Deinterlacing Using Hierarchical Motion Analysis

Qian Huang, Debin Zhao, Siwei Ma, Wen Gao, *Fellow, IEEE,*
and Huifang Sun, *Fellow, IEEE*

***Abstract*—A motion-compensated deinterlacing scheme based on hierarchical motion analysis is presented. According to deinterlacing steps, our contribution can be divided into four parts: motion estimation, motion state analysis, motion consistency analysis, and finer-grained interpolation. In motion estimation, we introduce a Gaussian noise model for choosing the best motion vector for each block, and make a tradeoff between utilizing previous de-interlaced frames and avoiding error propagation. A directional interpolation method is also introduced in this part for backward fields. In motion state analysis, we define two motion states for each pixel, thus achieve a compromise between traditional block-based strategies and the extreme pixel-based case. In motion consistency analysis, we propose to measure both the motion vector consistency and the motion state consistency in order to determine whether the previous two parts should be performed again with a different block size. In finer-grained interpolation, we utilize a combination of recursive median filters to generate the final results. Experimental results show that all of the proposed techniques are effective, either objectively or subjectively. As a result, we can achieve much higher image quality, with an average gain of about 1.83 dB in terms of peak signal-to-noise ratio. Moreover, the increased computation complexity is marginal.**

***Index Terms*—Deinterlacing, hierarchical motion analysis, motion consistency, noise model.**

I. Introduction

DEINTERLACING is a kind of format conversion that converts conventional interlaced signals such as those for broadcasting in TV due to bandwidth limitations to progressive signals such as those for computers. Nowadays, the demand for video deinterlacing grows day by day since we have piled up hundreds of thousands of valuable interlaced videos. In the past three decades, most researchers consider deinterlacing as the reverse process of the most common 2:1 interlacing [1] and aim at converting interlaced videos at 50/60 fields per second to progressive ones at 50/60 frames per second. The progressive format at 25/30 frames per second is usually not preferable because the display refresh rate should not be lower than 50 times per second in order to achieve continuous flicker-free motion [2]. We sort this kind of research into the first category. Recently, some other research perspectives have come into being. For example, R. Castagno *et al.* [3] focus on 50 Hz-to-75 Hz interlaced-to-progressive conversion. S. Muramatsu *et al.* [4] study invertible deinterlacing. M. Biswas *et al.* [5] stick to the performance analysis of deinterlacing system. C. Ballester *et al.* [6] propose to perform deinterlacing with inpainting. Since it is very difficult to categorize these researches, we simply classify them into the second category. As for the first category, motion compensation based deinterlacing algorithms have been developed as the most widely adopted ones [7]–[10], which comprise three typical stages: motion estimation, motion state analysis, and motion-compensated interpolation.

Motion estimation (ME) is one of the most important and computation-intensive parts of video codecs, thus it has been studied for years. However, ME algorithms used in video codecs usually estimate average motion over several field periods rather than motion between adjacent fields, e.g., the case when B pictures are used. Therefore, even the most efficient ME algorithms in video compression are not directly applicable to deinterlacing, since deinterlacing algorithms typically require only one single motion vector for each pixel, otherwise blurring might be introduced. Considering that pixel-based ME [11] is too complicated, most deinterlacing algorithms employ block-based ME algorithms, between either opposite-parity fields [12] or same-parity fields [13]. Opposite-parity ME outperforms same-parity ME when the vertical component of a motion vector is odd; however, it suffers from latent error propagation. Both the two kinds of ME are taken into account in [14]; however, the block sizes are fixed as most other algorithms do, which is not appropriate to predict or compensate various kinds of motion due to noise and the aperture problem. To tackle these problems, several methods are proposed to use variable block sizes [15]–[17]; however, all the available block sizes are still fixed (e.g., $16 \times 16 \rightarrow 8 \times 8 \rightarrow 4 \times 4$) hence the inherent problem cannot be solved radically.

Motion state analysis (MSA) is performed immediately after motion estimation. In this stage, motion vectors and texture information are typically utilized to determine whether and

Manuscript received October 19, 2008; revised February 2, 2009 and September 4, 2009. First version published March 15, 2010; current version published May 5, 2010. This work was supported in part by the National Basic Research Program of China (973 Program 2009CB320904) and the National Science Foundation of China (60833013). This paper was recommended by Associate Editor M. Comer.

Q. Huang is with the Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences (CAS), Beijing 100190, China and with the Graduate University of the Chinese Academy of Sciences, Beijing 100049, China (e-mail: qhuang@jdl.ac.cn).

D. Zhao is with the Department of Computer Science, Harbin Institute of Technology, Harbin 150001, China (e-mail: dbzhao@vilab.hit.edu.cn).

S. Ma and W. Gao are with the Institute of Digital Media, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China (e-mail: swma@pku.edu.cn; wgao@pku.edu.cn).

H. Sun is with the Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139 USA (e-mail: hsun@merl.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2010.2045807

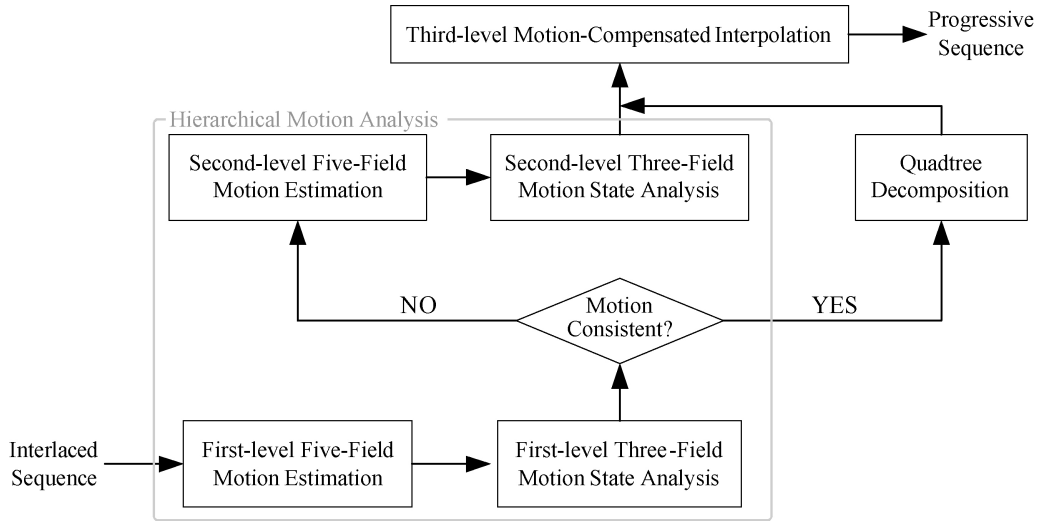


Fig. 1. Proposed deinterlacing scheme.

how a given region (typically a block) is moving. To achieve these goals, the sum of absolute difference (SAD) is usually used to measure the similarity along motion trajectories, with the help of other criteria such as the standard deviation (STD). The most intuitive motion states as indicated in [18] are stationary and moving. Some researchers proposed to define more motion states to make a finer-grained processing [19]–[21]. However, these approaches cannot work well for motions such as rotation, zoom, and occlusion. Chang *et al.* [22] and Huang *et al.* [23] tried to make the occlusion problem less suffering by using more reference fields, but the performance is still unsatisfactory for rotation and zoom cases (e.g., the *Mobile* sequence in [22]). Moreover, existing block-based MSA methods typically have only one single state for each motion-compensated block, which is not suitable for finer-grained interpolation. Pixel-based MSA [24] can be used to address these problems; however, the computation complexity is too high. Thus, a balance between conventional block-based MSA and pixel-based MSA is needed.

Motion-compensated interpolation (MCI) is the last stage of video deinterlacing. It should be performed based on the results of MSA, with the assistance of motion information and texture information of each region. Generally speaking, MCI methods can be divided into three categories: motion adaptive MCI [8], recursive MCI [25], [26], and generalized sampling theorem (GST)-based MCI [27], [28]. Motion adaptive MCI algorithms take noise into account; however, the typical assumption that an obtained image equals to the sum of original image and noise is not valid in many practical cases [29]. Recursive MCI algorithms show the best performance; however, they are threatened by latent error propagation. GST-based MCI algorithms never suffer from error propagation; however, artifacts might be introduced when the motion vectors are erroneous. Adaptive deinterlacing algorithms [23], [30] achieve better results by applying different methods to different regions; however, as for video sequences with motions such as rotation, zoom, and occlusion, finer-grained interpolation is required for better visual perception and lower probability of error propagation.

Based on the above discussions, a new block strategy is indispensable for more efficient ME, MSA, and MCI. As the experimental results have shown that two fixed-size block levels are generally good for video coding [31] and video deinterlacing [16], we propose a three-leveled block strategy that has two fixed-size block levels and one variable-size block level to achieve better prediction and motion compensation. In this paper, a motion-compensated deinterlacing scheme is presented based on hierarchical motion analysis, which is incarnated by the novel three-leveled block strategy. First, motion estimation and motion state analysis are performed successively for first-level blocks. Then motion vector consistency and motion state consistency are analyzed to decide whether to perform the second-level motion estimation and second-level motion state analysis. Finally, a finer-grained interpolation is proposed at the variable-size block level, which is introduced to better solve the aperture problem than the previous methods. The rest of this paper is organized as follows. The proposed deinterlacing scheme is described in Section II, where the first three subsections give the hierarchical motion analysis model and the last subsection deals with finer-grained interpolation. Experimental results are presented in Section III. Section IV draws our conclusion.

II. Proposed Deinterlacing Scheme

The proposed motion-compensated deinterlacing scheme is sketched in Fig. 1, which consists of four main modules, i.e., five-field motion estimation, three-field motion state analysis, motion consistency analysis for switching between the two fixed-size block levels, and finer-grained MCI at the third block level. Among them, the first three modules (five-field motion estimation, three-field motion state analysis, and motion consistency analysis) constitute the hierarchical motion analysis model.

In Fig. 1, the first-level motion estimation and motion state analysis differ from second-level ones only in block sizes.

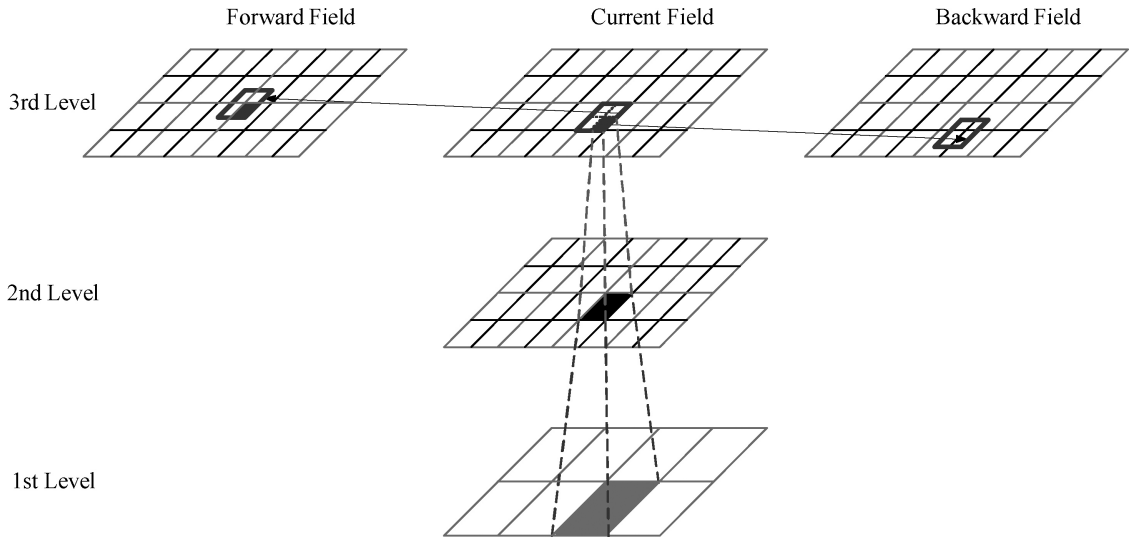


Fig. 2. Proposed three-leveled block strategy for deinterlacing.

The quadtree decomposition module equally divides a first-level block into four second-level blocks, each of which has the same motion vector and motion state with the first-level block. To interpret the third block level, we illustrate our block strategy in Fig. 2, where the first and second fixed-size block levels are represented by gray and black blocks, respectively. As can be seen from the third level, for each second-level fixed-size block in the current field, each of its reference blocks overlies at most four adjacent second-level fixed-size blocks that might have different motion vectors and motion states. Therefore, it might be better to subdivide the second-level fixed-size blocks according to motion vectors, rather than simply performing quadtree decomposition again.

Before discussing the individual modules in Fig. 1, we would like to introduce some background information at first. In [16], we have proposed an improved 3-D recursive search (3-DRS) block matching algorithm [32], which measures the matching error by

$$\varepsilon = \sum_{IB(x,\ y)} |f_{n+1}(x+dx,\ y+dy) - F_{n-1}(x-dx,\ y-dy)| \quad (1)$$

where $(x,\ y)$ designates the spatial position in the current field f_n and $(dx,\ dy)$ is the motion vector. $IB(x,\ y)$ is the interlaced block that contains $(x,\ y)$, whereas f_{n+1} and F_{n-1} are the backward reference field and the previously interpolated frame, respectively. Since F_{n-1} is a full frame and there is no vertical restriction on $F_{n-1}(x-dx,\ y-dy)$, we add a restriction that dy should be even in order to avoid error propagation. Unfortunately, it is not effective when the vertical component of a motion vector is odd. Moreover, noise is not considered. To address these problems, we further proposed a probabilistic motion estimation model in [23]. However, the block size for motion estimation was fixed so that it was not suitable for predicting different kinds of motion. Therefore, in this paper we propose to utilize two fixed-size block levels for better performance.

Now the relationships within Fig. 1 are clear. We first perform first-level motion estimation to get the motion vectors and then determine whether the first-level blocks are moving. After that, motion consistency is analyzed to determine whether motion estimation and motion state analysis should be performed again at the second-level; if not, quadtree decomposition is used to directly assign the motion vector and motion state of a large block to its sub-blocks. When all these are finished, a finer-grained interpolation is then applied. In the following subsections, the four main modules are introduced in sequence.

A. Five-Field Motion Estimation

In image/video processing, the following equation is typically assumed, taking additive white Gaussian noise into account

$$F_n(x,\ y) = G_n(x,\ y) + V_n(x,\ y) \quad (2)$$

where $F_n(x,\ y)$, $G_n(x,\ y)$, and $V_n(x,\ y)$ are the obtained image, original image, and Gaussian noise of the current nth picture, respectively. However, considering factors such as the ununiformity of camera sensitivity and uneven object illumination, a multiplicative error coefficient $e_n(x,\ y)$ is usually necessary to suppress the systematic degradation [29]

$$F_n(x,\ y) = e_n(x,\ y) \cdot G_n(x,\ y) + V_n(x,\ y). \quad (3)$$

Assuming the forward and backward reference positions of $(x,\ y)$ are $(x',\ y')$ and $(x'',\ y'')$, respectively, then we have

$$\begin{aligned} &F_n(x,\ y) - F_{n-1}(x',\ y') \\ &\quad = [e_n(x,\ y) - e_{n-1}(x',\ y')] \cdot G_n(x,\ y) \\ &\qquad + [V_n(x,\ y) - V_{n-1}(x',\ y')] F_{n+1}(x'',\ y'') - F_n(x,\ y) \\ &\quad = [e_{n+1}(x'',\ y'') - e_n(x,\ y)] \cdot G_n(x,\ y) \\ &\qquad + [V_{n+1}(x'',\ y'') - V_n(x,\ y)]. \end{aligned} \quad (4)$$

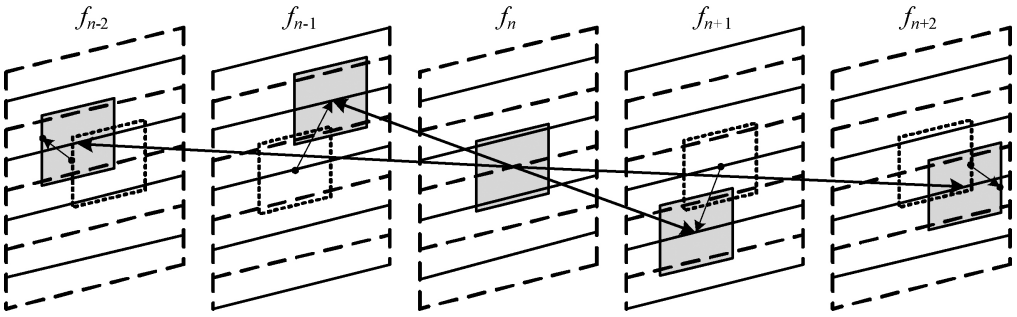


Fig. 3. Five-field motion estimation.

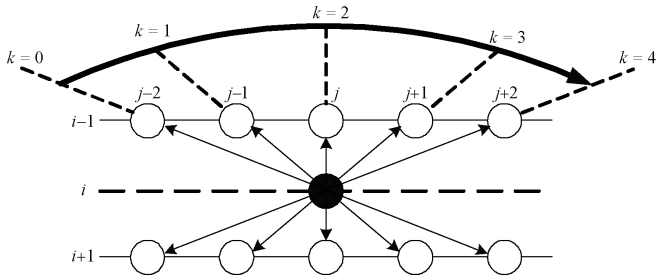


Fig. 4. Five directions for interpolation.

From the above two equations we can get the following:

$$\begin{aligned} &F_{n+1}(x'', y'') - \frac{e_{n+1}(x'', y'') - e_{n-1}(x', y')}{e_n(x, y) - e_{n-1}(x', y')} F_n(x, y) \\ &\quad + \frac{e_{n+1}(x'', y'') - e_n(x, y)}{e_n(x, y) - e_{n-1}(x', y')} F_{n-1}(x', y') \\ &= \left[V_{n+1}(x'', y'') - V_n(x, y)\right] - \frac{e_{n+1}(x'', y'') - e_n(x, y)}{e_n(x, y) - e_{n-1}(x', y')} \\ &\quad \left[V_n(x, y) - V_{n-1}(x', y')\right]. \end{aligned} \tag{5}$$

In general, it is not easy to get an accurate estimation of $e_n(x, y)$. Taking into account the self-correction ability (e.g., self-calibration) of modern cameras, we can make another assumption that $\{e_n\}$ is a decreasing arithmetical series along the motion trajectory, then we can obtain

$$\begin{aligned} &F_{n+1}(x'', y'') - 2F_n(x, y) + F_{n-1}(x', y') \\ &\quad = V_{n+1}(x'', y'') - 2V_n(x, y) + V_{n-1}(x', y'). \end{aligned} \tag{6}$$

Since $V_{n+1}(x'', y'')$, $V_n(x, y)$, and $V_{n-1}(x', y')$ are Gaussian noises, the left-hand side of (6) also obeys a Gaussian distribution. Therefore, based on the discussion in [23], the conditional probability that measures how well the current nth frame can be described by reference frames and motion vectors grows with the decrease of the following equation:

$$\frac{\left(\sum\limits_{B(x, y)} [F_{n+1}(x'', y'') - 2F_n(x, y) + F_{n-1}(x', y')]\right)^2 + delta}{\sum\limits_{B(x, y)} (F_{n+1}(x'', y'') - 2F_n(x, y) + F_{n-1}(x', y'))^2 + delta} \tag{7}$$

where $B(x, y)$ is the block that contains (x, y), and *delta* is a very small positive constant used to avoid division by zero. This criterion can be interpreted as follows. Let

$$Z_n(x, y) = F_{n+1}(x'', y'') - 2F_n(x, y) + F_{n-1}(x', y').$$

Then we can assume that $Z_n(x, y) \sim N(\mu(x, y), \sigma^2(x, y))$, which deduces $X = \dfrac{\sum\limits_{B(x, y)} \dfrac{Z_n(x, y) - \mu(x, y)}{\sigma(x, y)}}{\sqrt{Num}} \sim N(0, 1)$. On the other hand, $Y = \sum\limits_{B(x, y)} \left[\dfrac{Z_n(x, y) - \mu(x, y)}{\sigma(x, y)}\right]^2$ obeys the $\chi^2(Num)$ distribution. If we consider X and Y as independent, then $T = X/\sqrt{Y/Num}$ obeys the t distribution, whose probability density function curve is very similar to that of $N(0, 1)$. Therefore, the probability grows with the decrease of T's absolute value, which is exactly the square root of (7) when $\mu(x, y)$ equals 0 and $\sigma(x, y)$ equals 1.

Based on the above discussion, if we have several motion vector candidates for the current block, the one that minimizes (7) will be considered as best and extended to the other direction for further use. For interlaced sequences, the general three-frame case turns into the five-field case in Fig. 3, where four motion vector candidates for the current block can be calculated, including two between opposite-parity fields and two between same-parity fields. Note that the dotted blocks are co-located with the block in the current nth field.

The motion estimation process can avoid error propagation by referencing only original pixels. But in this case we cannot make full use of the previously de-interlaced frame. In the implementation, we make a tradeoff that the best motion vector for each block will be mapped to the previously interpolated frame F_{n-1}. Thus, the vertical component of the mapped motion vector can be either odd or even. In order to successfully extend it to the backward direction, we can temporarily de-interlace the backward field by an intra-field interpolation method (e.g., line averaging [7] or directional interpolations [33], [34]). In this way, we can not only benefit from the previous interpolation results when the motion vector is considered to be reliable but also have a great chance of avoiding error propagation. The latter can be interpreted as follows. First, we should assume that all the selected motion vectors before mapping to F_{n-1} are reliable, and that about 50% of the motion vectors have odd vertical components. Then the probability of making mistakes is no more than 50% in f_n. Now if we take F_{n-1} into account, we can find that the

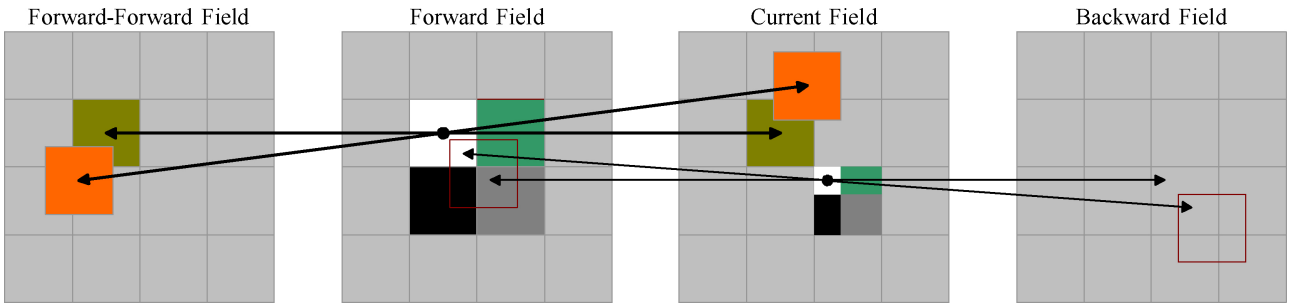


Fig. 5. Sketch map for motion state analysis.

interpolated pixels also have a chance of no more than 50% to be incorrect because half of the motion vectors calculated in f_{n-1} tend to be vertically even. So the error propagation probability in f_n is restricted to be no more than 25% when considering one more reference. By analogy we know that this phenomenon actually weakens with time. In Section II-D, we will also use some interpolating strategies to restrain error propagation.

For better performance, each to-be-interpolated pixel $(i,\ j)$ in the backward field is interpolated using a directional interpolation method modified from [33], as described by the following steps.

1) *Step 1:* Calculate five directional averages and five absolute directional differences as

$$\begin{cases} average(k) = \dfrac{f_0(i-1,\ j-2+k)+f_0(i+1,\ j+2-k)}{2} \\ difference(k) = |f_0(i-1,\ j-2+k) \\ \qquad\qquad - f_0(i+1,\ j+2-k)| \end{cases} \tag{8}$$

where k ranges from 0 to 4, representing the five directions in Fig. 4 from left to right.

2) *Step 2:* If the following equations hold, go to Step 3; otherwise $average(2)$ is used as the final result

$$\begin{cases} |average(2) - average(1)| \le 25 \\ |average(2) - average(3)| \le 25 \\ MAX(|average(2) - average(0)|, \\ \qquad |average(2) - average(4)|) > 25. \end{cases} \tag{9}$$

3) *Step 3:* Remove unnecessary directions by determining the starting direction *start* and the ending direction *end*:

```
start = end = 2;
if, (|difference (1) − difference (3) |≥ 40)
{
    start = 1;
    end = 3;
}
if (|difference (0) − difference (3) | ≥ 40
  && |difference (0) − difference (2) | ≥ 40)
    start = 0;
if (|difference (4) − difference (1) | ≥ 40
  && |difference (4) − difference (2) | ≥ 40)
    end = 4;
```

4) *Step 4:* Find the direction with minimal absolute difference from $k = start$ to $k = end$, and use the corresponding average as the interpolation result.

B. Three-Field Motion State Analysis

In Section I, we have mentioned that motion state analysis is generally a procedure after motion estimation. But we would like to emphasize that although motion vectors are available at this stage, we should not think motion estimation has already been finished. This is because the results of first-level motion state analysis will be used in the next subsection to decide whether the second-level motion estimation and subsequent second-level motion state analysis should be performed, as depicted in Fig. 1.

For each block in the current field, its forward reference might overlie one, two, or four adjacent blocks, depending on the motion vector. Fig. 5 gives an example when a current block is split into four sub-blocks. These sub-blocks may require different deinterlacing methods since their references belong to different blocks in the forward field. If only one motion state is defined for each block as existing algorithms do, we will take the risk of discarding all the previously computed motion state information. Therefore, in this subsection we propose to use both the currently detected motion state and the forwardly detected motion states for deinterlacing. This strategy is actually a tradeoff between traditional block-based cases and the extreme pixel-based case. Here are some explanations. Take the white sub-block in the current field for example when one more forward reference is considered, i.e., the forward-forward field. Depending on the motion vector in the forward field, the reference of its reference sub-block might also overlie at most four blocks in the forward-forward field. This means that the current block can be divided into more than four parts for processing if one more reference field is added. By this token, we can approach the extreme case of pixel-based motion state analysis if there are enough references and enough motion states are defined.

For clarity, only two motion states are defined in this subsection, i.e., *stationary* (0) and *moving* (1). For each pixel in the current field, we use *fullState* to denote the motion state detected in the current field, and *subState* to denote the motion state of its reference pixel detected in the forward field

$$\begin{cases} subState_n(x,\ y) = fullState_{n-1}(x',\ y') \\ fullState_n(x,\ y) = sgn\{1 - P_n(x,\ y) + P_n(x,\ y) \\ \qquad\qquad \cdot [1 - Q_n(x,\ y)] \cdot |R_n(x,\ y)|)\} \end{cases} \tag{10}$$

where the *sgn* function is defined as

$$sgn(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0. \end{cases} \tag{11}$$

$P_n(x, y)$, $Q_n(x, y)$ and $R_n(x, y)$ are formulated as

$$\begin{cases} P_n(x, y) = sgn[1 + sgn(SAD_{mot}(x, y) - SAD_{col}(x, y))] \\ Q_n(x, y) = sgn[1 + sgn(2 \times STD(x, y) - SAD_{col}(x, y))] \\ R_n(x, y) = sgn(x - x') \cdot sgn(y - y') \end{cases} \tag{12}$$

where $SAD_{mot}(x, y)$ and $SAD_{col}(x, y)$ are the average sums of absolute difference along motion trajectory and among co-located pixels, respectively; and $STD(x, y)$ is the standard deviation of the current block, as defined below. It is obvious that $\{P_n(x, y), Q_n(x, y), |R_n(x, y)|\} \subseteq \{0, 1\}$, hence (10) will not generate values other than 0 or 1

$$\begin{cases} SAD_{mot}(x, y) = \dfrac{1}{2} \displaystyle\sum_{IB(x, y)} [\, |f_n(x, y) - f_{n-1}(x', y')| \\ \qquad\qquad + |f_n(x, y) - f_{n+1}(x'', y'')| \,] \\ SAD_{col}(x, y) = \dfrac{1}{2} \displaystyle\sum_{IB(x, y)} [\, |f_n(x, y) - f_{n-1}(x, y)| \\ \qquad\qquad + |f_n(x, y) - f_{n+1}(x, y)| \,] \\ STD^2(x, y) = N \displaystyle\sum_{IB(x, y)} f_n^2(x, y) - \left[\sum_{IB(x, y)} f_n(x, y) \right]^2. \end{cases} \tag{13}$$

Note that the first field should be specially treated since $subState_0(x, y)$ is meaningless. The modified directional interpolation method described in Section II-A can be applied. For even better performance, we can assume a virtual previous field that equals the first field, and then perform intra-field interpolations on the first field to make it different. Thus, regular motion estimation and motion state analysis can be applied.

In the above computation, exactly three successive fields are used for the calculation of *fullState* due to the following facts. First, both fields of one interlaced frame are needed to theoretically ensure the quality while deinterlacing each field of the frame, as analyzed in [35]. So if the to-be-interpolated is a bottom field, the forward field is indispensable; otherwise, the backward field is required. Second, the experiments in [36] show that if only these two fields are used, the performance is often worse than the simplest one-field case. To do better, the number of fields should not be less than three; intuitively three successive fields can be used. Third, it can be seen from [37] that motion state analysis with three fields is generally better than that with two-fields, but it is usually unhelpful to use more than three fields.

C. Motion Consistency Analysis

At this point, the specific motion estimation and motion state analysis techniques have already been introduced for fixed-size block levels. The remaining problem before interpolation is how to dynamically switch between these two fixed-size block levels, (see Fig. 1). In this subsection, this problem is tackled by performing the motion consistency analysis, which includes the analysis on motion vector consistency and motion state consistency.

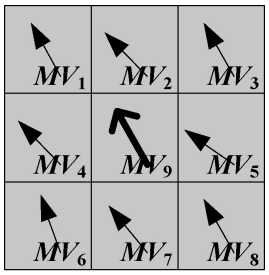


Fig. 6. 3×3 neighborhood for analysis on motion vector consistency.

Motion vector consistency is analyzed within the current field. For lower computation complexity, a 3×3 neighborhood is considered for each block, as shown in Fig. 6, where the bold arrow MV_9 refers to the motion vector of the current block. Intuitively, a consistent motion vector should act similar to its neighbors, in terms of both the motion vector value and the motion trajectory similarity. Therefore, MV_9 is considered to be consistent if the following equations are satisfied:

$$\begin{cases} \displaystyle\sum_{i=1}^{8} \left[sgn(|MV_i - MV_9|) \right] \le 2 \\ \displaystyle\sum_{i=1}^{9} sgn\left[1 + sgn\left(\frac{STD_i + delta}{SAD_i + delta} - 2 \right) \right] \ge 6 \end{cases} \tag{14}$$

where *delta* is a very small positive constant and the thresholds are experimentally set. The first equation means that the current motion vector should be equal to most of its neighbors, whereas the second equation is used to ensure whether most of these motion vectors are reliable. Here STD_i is the standard deviation defined as

$$STD_i^2 = N_i \sum_{IB(x_i, y_i)} f_n^2(x_i, y_i) - \left[\sum_{IB(x_i, y_i)} f_n(x_i, y_i) \right]^2 \tag{15}$$

where N_i and (x_i, y_i) are the number of pixels and an arbitrary pixel in the ith interlaced block, respectively. SAD_i is the average sum of absolute difference along motion trajectory

$$SAD_i = \frac{1}{2} \sum_{IB(x_i, y_i)} \left[|f_n(x_i, y_i) - f_{n-1}((x_i, y_i) + MV_i)| + |f_n(x_i, y_i) - f_{n+1}((x_i, y_i) - MV_i)| \right]. \tag{16}$$

As can be seen above, motion vector consistency is measured within the current field. On the contrary, the analysis on motion state consistency considers motion states in both the current field and the forward field, i.e., *subState* from the forward field and *fullState* in the current field. For a given pixel (x, y), its motion state is considered to be consistent if the following equations hold:

$$\begin{cases} fullState_n(x, y) = fullState_{n-1}(x', y') \\ subState_n(x, y) = subState_{n-1}(x', y') \end{cases} \tag{17}$$

where (x', y') is the forward reference pixel. These two equations can be easily understood and intuitively interpreted as: since each pixel has only two kinds of motion state, the motion state consistency should be acknowledged if these two kinds of motion state both remain the same along motion trajectory. Substituting the first equation of (10) into (17), the decision-making strategy can be equivalently rewritten as

$$\begin{cases} subState_n(x,\ y) = fullState_n(x,\ y) \\ subState_{n-1}(x',\ y') = fullState_{n-1}(x',\ y'). \end{cases} \tag{18}$$

These two equations are more compatible with the proposed idea. We can see clearly that for each pixel (x, y), the two kinds of motion state should be the same. Moreover, since the right-hand side of the first equation equals the left-hand side of the second equation according to (10), we actually mean that both of the two kinds of motion states should be kept the same along motion trajectory in at least three successive fields.

Finally, the motion consistency is defined as

$$MConsist_n(x, y) = MVConsist_n(x, y) \cdot MSConsist_n(x,\ y) \tag{19}$$

where $MVConsist_n(x, y)$ and $MSConsist_n(x, y)$ indicate the motion vector consistency and motion state consistency of pixel (x, y) in the current field n, respectively. Note that $MVConsist_n(x, y) \in \{0, 1\}$ and $MSConsist_n(x, y) \in \{0, 1\}$.

D. Finer-Grained Interpolation

From Fig. 1 we can see that all the first-level blocks are divided into four second-level blocks before interpolation. However, for a second-level block at this stage, the results of motion estimation and motion state analysis can be from either block levels. For example, if the first-level motion is considered as consistent, we will not do the second-level motion estimation and second-level motion state analysis, therefore we simply copy these information for the corresponding sub-blocks during quadtree decomposition; if the first-level motion is considered as inconsistent, the motion vector and motion state are calculated at the second-level. To differentiate the two fixed-size block levels, an additional superscript is used in the following interpolation.

The finer-grained interpolation strategy is defined as

$$\begin{aligned} I_n(x,\ y) = {} & I_n{}^{(1)}(x,\ y) \cdot MConsist_n{}^{(1)}(x,\ y) \\ & + I_n{}^{(2)}(x,\ y) \cdot \left[1 - MConsist_n{}^{(1)}(x,\ y)\right] \end{aligned} \tag{20}$$

where $MConsist_n{}^{(l)}(x, y)$ and $I_n{}^{(l)}(x, y)\,(l = 1, 2)$ are the motion consistency and interpolation algorithm for the current field n at the l-th fixed size block level, respectively. Note that the superscript l for $MConsist_n^{(l)}(x, y)$ is fixed to be one, as depicted in Fig. 1. As for the motion-compensated interpolation algorithm $I_n^{(l)}(x, y)$, several motion-compensated median filtering (MCMF) methods are selected

$$\begin{aligned} I_n{}^{(l)}(x,\ y) = {} & fullState_n{}^{(l)}(x,\ y) \cdot nonStationaryMF_n{}^{(l)}(x,\ y) \\ & + \left[1 - fullState_n{}^{(l)}(x,\ y)\right] \cdot stationaryMF_n{}^{(l)}(x,\ y) \end{aligned} \tag{21}$$

where $fullState_n{}^{(l)}(x, y) \in \{0, 1\}$ is the motion state calculated in current field n at the lth fixed size block level, as shown in Section II-B. $stationaryMF_n{}^{(l)}(x, y)$ and $nonStationaryMF_n{}^{(l)}(x, y)$ are the median filters for the stationary state and the moving state, respectively, as defined in (22), where F_{n-1} and F_{n+1} are the previously interpolated frame and the temporarily de-interlaced backward frame, respectively. The functions will be explained in (23) and (24)

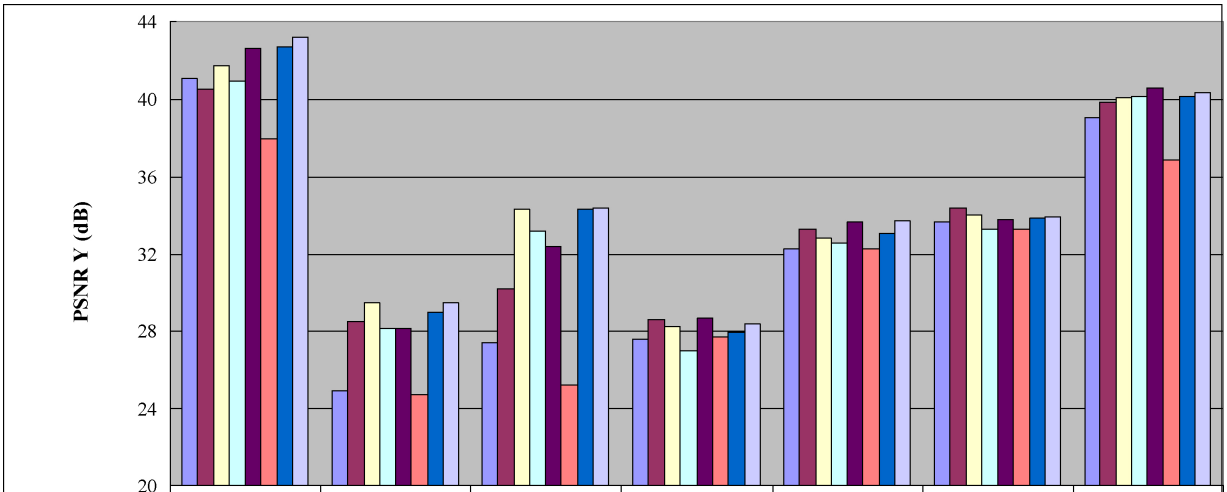
$$\begin{cases} stationaryMF_n{}^{(l)}(x,\ y) = Medians\left[F_{n-1}{}^{(l)}(x',\ y'),\right. \\ \quad F_{n+1}{}^{(l)}(x'',\ y''),\ f_n(x,\ y-1), \\ \quad \left. f_n(x,\ y+1),\ \dfrac{f_n(x,\ y-1) + f_n(x,\ y+1)}{2}\right]. \\ nonStationaryMF_n{}^{(l)}(x,\ y) \\ \quad = smooth_n{}^{(l)}(x,\ y) \cdot smoothMF_n{}^{(l)}(x,\ y) \\ \quad + \left[1 - smooth_n{}^{(l)}(x,\ y)\right] \cdot nonSmoothMF_n{}^{(l)}(x,\ y) \end{cases} \tag{22}$$

Some explanations are necessary here before further discussions. First, we would like to point out that motion acceleration and scene changes will have hardly any influence on the proposed motion estimation. This is because we have selected only one motion vector for each block out of many candidates (forward/backward, same-parity/opposite-parity) during the motion estimation procedure. For example, if there is a scene change at the current field, then a backward motion vector will be selected as the most reliable one. Second, although the unique motion vector is extended to another direction in Section II-A, the specific interpolation algorithms used in this subsection are relatively safe. For example, the definition of $stationaryMF_n{}^{(l)}(x, y)$ shows that "intra"-pixels have larger weights than "inter"-pixels. Even if one of the references is bad, the final result will generally not be influenced if the "intra"-pixels are similar.

In (22), $smooth_n{}^{(l)}(x, y)$ is used to determine whether a pixel (x, y) in the current field n is in a smooth region when considered in an lth level fixed-size block

$$smooth_n{}^{(l)}(x,\ y) = 1 - sgn\left(1 + sgn\left(\frac{STD_n{}^{(l)}(x,\ y) + delta}{SAD_n{}^{(l)}(x,\ y) + delta} - 2\right)\right) \tag{23}$$

$$\begin{aligned} & smoothMF_n{}^{(1)}(x,\ y) \\ & = Median\left(\frac{F_{n-1}{}^{(1)}(x',\ y') + F_{n+1}{}^{(1)}(x'',\ y'')}{2},\ edge_n(x,\ y),\right. \\ & \quad \left. f_n(x,\ y-1),\ f_n(x,\ y+1),\ \frac{f_n(x,\ y-1) + f_n(x,\ y+1)}{2}\right) \\ & smoothMF_n{}^{(2)}(x,\ y) = Median \\ & \left(F_{n-1}{}^{(2)}(x',\ y'),\ F_{n+1}{}^{(2)}(x'',\ y''),\ f_n(x,\ y-1),\ f_n(x,\ y+1),\right. \\ & \quad \left. \frac{f_n(x,\ y-1) + f_n(x,\ y+1)}{2}\right) \\ & nonSmoothMF_n{}^{(1)}(x,\ y) = Median \\ & \left(F_{n-1}{}^{(1)}(x',\ y'),\ F_{n+1}{}^{(1)}(x'',\ y''),\right. \\ & \quad \frac{F_{n-1}{}^{(1)}(x',\ y') + F_{n+1}{}^{(1)}(x'',\ y'')}{2}, \\ & \quad \left. f_n(x, y-1),\ f_n(x,\ y+1),\ \frac{f_n(x,\ y-1) + f_n(x,\ y+1)}{2}\right) \\ & nonSmoothMF_n{}^{(2)}(x,\ y) = Median \\ & \left(F_{n-1}{}^{(2)}(x',\ y'),\ F_{n+1}{}^{(2)}(x'',\ y''),\ f_n(x,\ y-1),\ f_n(x,\ y+1),\right. \\ & \quad \left. \frac{F_{n-1}{}^{(2)}(x',\ y') + F_{n+1}{}^{(2)}(x'',\ y'')}{2}\right) \end{aligned} \tag{24}$$



	News_CIF	Bus_CIF	Football_CIF	Mobile_CIF	Tempete_CIF	City_720P	Crew_720P
fiveTapMF	41.09	24.94	27.42	27.57	32.24	33.61	39.05
sixTapMF	40.52	28.53	30.24	28.63	33.30	34.38	39.83
Combination	41.73	29.50	34.32	28.22	32.86	34.02	40.09
AR [22,26]	40.97	28.18	33.24	27.00	32.53	33.27	40.16
RD [23]	42.59	28.18	32.40	28.65	33.63	33.78	40.58
Mohammadi [38]	37.93	24.73	25.25	27.72	32.22	33.26	36.84
Our_FDP [34]	42.66	28.98	34.32	27.94	33.04	33.85	40.12
Proposed_DI	43.17	29.45	34.35	28.40	33.75	33.96	40.38

Fig. 7. PSNR comparison between original progressive sequences and de-interlaced ones.

with the definitions of $STD_n^{(l)}(x, y)$ and $SAD_n^{(l)}(x, y)$ similar to (15) and (16), respectively. The remaining median filters for $nonStationaryMF_n^{(l)}(x, y)(l = 1, 2)$ are defined in equations (24), where $edge_n(x, y)$ is a simple edge-based line averaging (ELA) function

$$edge_n(x, y) = \frac{f_n(x-k, y-1) + f_n(x+k, y+1)}{2},$$
$$subject\ to\ k = arg \underset{t\in[0,2]}{MIN} |f_n(x-t, y-1) - f_n(x+t, y+1)|. \tag{25}$$

All the four median filters in equations (24) are defined for the moving state. For smooth regions, we intuitively assign larger weights to "intra"-pixels and smaller weights to "inter"-pixels so that cases such as motion acceleration and scene changes are less suffering, as explained before. However, for regions that are not smooth, "inter"-pixels along motion trajectory should have relatively larger weights because the intra neighbors tend to be different from the to-be-interpolated pixel. Then the definitions for $smoothMF_n^{(2)}(x, y)$ and $nonSmoothMF_n^{(2)}(x, y)$ can be understood. Now the remaining untouched issue is why the interpolation approach differs when using motions at different fixed-size block levels.

As seen from (23), smooth regions are those who have relatively small standard deviations compared to SAD. However, given a true motion vector, the SAD might be too small to be used to determine the smoothness. To overcome this deficiency, we make some changes when motion consistency is acknowledged. Based on the above discussions, $smoothMF_n^{(1)}(x, y)$ and $nonSmoothMF_n^{(1)}(x, y)$ will be applied if and only if the first fixed-size block level is considered as consistent. That is to say, the definitions for $smoothMF_n^{(1)}(x, y)$ and $nonSmoothMF_n^{(1)}(x, y)$ should consider the case when the result of (23) is incorrect. As for $smoothMF_n^{(1)}(x, y)$, we give more weights to "intra"-pixels by adding a simple ELA method. Therefore, it might be better when a "smooth" region contains some high frequency information such as edges. As for $nonSmoothMF_n^{(1)}(x, y)$, we give more weights to "intra"-pixels by adding an intra parameter in the median filter. In this way, the spatial weights (in the vertical direction) equal to the temporal weights (along motion trajectory). Note that there are two results when the number of parameters for median filtering is even. We simply take the average.

III. Experimental Results

In the following subsections, the proposed algorithm is evaluated both objectively and subjectively using various sequences. (The first-level and second-level block sizes are 16×16 and 8×8, respectively.) The computation complexity is also discussed. Note that the proposed motion estimation is implemented based on 3-DRS.

A. Objective Performance

Fig. 7 compares the objective performance of several deinterlacing algorithms on seven sequences, including five common intermediate format (CIF) sequences (*News*, *Bus*,

TABLE I

PSNR COMPARISON FOR EVALUATING THE PROPOSED MOTION ESTIMATION

Sequence	*Foreman*	*News*	*Bus*	*Football*	*Mobile*	*Paris*	*Tempete*	*Average*
Forward motion estimation	36.57	42.02	26.92	32.16	26.94	36.44	32.91	33.42
Five-field ME + Motion Vector Reliability [14]	36.46	42.57	27.62	32.72	27.79	36.99	33.54	33.96
Five-field ME + old Gaussian [23]	37.17	42.59	28.18	32.4	28.65	37.3	33.63	34.27
Proposed first-level ME	37.27	42.59	28.4	32.48	28.92	37.43	33.66	34.39

TABLE II

PERFORMANCE OF THE PROPOSED MOTION STATE ANALYSIS AND MOTION CONSISTENCY ANALYSIS

Sequence	*Foreman*	*News*	*Bus*	*Football*	*Mobile*	*Paris*	*Tempete*	*Average*
PSNR gain	0.14 dB	0.55 dB	0.52 dB	1.55 dB	0.05 dB	0.3 dB	0.01 dB	0.45 dB

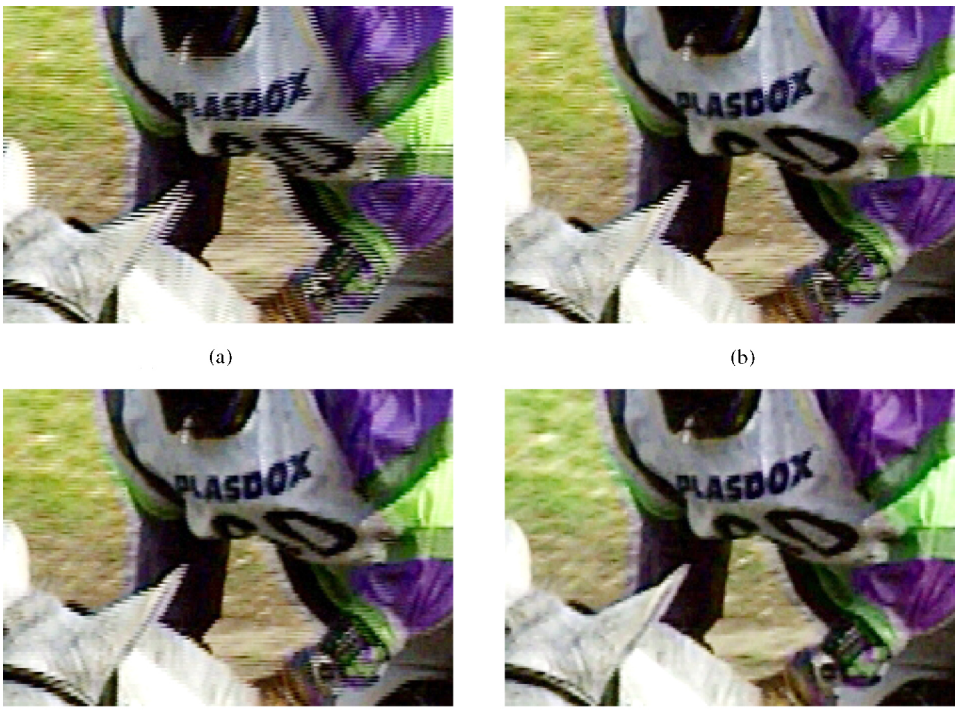


Fig. 8. Subjective performance on the *Horseriding* sequence with SD resolution. (a) Original sequence. (b) *fiveTapMF*. (c) *sixTapMF*. (d) Proposed.

TABLE III

COMPUTATION COMPLEXITY COMPARISON IN SECOND

Sequence	*Foreman*	*News*	*Bus*	*Football*	*Mobile*	*Paris*	*Tempete*	*Average*
Time of *fiveTapMF*	17.219	17.063	17.297	17.031	17.547	17.422	17.422	17.286
Time of *sixTapMF*	17.343	17.125	17.282	17.031	17.719	17.562	17.500	17.366
First-level interpolation	19.235	19.094	19.235	18.922	19.641	19.391	19.672	19.313
Second-level interpolation	24.140	24.797	23.063	23.438	24.360	24.234	24.156	24.027
Proposed interpolation	25.373	27.676	22.899	19.414	27.726	28.315	27.916	25.617
Time of AR [22], [26]	21.828	21.375	21.500	21.375	22.297	21.828	22.110	21.759
Time of [23]	22.781	21.985	21.906	21.609	22.750	22.140	22.219	22.199

Fig. 9. Subjective performance on the *MobileCalendar* sequence with SD resolution. (a) Original sequence. (b) *fiveTapMF*. (c) *sixTapMF*. (d) Proposed.

Football, *Mobile*, and *Tempete*, 150 frames) and two 720P sequences (*City* and *Crew*, 50 frames). We first perform interlacing on the progressive sequences and then de-interlace the resultant interlaced ones. Due to the lack of better substitutes, peak signal-to-noise ratio (PSNR) comparisons are made between original progressive sequences and the corresponding de-interlaced ones. The first method, *fiveTapMF*, is the objectively best five-tap median filter proposed in Section II-D, whereas *sixTapMF* is the six-tap median filter defined in (24). The method *Combination* is a simple combination of *fiveTapMF* and *sixTapMF*, without hierarchical motion analysis. The last two rows are based on our scheme. Note that the method *Our_FDP* differs from the proposed method in that it uses the four directional patterns [34] for backward interpolation.

From the first three rows in the data table, we can see that the combination of five-tap filters and the six-tap filter is objectively much more reliable than each of the components. Since the same motion estimation, motion state analysis, and motion consistency analysis are used, we can conclude that the performance of the proposed finer-grained interpolation is good. Note that this combination also outperforms adaptive recursive (AR) [22], [26] and Mohammadi [38], and is comparable to robust deinterlacing [23] which is much more complex. When the proposed hierarchical motion analysis and the proposed finer-grained interpolation are combined, we get the *Proposed_DI* method and it achieves more reliable results, especially on sequences *News*, *Tempete*, and *Crew*. Therefore, both the hierarchical motion analysis (including motion estimation, motion state analysis, and motion consistency analysis) and the finer-grained interpolation are objectively effective. The last two rows indicate that the proposed backward interpolation method also outperforms the one using four directional patterns.

Table I measures the effect of the motion estimation stage in terms of PSNR. As can be seen from (4), two fields are not enough to remove the factor e, which leads to the fact that two-field ME being worse than five-field ME. Among the five-field cases, the proposed algorithm makes full use of previously interpolated pixels both forwardly and backwardly, as explained in Section II-A, thus achieves the best performance. Note that although the PSNR gain between the last two rows is relatively small, we have successfully made the deinterlacing results more robust (e.g., on sequences *Bus* and *Mobile*) and set up a better base for the subsequent improvement. Since the second-level motion estimation has not been carried out at this stage to make the comparison fair, the actual performance of the proposed motion estimation should be even better than listed.

As depicted in Fig. 1, when the second-level motion state analysis begins, the analysis on motion consistency has already come to an end. Therefore, it is difficult to tell the accurate performance of motion state analysis from motion consistency analysis. Fortunately, from another point of view, the performance of motion consistency analysis actually reflects the efficiency of the full version of motion state analysis. Table II shows the overall PSNR gain of motion state analysis and motion consistency analysis. We are happy to see that

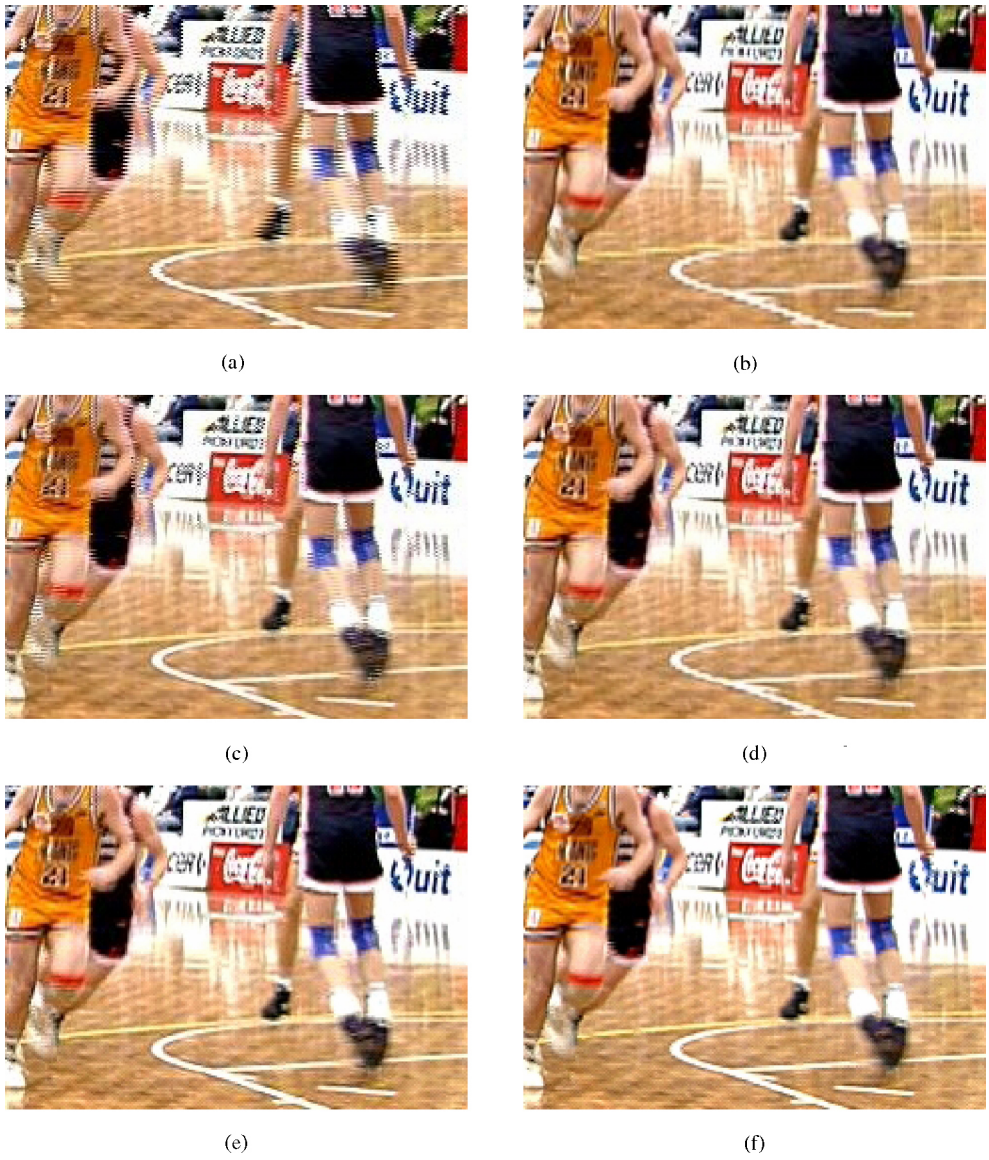


Fig. 10. Subjective performance on the *Basketball* sequence with SD resolution. (a) Original sequence. (b) Line averaging. (c) Edge-based median filtering [14]. (d) Adaptive recursive [22], [26]. (e) Method of [23]. (f) Proposed.

significant gain has been achieved on sequence *Football* which has tough motions for deinterlacing.

B. Subjective Performance

The proposed deinterlacing scheme is in fact a combination of median filters. In this subsection, we first give a subjective evaluation of the finer-grained interpolation by comparing the proposed scheme to its individual components in Figs. 8 and 9, and then make an overall evaluation against other algorithms.

Fig. 8 compares the subjective performance on SD sequence *Horseriding*. The "Original Sequence" is generated by simply merge two interlaced fields together hence shows the worst subjective performance. *fiveTapMF* and *sixTapMF* are the same as those in the previous subsection. Although many artifacts are removed, the alphanumeric characters and the horse ears in Fig. 8(b) are still not satisfactory. In Fig. 8(c), things turn better but it seems that some noise is introduced around the English letters on the clothes. Fig. 8(d) is actually generated by the combination of *fiveTapMF* and *sixTapMF*. We are happy to see that although only simple median filtering methods are used, the picture quality degradation from the most efficient high-quality deinterlacing algorithms is negligible (especially when displayed consecutively).

As can be seen from (24), the adopted median filters consider mainly two directions: motion trajectory direction and vertical direction. Fig. 8 has shown a scene when the motion is not vertical. A more extreme case is considered in Fig. 9, which is one part of the SD sequence *MobileCalendar*. In the original interlaced sequence shown in Fig. 9(a), serrations can be observed. But the numbers are generally correct in shape. Fig. 9(b) and (c) removes many serrations, but severe shape distortion is also obtained due to three reasons. First, the dominant motion direction is vertical, thus the up and down pixels are approximately on the motion trajectory. Both *fiveTapMF* and *sixTapMF* degrade into variations of line averaging, without considering the edge information. Second,

the usage of up and down pixels is not very reliable as we do not know the vertical speed. Therefore, both *fiveTapMF* and *sixTapMF* are fallible. Third, the scene itself has very strong foreground-background contrast, so even small distortions are very obvious. In Fig. 9(d), number "9" is de-interlaced much better as we take edge information into account. As for number "2," the background artifacts are gone due to region-based decision and finer-grained interpolation.

Fig. 10 illustrates the efficiency on the SD sequence *Basketball*. Fig. 10(b) indicates that line averaging removes most of the interlacing artifacts, but tends to fail on non-horizontal edges even if the moving direction is not vertical. Edge-based median filtering [14] can be used to restrain this problem. But as shown in Fig. 10(c), the overall result is generally not smooth. The result of AR [22], [26] is much smoother; however, AR is often inferior on diagonal edges, as can be seen from Fig. 10(d). In Fig. 10(e), the advantages of adaptive recursive and edge-based median filtering are combined thus the visual quality is rather good. The proposed deinterlacing scheme only utilizes simple median filters; nevertheless, the subjective effect in Fig. 10(f) is quite comparable to that in Fig. 10(e).

From Figs. 8–10 we can see that advisable decisions can be made even when all the candidate interpolation methods are not very reliable. This can to some extent reflect the efficiency of the proposed hierarchical motion analysis.

C. Computation Complexity

The computation complexity is measured in seconds in Table III, where each of the seven CIF sequences has 150 frames. In the first five rows, the complexity of the proposed deinterlacing scheme is evaluated, whereas in the last two rows, two motion-compensated deinterlacing methods are also measured for reference. In the experiment, we choose the first-level block size as 16×16 and the second-level block size as 8×8. Accordingly, the fixed-size methods are also implemented with block size 16×16. Note that for a given interpolation method, we may have quite different running time when implementing with different software and hardware conditions, as well as tricks. In this subsection, an IBM T43 AZ9 laptop (Intel Pentium-M 1.73 GHz, 1 GB) is used to run the test.

For the first two rows, the running time is relatively short since finer-grained interpolation is not needed. The next two rows indicate that block size matters. For a larger block size, much less motion vector candidates will be computed and analyzed, therefore much time can be saved. The fifth row illustrates that the proposed scheme usually takes longer time than the cases when block sizes are fixed. This is because when the second-level information is used, the corresponding first-level information has already been calculated and analyzed. Statistics show that 19.72%, 39.52%, 8.66%, 2.83%, 25.78%, 37.29%, and 34.23% of the second-level motion information is used for sequences *Foreman*, *News*, *Bus*, *Football*, *Mobile*, *Paris*, and *Tempete*, respectively. Therefore, the running time on the fifth row is especially small for sequences *Bus* and *Football*, and relatively large for *News*, *Paris*, and *Tempete*. On average the running time of our proposed scheme is about 6.6% more than that of the second-level case, and about 32.64% more than that of the first-level case.

IV. Concluding Remarks

In this paper, we have proposed a motion-compensated deinterlacing scheme using hierarchical motion analysis. First, a hierarchical motion analysis model is presented based on a novel block strategy, which differs from conventional methods in that we have a variable-size block level based on motion vectors and can even approach the extreme of pixel-based interpolation when sufficient reference fields and motion states are available. The proposed hierarchical motion analysis mainly comprises three stages. The first stage is five-field motion estimation, where a Gaussian noise model is introduced for choosing the unique motion vector for each fixed-size block, either between same-parity fields or between opposite-parity fields. The second stage is three-field motion state analysis, where two kinds of motion states are defined for each pixel. The last stage is motion consistency analysis that takes both motion vector consistency and motion state consistency into account. Then several recursive median filters are used for finer-grained interpolation based on the motion consistency and texture information. Experimental results show that the objective performance is very good in terms of PSNR, whereas the subjective loss from existing best methods is negligible. Moreover, the increased computation complexity is marginal.

Acknowledgment

The authors would like to thank Dr. Q. Wang and the anonymous reviewers for valuable suggestions and L. Fu for helpful discussions on directional interpolation.

References

[1] Y. Wang, J. Ostermann, and Y. Zhang, "Video formation, perception, and representation," *Video Processing and Communications*, Upper Saddle River, NJ: Prentice Hall, 2002, pp. 12–14.

[2] R. Steinmetz and K. Nahrstedt, "Video and animation," *Multimedia: Computing, Communications and Applications*. Upper Saddle River, NJ: Prentice Hall, 1995, p. 85.

[3] R. Castagno, P. Haavisto, and G. Ramponi, "A method for motion adaptive frame rate up-conversion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 5, pp. 436–446, Oct. 1996.

[4] S. Muramatsu, T. Ishida, and H. Kikuchi, "Invertible deinterlacing with sampling-density preservation: Theory and design," *IEEE Trans. Signal Process.*, vol. 51, no. 9, pp. 2343–2356, Sep. 2003.

[5] M. Biswas, S. Kumar, and T. Q. Nguyen, "Performance analysis of motion-compensated de-interlacing systems," *IEEE Trans. Image Process.*, vol. 15, no. 9, pp. 2596–2609, Sep. 2006.

[6] C. Ballester, M. Bertalmío, V. Caselles, L. Garrido, A. Marques, and F. Ranchin, "An inpainting-based deinterlacing method," *IEEE Trans. Image Process.*, vol. 16, no. 10, pp. 2476–2491, Oct. 2007.

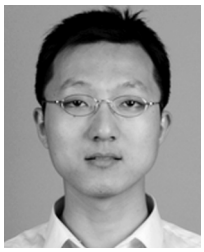
[7] G. de Haan and E. B. Bellers, "Deinterlacing—An overview," *Proc. IEEE*, vol. 86, no. 9, pp. 1839–1857, Sep. 1998.

[8] S. Yang, Y. Y. Jung, Y. H. Lee, and R. H. Park, "Motion compensation assisted motion adaptive interlaced-to-progressive conversion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 9, pp. 1138–1148, Sep. 2004.

[9] G. Jeon and J. Jeong, "Designing Takagi-Sugeno fuzzy model-based motion adaptive deinterlacing system," *IEEE Trans. Consum. Electron.*, vol. 52, no. 3, pp. 1013–1020, Aug. 2006.

[10] M. Li and T. Nguyen, "A de-interlacing algorithm using Markov random field model," *IEEE Trans. Image Process.*, vol. 16, no. 11, pp. 2633–2648, Nov. 2007.

[11] J. Deame, "Motion compensated de-interlacing: The key to the digital video transition," in *Proc. Soc. Motion Picture Television Engineers Tech. Conf.*, New York, 1999.
[12] G. de Haan and P. W. A. C. Biezen, "Sub-pixel motion estimation with 3-D recursive search block-matching," *Signal Process. Image Commun.*, vol. 6, no. 3, pp. 229–239, Jun. 1994.
[13] B. D. Choi, J. W. Han, C. S. Kim, and S.-J. Ko, "Motion-compensated frame interpolation using bilateral motion estimation and adaptive overlapped block motion compensation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 4, pp. 407–416, Apr. 2007.
[14] Q. Huang, W. Gao, D. Zhao, and Q. M. Huang, "An edge-based median filtering algorithm with consideration of motion vector reliability for adaptive video deinterlacing," in *Proc. IEEE Int. Conf. Multimedia Expo*, Toronto, Canada, Jul. 2006, pp. 837–840.
[15] R. A. Braspenning and G. de Haan, "Efficient motion estimation with content-adaptive resolution," in *Proc. Int. Symp. Consum. Electron.*, Ilmenau, Germany, Sep. 2002, pp. E29–E34.
[16] S. Li, J. Du, D. Zhao, Q. Huang, and W. Gao, "An improved 3-DRS algorithm for video de-interlacing," in *Proc. Picture Coding Symp.*, Beijing, China, Apr. 2006.
[17] I. Kim, T. Jeong, and C. Lee, "Deinterlacing based on motion compensation with variable block sizes," in *Proc. SPIE*, vol. 6312. Aug. 2006, pp. B1–B8.
[18] Y. Y. Jung, S. Yang, and P. Yu, "An effective de-interlacing technique using two types of motion information," *IEEE Trans. Consum. Electron.*, vol. 49, no. 3, pp. 493–498, Aug. 2003.
[19] S. G. Lee and D. H. Lee, "A motion-adaptive de-interlacing method using an efficient spatial and temporal interpolation," *IEEE Trans. Consum. Electron.*, vol. 49, no. 4, pp. 1266–1271, Nov. 2003.
[20] S. F. Lin, Y. L. Chang, and L. G. Chen, "Motion adaptive interpolation with horizontal motion detection for deinterlacing," *IEEE Trans. Consum. Electron.*, vol. 49, no. 4, pp. 1256–1265, Nov. 2003.
[21] C. C. Lin, M. H. Sheu, H. K. Chiang, and C. Liaw, "Motion adaptive de-interlacing with horizontal and vertical motions detection," in *Proc. Pacific-Rim Conf. Multimedia*, LNCS 3767. 2005, pp. 291–302.
[22] Y. L. Chang, S. F. Lin, C. Y. Chen, and L. G. Chen, "Video de-interlacing by adaptive 4-field global/local motion compensated approach," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 12, pp. 1569–1582, Dec. 2005.
[23] Q. Huang, W. Gao, D. Zhao, and H. Sun, "An efficient and robust adaptive deinterlacing technique," *IEEE Trans. Consum. Electron.*, vol. 52, no. 3, pp. 888–895, Aug. 2006.
[24] G. G. Lee, D. W. C. Su, H. Y. Lin, and M. J. Wang, "Multiresolution-based texture adaptive motion detection for de-interlacing," in *Proc. IEEE Int. Symp. Circuits Syst.*, Kos, Greece, 2006, p. 4.
[25] F. M. Wang, D. Anastassiou, and A. N. Netravali, "Time-recursive deinterlacing for IDTV and pyramid coding," *Signal Process. Image Commun.*, vol. 2, no. 3, pp. 365–374, Oct. 1990.
[26] G. de Haan and E. B. Bellers, "Deinterlacing of video data," *IEEE Trans. Consum. Electron.*, vol. 43, no. 3, pp. 819–825, Aug. 1997.
[27] L. Vanderdorpe, L. Cuvelier, B. Maison, P. Queluz, and P. Delogne, "Motion-compensated conversion from interlaced to progressive formats," *Signal Process. Image Commun.*, vol. 6, no. 3, pp. 193–211, Jun. 1994.
[28] W. H. A. Bruls and C. Ciuhu, "Bridging the interlace and progressive controversy using a progressive enhancement stream on top of the interlace stream and a new de-interlace algorithm," in *Proc. IEEE Int. Conf. Consum. Electron.*, 2005, pp. 5–6.
[29] M. Sonka, V. Hlavac, and R. Boyle, "Image preprocessing," *Image Processing, Analysis, and Machine Vision*, 2nd ed. New York: Brooks/Cole, 1999, pp. 56–57.
[30] D. Wang, A. Vincent, and P. Blanchfield, "Hybrid de-interlacing algorithm based on motion vector reliability," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 8, pp. 1019–1025, Aug. 2005.
[31] Z. Yang, W. Gao, and Y. Liu, "Performance-complexity analysis of high resolution video encoder and its memory organization for DSP implementation," in *Proc. IEEE Int. Conf. Multimedia Expo*, Toronto, Canada, 2006, pp. 1261–1264.
[32] G. de Haan, P. W. Biezen, H. Huijgen, and O. A. Ojo, "True-motion estimation with 3-D recursive search block matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, no. 5, pp. 368–379, Oct. 1993.
[33] H. Y. Lee, J. W. Park, T. M. Bae, S. U. Choi, and Y. H. Ha, "Adaptive scan rate up-conversion system based on human visual characteristics," *IEEE Trans. Consum. Electron.*, vol. 46, no. 4, pp. 999–1006, Nov. 2000.
[34] H. Ku and T. Hou, "An efficient directional interpolated algorithm for video deinterlacing," *IEICE Electron. Express*, vol. 6, no. 5, pp. 211–217, Mar. 2009.
[35] P. Delogne, L. Cuvelier, B. Maison, B. Van Caillie, and L. Vandendorpe, "Improved interpolation, motion estimation and compensation for interlaced pictures," *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 482–491, Sep. 1994.
[36] T. Koivunen, "Motion detection of an interlaced video signal," *IEEE Trans. Consum. Electron.*, vol. 40, no. 3, pp. 753–760, Aug. 1994.
[37] B. Heng, "Application of deinterlacing for the improvement of surveillance video," M.S. thesis, Dept. Elect. Eng. Comput. Sci., Massachusetts Instit. Technol., Cambridge, MA, Jun. 2001.
[38] H. Mohammadi, P. Langlois, and Y. Savaria, "A five-field motion compensated deinterlacing method based on vertical motion," *IEEE Trans. Consum. Electron.*, vol. 53, no. 3, pp. 1117–1124, Aug. 2007.



Qian Huang received the B.S. degree in computer science from Nanjing University, Nanjing, China, in 2003. He is currently working toward the Ph.D. degree in computer science, specifically, multimedia technology from the Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences (CAS), Beijing, China.

From April 2008 to March 2009, he was a Research Intern with the Intel China Research Center, Beijing, China. His current research interests include video processing and video surveillance.

Debin Zhao received the B.S., M.S., and Ph.D. degrees in computer science from the Harbin Institute of Technology (HIT), Harbin, China, in 1985, 1988, and 1998, respectively.

Dr. Zhao was a Lecturer from 1989 to 1993 and an Associate Professor from 1993 to 2000 in the Department of Computer Science, HIT. He is currently a Professor with the Department of Computer Science, HIT, and also an Adjunct Professor with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. He has published over 200 technical articles in refereed journals and conference proceedings. His current research interests include image and video coding, video processing, video streaming and transmission, and pattern recognition.

Dr. Zhao was the recipient of three National Science and Technology Progress Awards of China (Second Prize).

Siwei Ma received the B.S. degree in computer science from Shandong Normal University, Jinan, China, in 1999, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2005.

From 2005 to 2007, he was a Post-doctorate with the University of Southern California, Los Angeles. Then he joined the Institute of Digital Media, School of Electronics Engineering and Computer Science, Peking University, Beijing, China, where he is currently an Associate Professor. He has published over 30 technical articles in refereed journals and proceedings. His current research interests include image and video coding, video processing, video streaming, and transmission.

Wen Gao (M'92–SM'05–F'09) received the B.S. and M.S. degrees in computer science from Harbin University of Science and Technology and Harbin Institute of Technology (HIT), China, in 1982 and 1985, respectively, and received the Ph.D. degree in computer science from HIT, China and in electronics engineering from the University of Tokyo, Tokyo, Japan, in 1988 and 1991, respectively.

Before joining Peking University, Beijing, China, he was a Full Professor of computer science with the Harbin Institute of Technology, Harbin, China from 1991 to 1995, and with the Chinese Academy of Sciences (CAS), Beijing, China, from 1996 to 2005. With CAS, he served as a Professor (1996–2005), the Managing Director of the Institute of Computing Technology

(1998–1999), the Executive Vice President of the Graduate School of CAS (2000–2004), and a Vice President (2000–2003) of the University of Science and Technology of China, Hefei, China. He is currently a Professor of computer science with Peking University. He has published extensively, including four books and over 500 technical articles in refereed journals and conference proceedings. His current research interests include image processing, video coding and communication, pattern recognition, multimedia information retrieval, multimodal interface, and bioinformatics.

Dr. Gao is the Editor-in-Chief of the *Journal of Computer* (a journal of the China Computer Federation), an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA, an Associate Editor of the IEEE TRANSACTIONS ON AUTONOMOUS MENTAL DEVELOPMENT, an Area Editor of the *EURASIP Journal of Image Communications*, and an Editor of the *Journal of Visual Communication and Image Representation*. He chaired a number of prestigious international conferences on multimedia and video signal processing, and also served on the advisory and technical committees of numerous professional organizations.

Huifang Sun (S'83–M'85–SM'93–F'01) received the B.S. degree from the Harbin Military Engineering Institute, Harbin, China, in 1967, and the Ph.D. degree in electrical engineering from the University of Ottawa, Ottawa, Canada, in 1986.

He was an Associate Professor with Fairleigh Dickinson University, Madison, NJ, from 1986 to 1990, before joining Sarnoff Research Laboratories, Princeton, NJ, in 1990, where he was the Technology Leader of digital video communication. In 1995, he joined Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, as a Senior Principal Technical Staff and was promoted to the Vice President and Research Fellow of MERL in 2003 and is currently a Research Fellow. He has coauthored two books and more than 150 journal and conference papers. He holds 59 U.S. patents. His current research interests include digital video/image compression and digital communication.

Dr. Sun was the recipient of the Technical Achievement Award in 1994 at Sarnoff Laboratories for Grand Alliance high definition TV development, the Best Paper Award of the IEEE TRANSACTIONS ON CONSUMER ELECTRONICS in 1992, the Best Paper Award of the 1996 International Conference on Consumer Electronics, and the Best Paper Award of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (IEEE TCSVT) in 2003. He is now an Associate Editor of IEEE TCSVT and was the Chair of the Visual Processing Technical Committee of the IEEE Circuits and System Society.