Express Letters

A Simple and Efficient Search Algorithm for Block-Matching Motion Estimation

Jianhua Lu and Ming L. Liou

***Abstract*— The three-step search (TSS) algorithm for block-matching motion estimation, due to its simplicity, significant computational reduction, and good performance, has been widely used in real-time video applications. In this paper, a new search algorithm is proposed for further reduction of computational complexity for motion estimation. It will be shown that the proposed algorithm is simple and efficient and requires about one half of the computation for TSS while keeping the same regularity and good performance.**

***Index Terms*— Image sequence analysis, motion compensation, video signal processing.**

I. Introduction

Motion estimation plays an important role in digital video compression. However, its inherent computational complexity poses great challenge for real-time codec implementation. In recent years, studies on fast block-matching algorithms for significantly reducing the computational complexity have gained more and more attention and many effective algorithms have been proposed [1]–[4]. One of the most popular algorithms is the three-step search (TSS) algorithm.

The advantages of the TSS lie in its simplicity, significant computation reduction, and good performance. The principle which supports the TSS is based on the *unimodal error surface assumption*, namely, the block-matching error increases *monotonically* as the search moves away from the position of global minimum error. In fact, this assumption is not always true due to reasons such as the aperture problem and the inconsistent block segmentation of moving object and background, etc. However, experimental results show that the TSS still provides a near-optimal performance even when the assumption does not hold exactly true [5], [6]. Therefore TSS has been widely accepted as one of the best fast block-matching algorithms for low bit-rate real-time video applications and is recommended by CCITT RM 8 [7] and MPEG SM 3 [8].

On the other hand, it is still of interest to further study the TSS in order to improve the performance and/or further reduce the computational complexity. In [1], a new three-step search (NTSS) algorithm has been proposed to improve the performance while keeping the computational complexity similar to the original TSS. This paper proposes a simple and efficient search (SES) algorithm which aims at further reducing the computational complexity. It will be shown that the proposed SES can further speed up the TSS by a factor of two and maintain its regularity and good performance comparable with the TSS.

Manuscript received July 19, 1996; revised October 16, 1996. This paper was recommended by Associate Editor Y.-Q. Zhang. This work was supported by the Hongkong Telecom Institute of Information Technology.

The authors are with the Electrical and Electronic Engineering Department, the Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong.

Publisher Item Identifier S 1051-8215(97)02240-4.

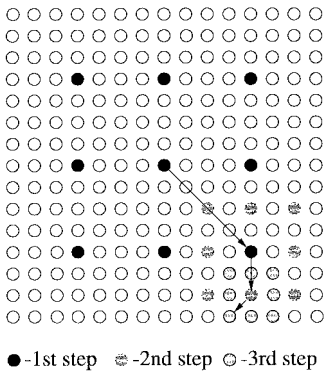

Fig. 1. The TSS procedure ($W = 7$). The motion vector is (3,7) in this example.

II. The Proposed SES Algorithm

A. Preliminaries

Block-matching motion estimation is performed between the current frame and a previously processed frame of a video sequence. The current frame is divided into nonoverlapped square blocks of pixels with size $N \times N$ and each block has a corresponding search area in the previously processed frame which has the size $(2W + N + 1) \times (2W + N + 1)$, where W is the maximum displacement in pixels along both horizontal and vertical directions. Then, for each current block, we look for the block in the search area that best matches the current block. The matching criterion is based on the mean absolute difference (MAD), which can be expressed as

$$MAD(x,y) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} |F_c(k+i,l+j) - F_p(k+x+i,l+y+j)| \tag{1}$$

where $F_c(\cdot,\cdot)$ and $F_p(\cdot,\cdot)$ denote the pixel intensity in the current frame and previously processed frame, respectively, (k,l) is the coordinates of the upper left corner pixel of the current block, and (x,y) represents the displacement in pixels which is relative to the position of current block. After checking each location in the search area, the motion vector is then determined as the (x,y) at which the MAD has the minimum value, i.e., minimum error.

B. Fast Search Algorithms and the TSS

With the full search (FS), each block-matching requires a total of $(2W + 1)^2 \times N^2$ absolute differences, which implies a huge computation for the encoding process. To reduce the computational complexity, many effective algorithms have been proposed which either restrict the number of checking points, such as in [2] and [3], or decrease the computation of absolute differences in (1) by subsampling and other techniques [4], or a combination of both.

The TSS is the algorithm that limits the number of checking points in a search area. In Fig. 1, an example is shown to illustrate the procedure of the TSS for $W = 7$. It must be noted that, though the TSS is originally proposed for low-bit rate video applications where the search window is relatively small, which is typically limited at $W = 7$, its procedure can be extended for cases with $W > 7$ [3]

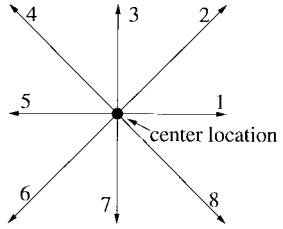

Fig. 2. Search directions in a search pattern of the TSS.

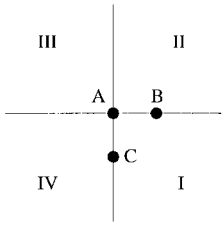

Fig. 3. Search pattern in the first phase of each step of the SES.

while the number of steps becomes more than three. In general, given W, the number of steps required is

$$L = [log_2(W + 1)] \tag{2}$$

where $[x]$ denotes the smallest integer which is larger than or equal to x. Consequently, the step-size (distance between pixels in a search step) for nth step is given by

$$ss(n) = 2^{L-n}. \tag{3}$$

It is shown that the TSS uses a uniformly distributed search pattern in each step and hence exhibits simplicity and regularity. In particular, the number of checking points is nine for the first step and eight for the subsequent steps (excluding the location which is already checked in the previous step). For $W = 7$, the total number of checking points for TSS is 25. Therefore, compared with 225 checking points in the FS, the speed up ratio for TSS is nine.

C. The Proposed SES Algorithm

First, we define the direction from center location to a surrounding location as a *search direction.* Then, in the TSS, there are a total of eight search directions, as shown in Fig. 2, involved in each step. It is observed that, due to uniformly distributed search pattern, each direction has an opposite direction which is also searched. This is not effective under the unimodal error surface assumption on which the TSS is based. Recall that the unimodal error surface has the property that the block-matching error increases monotonically as the search is along the direction away from the position of the global minimum error, which in turn implies that minimum cannot occur in two directions opposite to each other *simultaneously*. In other words, for the search pattern in TSS, at most half of the total eight directions need actually be searched in each step, and thus, the computational complexity can be further reduced. On the other hand, to determine which directions are to be chosen, additional computation is needed.

A closer observation indicates that we can first check two arbitrary but orthogonal directions (say directions 1 and 7 in Fig. 2) and then restrict the search area of each step in certain quadrant which contains only three search directions. In that sense, we propose the SES algorithm.

The SES algorithm also uses (2) and (3) to determine the number of steps required and the step-size used. The innovation is that each step is further divided into two phases: the first phase is to select a search quadrant; the second phase is to find the minimum error location in the selected quadrant.

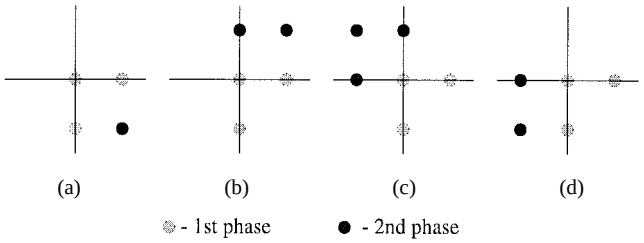

Fig. 4. Search patterns corresponding to each selected quadrant: (a) quadrant I, (b) quadrant II, (c) quadrant III, and (d) quadrant IV.

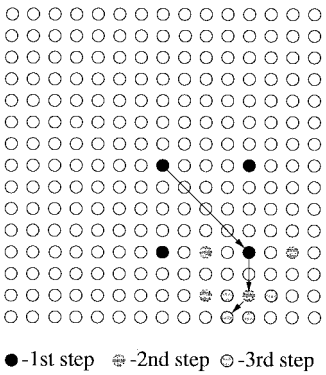

Fig. 5. The SES procedure $(W = 7)$. The motion vector is (3,7) in this example.

Consider the nth step. In the first phase, we compute the MAD's of A, B, and C as shown in Fig. 3, where A is the center location, B and C are the locations ss(n) pixels away from A along the horizontal and vertical directions, respectively. Note that the directions from A to B and from A to C correspond to directions 1 and 7 in Fig. 2, respectively. Furthermore, we denote MAD (X) as the MAD of location X and label four quadrants I, II, III, and IV as shown. The rule for determining a search quadrant can be described as follows.

If MAD (A) $\geq$ MAD (B) and MAD (A) $\geq$ MAD (C), I is selected;
If MAD (A) $\geq$ MAD (B) and MAD (A) $<$ MAD (C), II is selected;
If MAD (A) $<$ MAD (B) and MAD (A) $<$ MAD (C), III is selected;
If MAD (A) $<$ MAD (B) and MAD (A) $\geq$ MAD (C), IV is selected.

The patterns of checking points during the second phase of each step in each selected quadrant are shown in Fig. 4. Fig. 5 shows an example for the SES corresponding to the TSS example shown in Fig. 1.

It is observed that the SES follows the same procedure in each step and thus retains the regularity comparable with the TSS. In particular, as shown in Fig. 4, the number of checking points for the SES is five on average in the first step and four on average in the subsequent steps (excluding the location which is checked in the previous step). Therefore, the total number of checking points for each block-matching is further reduced compared with the TSS. Table I compares the computational complexity and speed-up ratio of three search algorithms at different values of W. It can be shown that the proposed SES can further speed up the TSS by a factor about two.

III. Simulation and Comparison

To verify the usefulness of the SES, we compare it with three other algorithms, namely, the FS, TSS, and the one-at-a-time search (OTS) [6], using a 5-s segment of the Football and 10-s segment of the Tennis as the test sequences. Both sequences are in the common

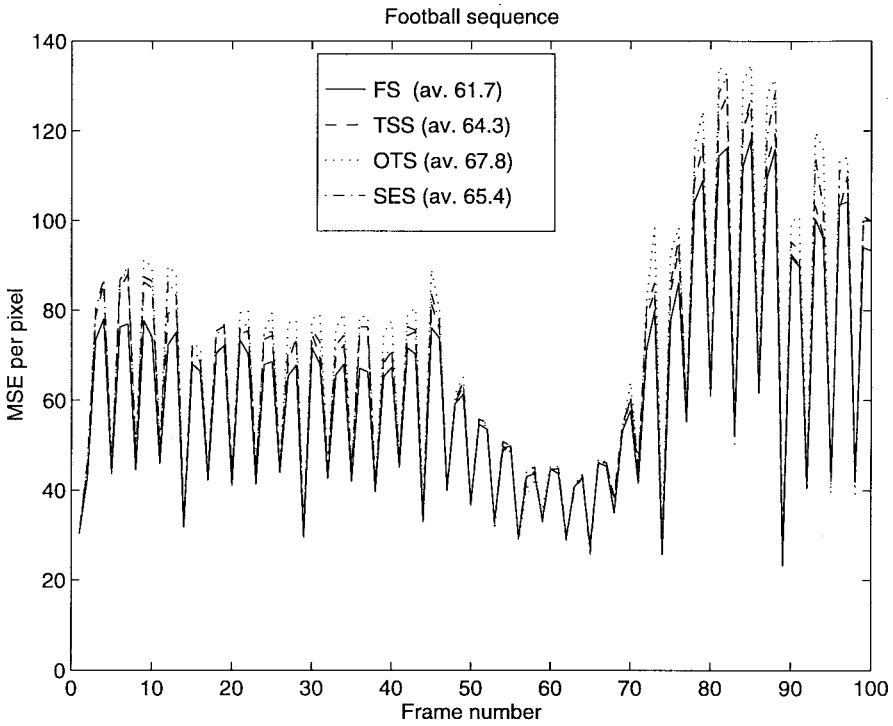

Fig. 6. Performance comparison in terms of MSE per pixel.

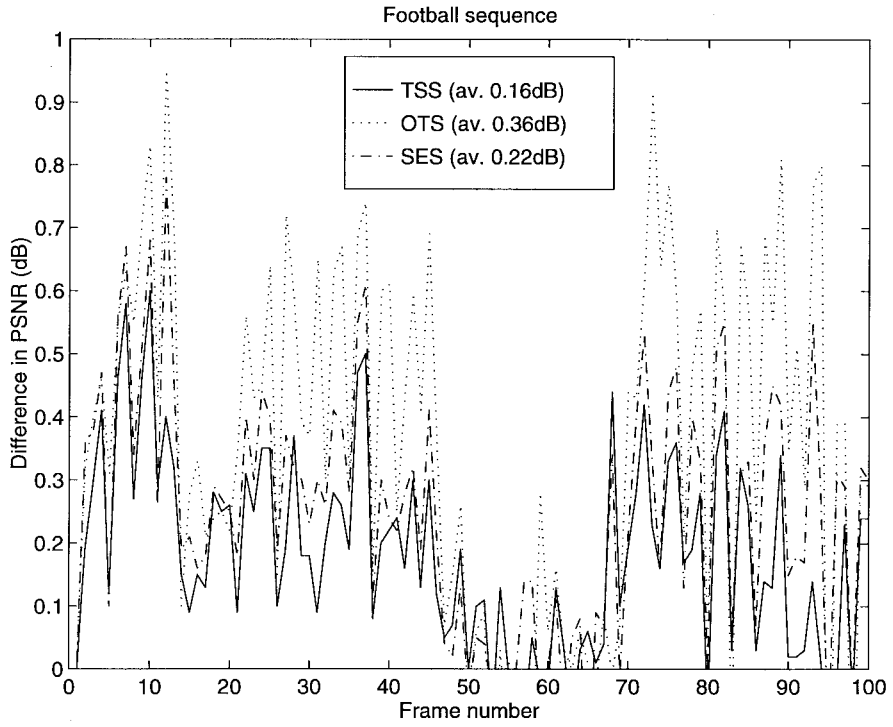

Fig. 7. Performance comparison in terms of the difference in PSNR with respect to FS.

intermediate format (CIF) (30 frames/s with 352 $\times$ 240 pixels/frame). It is noted that these sequences contain different combinations of still, slow, and fast moving objects, camera zoom, and panning. Moreover, only the results obtained with the first 100 frames of both sequences are presented, since they are sufficiently representative of the overall results. The parameters used in the simulations are listed in Table II. Performance comparison among different search algorithms is based on peak signal-to-noise ratio (PSNR) and mean square error (MSE) per pixel of the motion-compensated frames.

Fig. 6 compares the performance of different search algorithms in terms of MSE for the Football sequence. It is observed that the SES and TSS exhibit very close performances which are better than that of the OTS. Also, both the SES and TSS are worse than the FS, but the difference in MSE is very small. A clearer comparison by showing

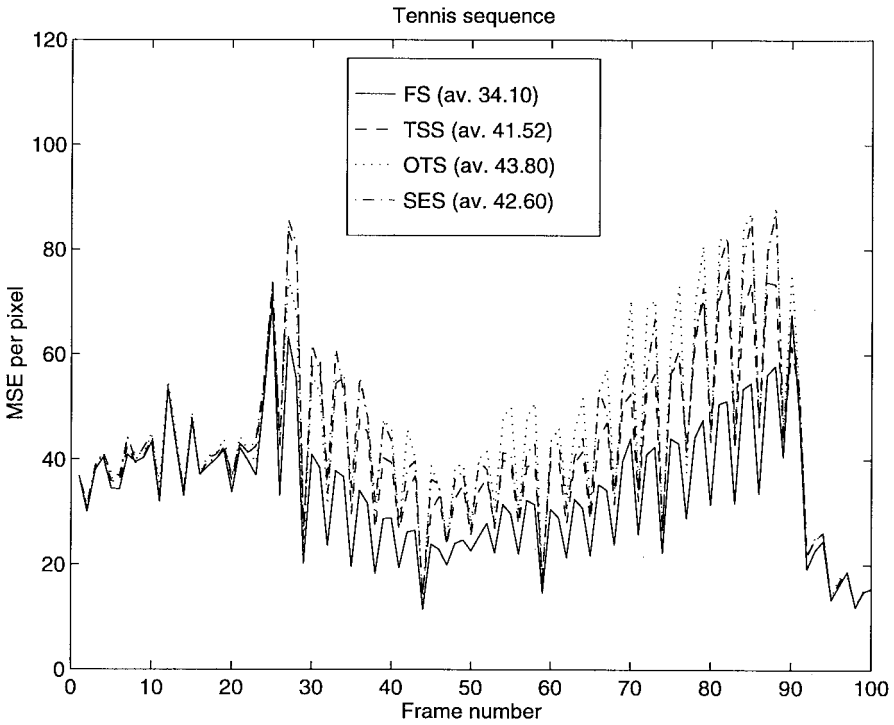

Fig. 8. Performance comparison in terms of MSE per pixel.

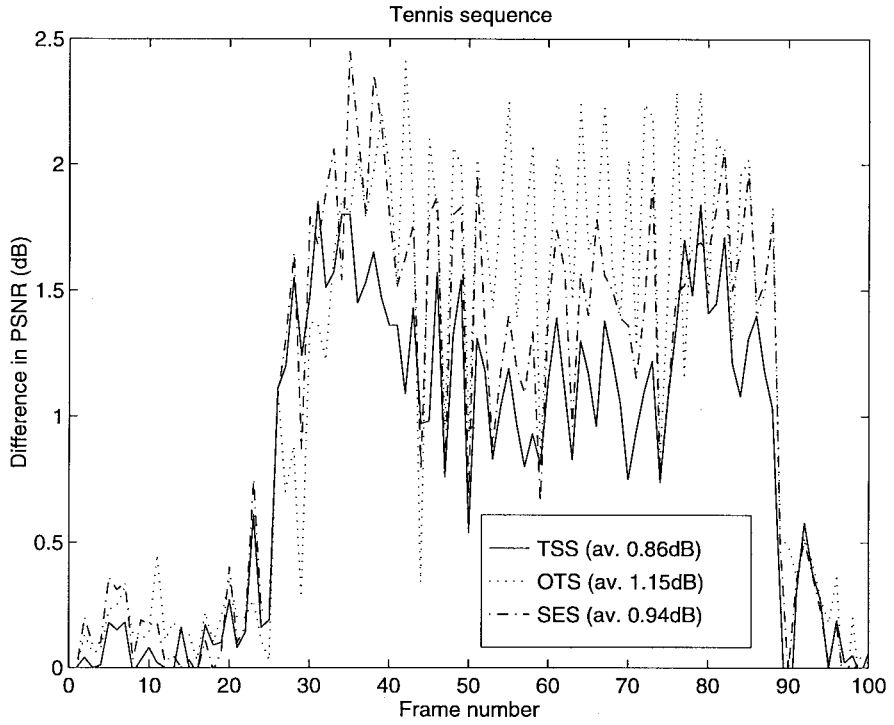

Fig. 9. Performance comparison in terms of the difference in PSNR with respect to FS.

the difference in PSNR with respect to the FS is given in Fig. 7. In particular, the average PSNR differences for the TSS, SES, and OTS are shown to be 0.16 dB, 0.22 dB, and 0.36 dB, respectively, which in turn indicate that the SES has similar performance as TSS and better performance than the OTS. Similarly, Fig. 8 demonstrates the MSE performance of different search algorithms in the case of Tennis sequence. Note that all the three fast search algorithms exhibit more performance degradation than in the case of Football sequence, but the overall values of MSE remain reasonably small and thus can produce acceptable quality. Also, the MSE for both the SES and TSS are very close. Likewise, Fig. 9 indicates that the SES and TSS have comparable performance in terms of the difference in PSNR with respect to the FS. Therefore, we conclude that the SES can provide good performance comparable with the TSS.

TABLE I
COMPARISON OF COMPUTATIONAL COMPLEXITY OF THE SEARCH ALGORITHMS

Algorithm	W = 7		W = 14		W = 21	
	Number of Points	Speed-up Ratio	Number of Points	Speed-up Ratio	Number of Points	Speed-up Ratio
FS	15 × 15	1	29 × 29	1	43 × 43	1
TSS	25	9.0	33	25.5	41	45.1
SES	13	17.3	17	49.5	21	88.0

TABLE II
PARAMETERS FOR SIMULATIONS

Block Size	16 × 16 (i. e., N = 16)
Matching Criterion	MAD
Simulation Package	MPEG I simulation program [9]
Coding Rate	1300.0 kbps
Group Size	15 frames
Frame Arrangement	IBBPBBPBBPBBPBBIBBP ···
Values of W	7, 14 and 21 for 1, 2 and 3 frame distance, respectively
Search Algorithms	FS, TSS, SES, OTS

In addition, since the SES algorithm maintains the similar regularity and parallelism comparable with the TSS, we can use a similar structure as proposed in [5] for its VLSI implementation. Likewise, by using the statistical motion vector distribution information as proposed in [1], the performance of the SES can be further improved. Finally, the SES algorithm should be an excellent candidate for real-time software-based implementation of codecs for low bit-rate applications.

IV. CONCLUSIONS

In this paper, we have proposed an SES algorithm for fast block-matching motion estimation. It has been shown that the SES can further speed up the TSS by a factor about two while maintaining regularity and parallelism. Experimental results based on simulations, which are performed by using MPEG-1 simulation program and two test sequences, Football and Tennis, demonstrates that the SES can provide good performance comparable with the TSS in terms of PSNR and MSE. In particular, by imposing the similar algorithm as proposed in [1], the performance of the SES can be further improved.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments and suggestions.

REFERENCES

[1] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 438–442, Aug. 1994.
[2] L. G. Chen, W. T. Chen, Y. S. Jehng, and T. D. Chiueh, "An efficient parallel motion estimation algorithm for digital image processing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, pp. 378–385, Dec. 1991.
[3] K. H.-K. Chow and M. L. Liou, "Genetic motion search algorithm for video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 440–445, Dec. 1993.
[4] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 148–157, Apr. 1993.
[5] H. M. Jong, L. G. Chen, and T. D. Chiueh, "Parallel architectures for 3-step hierarchical search block-matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 407–416, Aug. 1994.
[6] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans. Commun.*, vol. COM-33, pp. 888–896, Aug. 1985.
[7] CCITT SGXV, "Description of reference model 8 (RM8)," Document 525, Working Party XV/4, Specialists Group on Coding for Visual Telephony, June 1989.
[8] MPEG, "ISO CD11172-2: Coding of moving pictures and associated audio for digital storage media at up to 1. 5 Mbits/s," Nov. 1991.
[9] Draft International Standard ISO/IEC DIS 11172, "Coding of moving pictures and associated audio for digital storage media at up to 1. 5 Mbits/s," Nov. 1992

A Deblocking Algorithm for JPEG Compressed Images Using Overcomplete Wavelet Representations

Zixiang Xiong, Michael T. Orchard, and Ya-Qin Zhang

***Abstract*— This paper introduces a new approach to deblocking of JPEG compressed images using overcomplete wavelet representations. By exploiting cross-scale correlations among wavelet coefficients, edge information in the JPEG compressed images is extracted and protected, while blocky noise in the smooth background regions is smoothed out in the wavelet domain. Compared with the iterative methods reported in the literature, our simple wavelet-based method has much lower computational complexity, yet it is capable of achieving the same peak signal-to-noise ratio (PSNR) improvement as the best iterative method and giving visually very pleasing images as well.**

***Index Terms*—Blocking effect, denoising, JPEG, wavelets.**

I. INTRODUCTION

With the widespread adoption of the JPEG standard for still image compression [1], there has been interest in deblocking of JPEG compressed images which demonstrate blocking effects, especially at low bit rates. This blocking effect is due to the underlying block-based approach JPEG takes that processes each block independently. At low bit rates, the JPEG compression process introduces large high-frequency quantization error to each individual block, resulting in discontinuity across block boundaries in the JPEG compressed image, hence, the annoying blocking effects.

To remove this blocking effect, several deblocking techniques have been proposed in the literature as postprocess mechanisms after JPEG compression, depending on the angle from which the deblocking problem is tackled. If deblocking is viewed as an inverse (or estimation) problem, the simplest solution is probably just to lowpass the blocky JPEG compressed image, as was done in [2]. More sophisticated approaches involve iterative methods such as

Manuscript received May 30, 1996; revised December 2, 1996. This paper was recommended by Associate Editor J. Brailean.

Z. Xiong and M. T. Orchard are with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA.
Y.-Q. Zhang is with the David Sarnoff Research Center, Princeton, NJ 08543 USA.

Publisher Item Identifier S 1051-8215(97)02241-6.