# High Performance De-interlacing Algorithm Based on Region Adaptive Interpolation Filter

*Yang, Yang ** Chen, Xiangdong ***Wang, Jin ****Jeong, Jechang

Department of Electronics and Computer Engineering,
Hanyang University, 17 Haengdang-dong, Seongdong-gu, Seoul, Korea
E-mail: yangyang02566@163.com

## Abstract

In order to convert interlaced video into progressive scanning format, this paper proposed a high performance de-interlacing algorithm based on region adaptive interpolation filter design. Specifically, usage of the 6-tap filter is only for the most complex region, but for the smooth and regular edge region, much more correlated filter such as 2-tap or 4-tap filter should be used instead. According to the experimental results, the proposed algorithm has achieved noticeably good performance.

Keywords: De-interlacing, filter design, directional interpolation.

## 1. Introduction

De-interlacing is the conversion of video from interlaced format to progressive format [1]. The interlaced format is economical to save transmission bandwidth and system costs. However, it has defects such as line crawling and inter-line twitter in pictures with fine details. The goal of de-interlacing is to reconstruct the missing lines of pixels in interlaced images, reduce aliasing, and increase the vertical resolution of the images.

In order to overcome these problems, many de-interlacing techniques have been proposed during the last two decades. These techniques can be roughly divided into two categories: inter-field(temporal) and intra-field(spatial) algorithms. Actually, the inter-field algorithm shows better performance than intra-field algorithm, but it requires a mass of calculation and memory for implementation, so it is not suitable for real-time communication. Consequently, the intra-field algorithm is widely being applied.

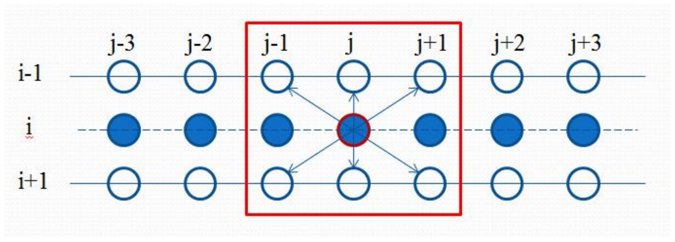
The most typical method among intra-filed algorithm is Edge-based Line Average(ELA)[3]. This method extracts edge information first and then determines its direction to use appropriate pixels for interpolation. However, it just calculates average values between lines after simple edge direction extraction, and it is much sensitive to small variations of pixel values and uses incorrect edge information. To resolve these problems, plentiful improved methods which based on ELA are proposed, such as Efficient Edge-based Line Average(EELA)[4] and Modified Edge-based Line Average(MELA)[5]. Even though these methods show a little bit better performance than ELA, there still has been limitation in complex edge region. Based on this point, another algorithm, fixed directional interpolation filter(FDIF) [2], which apply sinc interpolation filter and weighted distance scheme has been presented. This algorithm can highly improve the performance of de-interlacing images, especially for the images which consist of much complex edge information. But for the smooth region, this algorithm looks too onerous that it may cost a good deal of time to achieve a nearly result.

To surmount this issue, the paper presents a high performance de-interlacing algorithm which is based on region adaptive interpolation filter design. The outline of the paper is as follows. Section 2 introduces the previous methods such as ELA, EELA, MELA. And the proposed method is recommended in details during Section 3. In Section 4, experimental results are presented and Section 5 gives the conclusion.

## 2. The Existing Methods

### A. ELA

The ELA method is a traditional edge-based method which considering three directional correlations between two adjacent interlacing lines. In this method, a 3x3 window is shown in Fig.1 in which detects vertical, diagonal, and antidiagonal directional correlations. The directional correlation measurement C(k) is



determined by equation (1).

Fig.1 3x3 window in ELA method

$$C(k) = | P(i-1, j-k) - P(i+1, j+k) |, k = -1,0,1 \qquad (1)$$

Among C(-1), C(0), and C(1), the lowest value shows the highest correlation, which is determined as equation (2).

$$C(m) = \arg\min(C(k)), k = -1,0,1 \qquad (2)$$

Then, the current pixel is interpolated by equation (3).

$$P(i, j) = [P(i-1, j-m) + P(i+1, j+m)]/2 \qquad (3)$$

### B. EELA

Even though ELA provides a good result in the region where the edge can be correctly estimated, it still gives a poor performance in high spatial frequency region due to the edge detection errors. So, several methods have been proposed, such as EELA, which introduces two more directional correlations are defined shown in Fig.2.

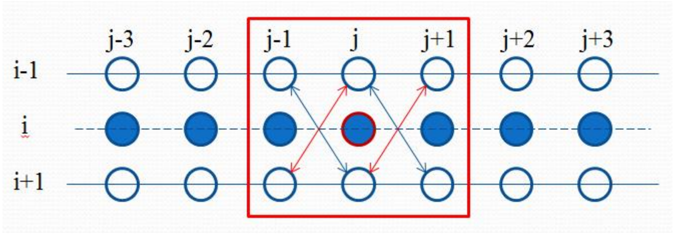


Fig.2 3x3 window in EELA method

$$R = |P(i-1, j-1) - P(i+1, j)| + |P(i-1, j) - P(i+1, j+1)| \quad (4)$$

$$L = |P(i-1, j) - P(i+1, j-1)| + |P(i-1, j+1) - P(i+1, j)| \quad (5)$$

Equation (4) and (5) denote the differences between the two edge directions, and then compare them to select candidate pixels which are used to interpolate. There are three cases for consideration as follows. Here, C(-1), C(0) and C(1) are defined as equation (1).

Case1: R<L

$$\begin{cases} P(i,j) = [P(i-1,j) + P(i+1,j)]/2, & C(0) \le C(1); \\ P(i,j) = [P(i-1,j-1) + P(i+1,j+1)]/2, & C(0) > C(1). \end{cases}$$

Case2: R=L

Apply ELA method.

Case3: R>L

$$\begin{cases} P(i,j) = [P(i-1,j) + P(i+1,j)]/2, & C(0) \le C(-1); \\ P(i,j) = [P(i-1,j+1) + P(i+1,j-1)]/2, & C(0) > C(-1). \end{cases}$$

## C. MELA

The EELA method recommends two directional measurements in order to obtain accurate edge directions as much as possible, but it cannot provide a good PSNR result both in thin edges and textured regions. The MELA method can get better performance than ELA and EELA, due to the three directional detections. The extraction of edge direction in MELA is very similar as EELA, but one more direction is added as follows.

$$R = [|P(i-1, j-1) - P(i+1, j)| + |P(i-1, j) - P(i+1, j+1)|]/2 \quad (6)$$

$$L = [|P(i-1, j) - P(i+1, j-1)| + |P(i-1, j+1) - P(i+1, j)|]/2 \quad (7)$$

$$V = [|P(i-1, j-1) - P(i+1, j-1)| + |P(i-1, j) - P(i+1, j)| + |P(i-1, j+1) - P(i+1, j+1)|]/3 \quad (8)$$

Equation (6), (7) and (8) explain the three directional correlations, there are also three cases for application.

$$\begin{cases} P(i,j) = [P(i-1,j-1) + P(i+1,j) + P(i-1,j) + P(i+1,j+1)]/4, \\ R = \min\{R,L,V\} \,\&\&\, C(1) < C(0); \\ P(i,j) = [P(i-1,j) + P(i+1,j-1) + P(i-1,j+1) + P(i+1,j)]/4, \\ L = \min\{R,L,V\} \,\&\&\, C(-1) < C(0); \\ P(i,j) = [P(i-1,j) + P(i+1,j)]/2, \\ \textit{Otherwise.} \end{cases}$$

Where, C(-1), C(0) and C(1) also come from equation (1).

## D. FDIF

This method employs a sinc interpolation filter, which is a 1-D 6-tap interpolation filter with fixed coefficients. Furthermore, it applies an adaptive weighting scheme according to the distance between the current pixel to be interpolated and neighboring pixels. Before adopting the sinc interpolation filter, the edge direction should be determined using MELA method. The 11x11 window in FDIF method is shown in Fig.3.

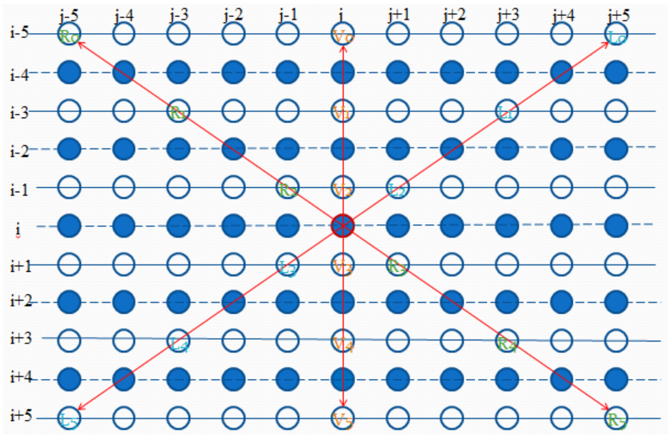


Fig.3 11x11 window in FDIF method

The three cases in FDIF method is described as follows. By the way, C(-1), C(0) and C(1) are still determined by equation (1).

$$\begin{cases} P(i,j) = \dfrac{C(1)}{C(0)+C(1)} \cdot \sum_{m=0}^{5} h(m)\cdot R(m) + \dfrac{C(0)}{C(0)+C(1)} \cdot \sum_{m=0}^{5} h(m)\cdot V(m), \\ R = \min\{R,L,V\} \,\&\&\, C(1) < C(0); \\ P(i,j) = \dfrac{C(-1)}{C(-1)+C(0)} \cdot \sum_{m=0}^{5} h(m)\cdot L(m) + \dfrac{C(0)}{C(-1)+C(0)} \cdot \sum_{m=0}^{5} h(m)\cdot V(m), \\ L = \min\{R,L,V\} \,\&\&\, C(-1) < C(0); \\ P(i,j) = \sum_{m=0}^{5} h(m)\cdot V(m), \\ \textit{Otherwise.} \end{cases}$$

The coefficients of sinc function are h={3,-17,78,78,-17,3}/128. Where R(m), L(m) and V(m) denote the antidiagonal, diagonal and vertical direction pixels, respectively, which also can be found in Fig.3.

## 3. The Proposed Method

FDIF algorithm presents a better performance than the other existing algorithms, particularly for the images which containing much complex edge information. When the current pixel which located in irregular region should be interpolated, the window size can be enlarged to enhance the correlation between the current pixel and the neighboring pixels, and it can approximately estimate the target as much as possible. However, for the smooth and regular region, a smaller window size is enough to satisfy the directional correlation requirements. Meanwhile due to most of an image with less complex directional correlation, for the smooth region, Line Average(LA) algorithm is a good choice. In our proposed method, with the purpose of reducing the CPU time, dividing the image into three regions by equation (9) primarily.

$$D=[|P(i-1,j-1)-P(i+1,j+1)|+|P(i-1,j)-P(i+1,j)|+ \\ |P(i-1,j+1)-P(i+1,j-1)|]/3 \quad (9)$$

And then, the threshold values are chosen experimentally. Based on the two threshold values, the whole image can be segmented into three regions, such as smooth region, regular edge region and irregular or texture region. The three cases are described in details as follows.

Case1: D<=th1

This region can be determined as smooth area, and a 2-tap fixed interpolation filter with coefficients $h_1$=(1,1)/2 is applied. The same as Fig.4 represents.

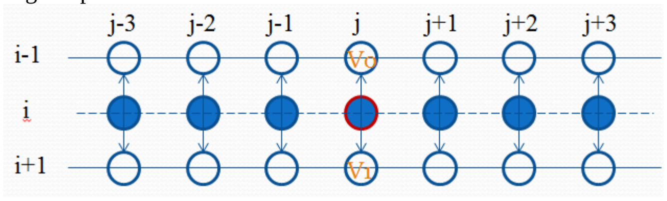


Fig.4 2-tap filter interpolation for smooth region

The interpolation equation is shown as follows, and here the notation V(m) denotes the pixels which using to be interpolated, it is also described in Fig.4.

$$P(i,j)=\sum_{m=0}^{1}h_1(m)\cdot V(m)$$

Case2: D>th1 && D<=th2

This region may contains of regular edge information, and we apply 4-tap fixed Winer filter with coefficients $h_2$=(-1,5,5,-1)/8 to interpolate the missing pixels. A 7x7 window is used in this condition as shown in Fig.5. The interpolation formulas are described below. And R(m), V(m), L(m) also denote the pixels which should be applied for interpolation, meanwhile C(-1), C(0) and C(1) like we previous mentioned that still come from equation (1).

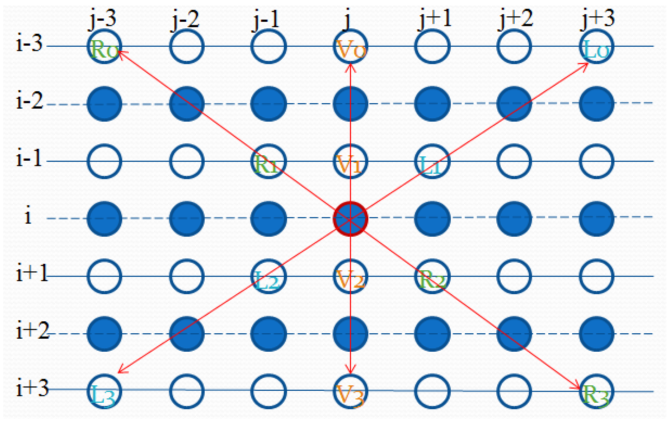


Fig. 5 4-tap filter interpolation for regular edge region

$$\begin{cases} P(i,j)=\dfrac{C(1)}{C(0)+C(1)}\cdot X+\dfrac{C(0)}{C(0)+C(1)}\cdot Z, \\ R=\min\{R,L,V\}\,\&\&\,C(1)<C(0); \\ P(i,j)=\dfrac{C(-1)}{C(-1)+C(0)}\cdot Y+\dfrac{C(0)}{C(-1)+C(0)}\cdot Z, \\ L=\min\{R,L,V\}\,\&\&\,C(-1)<C(0); \\ P(i,j)=Z, \\ Otherwise. \end{cases}$$

And the X, Y, Z equations are defined as follows.

$$\begin{cases} X=(\sum_{m=0}^{3}h_2(m)\cdot R(m)+4)/8 \\ Y=(\sum_{m=0}^{3}h_2(m)\cdot L(m)+4)/8 \\ Z=(\sum_{m=0}^{3}h_2(m)\cdot V(m)+4)/8 \end{cases}$$

Case3: D>th2

This region may be much complex edge area, and a 6-tap fixed Winer filter with coefficients $h_3$=(1,-5,20,20,-5,1)/32 to be used. A 11x11 window should be employed for de-interlacing. And the figure is same as Fig.3.

In this situation, the edge extraction method and the conditions are same as Case2, but for the X, Y, Z equations, there is a little bit difference between them by the coefficients of the 6-tap fixed Winer filter.

$$\begin{cases} X=(\sum_{m=0}^{5}h_3(m)\cdot R(m)+16)/32 \\ Y=(\sum_{m=0}^{5}h_3(m)\cdot L(m)+16)/32 \\ Z=(\sum_{m=0}^{5}h_3(m)\cdot V(m)+16)/32 \end{cases}$$

## 4. Experimental Results

In this section, we present experimental results from region adaptive interpolation filter design. Experimentally, we set the threshold of D as th1=10, th2=15.

Here, we compare the PSNR values for still sequences between different de-interlacing methods in Table 1. The proposed algorithm increases 0.18 DB in average than FDIF method. Meanwhile, due to the intersected region, the proposed method can reduce the CPU time as much as 78.57%, which is shown in Table 2.

Fig. 6(a) shows the original image of Airplane with the size 512x512. Fig. 6(b) is the partial image of (a) during the red rectangular. Fig. 6(c)-(g) are the de-interlacing results by the existing algorithms. The result of the proposed algorithm is described in Fig. 6(h).

Tabel 1 PSNR Comparison for Still Sequences (DB)

| **Method** | Airplane | Jets | Lena | Peppers | Toys | Average |
|---|---|---|---|---|---|---|
| **LA** | 34.25 | 39.12 | 37.67 | 33.80 | 33.35 | 35.64 |
| **ELA** | 33.00 | 38.98 | 36.03 | 34.10 | 32.47 | 34.92 |
| **EELA** | 33.32 | 39.10 | 36.97 | 34.18 | 33.03 | 35.32 |
| **MELA** | 34.12 | 39.32 | 37.96 | 34.15 | 33.34 | 35.78 |
| **FDIF** | 34.70 | 39.43 | 38.19 | 33.85 | 33.71 | 35.98 |
| **Proposed** | 34.81 | 39.64 | 38.34 | 34.10 | 33.93 | 36.16 |

Tabel 2 Average Results of CPU Time for Still Sequences(s/frame)

| **Method** | Airplane | Jets | Lena | Peppers | Toys | Average |
|---|---|---|---|---|---|---|
| **LA** | 0.01 | 0.03 | 0.01 | 0.02 | 0.01 | 0.02 |
| **ELA** | 0.06 | 0.14 | 0.06 | 0.06 | 0.06 | 0.08 |
| **EELA** | 0.07 | 0.20 | 0.07 | 0.08 | 0.07 | 0.10 |
| **MELA** | 0.30 | 1.11 | 0.31 | 0.32 | 0.31 | 0.47 |
| **FDIF** | 0.51 | 2.02 | 0.55 | 0.55 | 0.55 | 0.84 |
| **Proposed** | 0.16 | 0.30 | 0.15 | 0.15 | 0.13 | 0.18 |

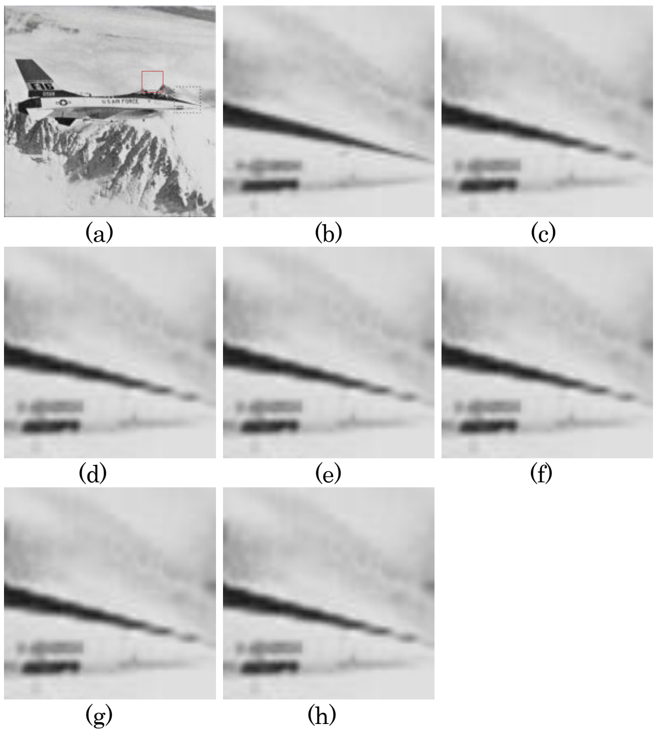


Fig. 6 Original image and several de-interlacing results. (a) Original image of Airplane 512x512. (b) Original partial image. De-interlacing results of (c) LA, (d) ELA, (e) EELA, (f) MELA, (g) FDIF, (h) the proposed method.

## 5. Conclusion

This paper presents a high performance de-interlacing algorithm based on the region adaptive filter design. The proposed method first separates the smooth, regular edge and complex edge region by simple correlation between 6 pixels during a 3x3 window, and then using the 2-tap, 4-tap and 6-tap filters for the three regions respectively. Because most of an image is less complex area, so the region segmentation can highly increase the efficiency. Experimental results show the proposed algorithm has a better performance than the previous algorithms.

## 6. Acknowledgments

This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency) (NIPA-2011-C1090-1100-0010).

## 7. References

[1] G. de Haan and E. B. Bellers, “Deinterlacing-An overview,” *Proc. IEEE*, vol. 86, no. 9,pp. 1839-1857, Sep. 1998.
[2] S. Hong, S. Park, J. Jang and J. Jeong, “Deinterlacing algorithm using fixed directional interpolation filter and adaptive distance weighting scheme,” *Proc. SPIE Optical Engineering*, vol. 50, no.6, June 2011.
[3] C. J. Kuo, C. Liao and C. C. Lin, “Adaptive interpolation technique for scanning rate conversion,” *IEEE Trans. Circuits and Syst. Video Technol.* vol. 6, no. 3, 317-321, 1996.
[4] T. Chen, H. R. Wu and Z. H. Yu, “Efficient deinterlacing algorithm using edge-based line average interpolation,” *Opt. Eng.* vol. 39, no.8, 2101-2105, 2000.
[5] W. Kim, S. Jin and J.Jeong, “Novel intra deinterlacing algorithm using content adaptive interpolation,” *Proc. IEEE, Transaction on Consumer Electronics*, vol. 53, no. 3, pp. 1036-1043, 2007.
[6] S. F. Lin, Y. L. Chang and L. G. Chen, “Motion Adaptive De-interlacing with Horizontal Motion Detection and ELA with Median,” *Proc. International Symposium on Circuits and System*, Bangkok, Thailand, pp.696-699, 2003.
[7] Y. Y. Jung, B. T. Choi, Y. J. Park and S. J. Ko, “An Effective De-interlacing Technique Using Motion Compensation Interpolation,” *Proc. IEEE, Transaction on Consumer Electronics*, vol. 46, no. 3, pp. 460-466, 2001.
[8] A. M. Tourapis, O.C. Au and M. L. Liou, “Advanced De-interlacing Techniques with the Use of Zonal Based Algorithm,” *Proc. Visual Communications and Image Processing*, San Jose, CA, pp.949-958, Jan.2001.
[9] 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,” *Proc. IEEE Trans. Circuits Syst. Video Technol*., vol. 15, no. 12, pp. 1569–1582, Dec. 2005.
[10] T. S. Chong, O. C. Au, T. W. Chan and W. S. Chau, “A Spatial-Temporal De-interlacing Algorithm,” *Proc. IEEE Int. Conf. Multimedia Expo*, pp. 249–252, Jul. 2005.
[11] H. Yoo and J. Jeong, “Direction-oriented Interpolation and Its Application to De-interlacing,” *Proc.IEEE Trans. Consum. Electron.*, vol. 48, no. 4, pp. 954–962, Nov. 2002.
[12] M. K. Park, M. G. Kang, K. Nam and S. G. Oh, “New Edge Dependent De-interlacing Algorithm Based on Horizontal Edge Pattern,” *Proc.IEEE Trans. Consum. Electron.*, vol. 49, no. 4, pp. 1508–1512, Nov. 2003.