

A New Source-Side De-interlacing Method for High Quality Video Content

Yulai Bi^(✉), Xiaoyun Zhang, and Zhiyong Gao

Institute of Image Communication and Network Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
yulaibi9351@gmail.com, {xiaoyun.zhang,zhiyong.gao}@sjtu.edu.cn

Abstract. Traditionally, de-interlacing is implemented at the terminal-side as a post processing step, however this paper proposes a novel and advanced source-side de-interlacing method for high quality video content. It is an adaptive integration of texture-based intra-interpolation and inter-field motion compensation. Firstly, intra-interpolation based on texture similarity is proposed for wider search scope and better reconstruction performance. Motion compensation is strictly limited by vertical motion conditions and uniform motion check. Then, adaptive intra or inter mode decision is presented to further remove the compensated artifacts. Experiments show that the proposed technique can produce more convincing de-interlaced videos with less artifacts on both subjective and objective evaluation in comparison with other methods, including the leading multimedia tool ffmpeg. The parallelism nature of the proposed technique also makes it suitable for acceleration implementation on GPUs and thus is applicable for real time systems.

Keywords: De-interlacing · Texture-based · Motion compensation · Adaptive integration

1 Introduction

Interlacing was once widely used as the standard of TVbroadcast in analog television systems to meet the limit requests of transmission bandwidth and scanning frequency. De-interlacing is thus a necessary and important technology for displaying interlaced video on progressive scan displays without obvious visual artifacts, such as edge flickers and jagged effects.

In digital television system, the terminal-side de-interlacing has been widely used and implemented in the post-processing chips or set-top boxes. Due to the cost and hardware limitation of terminal devices, most video post-process chips in TVs make a tradeoff between computation complexity and video quality when developing de-interlacing algorithms.

With the development of broadcasting, Internet transmission and display technique, progressive video signal can be transmitted directly and displayed on all kinds of progressive displays. However, there are still a great amount of legacy

interlaced video, such as 1080/50i, which are very popular and valuable content for audience. Hence, it is essential and worthwhile to conduct a high-quality de-interlacing process at the source-side, and the de-interlaced progressive video is transmitted and displayed directly for consumers.

Generally, de-interlacing methods can be classified into three categories: intra-field de-interlacing, inter-field de-interlacing, and adaptive de-interlacing. They differ on how spatial and temporal information is exploited to calculate the missing pixels.

The intra-field de-interlacing methods only use the information within the current field to interpolate the missing pixels. A well-known intra-field de-interlacing method is the edge-based line-averaging (ELA) [1], which interpolates the missing pixels after a simple edge direction estimation. It generally provides satisfactory results at the regions where edge detection can be estimated accurately. Many optimized algorithms (eg. [2,3]) based on ELA have been proposed and widely used in early digital televisions because of the low computation. In addition, other intra-field de-interlacing methods based on edge pattern or texture similarity (eg. [4–6]) have also been proposed.

The inter-field de-interlacing methods, also called motion compensation methods (MC), reconstruct the image using temporal information in adjacent fields. In these methods (eg. [7–9]), a local motion vector is first estimated. And temporal interpolation is then performed. However, the motion compensation methods involves a heavy calculation burden, and the quality is not robust since it is difficult to keep the accuracy of motion estimation in rotation, zooming and fast motion scenarios.

The adaptive de-interlacing methods can be sub-classified into two categories: motion adaptive (MA) and motion compensated adaptive (MCA). MA methods exploit inter-field information in static areas and intra-interpolation in moving areas to reconstruct vertical resolution. And MCA (eg. [10–13]) further make use of inter-field information in both statics and some specific moving areas. The key point in adaptive methods is to obtain reliable motion vectors and make mode decision correctly.

In this paper, we propose a novel source-side de-interlacing framework based on software implementation and GPU acceleration for high quality video content application, such as IP-based video on demand. In the proposed method, intra-interpolation is based on local texture similarity, and motion compensation is strictly limited by vertical motion conditions and uniform motion check. More importantly, an adaptive integration based on the reliability of motion compensation is used to further reduce visual artifacts. Experimental results show both subjectively and objectively that the proposed technique can produce convincing de-interlaced videos in comparison with other state-of-art techniques.

This paper is organized as follows. The motivation for the proposed de-interlacing method is discussed in Sect. 2. The proposed algorithm is presented in Sect. 3. Section 4 shows the experimental results. And Sect. 5 gives the conclusion.

2 Motivation

2.1 ELA and Its Jagged Effects

In this section, we introduce the traditional intra-interpolation algorithm, ELA. The ELA-based algorithm is widely used because of its simple calculation. It interpolates the missing pixels using edge directional correlations between two neighboring lines $j - 1$ and $j + 1$. The ELA $5+5$ algorithm using 5 directions is shown in Fig. 1. Pixel $X(i, j)$ represents the missing pixel and is interpolated along the best direction with the smallest difference.

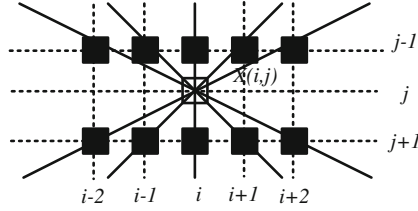


Fig. 1. ELA structure

The ELA method provides a good performance in regions where the edge direction can be detected accurately. However, interpolation errors usually exist since it only focuses on recovering a single pixel that is the most similar to its neighbor pixels without considering the texture structure in its region. And wider search such as 9 directions are generally used for finer edge interpolation in ELA, but it also tend to bring more incorrect results with visual artifacts.

Figure 2 shows a graduated black edge Fig. 10(a), and its interlaced field Fig. 10(b). The white lines represent the missing lines of the current field. Pixel represents the pixel to be interpolated, and a black pixel for is expected according to the ground truth Fig. 10(a). But the ELA algorithm selects a wrong direction along the red line and interpolates a light gray pixel for in Fig. 10(c), which we call jagged effects.

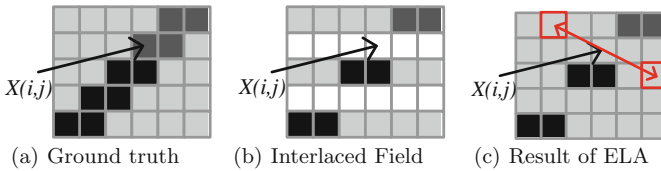


Fig. 2. Jagged effects of ELA (Color figure online)

Thus, in order to avoid jagged effects and get better edge direction estimation with wider search area for small angle edge directions, an intra-interpolation based on texture structural similarity is needed and will be proposed in Sect. 3.1.

2.2 Artifacts Introduced by Motion Compensation

The intra-interpolation can recover the missing pixel lines using the existing pixel lines of the current field. But it may cause blurring and distortion, especially in static areas. Thus motion adaptive is proposed to reconstruct the real missing information in static and slow motion areas using the neighboring fields. The key and difficult point of motion adaptive is to decide whether motion compensation is used. In most adaptive algorithms, the missing pixels are reconstructed by motion compensation if the compensated information is found in the searching scope. And in other regions, intra-interpolation is applied.

Theoretically, the adaptive method provides better performance by using neighboring fields, especially in static and slow motion areas. However, in practice, more artifacts are caused because of the worse robustness in motion compensation. Statistics show that the artifacts are related to the severity of compensation conditions. If we strictly control the compensation conditions, artifacts rate is low, but the compensated information we get is less, and the motion adaptive de-interlacing is basically the same with intra-interpolation de-interlacing. On the contrary, if we relax the conditions, more compensated information can be found, but more artifacts may be introduced. Thus, the accuracy of motion estimation and compensation need to be improved, and a trade-off criterion is required to integrate the intra-interpolation and compensated information adaptively.

3 The Proposed Method

3.1 Texture-Based Intra-interpolation

In order to improve the accuracy of edge estimation, we proposed a new intra-interpolation method using block-based texture similarity, which was first introduced by direction-oriented interpolation (DOI) algorithm [4]. DOI introduces a spatial direction vector (*SDV*) so that finer resolution and higher accuracy of the edge-direction can be obtained. In Fig. 3, we re-illustrate the concept of *SDV* and introduce the concepts of upper matching block (*UMB*) and lower matching block (*LMB*).

In Fig. 3, the white pixels represent the missing lines, the black and gray pixels represent the reference lines, and the black pixels represent the texture structure. Pixel $X(i, j)$ represents the current pixel to be interpolated. B_X is the local block centered at $X(i, j)$. The blue block upon line $j - 3$ and $j - 1$, that best matched with B_X , called Upper Matching Block (*UMB*). The upper *SDV*, SDV_U (the blue vector), is the vector that points from B_X to *UMB*. The *LMB* (the green block) and SDV_L (the green vector) are defined in a similar way.

The *UMB* and *LMB* are selected according to the smallest sum of absolute differences (SAD), and then SDV_U and SDV_L are calculated from the location of *UMB* and *LMB*. In this framework, more candidate directions can be searched, and 21 candidate directions is used in our algorithm, which can bring much better interpolation results along small angle edges.

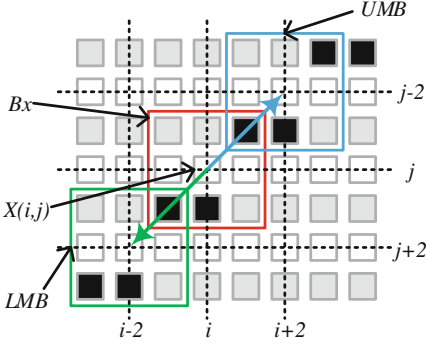


Fig. 3. Intra-texture-interpolation (Color figure online)

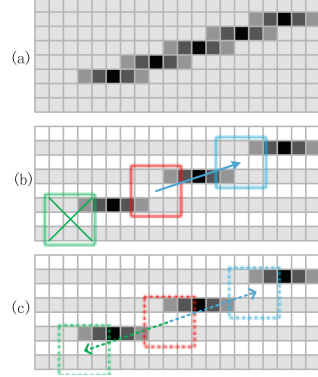


Fig. 4. An example at the ends of a line

In case of edge regions, there are two kinds of relationship between these parameters. In normal edge regions like Fig. 3, the directions of SDV_U and SDV_L are nearly opposite, which means the sum of these two vectors is close to 0. In this case, the edge direction is explicitly estimated. And $X(i, j)$ is interpolated according to the direction via Eq. 1.

$$X(i, j) = \frac{1}{2} [X(i + \frac{1}{2}SDV_{U_x}, j - 1) + X(i + \frac{1}{2}SDV_{L_x}, j + 1)] \quad (1)$$

In some edge regions like Fig. 4, the current block is located near the end of texture. In this case, only UMB (or LMB) can be found, as shown in Fig. 4(b). So SDV_U and SDV_L are absolutely not opposite. Here, we use the estimated direction of the pixels next to it as a reference direction vector (SDV_{ref}), shown in Fig. 4(c). If SDV_U (or SDV_L) is correlated to SDV_{ref} , we interpolate $X(i, j)$ using the SDV_U or SDV_L instead.

The edge direction estimation above is built on the hypothesis of non-horizontal edge regions. By using the texture structural similarity, the edge direction can be estimated more accurately, and the reconstruct performance in non-horizontal edge regions is perfect. However, there still exist horizontal edge regions and smooth regions. Thus, a pre-processing is needed before edge estimation for these regions.

$$X(i, j) = \frac{1}{2} [X(i, j - 1) + X(i, j + 1)] \quad (2)$$

In edge direction estimation shown in Fig. 3, horizontal edge is ignored. And in horizontal edge regions, no correct similar blocks can be found. So we also need to consider horizontal edge direction before edge estimation by calculating the SAD between current block and the left block (the right block). If SAD_{left} and SAD_{right} are both small enough, then the texture is considered to be horizontal and line averaging in Eq. 2 is applied.

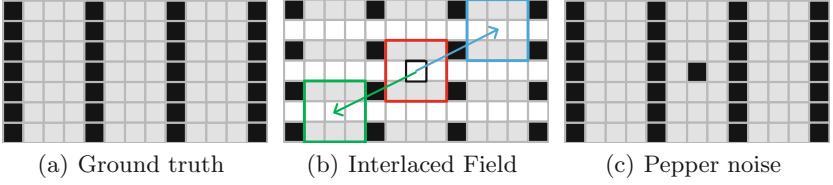


Fig. 5. The salt-and-pepper problem in smooth regions

For smooth regions, edge estimation may cause unwanted pepper noise. As shown in Fig. 5, similar blocks in non-local areas may be found, then wrong edge direction is estimated at a smooth region. In this case, unwanted pepper noise may be interpolated, as shown in Fig. 10(c).

Thus, we calculate the variance within a local region to distinguish smooth area. If the variance is small enough, then we consider it as a smooth region, and $X(i, j)$ is interpolated by region averaging in Eq. 3.

$$X(i, j) = \frac{1}{6} \sum_{k=-1}^{k=1} [X(i + k, j - 1) + X(i - k, j + 1)] \quad (3)$$

With the texture based framework above, the proposed methods not only increase the accuracy of direction estimation, but also provide a good framework for wider searching scope, which make it have better performance in both smooth regions and edge regions (including small angle edge regions), without jagged effects in small angle edges and pepper noise in smooth regions.

3.2 Motion Compensation

The texture based intra-interpolation performs well in reconstructing a single interlaced image without jagged effects and pepper noise. However, the problem of edge flickers still exists. The reason lies in the fact that different information may be absent in neighboring odd and even fields, so the texture based intra-interpolation may result different de-interlaced frames for the neighboring odd and even fields. Then edge flicker is caused when watching such de-interlaced frames continuously. Figure 6 shows an example of a stationary texture region with edge flicker.

To solve the problem of edge flickers, motion compensation (MC) algorithm is needed, in which motion estimation and compensation decision are two important parts.

In order to improve the accuracy of motion estimation, many algorithms calculate the sum of absolute difference (SAD) or the mean absolute difference (MAD) between the current block and candidate blocks in the same parity neighboring fields. Different search methods have also been proposed to decrease the computational complexity.

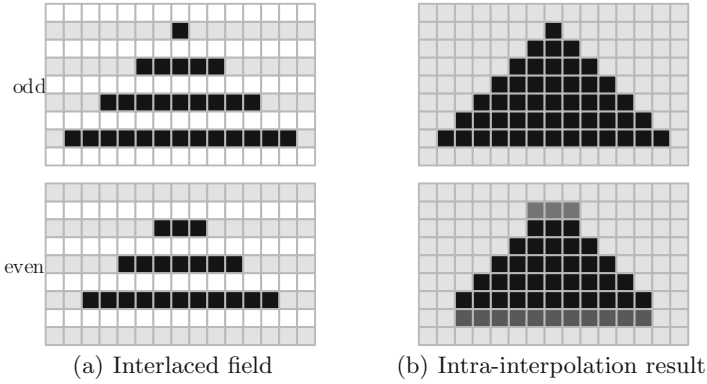


Fig. 6. Edge flicker in texture based intra-interpolation

In the proposed algorithm, four neighboring fields are used as reference fields for both backward and forward motion estimations. And we choose full search method for the best results, and the optimal matched block (OMB) in the same parity reference field is selected by the smallest SAD.

As for compensation decision part, a check of whether the motion vector (MV) can be used for compensation is implement. [9, 10] proposed a detail analysis of MC principle and we proposed a more effective version based on it, shown in Fig. 7. Figure 7(a) shows the original object.

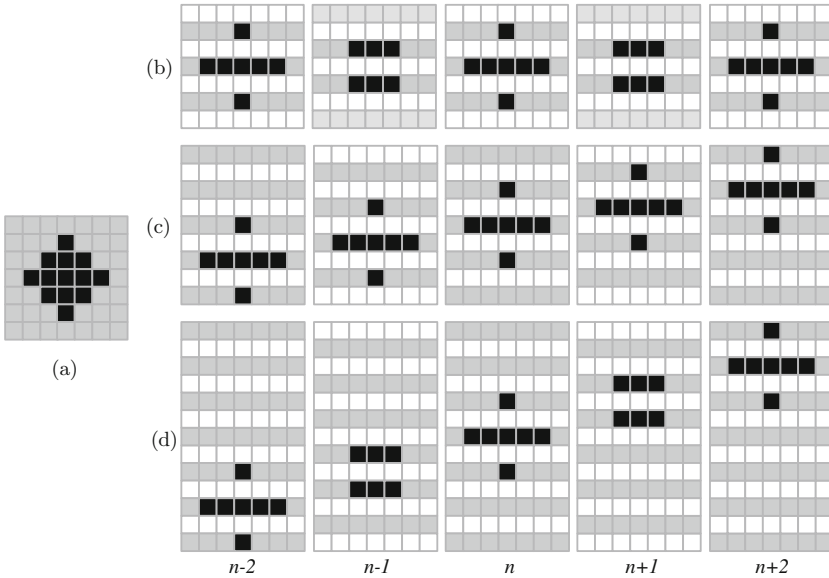


Fig. 7. A detail analysis of MC principle

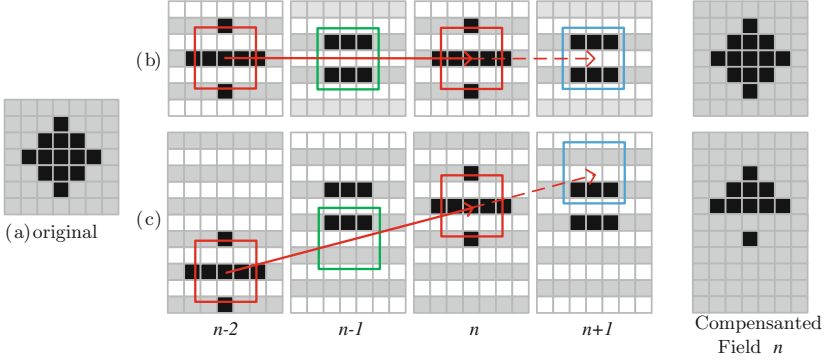


Fig. 8. Uniform motion and non-uniform motion (Color figure online)

We search the OMB for field n in the same parity field $n-2$ (and $n-2$), and calculate MV between field $n-1$ and n (field n and $n+1$). When the vertical motion is even, shown in Fig. 7(b) (vertical motion = 0) and Fig. 7(d) (vertical motion = 2), then the lost information of current field n can be found in opposite parity field $n-1$ (and $n+1$). This information may help in vertical resolution improvement.

When the vertical motion between field $n-1$ and n (field n and $n+1$) is odd, as shown in Fig. 7(c) (vertical motion = 1), then no compensated information can be found since the wanted information has been lost in every field.

For those blocks, which meet vertical motion conditions above, a reliability check of the estimated MV is still needed. The motion estimation and compensation based on a uniform motion is reliable. Figure 8 shows the different backward compensation performance for uniform and non-uniform motion. MV (the red vector) is estimated by the two red matching blocks, and the green block is the compensated information along the estimated MV. The compensation based on uniform motion improves vertical resolution in Fig. 8(b), but the compensation for non-uniform motion just bring artifacts in Fig. 8(c).

In order to check the reliability of the estimated MV, we evaluate the motion consistency by calculating SAD between blocks in field $n-1$ and $n+1$ (the green one and the blue one in Fig. 8) along the estimated MV. If the SAD is smaller than a threshold, the MV is considered to be uniform and thus reliable, and its compensated information can really help in vertical resolution improvement. Otherwise, which means non-uniform motion, the MV is considered to be unreliable and the compensation should be ignored. The threshold depends on a coefficient multiplied SAD between the current block and the OMB (the SAD between two red blocks in Fig. 8). The coefficient should be between 1 and 2 [7].

3.3 Overall Algorithm

The flow chart of complete proposed method is shown in Fig. 9. First, the texture-based intra-interpolation is implemented. Then some reliable compensated

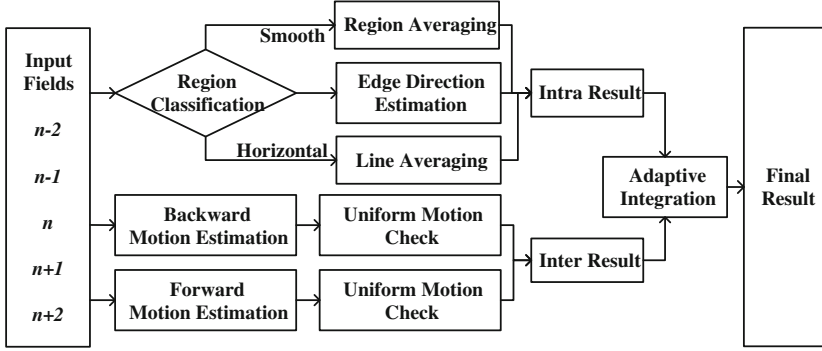


Fig. 9. Flow chart of the proposed method

information is provided via motion estimation and reliability check. Adaptive integration is finally constructed. In order to further reduce the artifacts introduced by motion compensation, we propose an adaptive integration for inter and intra results.

$$X = \omega_{intra}X_{intra} + \omega_{inter}X_{inter} \quad (4)$$

$$\omega_{intra} = \frac{1}{1 + e^{-\left(\frac{SAD_{OMB} - \mu}{\sigma}\right)}} \quad (5)$$

$$\omega_{inter} = 1 - \omega_{intra} \quad (6)$$

The weights of intra and inter are calculated by the sigmoid function Eqs. 5 and 6. μ is the average SAD that can be accepted, σ controls the steepness of the sigmoid curve. ω_{intra} ranges from 0 to 1. If SAD between OMB and current block is small, which means the reliability of inter results is higher, then ω_{intra} can be set smaller and ω_{inter} can be set larger. Otherwise, ω_{intra} is smaller and ω_{inter} can be set larger. The SAD used here is allocated to every pixels in the block.

The adaptive integration in the proposed framework can not only reconstruct more real missing information, but also keep the robustness of texture-based intra-interpolation, which performs a convincing de-interlacing result.

4 Experimental Results

Several 1080P and 720P 4:2:0 YUV sequences are used to evaluate the performance of proposed method. The progressive sequences are first interlaced and then de-interlaced by six de-interlacing techniques: ELA [1], FuzzyELA [3], DOI [4], FPMC [9], FFMPEG and the proposed algorithm. Several local image details are provided to show subjective comparisons, and the peak signal-to-noise ratio (PSNR) of the major luminance channel Y are used to measure the objective performances.



Fig. 10. Subjective results

Figure 10 shows the subjective comparisons in sequence “Cactus”. The ELA method has a poor performance even in some obvious edges. FuzzyELA cannot reconstruct the edges with small-angle. DOI performs well at obvious edges but provides poor quality in non-edge regions. FFMC provides a good performance in regions that satisfy motion compensation conditions. FFMPEG performs well in different regions, but jagged artifacts and errors still exists in some obvious edge regions. The proposed method offers the most robust quality in both non-edge and edge regions.

Table 1 shows the objective comparisons of PSNR in five sequences. We can see that the proposed method can achieve gain in PSNR with different level, which has 0.1.8dB gain when compared with FFMPEG.

The parallelism nature of the proposed framework makes it easy to allow an accelerated implementation on GPU equipment. We have developed an accelerated implementation on GPU “GeForce GTX 750Ti”. The experimental results show that at least 10 fields per second can be de-interlaced for 1080i video. Thus a real-time software de-interlacing system is expected on more powerful GPUs.

Table 1. Performance comparison in PSNR(dB)

Name	ELA5+5	FuzzyELA	DOI	FFMC	ffmpeg	Ours
Cactus	32.5	35.6	35.4	36.3	36.0	36.5
Kimono	37.5	42.3	41.9	41.1	41.1	42.4
Mobcal	26.3	33.4	32.4	30.5	31.9	33.7
Stockholm	25.7	30.1	27.9	29.6	30.1	29.9
Parkrun	22.1	25.9	24.2	26.5	26.9	26.9

5 Conclusion

This paper presents a novel compensated motion adaptive de-interlacing algorithm, with a texture-based intra-interpolation, an effective five-field-motion compensation and an adaptive integration. The algorithm further decreases the interpolation errors and artifacts, and has a robust performance in both edge and non-edge regions, especially in small-angle regions. In addition, an accelerated implementation of the proposed method on GPU are also provided.

Acknowledgment. This work was supported in part by National Natural Science Foundation of China (61133009, 61301116, 61221001), the Shanghai Key Laboratory of Digital Media Processing and Transmissions (STCSM 12DZ2272600).

References

1. Kuo, C.J., Liao, C., Lin, C.C.: Adaptive interpolation technique for scanning rate conversion. *IEEE Trans. Circ. Syst. Video Technol.* **6**(3), 317–321 (1996)
2. Jeon, G., Min, Y.J., Anisetti, M., Bellandi, V., Damiani, E., Jeong, J.: Specification of the geometric regularity model for fuzzy if-then rule-based deinterlacing. *J. Disp. Technol.* **6**(6), 235–243 (2010)
3. Brox, P., Baturone, I., Sanchez-Solano, S., Gutierrez-Rios, J.: Edge-adaptive spatial video de-interlacing algorithms based on fuzzy logic. *IEEE Trans. Consum. Electron.* **60**(3), 375–383 (2014)
4. Yoo, H., Jeong, J.: Direction-oriented interpolation and its application to de-interlacing. *IEEE Trans. Consum. Electron.* **48**(4), 954–962 (2002)
5. Lin, S.F., Chang, Y.L., Chen, L.G.: Motion adaptive de-interlacing by horizontal motion detection and enhanced ela processing. In: *International Symposium on Circuits and Systems*, pp. II-696–II-699 (2003)
6. Kim, W., Jin, S., Jeong, J.: Novel intra deinterlacing algorithm using content adaptive interpolation. *IEEE Trans. Consum. Electron.* **53**(3), 1036–1043 (2007)
7. Mohammadi, H.M., Langlois, P., Savaria, Y.: A five-field motion compensated deinterlacing method based on vertical motion. *IEEE Trans. Consum. Electron.* **53**(3), 1117–1124 (2007)
8. Ding, Y., Yan, X.: A robust motion estimation with center-biased diamond search and its parallel architecture for motion-compensated de-interlace. *J. Supercomput.* **58**(1), 68–83 (2011)

9. Mohammadi, H.M., Savaria, Y., Langlois, J.M.P.: Enhanced motion compensated deinterlacing algorithm. *IET Image Process.* **6**(8), 1041–1048 (2012)
10. Chang, Y.L., Chen, C.Y., Lin, S.F., Chen, L.G.: Four field variable block size motion compensated adaptive de-interlacing, vol. 2, pp. ii/913–ii/916 (2005)
11. Chang, Y.L., Lin, S.F., Chen, C.Y., Chen, L.G.: Video de-interlacing by adaptive 4-field global/local motion compensated approach. *IEEE Trans. Circ. Syst. Video Technol.* **15**(12), 1569–1582 (2006)
12. Sun, H., Liu, Y., Zheng, N., Zhang, T.: Motion compensation aided motion adaptive de-interlacing for real-time video processing applications. In: *Conference on Circuits*, pp. 1523–1527 (2008)
13. Trocan, M., Mikovicova, B., Zhanguzin, D.: An adaptive motion-compensated approach for video deinterlacing. *Multimedia Tools Appl.* **61**(3), 819–837 (2011)

Digital TV and Wireless Multimedia Communication
13th International Forum, IFTC 2016, Shanghai, China,
November 9-10, 2016, Revised Selected Papers
Yang, X.; Zhai, G. (Eds.)
2017, XII, 394 p. 221 illus., Softcover
ISBN: 978-981-10-4210-2