	31.06	30.06	29.15	28.37
P_J	36.38	33.04	31.11	30.06	29.16	28.38
T_J/T_B	12.6%	15.3%	12.3%	12.9%	12.3%	11.4%
P_J/P_B	99.7%	100%	100%	100%	100%	100%

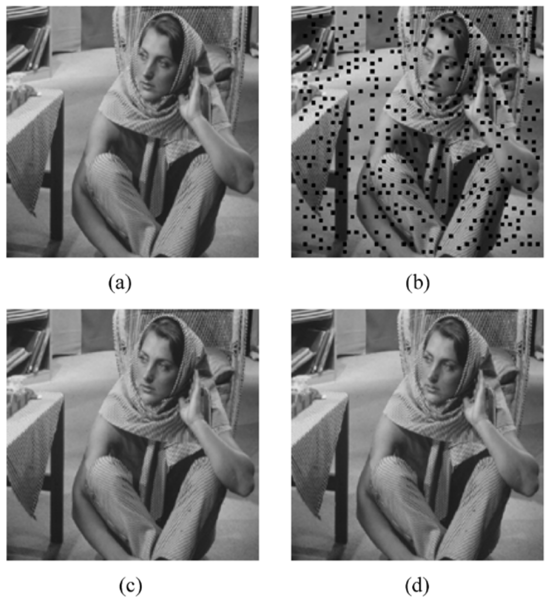


Fig. 7. Comparison of the restoration results for “*Barb*” with block loss rate of 10%. Block size is 8×8. (a) Original image *“Barb.”* (b) Damaged image “*Barb*,” $\mathrm{PSNR} = 16.37$. (c) Restored image by BNM, $\mathrm{PSNR} = 35.29$. (d) Restored image by JLBNM, $\mathrm{PSNR} = 35.00$.

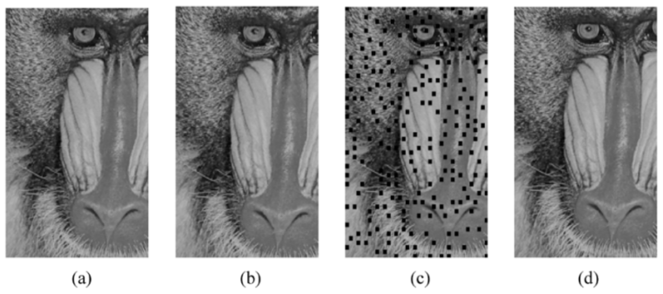


Fig. 8. Illustration of the restoration results of one enlarged area for image "*Baboon*." Block size is 8 × 8. (a) Original enlarged part. (b) Damaged part with 10% block loss rate, $\mathrm{PSNR} = 16.44$. (c) Restored image by BNM, $\mathrm{PSNR} = 30.05$. (d) Restored image by JLBNM, $\mathrm{PSNR} = 30.06$.

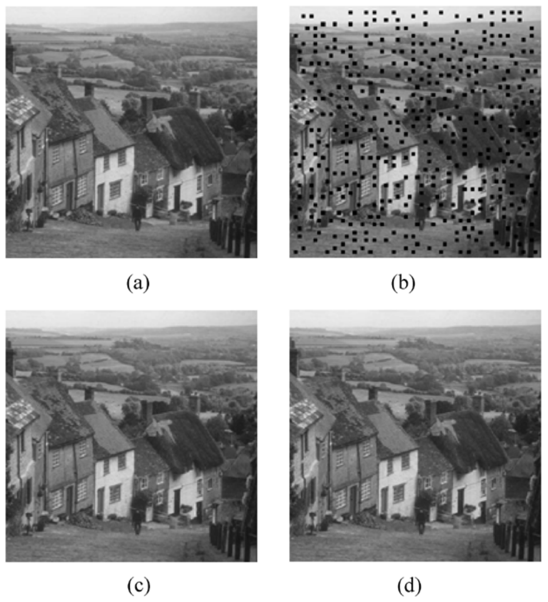


Fig. 9. Illustration of the restoration results for another image "*Goldhill*" with block loss rate 10%. Block size is 8 × 8. (a) Original image. (b) Damaged image "*Goldhill*," $\mathrm{PSNR} = 16.51$. (c) Restored image by BNM $\mathrm{PSNR} = 35.28$. (d) The restored image by JLBNM, $\mathrm{PSNR} = 34.83$.

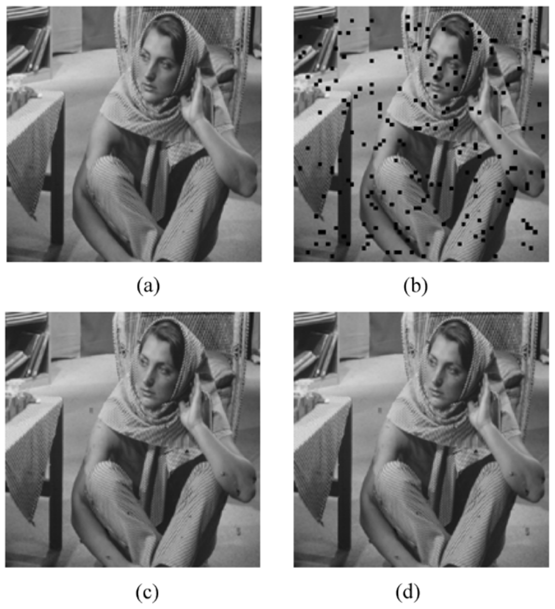


Fig. 10. Illustration of the restoration results for "*Barb*" with contiguous block losses. (a) Original image. (b) Damaged "*Barb*" with the 5% continuous block loss rate, $\mathrm{PSNR} = 19.47$. (c) Restored image by BNM $\mathrm{PSNR} = 35.02$. (d) Restored image by JLBNM $\mathrm{PSNR} = 35.01$.

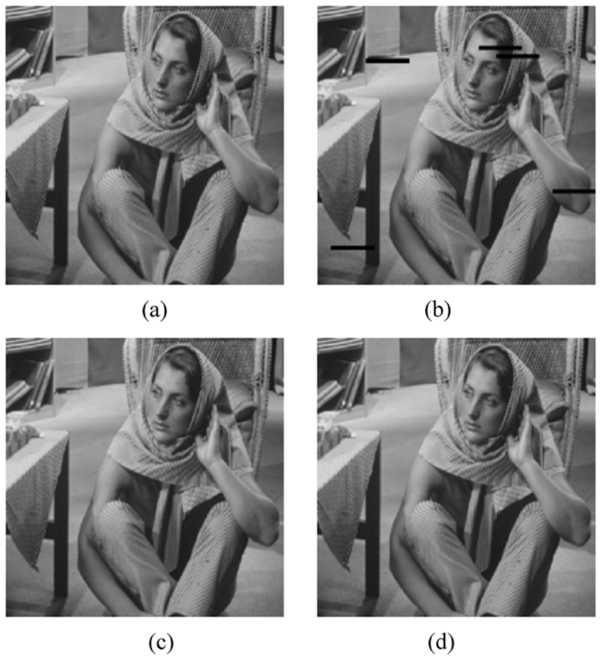


Fig. 11. Restoration results for "*Barb*" with several damaged blocks connecting together. Block size is 8 × 8. (a) Original image. (b) "*Barb*" with 1% continuous block loss ratio, $\mathrm{PSNR} = 25.93$. (c) Restored image by BNM, $\mathrm{PSNR} = 42.42$. (d) Restored image by JLBNM $\mathrm{PSNR} = 42.23$.

by the optimized JLBNM can be reduced to 13%–17% compared with that by the original BNM, and the loss of PSNR is only 1%–3%.

Group 4: Comparison of Subjective Quality: To show subjective quality of restored images by JLBNM, we display restoration results for some standard images ("*Barb*," "*Goldhill*," and "*Baboon*"). The cases of damaged blocks are grouped into isolated damaged blocks, contiguous damaged blocks, several damaged blocks connected together, and several whole lines damaged for image blocks with 8 × 8 and 16 × 16 pixels, respectively. Restoration results for images with blocks of 8 × 8 pixels are shown in Figs. 7–12. Figs. 7–9 indicate images with isolated damaged blocks. The damaged images with a block loss rate of 10% are given in Figs. 7–9(b). The restored images by BNM and JLBNM are shown in Figs. 7–9(c) and (d), respectively. Notice that Figs. 8(a)–(d) indicate an enlarged part of "*Baboon*" to show the difference more clearly. These images contain different details, such as a sharp edge, texture, stripe areas and small objects. The results show that the visual images recovered by JLBNM are very good. Fig. 10 shows images with some contiguous lost blocks. Typical restoration results for images with several damaged blocks linked together are shown in Fig. 11. Restoration results for images with several whole lines damaged are given in Fig. 12. From these results, we can see that the visual quality of the restored images is very good even when the areas contain a great deal of detailed information. Furthermore, when missing blocks in the image are connected into large black regions, they are still recovered very well by our JLBNM. Figs. 13 to 15 show images with

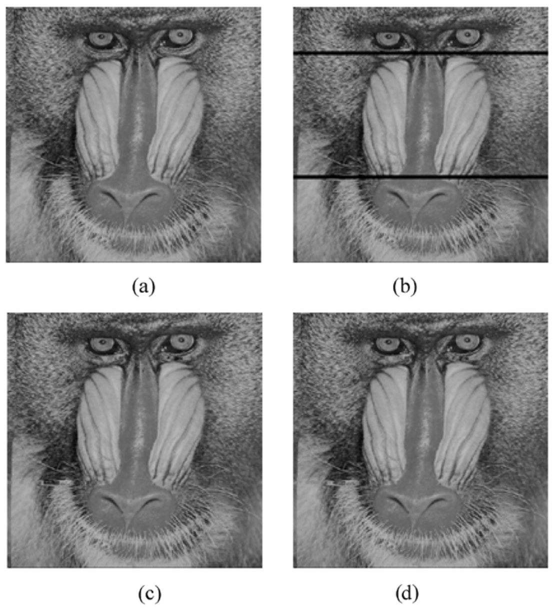


Fig. 12. Restoration results for "*Baboon*" with two whole lines losses. Block size is 8×8. (a) Original image. (b) "*Baboon*" with 3% continuous block loss ratio, $\mathrm{PSNR} = 12.16$. (c) Restored image by BNM, $\mathrm{PSNR} = 30.82$. (d) Restored image by JLBNM $\mathrm{PSNR} = 31.02$.

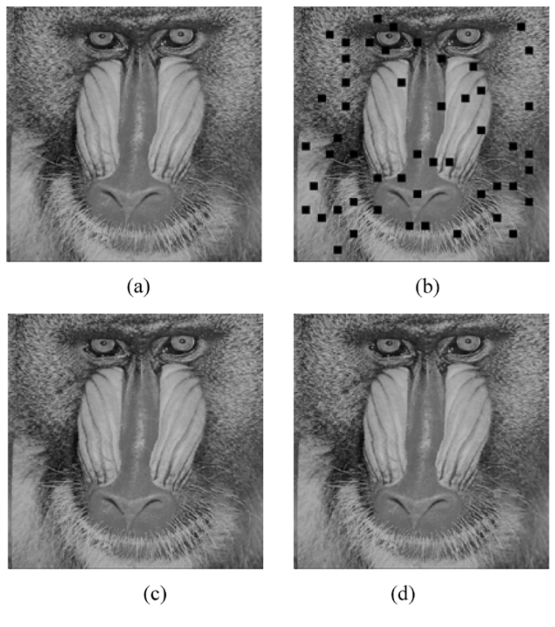


Fig. 13. Restoration results for "*Baboon*" with block loss rate of 5%. Block size is 16×16. (a) Original image *"Baboon"*. (b) Damaged image "*Baboon*," $\mathrm{PSNR} = 19.46$. (c) Restored image by BNM $\mathrm{PSNR} = 31.40$. (d) Restored image by JLBNM, $\mathrm{PSNR} = 31.15$.

blocks of 16×16 pixels. Fig. 13 shows the case of isolated damaged blocks. Typical restoration results for image with several blocks linked together are shown in Fig. 14. Restoration results for images with several whole lines damaged are shown in Fig. 15. From these results, we can see that good image quality can be achieved for images with large blocks damaged by our JLBNM.

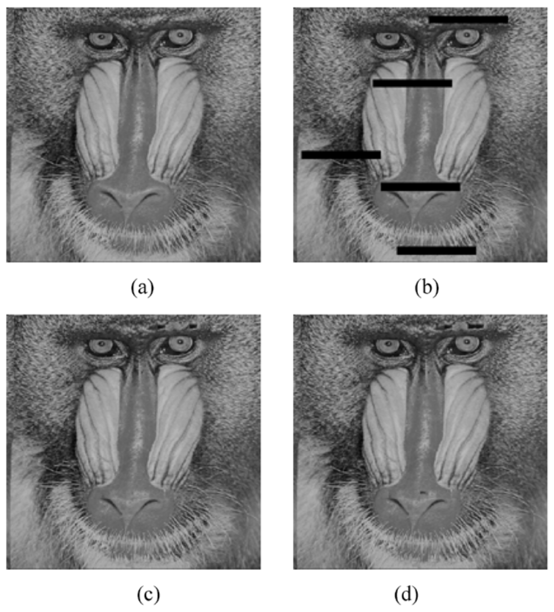


Fig. 14. Restoration results for "*Baboon*" with several damaged blocks connecting together. Block size is 16×16. (a) Original image. (b) "*Baboon*" with 5% continuous block loss ratio, $\mathrm{PSNR} = 19.81$. (c) Restored image by BNM, $\mathrm{PSNR} = 32.92$. (d) Restored image by JLBNM $\mathrm{PSNR} = 32.83$.

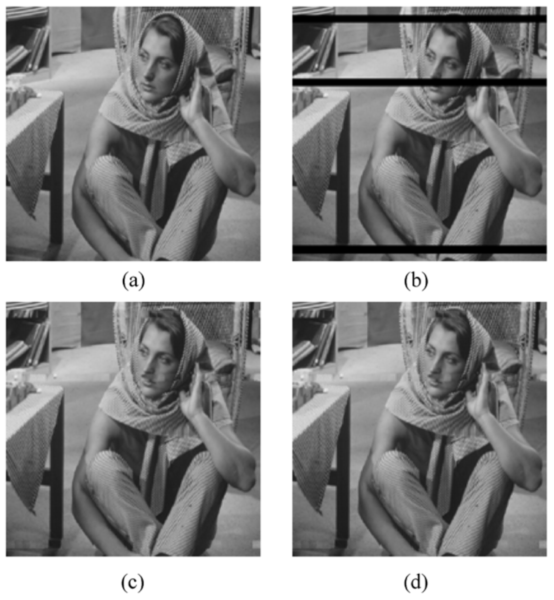


Fig. 15. Restoration results for "*Barb*" with three whole lines losses. Block size is 16×16. (a) Original image. (b) "*Barb*" with 10% continuous block loss ratio, $\mathrm{PSNR} = 16.43$. (c) Restored image by BNM, $\mathrm{PSNR} = 30.45$. (d) Restored image by JBNM $\mathrm{PLSNR} = 30.25$.

VII. Conclusion

An algorithm JLBNM for image restoration is developed in order to reduce computation overhead of the BNM algorithm. In the JLBNM algorithm, two kinds of search processes have been applied alternately with different levels of carefulness in the search process; some candidate blocks are abandoned early so that large number of pixels need not to be processed in such candidate windows; the search process for the best matching block is terminated early when an acceptable block is found so that some windows need not be processed. The objective in the development of JLBNM is to reduce restoration time while keeping the high quality of the restored images. Theoretical analysis and experimental results show that the proposed image restoration algorithm JLBNM can reduce the processing time of the original BNM significantly while having restored images of high quality. The processing time by JLBNM can be reduced to 13%–17% of that by the original BNM. The loss of PSNR by JLBNM is about 1%–3% compared with the PSNR achieved by the original BNM. It is one of the algorithms that produces the highest quality of restored images among existing restoration algorithms.

REFERENCES

[1] Z. Wang, Y. L. Yu, and D. Zhang, "Best neighborhood matching: An information loss restoration technique for block-based image coding systems," *IEEE Trans. Image Process.*, vol. 7, no. 7, pp. 1056–1061, Jul. 1998.

[2] G. Walleye, "The JPEG still picture compression standard," *Commun. ACM*, vol. 34, pp. 30–44, Apr. 1991.

[3] M. Liou, "Overview of the $p * 64$ kbit/s video coding standard," *Commun. ACM*, vol. 34, pp. 59–63, Apr. 1991.

[4] D. Le Gall, "MPEG: A video compression standard for multimedia applications," *Commun. ACM*, vol. 34, pp. 46–58, Apr. 1991.

[5] X. Lee, Y. Q. Zhang, and A. Leon-Garcia, "Information loss recovery for block-based image coding techniques—A fuzzy logic approach," *IEEE Trans. Image Process.*, vol. 4, no. 3, pp. 259–273, Mar. 1995.

[6] S. S. Hemami and T.H. Y. Meng, "Transform coded image reconstruction exploiting interblock correlation," *IEEE Trans. Image Process.*, vol. 4, no. 7, pp. 1023–1027, July 1995.

[7] H. Sun and W. Kwok, "Concealment of damaged block transform coded images using projections onto convex sets," *IEEE Trans. Image Process.*, vol. 4, no. 4, pp. 470–477, Apr. 1995.

[8] W. M. Lam and A. R. Reibman, "An error concealment algorithm for image subject to channel errors," *IEEE Trans. Image Process*, vol. 4, no. 5, pp. 533–542, May 1995.

[9] Z. Wang and D. Zhang, "Restoration of impulse noise corrupted images using long-range correlation," *IEEE Trans. Signal Process.*, vol. 5, no. 1, pp. 4–7, Jan. 1998.

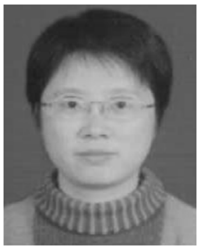
[10] C. W. Chen and Z. H. Sun, "Uniform trellis coded quantization for image transmission over noisy Channels," *Signal Process.: Image Commun.*, vol. 14, no. 6/8, pp. 575–584, May 1999.

[11] I. Rhee and S. R. Joshi, "FEC-based loss recovery for interactive video transmission," in *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, Florence, Italy, June 1999, pp. 250–256.

[12] F. Kishino, K. Matanabe, Y. Hayashi, and H. Yasuda, "Variable bit rate coding of video signals for ATM networks," *IEEE J. Select. Areas Commun.*, vol. 7, no. 5, pp. 801–806, Jun. 1989.

[13] M. Ghanbari and C. Hughes, "Packing coded video signals into ATM cells," *IEEE/ACM Trans. Networking*, vol. 1, no. 5, pp. 505–508, Oct. 1993.

[14] D. Zhang and Z. Wang, "Image information restoration based on long-range correlation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 5, pp. 331–341, May 2002.



Wen Li received the B.Sc. and M.Sc. degrees in computer science, both from the Department of Computer Science, Inner Mongolia University, China, in 1998 an 1991, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2001.

She is an Associate Professor in the Department of Computer Science, Inner Mongolia University. She has published more than 15 technical papers thus far, and her research interests include computer architectures, algorithms, image processing, and parallel processing.

David Zhang (M'90–S'92–SM'95) graduated in computer science from Peking University, Peking, China, in 1974, received the M.Sc. degree in computer science and engineering from the Harbin Institute of Technology (HIT), Harbin, China, in 1983, the Ph.D. degree from the same institution in 1985, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, Ontario, Canada, in 1994.

From 1986 to 1988, he was first a Postdoctoral Fellow at Tsinghua University, Tsinghua, China, and then an Associate Professor at the Academia Sinica, Beijing, China. He is currently with the Hong Kong Polytechnic University, where he is the Founding Director of the Biometrics Technology Centre (UGC/CRC) (www4.comp.polyu.edu.hk/~biometrics/), a body supported by the Hong Kong SAR Government. He also serves as an Adjunct Professor at Tsinghua University, Shanghai Jiao Tong University, Harbin Institute of Technology, and the University of Waterloo. His research interests include automated biometrics-based authentication, pattern recognition, and biometric technology and systems.

Dr. Zhang is the Founder and Editor-in-Chief of the *International Journal of Image and Graphics (IJIG)* (www.worldscinet.com/ijig/ijig.shtml); a Book Editor for the Kluwer International Series on Biometrics (KISB) (www.wkap.nl/prod/s/KISB); a Program Chair for the First International Conference on Biometrics Authentication (ICBA), Associate Editor of more than ten international journals, including the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS A, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS C, and *Pattern Recognition*, and is the author of more than 130 journal papers, twenty book chapters, and ten books. As a principal investigator, he has brought to fruition many biometrics projects and won numerous prizes since 1980. In 1984, his Fingerprint Recognition System won the National Scientific Council of China's third prize, and in 1986, his Real-Time Remote Sensing System took the Council's first prize. In 2002, his Palmprint Identification System won a Silver Medal at the Seoul International Invention Fair, followed in 2003 by taking a Special Gold Award, a Gold Medal, and a Hong Kong Industry Award. He holds a number of patents in both the United States and China, and is a current Croucher Senior Research Fellow.

Zhiyong Liu received the Ph.D. Degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 1987.

He has worked as an Assistant Professor, Associate Professor, and Full Professor in the Institute of Computing Technology, Chinese Academy of Sciences, since 1987. Since 1995, he has been the Deputy Director General (Executive) of the Directorate of Information Sciences, National Natural Science Foundation of China. He worked as a Postdoctoral Fellow and Visiting Scientist with the Pennsylvania State University, State College, the Massachusetts Institute of Technology, Cambridge, the University of Alberta, Canada, and PolyU, CityU, and HKU, Hong Kong. He served as Referee, Editor, and Invited Editor for some academic journals, Program/Organization Committee Member/Chairman and Advisory Committee Member for some international conferences, Steering Expert Committee Member for some research initiatives, and Investigator/Principal Investigator of some projects on computer control systems, computer architectures, and algorithms. He holds some patents, has published more than 70 technical papers, a book, and a number of articles on technical review, science policy, and administrations. His research interests include computer architectures, algorithms, interconnection networks, and parallel processing.

Xiangzhen Qiao received the M.S. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 1981.

She is a Full Professor with the Institute of Computing Technology, Chinese Academy of Sciences. Her main research interests include parallel algorithms, high performance computing, and parallel processing.