Optical Engineering

OpticalEngineering.SPIEDigitalLibrary.org

Efficient deinterlacing method using simple edge slope tracing

Sajid Khan
Dongho Lee

SPIE.

Efficient deinterlacing method using simple edge slope tracing

Sajid Khan and Dongho Lee*
Hanyang University ERICA Campus, Department of Electronics and Communication Engineering, 1271 Sa-3-dong, Ansan 426-791, Republic of Korea

Abstract. This paper presents a low-complexity interpolation method that minimizes image quality losses at edges, which are easily perceivable by the human eye. Deinterlacing, which converts an interlaced video into a progressive video, is a problem in image interpolation that doubles the number of vertical lines. Applying averaging, or any linear algorithm, achieves time-efficient deinterlacing but produces artifacts. However, applying other complex methods tends to reduce unwanted artifacts but at the cost of high computation time. The proposed deinterlacing scheme is based on an algorithm called "edge slope tracing" which simply predicts the slope on the basis of information on adjacent slopes. Predicted slopes are used to perform deinterlacing in slope-based line averaging. The simulation results show that this scheme provides better results and reduces complexity compared to conventional state-of-the-art algorithms. [DOI: 10.1117/1.OE.54.10.103108]

Keywords: deinterlacing; upscaling; edge; slope tracing; progressive scanning.

Paper 150752 received Jun. 3, 2015; accepted for publication Sep. 25, 2015; published online Oct. 20, 2015.

1 Introduction

In digital image processing, interpolation is used to produce a higher resolution image from a low-resolution image. Image interpolation is applied in many areas in the field of image processing, computer vision, and video format conversion for digital systems. Televisions and monitors are now more flexible at displaying different video formats than they were in the past. This interpolation has become a primary technique to support these digital displays.

The human eye is very sensitive to the edges of images, in particular. In recent years, interpolation algorithms that take the edge's direction and gradient into account have been proposed. Although such edge-based interpolation algorithms perform better than conventional nonadaptive methods in the subjective perception of image quality, they have limitations in retaining edge information because they cannot accurately identify the edges of the pixels to be interpolated.

Deinterlacing, which converts interlaced video into progressive video, is a problem for image interpolation; it doubles the number of vertical lines.[1–15] In general, TV broadcasters choose interlaced scanning as the main standard for video formats.[16] Since it alternately scans two fields by dividing a frame into two parts, interlacing may cause video quality degradation such as flickering. Because many devices such as DTV, UDTV, or other monitors use progressive scanning, the deinterlacing problem has become serious because it converts interlaced scanned videos into progressive scanned videos.

One representative method for deinterlacing is the edge-based line average (ELA),[3] whose simple structure improves the edges of images. However, when incorrect directions are identified for edges, ELA can result in video quality degradation. To resolve this issue, several other ELA-improved methods have been suggested. Of these, efficient edge-based line average (EELA)[4] and low-complexity deinterlacing[5] effectively increase the accuracy of edge direction determination. In addition, fine directional deinterlacing[8] and edge-preserving directional deinterlacing[10] are suggested to solve difficult interpolation problems in a low-sloped edge domain involving video quality degradation. Binary patterns,[9] gradient-guided deinterlacing (GGD),[12] deinterlacing with closeness and similarity[14] and the moving least-squares method (MLSM)[15] have also been suggested as methods to accurately restore various slopes. Nevertheless, when needed for high performance, such methods have some shortcomings in that they experience performance limitations or are difficult to implement in real time due to their high levels of complexity.

In this paper, a new intrafield deinterlacing algorithm based on edge slope tracing (EST) is proposed. EST predicts the slope of the current pixel on the basis of the information obtained from the slope of the adjacent pixel. This method, however, has some serious problems with interpolation for deinterlacing, which diverges when it fails to trace thin lines or edges. To solve these problems, correction techniques consisting of two-way interpolation, thin line correction, and window-based correction are applied. The complexity of the proposed algorithm is closer to those of linear filters such as ELA and is far less than those of other edge-based methods. Simulation results show that the proposed algorithm provides better results compared to other conventional methods proposed to date.

The rest of this paper is organized as follows. Section 2 describes some of the conventional methods. Section 3 describes the proposed EST-based deinterlacing algorithm. Section 4 describes the performance analysis. Section 5 is the conclusion.

*Address all correspondence to: Dongho Lee, E-mail: dhlee77@hanyang.ac.kr

2 Conventional Methods

In this section, conventional deinterlacing methods are discussed. For instance, GGD[12] predicts the gradients of missing pixels for interpolating unknown pixel intensity values. The algorithm is based on minimization of the energy function using

$$D(f_p) = \omega_1 D_1(f_p) + \omega_2 D_2(f_p) + \omega_3 D_3(f_p), \quad (1)$$

where ω_1, ω_2, and ω_3 are the weights assigned to each term. $D_1(f_p)$ and $D_2(f_p)$ are calculated using a summation of gradients and the intensities of known neighboring pixels, whereas $D_3(f_p)$ is based on the absolute sum of the upper and lower horizontal distances from an interpolating pixel that defines all possible edge orientations. Since $D(f_p)$ is based on various combinations of all neighboring pixels, it increases the complexity of the algorithm.

Low-complexity interpolation method (LCIM)[5] is a simple low-complexity algorithm that interpolates a missing pixel along one of the four directions (horizontal, vertical, first diagonal, and second diagonal) on the basis of minimum absolute differences. Since LCIM covers only four slopes, it fails to recover gentle-slope edges, which results in small slope variation.

MLSM[15] is based on a coefficient calculation using matrix multiplication operations as follows:

$$\hat{f}(x_0, y_0) = a_{00}^* + \frac{a_{01}^*}{\lambda^2} + \frac{a_{10}^*}{\lambda^2} + \frac{a_{11}^*}{\lambda^4}, \quad (2)$$

where coefficient matrix a can be calculated by matrix multiplication involving the known intensities of neighboring pixels. MLSM uses predetermined matrix multiplication for all unknown pixels that results in a reduction of complexity to a great extent, but does not provide good performance due to its limitation.

3 Proposed Method

The proposed algorithm is based on a simple efficient technique, EST, which predicts the present slope change using previous slope information. The slope value obtained by EST is used for slope-based interpolation. To remove unwanted artifacts, correction techniques consisting of two-way interpolation and window-based correction are applied. EST and slope-based interpolation are explained in detail followed by a detailed description of the complete proposed algorithm.

3.1 *Edge Slope Tracing*

The existing edge-based interpolation methods are either highly complex or suffer from problems with gently sloped edges. To resolve the issue with interpolating along gentle slopes using averaging techniques, a large mask is needed to detect the slope and to interpolate with the help of the detected slope. Such types of techniques may result in heavy calculations, consuming a large amount of execution time due to the many correlation calculations. This kind of technique may produce an erroneous edge selection if the mask is not large enough. In this paper, a very low-complexity algorithm is used for slope calculation by predicting the current slope on the basis of the slope information of the previous pixel.

Most of the edges experience a gradual change in the slope domain; in other words, new slopes are created by slightly changing the previous slope. EST searches slopes in the edge domain using the slope information of adjacent pixels; this results in the efficient calculation of the edge slope with greatly reduced complexity. Using EST, there is no need to calculate correlations or to use a large mask since the

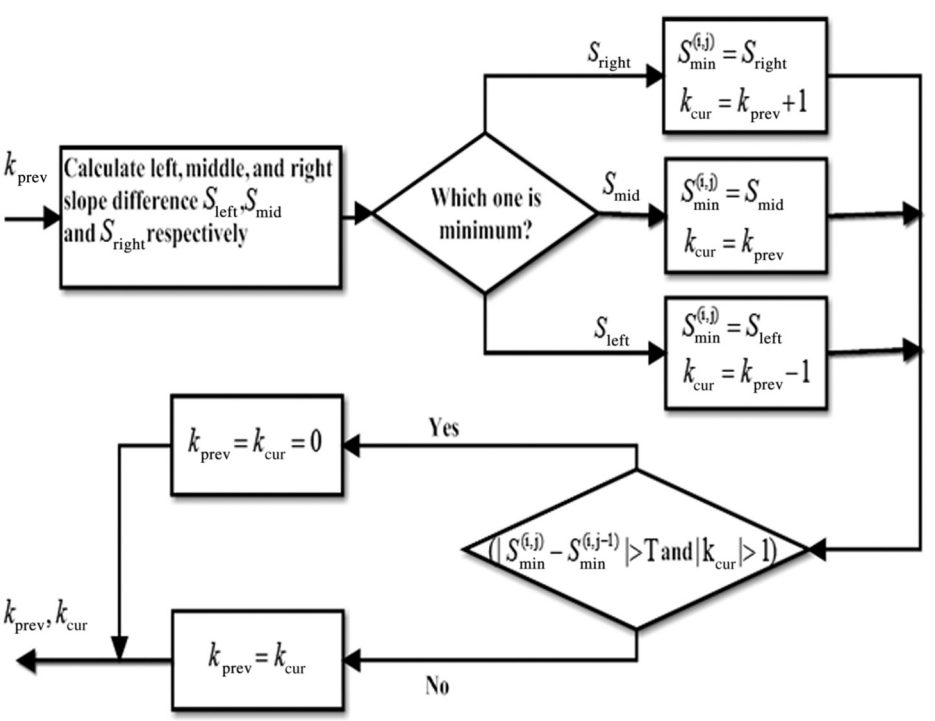

Fig. 1 Flow chart of the edge slope tracing (EST) process.

method depends totally on the slope information of the adjacent pixel which has been recursively calculated.

The flow chart of the entire EST process is shown in Fig. 1. Slope k_{cur} of the current pixel is calculated on the basis of the left, right, and middle slopes calculated using slope k_{prev} (the slope of the adjacent pixel). A value of $k_{prev} = 0$ is used as an initial slope for every first pixel of an image, or a new row, or after a discontinuity. For other pixels, the middle (S_{mid}), left (S_{left}), and right (S_{right}) intensity differences that are used in calculating k_{cur} are given as follows:

$$\begin{cases} S_{mid} = |I(i-1, j+k_{prev}) - I(i+1, j-k_{prev})|, \\ S_{left} = |I(i-1, j+k_{prev}-1) - I(i+1, j-k_{prev}+1)|, \\ S_{right} = |I(i-1, j+k_{prev}+1) - I(i+1, j-k_{prev}-1)|. \end{cases} \tag{3}$$

Two examples of calculating the middle, left, and right slopes using different k_{prev} values are shown in Fig. 2. For all calculations, it is assumed that the current pixel is at image location (i, j). Current slope k_{cur} is calculated by incrementing, decrementing, or using the same value of previous slope k_{prev} depending on the conditions, given as follows:

$$\begin{cases} k_{cur} = k_{prev} - 1, & \text{if } \min(S_{left}, S_{mid}, S_{right}) = S_{left} \\ k_{cur} = k_{prev} + 1, & \text{if } \min(S_{left}, S_{mid}, S_{right}) = S_{right} \\ k_{cur} = k_{prev}, & \text{else} \end{cases} . \tag{4}$$

To avoid the production of unwanted artifacts in case a wrong value exists for the reference to previous slope k_{prev}, a resetting criterion is used in which the value of the reference to the previous slope is reset to zero if the condition ($|S_{min}^{(i,j)} - S_{min}^{(i,j-1)}| > T$ and $|k_{cur}| > 1$) is satisfied. A large value of $|S_{min}^{(i,j)} - S_{min}^{(i,j-1)}|$ indicates that the region is not smooth, thus the current slope cannot be predicted on the basis of the information of the previous slope. A large value of T may fail in an efficient reset of the reference slope, whereas a very small value may result in a frequent reset of the reference slope. $T = 10$ is used on the basis of extensive simulations.

3.2 *Slope-Based Interpolation*

On the basis of slope k_{cur} calculated using EST, the pixel intensity at location (i, j) can be interpolated using

$$I(i, j) = \frac{I(i-1, j+k_{cur}) + I(i+1, j-k_{cur})}{2}. \tag{5}$$

Figure 3 shows different examples of the calculation of the interpolated pixel at location (i, j) using different values of k_{cur}. Small gray circles show pixels that are used in slope-based interpolation, whereas large gray circles show pixels that are interpolated by averaging small gray pixels.

3.3 *Entire Algorithm with Artifact Reduction*

The flow chart of the entire proposed algorithm is shown in Fig. 4. First of all, it detects if a pixel to be interpolated is at a vertical or thin edge. For vertical or thin edges, line averaging works well, whereas for other slopes, slope-based interpolation works well and produces good results on the basis of slope prediction.

The differences for deciding a vertical slope (90 deg or ± 75 deg) in Fig. 5 are given as follows:

$$d_1 = |I(i-1, j-1) - I(i+1, j-1)| + |I(i-2, j) - I(i+1, j)| + |I(i-1, j+1) - I(i+1, j+1)|, \tag{6}$$

$$d_2 = |I(i-1, j-1) - I(i+1, j)| + |I(i-1, j) - I(i+1, j+1)|, \tag{7}$$

$$d_3 = |I(i-1, j) - I(i+1, j-1)| + |I(i-1, j+1) - I(i+1, j)|. \tag{8}$$

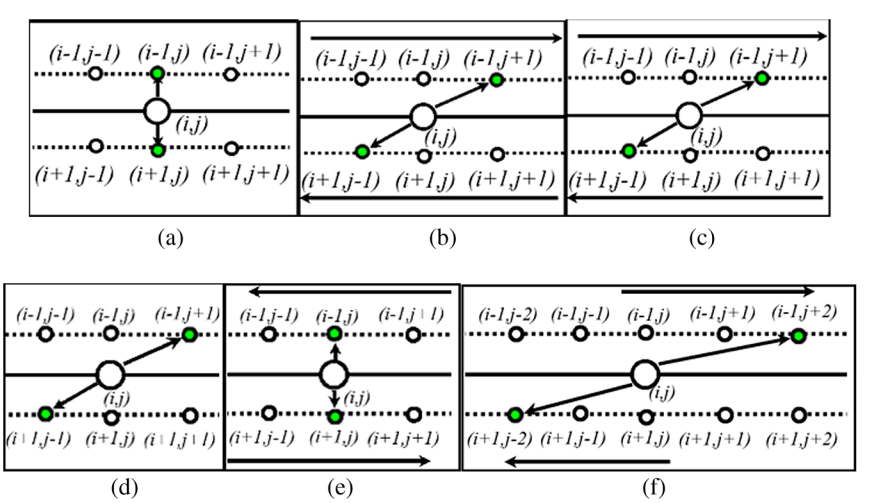

Fig. 2 Examples of slope calculation: (a), (b), and (c) are middle, left, and right slopes using a previous slope = 0. (d), (e), and (f) are middle, left, and right slopes using a previous slope = 1.

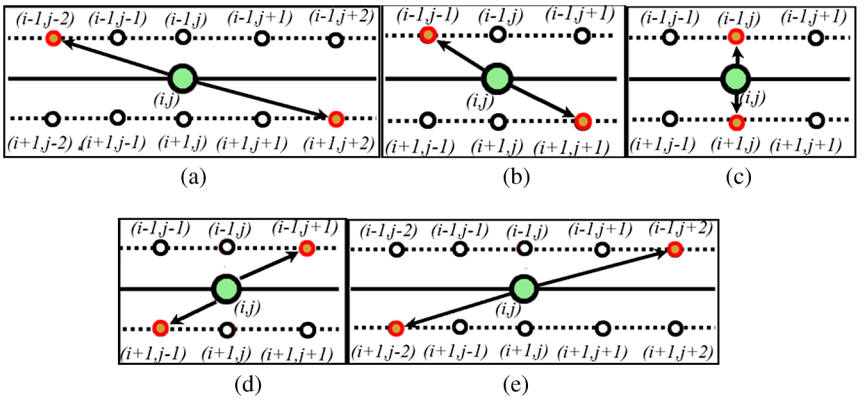

Fig. 3 Examples of slope-based interpolation using different slopes: (a) −2, (b) −1, (c) 0, (d) 1, and (e) 2.

Line averaging is applied to the pixel at location (i, j) if the condition $\min(d_1, d_2, d_3) < \text{Th}$ is satisfied, using $\text{Th} = 20$. In other locations, where $\min(d_1, d_2, d_3) > \text{Th}$, it is considered as a nonvertical edge and slope-based interpolation is applied.

Slope-based interpolation sometimes destroys very thin lines or edges. If a very thin edge separates two uniform regions with similar intensity values, few pixels from these thin edges are subjected to distortion since there is a chance that the difference among pixels in those two uniform regions is smaller than that along a thin edge. In such cases, slope-based interpolation selects pixels from those two uniform regions. To avoid such failure, if at least two differences among S_{right}, S_{left}, and S_{mid} given in Eq. (3) appear to be smaller than the threshold, it is considered as a thin edge and line averaging is applied. Hence, pixels at

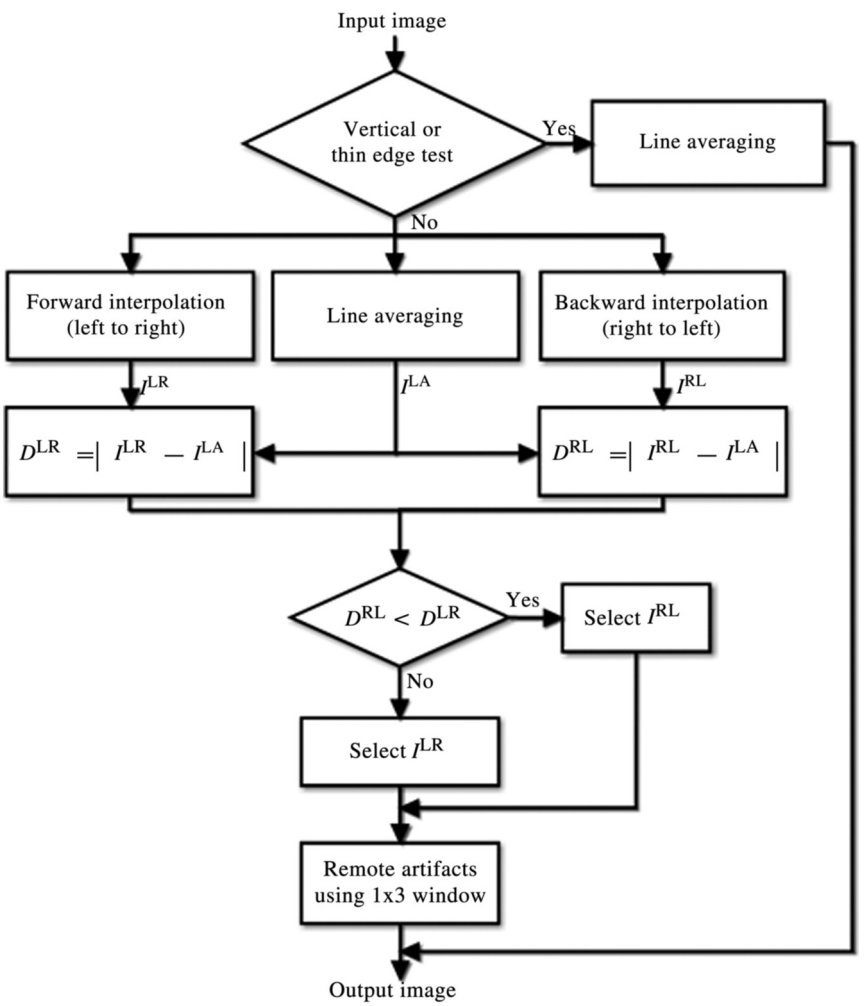

Fig. 4 Flow chart of the proposed interpolation method.

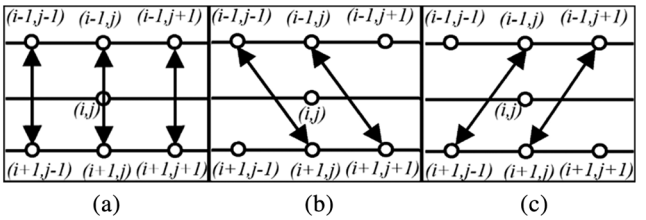

Fig. 5 Differences for determining ±75 or 90 deg slope: (a) d_1, (b) d_2, and (c) d_3.

vertical or thin edges are interpolated using line averaging. Slope-based interpolation is applied to the remainder of the regions.

Pixels that are not detected as vertical or thin edge pixels are interpolated along both the left to right (forward interpolation) and right to left (backward interpolation) directions to calculate the forward and backward interpolated images, I^{LR} and I^{RL}, respectively. The proposed method fails to track the slope when edges abruptly change in direction. I^{LR} and I^{RL} are required for the removal of unwanted artifacts. The

Fig. 6 Selected test images for evaluations.

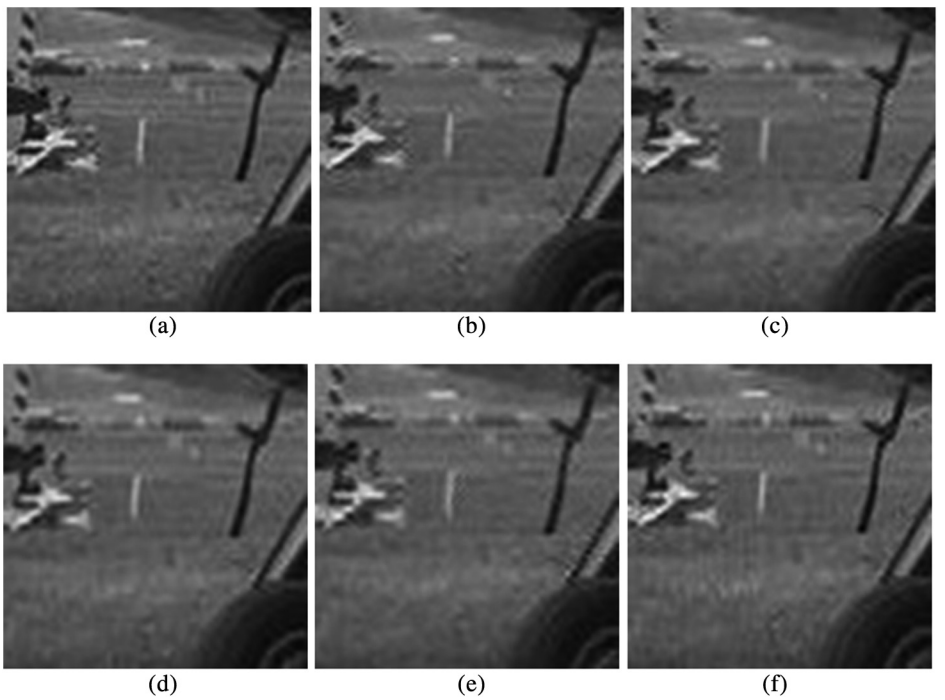

Fig. 7 Cropped regions after deinterlacing: (a) original, (b) edge-based line average (ELA), (c) gradient-guided deinterlacing (GGD), (d) LCIM, (e) moving least-squares method (MLSM), and (f) the proposed method.

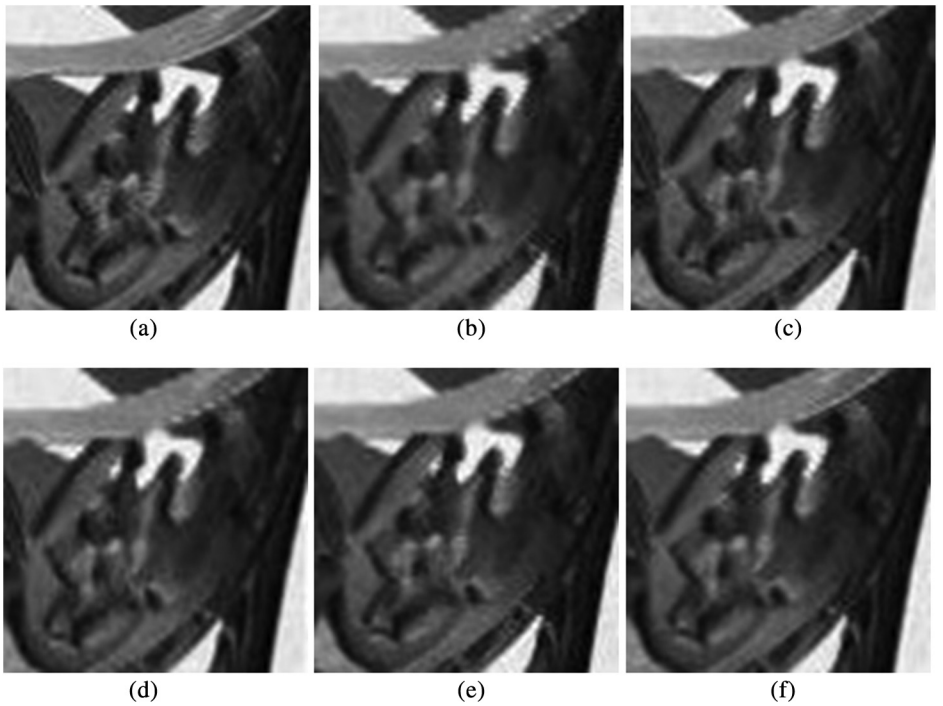

Fig. 8 Cropped regions after deinterlacing: (a) original, (b) ELA, (c) GGD, (d) LCIM, (e) MLSM, and (f) the proposed method.

decision to use the right to left or left to right interpolation is based on the following differences:

$$D^{\mathrm{LR}} = |I^{\mathrm{LR}} - I^{\mathrm{LA}}|, \tag{9}$$

$$D^{\mathrm{RL}} = |I^{\mathrm{RL}} - I^{\mathrm{LA}}|, \tag{10}$$

where I^{LA} is the image interpolated using line averaging. On the basis of D^{LR} and D^{RL}, interpolated image I is obtained using interpolated values in I^{LR} that correspond to small differences in D^{LR} compared to D^{RL} and vice versa.

The above method results in an image that provides good results along edges and other detailed regions; however, for some cases, the above method results in the production of

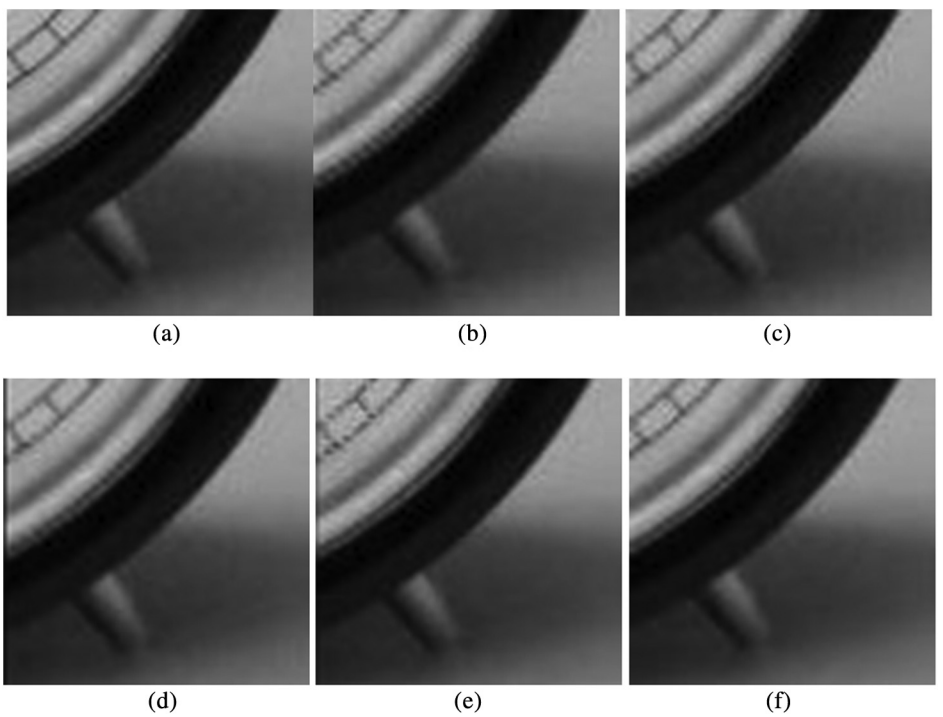

Fig. 9 Cropped regions after deinterlacing: (a) original, (b) ELA, (c) GGD, (d) LCIM, (e) MLSM, and (f) the proposed method.

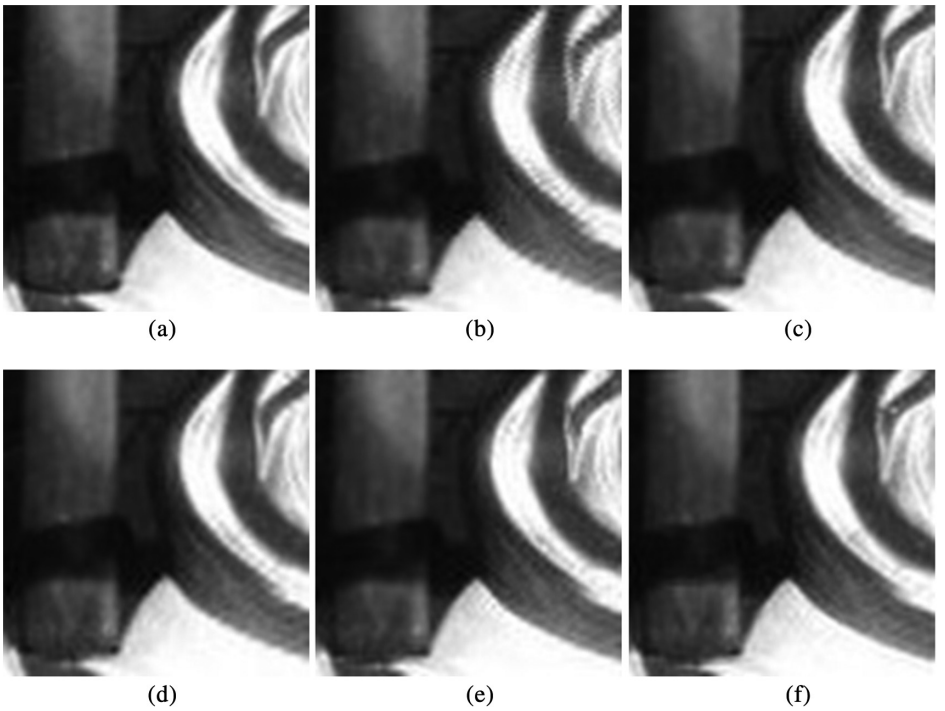

Fig. 10 Cropped regions after deinterlacing: (a) original, (b) ELA, (c) GGD, (d) LCIM, (e) MLSM, and (f) the proposed method.

unwanted artifacts in the form of a single or pair of pixels in some regions. For removal of such remaining artifacts, a 1×3 pixel window is moved across I at every interpolated location, (i, j), and all three pixel intensities, $I(i, j-1)$, $I(i, j)$, and $I(i, j+1)$, are compared with $I^{\mathrm{LA}}(i, j)$. To remove the uncertainty, an intensity that is closer to $I^{\mathrm{LA}}(i, j)$ is used to the replace the intensity value of I at location (i, j) to obtain the final interpolated image.

4 Performance Evaluation

The performance of the proposed algorithm is evaluated by applying ELA, GGD, LCIM, MLSM, and the proposed algorithm to 14 test images which is shown in Fig. 6. We performed both subjective and objective tests to quantitatively compare the quality of the images created with different methods and the related computational costs. MATLAB 2015a was used for collecting all results, and the tic and toc

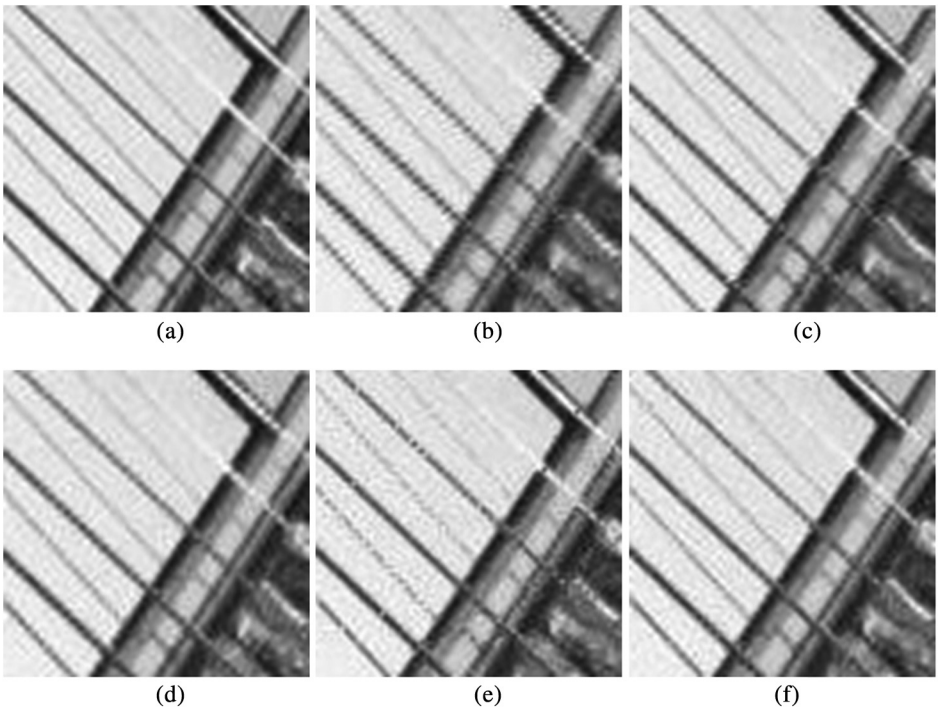

Fig. 11 Cropped regions after deinterlacing: (a) original, (b) ELA, (c) GGD, (d) LCIM, (e) MLSM, and (f) the proposed method.

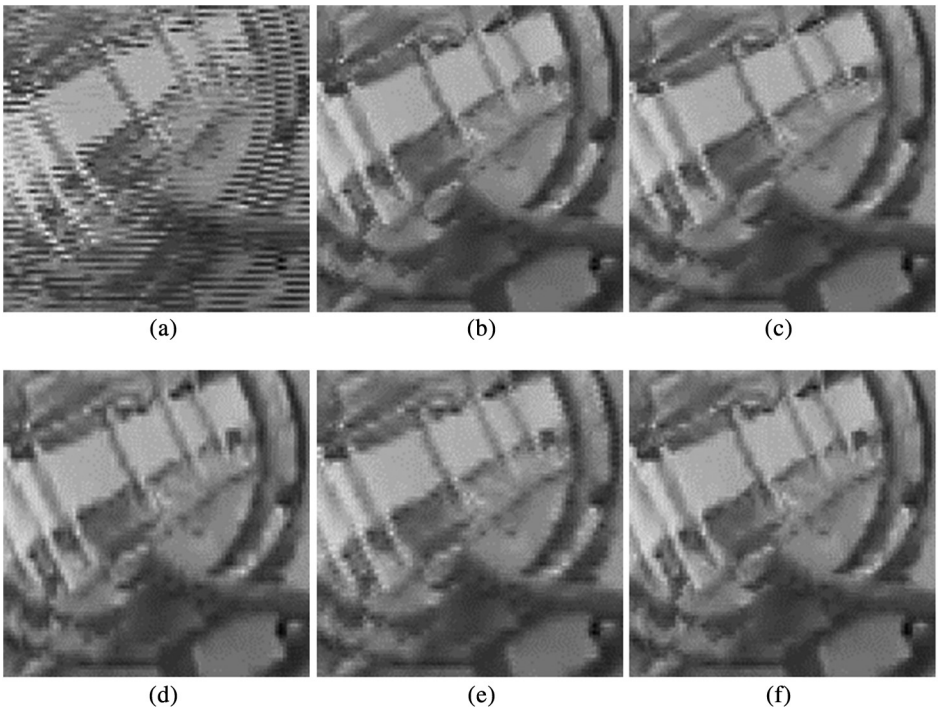

Fig. 12 Cropped regions after deinterlacing: (a) original interlaced frame, (b) ELA, (c) GGD, (d) LCIM, (e) MLSM, and (f) the proposed method.

functions of MATLAB were used to calculate execution time. The optimization levels of all implemented conventional methods were the same.

Figure 6 shows test images selected for evaluation since they represent varieties of patterns. The set of images in Fig. 6 contains grayscale images, but the proposed algorithm can also be applied to color images by either treating each red, green, and blue channel of an RGB image individually or by calculating the slopes for one channel and using the same slopes for the other two channels to reduce execution time.

For comparison of interpolation algorithms, subjective evaluation is very important, because the efficiency of an interpolation algorithm can be analyzed by observing edges and other details. Figures 7–11 show some cropped regions of test images subjected to deinterlacing. The ELA, LCIM, and MLSM methods failed to restore edges distorted due to downscaling of the original images, thereby producing jagged edges. ELA and LCIM failed to restore most of the smooth edge patterns, since they were designed to be best for vertical and diagonal edges. However, GGD recovered many cases of smooth edges because it used gradient prediction. GGD showed a similar performance to the proposed method, preserving edges in the presence of minor jagging as can be seen in Figs. 7–11. The proposed algorithm efficiently restores any edges, including thin lines, compared to other algorithms. For all cases, the proposed algorithm more efficiently restored details and edges compared to all other algorithms.

A video sequence of a fast-moving roller coaster was also used for performance evaluation. Figure 12(a) shows a cropped region of an interlaced frame which includes disparities in two different fields. The ELA, GGD, LCIM, and MLSM methods failed to restore some thin or gentle edge patterns, while the proposed algorithm recovered many cases of edges, but produced a few artifacts as well.

Although an interpolation algorithm can be analyzed mainly by using subjective evaluation, the peak signal to

Table 1 Comparison of the PSNR (dB) for different images.

	ELA	GGD	LCIM	MLSM	Proposed
Airplane	31.74	31.73	31.99	30.30	32.38
Parrot	35.92	36.42	35.63	32.50	36.95
Windows	26.14	26.32	26.60	25.95	26.84
Flower	34.37	34.68	34.15	31.55	35.04
Bench	32.50	32.51	31.39	28.45	32.37
Pelican	30.89	31.22	31.18	30.23	31.46
Clock	34.93	35.43	32.39	27.67	36.10
Butterfly	31.26	31.66	31.15	28.04	32.14
Bug	35.69	35.78	33.73	29.47	36.42
Horse toy	33.47	33.67	33.37	31.11	34.09
Sunflower	37.05	37.16	37.58	34.91	38.00
Guitar	31.90	32.40	32.31	28.94	33.63
House	30.96	31.25	31.36	29.29	31.67
Pearls	32.47	32.29	31.52	28.20	32.97
Average	32.81	33.04	32.45	29.76	33.40

Table 2 Comparison of complexity by elapsed CPU time (s).

	ELA	GGD	LCIM	MLSM	Proposed
Airplane	0.034	3.387	0.809	2.081	0.093
Parrot	0.025	3.378	0.812	2.051	0.046
Windows	0.026	3.347	0.814	2.055	0.084
Flower	0.025	3.352	0.813	2.066	0.062
Bench	0.016	2.234	0.545	1.365	0.041
Pelican	0.019	2.204	0.549	1.389	0.056
Clock	0.017	2.260	0.545	1.357	0.032
Butterfly	0.017	2.233	0.537	1.334	0.043
Bug	0.016	2.231	0.554	1.348	0.038
Horse toy	0.017	2.238	0.551	1.352	0.036
Sunflower	0.017	2.226	0.538	1.349	0.031
Guitar	0.016	2.258	0.542	1.354	0.038
House	0.017	2.231	0.547	1.358	0.04
Pearls	0.017	2.245	0.559	1.357	0.041
Average	0.020	2.559	0.622	1.558	0.049

noise ratio (PSNR), calculated as in Eq. (11), is also used for objective comparisons with other methods.

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{\mathrm{Max}^2}{\mathrm{MSE}}\right), \tag{11}$$

where MSE is the mean square error.

Table 1 shows the average PSNR for 14 test images when deinterlaced using ELA, GGD, LCIM, MLSM, and the proposed method. The results show that proposed method provides the highest PSNR in all cases. An image deinterlaced using the proposed method provides a PSNR that is higher than that of the existing methods: more than 0.6 dB higher than ELA, more than 1 dB higher than LCIM, more than 0.4 dB higher than GGD, and more than 3.4 dB higher than MLSM. ELA, like other simple averaging methods, provides a high PSNR even though it fails in recovering details and edges.

Table 2 shows a comparison of the execution times for all six algorithms applied to 14 test images. The execution time of ELA is the best among all algorithms since it requires few comparators and additions. ELA is almost 2 times faster than that of the proposed algorithm. The execution time for the proposed algorithm is almost 13 times faster than LCIM, which is the second most time-efficient method. GGD and MLSM are very complex compared to the proposed algorithm.

Figure 13 shows performance comparisons for 60 frames of the roller coaster video when deinterlaced using ELA, GGD, LCIM, MLSD, and the proposed method. The results show that the proposed method provides the highest PSNR for all frames as shown in Fig. 13(a). For a few frames, the PSNR of GGD is close to that of the proposed algorithm; however, for most cases, the PSNR of proposed method is far better than all other algorithms. Elapsed CPU time comparison is also shown in Fig. 13(b). The execution time of the proposed algorithm is close to ELA, but far better than that of the other algorithms.

The reason the proposed algorithm is so simple is because it is based only on additions and comparisons. The proposed method using EST requires just two lines of memory, 18 additions, no multiplications, and nine comparators.

5 Conclusion

In this paper, an efficient edge-based deinterlacing method is proposed. A technique called EST that predicts the present slope on the basis of the information of previous slopes is introduced and used for deinterlacing. The proposed deinterlacing offers high performance at very low complexity. Moreover, to improve the accuracy of EST-based deinterlacing, a slope resetting criteria, the application of two-way EST-based deinterlacing, and compensation for thin lines are also applied. Simulation results showed that restorations of edges were clearer in the proposed algorithm compared to most state-of-the-art conventional algorithms, whereas simple deinterlacing of an image without calculation of coefficients or the need for large correlation masks, unlike other conventional algorithms, demonstrates the superiority of the proposed algorithm on the basis of low complexity.

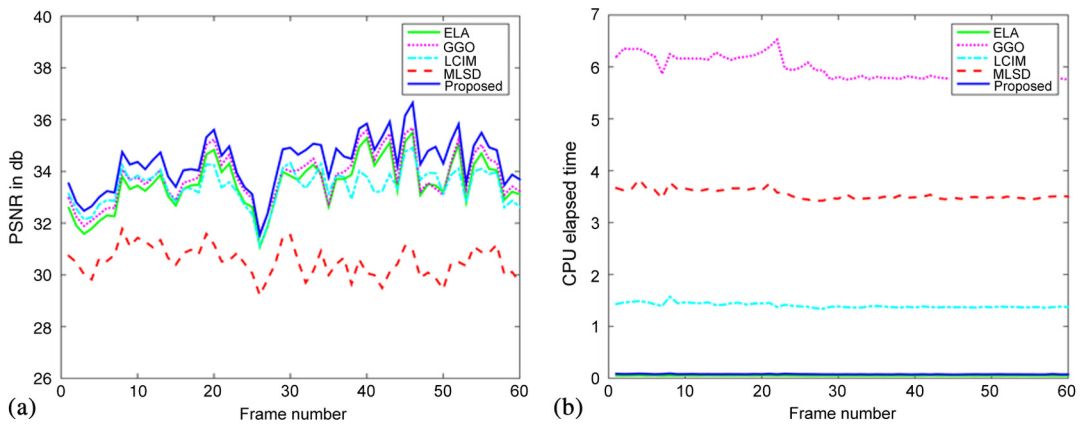

Fig. 13 Comparison for a video sequence: (a) PSNR and (b) elapsed CPU time.

References

1. F. Yao and S. Li, "Overview of scan format conversion for digital video," in *Int. Conf. Artificial Intelligence, Management Science and Electronic Commerce*, pp. 4123–4126 (2011).
2. J. F. Stromeyer and S. Klein, "Spatial frequency channels in human vision as asymmetric (edge) mechanisms," *Vision Res.* **14**(12), 1409–1420 (1974).
3. T. Doyle, "Interlaced to sequential conversion for EDTV applications," in *2nd Int. Workshop Signal Proc. HDTV*, pp. 412–430 (1998).
4. T. Chen, H. R. Wu, and Z. H. Yu, "Efficient deinterlacing algorithm using edge-based line average interpolation," *Opt. Eng.* **39**(8), 2101–2105 (2000).
5. P. Y. Chen and Y. H. Lai, "A low-complexity interpolation method for deinterlacing," *IEICE Trans. Inf. Syst.* **E90-D**(2), 606–608 (2007).
6. M. K. Kim and J. C. Jeong, "An efficient deinterlacing algorithm using the new edge-directed interpolation," *J. Korean Soc. Broadcast Eng.* **12**(2), 185–192 (2007).
7. S. H. Keller, F. Lauze, and M. Nielsen, "Deinterlacing using variational methods," *IEEE Trans. Image Process.* **17**(11), 2015–2028 (2008).
8. S. Jin, W. Kim, and J. Jeong, "Fine directional de-interlacing algorithm using modified Sobel operation," *IEEE Trans. Consum. Electron.* **54**(2), 857–862 (2008).
9. D. H. Lee, "A new edge-based intra-field interpolation method for deinterlacing using locally adaptive-thresholded binary image," *IEEE Trans. Consum. Electron.* **54**(1), 110–115 (2008).
10. S. Yang, D. Kim, and J. Jeong, "Fine edge-preserving deinterlacing algorithm for progressive display," *IEEE Trans. Consum. Electron.* **55**(3), 1654–1662 (2009).
11. X. Chen, G. Jeon, and J. Jeong, "Filter switching interpolation method for deinterlacing," *Opt. Eng.* **51**, 107402 (2012).
12. B. Jin, J. G. Kuk, and N. I. Cho, "A gradient guided deinterlacing algorithm," in *Int. Conf. Image Processing*, pp. 853–856 (2012).
13. D. H. Lee, "A simple, high performance edge-adaptive deinterlacing algorithm with very low complexity," in *IEEE Int. Conf. Consumer Electronics*, pp. 636–637 (2012).
14. J. Wang, G. Jeon, and J. Jeong, "Efficient adaptive deinterlacing algorithm with awareness of closeness and similarity," *Opt. Eng.* **51**(1), 017003 (2012).
15. J. Wang, G. Jeon, and J. Jeong, "Moving least-square method for interlaced to progressive scanning format conversion," *IEEE Trans. Circuits Syst. Video Technol.* **23**(11), 1865–1872 (2013).
16. T. Fukinuki, "Television: past, present, and future," *Proc. IEEE* **86**(5), 998–1004 (1998).

Sajid Khan received his BS degree in telecom engineering from FAST University Peshawar campus in 2011. He is currently pursuing his PhD in the Department of Electronics and Communication Engineering at the Hanyang University ERICA Campus. His current research interests include image interpolation, edge detection, biomedical image processing, and image denoising.

Dongho Lee received his MS and PhD degrees in electrical and computer engineering from the University of Texas at Austin in 1988 and 1991, respectively. From June 1991 to February 1994, he worked as a senior engineer at LG Electronics involving the development and implementation of DTV systems. Since 1994, he has been a professor in the Department of Electronics and Communication Engineering at the Hanyang University ERICA campus with research interests including digital image processing and pattern recognition.