

# Effat University Repository

## Video De-interlacing: From Spatial to Adaptive Based Motion Detection

Authors	Salem, Nema;ElBadawy, ElSayed
Publisher	The International Institute of Informatics and Cybernetics, IIS
Download date	2026-03-17 06:55:45
Link to Item	<a href="http://hdl.handle.net/20.500.14131/689">http://hdl.handle.net/20.500.14131/689</a>

# Video De-interlacing: From Spatial to Adaptive Based Motion Detection

Nema M. Elfaramawy

Dept. Of Elect Eng., Faculty of Eng., Alexandria University, Alexandria 21544, Alexandria Egypt.

E-mail: [n\\_elfaramawy@yahoo.com](mailto:n_elfaramawy@yahoo.com)

El-Sayed A. El-Badawy

Alexandria Higher Institute of Engineering & Technology, Alexandria 21311, Egypt.

& Dept. of Elect Eng., Faculty of Eng., Alexandria University, Alexandria 21544, Alexandria Egypt

Senior Member IEEE, Member of the Optical Society of America (OSA), E-mail: [sbadawy@ieee.org](mailto:sbadawy@ieee.org)

## ABSTRACT

While interlacing succeeds in reducing the transmission bandwidth, it introduces a number of high frequency artifacts that can be distracting to the human eye, such as flicker and thin vertical lines. Thus, a lot of de-interlacing algorithms are developed. Subjective and objective assessments to spatial and temporal de-interlacing techniques are given in this paper. Combining benefits of the spatial and temporal algorithms, a motion detector is used to segment the frame into static and dynamic portions. The dynamic is segmented into slow- and high-motion pixels. Finally, an adaptive de-interlacing algorithm based on the results obtained from the motion detector is proposed and judged.

**Keywords:** Video Processing, De-interlacing, Spatial and Temporal Filtering, Motion Detection.

## 1. INTRODUCTION

Digital video representation provides flexibility for processing, enables integration, and can be transmitted with lower band width (BW) than analog. It can be classified as interlaced or non-interlaced (progressive scan) type. In 1941, the NTSC in the US introduced an interlacing-based television (TV) that was used in the US exclusively until the introduction of high definition television (HDTV) in 1995. PAL and SECAM are interlacing-based TV used in Europe. As a result, interlacing is still widely used in video systems. NTSC has 525 horizontal lines but only 486 of these lines are visible on the screen, the rest are in the sync pulse, which is not visible. Each horizontal line is made up of 720 pixels. This gives NTSC a total screen resolution of 720 x 486 pixels with aspect ratio of 4:3. The pixels are not squares with an aspect ratio of  $(4/720)/(3/486) = 9:10$  causing problems when displaying on a computer [1]. A frame is a single photographic image in a motion picture. The size of a film frame varies depending on the film format from 4.8 by 3.5 mm<sup>2</sup> to 69.6 by 48.5 mm<sup>2</sup>. The larger the frame size is, the sharper the image will appear. Computers display using progressive scan, in which all lines in a frame are displayed in one pass providing better final picture quality. Moreover, thin lines and small text are more likely to flicker on an interlaced display. An interlaced sequence is one in which the display alternates between scanning the *even lines* and *odd lines* of the corresponding progressive frames. The standard convention is to start numeration at zero (1<sup>st</sup> line is even). The term *field* is used (rather than frame) to differentiate between interlacing and progressive. Interlacing *even/top field* means a field containing all the even lines of one frame and the *odd/bottom field* means a field containing all the odd lines of that frame [2].

$F(x,y,n)$  denotes the pixel in a progressively-scanned 3-D video sequence at horizontal position  $x$ , vertical position  $y$ , and time  $n$ . A generated interlaced  $F_i(x,y,n)$  and the de-interlacing output  $F_o(x,y,n)$  are [2]:

$$F_i(x,y,n) = \begin{cases} F(x,y,n), & (y \bmod 2 = n \bmod 2) \\ 0 & \text{otherwise} \end{cases}$$
$$\text{mod}(M) = \begin{cases} 0 & M \text{ even} \\ 1 & M \text{ odd} \end{cases}$$
$$F_o(x,y,n) = \begin{cases} F_i(x,y,n), & (y \bmod 2 = n \bmod 2) \\ \tilde{F}(x,y,n), & \text{otherwise} \end{cases}$$

where  $\tilde{F}(x,y,n)$  represents the estimated missing lines. The existing even or odd, lines in the original fields are directly transferred to the output frame.

US and Japan use a 60-Hz power cycle, and so their early TVs were only able to display at a 30 frame per second. Interlacing two 30 field per second achieved an effective 60 frame per second, which solved the problem of low bandwidth and was high enough to provide adequate persistence of vision. The European standards use interlacing at 25 frames per second to achieve an effective 50 fields per second.

If slowing down or holding a frame or viewing video on a progressively scanned monitor are required without flickering or visual stuttering, then fields should be converted into complete frames requiring interlaced to progressive conversion, known as *de-interlacing algorithms*.

There are various methods for de-interlacing as *Weave* method that combines the two fields into one progressive frame leading to feathering artifacts. *Bob* method in which one field is discarded and the remaining field is doubled. This eliminates any feathering but reduces the resolution to half, so non-moving objects look worse [2]. *Blend* method in which the two fields are blurred together to make the interlace artifacts less noticeable. This causes a ghost effect in motion areas. *In adaptive* method, a weave interpolation is applied to the non-moving areas, while either one field's information is thrown away and new data is interpolated or the fields are blended for the moving areas [3].

Thus, de-interlacing algorithms should be adapted to give better visual quality, acceptable computational complexity for system integration and hardware cost, and be served as a base for other scan rate conversions.

Over the last two decades, many de-interlacing algorithms have been proposed. They range from simple spatial (intra-field) interpolation, via directional dependent filtering, up to advanced motion-detection interpolation. Some are already available in products,

while others wait when industry can justify their complexity and cost [3-11]. This paper reviews most of these algorithms and compares their behavior. RMSE, SNR, and PSNR are used as video quality measurements. The disadvantages of each algorithm are shown using screen photographs. This paper may help industry in taking decision about the best algorithm to be used.

The paper is organized as follows: The spatial de-interlacing algorithms are discussed in Section 2. Section 3 discusses the inter-frame (temporal) algorithms. Using the benefits of spatial and those of the temporal algorithms, adaptive de-interlacing algorithm based motion detection is discussed in Section 4. Section 5 presents the performance evaluation of each algorithm. Finally, discussion and conclusion are given in Section 6.

## 2. SPATIAL DE-INTERLACING ALGORITHMS

They only use pixels in the current field to construct the missing lines. They are simple and there's no need of storage elements but their performance is limited due to the high temporal correlation that exists between successive fields. The most basic algorithms are line-repetition, and linear interpolation. One of the advanced algorithms is the parametric image modeling which models a small region of an image through a set of parameters and basis equations. Another advanced algorithm that employ the correlation between the pixels on the same edge called **edge-based line average (ELA)**. We developed Matlab codes to extract even and odd fields from each progressive frame to be used with all of the following algorithms, see Figure 1.

### 2.1 Line Repetition

The missing lines are generated by repeating the line directly above or below the missing line. Mathematically:

$$F_o(x, y, n) = \begin{cases} F_i(x, y, n) & \text{even} \\ F_i(x-1, y, n) & \text{otherwise} \end{cases}$$

$$F_o(x, y, n) = \begin{cases} F_i(x, y, n) & \text{odd} \\ F_i(x+1, y, n) & \text{otherwise} \end{cases}$$

This interpolation has poor performance especially at vertical details; those are incorrectly reconstructed leading to shaking in the video sequences. The rate of this vibration is in direct relation to the frame rate. At high frame rates, this shaking may be so fast that it becomes un-noticeable to the human eye. While at lower frame rates, this effect becomes very noticeable and extremely annoying (Figure 2a).

### 2.2 Linear Interpolation

The missing lines are constructed by averaging the lines directly above and below. Mathematically:

$$F_o(x, y, n) = \begin{cases} F_i(x, y, n) & \text{mod}(y, 2) = \text{mod}(n, 2) \\ \frac{F_i(x-1, y, n) + F_i(x+1, y, n)}{2} & \text{otherwise} \end{cases}$$

Figure 2b, shows a slightly smoother image than line repetition but still has poor performance as artifacts.

### 2.3 Martinez and Lim Algorithm

Figure 3 illustrates the algorithm that begins by selecting a group of 10 pixels, 5 on each line, surrounding the missing pixel [12]. The parameters for the image model are then estimated for the selected group of pixels, and

these parameters are used in estimation. Finally, the interpolation is done along the estimated direction. In the figure the missed pixel at the center is interpolated by averaging the two pixels along the edge direction.

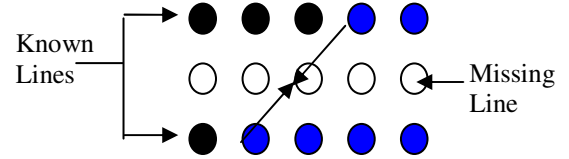


Figure 3: Illustration of Martinez-Lim Algorithm.

It is less susceptible to noise but still suffers from some jitter effects (Figure 2c).

### 2.4 Edge-Based Line Average (ELA)

The missed pixel is approximated by the interpolation of its adjacent pixels on  $n$ -directions [13]. In the 3-tap ELA algorithm, three directional differences  $D_1$ ,  $D_2$ , and  $D_3$  around the missed pixel are computed and the interpolation is done along the minimum direction with the highest correlation as follows:

$$D_1 = \text{abs}[f(x-1, y-1, n) - f(x+1, y+1, n)],$$

$$D_2 = \text{abs}[f(x-1, y, n) - f(x+1, y, n)], \quad \text{and}$$

$$D_3 = \text{abs}[f(x-1, y+1, n) - f(x+1, y-1, n)].$$

$$F_o(\bar{x}, n) = \begin{cases} f(\bar{x}, n), & y \bmod 2 = n \bmod 2, \\ \left. \begin{aligned} & \frac{f(\bar{x} - \begin{bmatrix} 1 \\ 1 \end{bmatrix}, n) + f(\bar{x} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}, n)}{2}, & \text{if } D_{\min} = D_1 \\ & \frac{f(\bar{x} - \begin{bmatrix} 1 \\ 0 \end{bmatrix}, n) + f(\bar{x} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}, n)}{2}, & \text{if } D_{\min} = D_2 \\ & \frac{f(\bar{x} - \begin{bmatrix} 1 \\ -1 \end{bmatrix}, n) + f(\bar{x} + \begin{bmatrix} 1 \\ -1 \end{bmatrix}, n)}{2}, & \text{if } D_{\min} = D_3 \end{aligned} \right\} \text{otherwise}$$

It detects edges and generates a high-contrast frame. Enhanced ELA uses 5 pixels instead of 3. The interpolation is done along the smallest-difference direction (of 5):

$$D_1 = \text{abs}[f(x-1, y-2, n) - f(x+1, y+2, n)]$$

$$D_2 = \text{abs}[f(x-1, y-1, n) - f(x+1, y+1, n)]$$

$$D_3 = \text{abs}[f(x-1, y, n) - f(x+1, y, n)]$$

$$D_4 = \text{abs}[f(x-1, y+1, n) - f(x+1, y-1, n)]$$

$$D_5 = \text{abs}[f(x-1, y+2, n) - f(x+1, y-2, n)]$$

$$F_o(\bar{x}, n) = \begin{cases} f(\bar{x}, n), & y \bmod 2 = n \bmod 2, \\ \left. \begin{aligned} & \frac{1}{2} (f(\bar{x} - \begin{bmatrix} 1 \\ 2 \end{bmatrix}, n) + f(\bar{x} + \begin{bmatrix} 1 \\ 2 \end{bmatrix}, n)), & \text{if } D_{\min} = D_1 \\ & \frac{1}{2} (f(\bar{x} - \begin{bmatrix} 1 \\ 1 \end{bmatrix}, n) + f(\bar{x} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}, n)), & \text{if } D_{\min} = D_2 \\ & \frac{1}{2} (f(\bar{x} - \begin{bmatrix} 1 \\ 0 \end{bmatrix}, n) + f(\bar{x} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}, n)), & \text{if } D_{\min} = D_3 \\ & \frac{1}{2} (f(\bar{x} - \begin{bmatrix} 1 \\ -1 \end{bmatrix}, n) + f(\bar{x} + \begin{bmatrix} 1 \\ -1 \end{bmatrix}, n)), & \text{if } D_{\min} = D_4 \\ & \frac{1}{2} (f(\bar{x} - \begin{bmatrix} 1 \\ -2 \end{bmatrix}, n) + f(\bar{x} + \begin{bmatrix} 1 \\ -2 \end{bmatrix}, n)), & \text{if } D_{\min} = D_5 \end{aligned} \right\} \text{otherwise}$$

Results of 3- and 5-tap ELA are shown in Figures 2d and 2e. This method is widely used because of its low computational requirements and easy implementation in hardware.

### 3. TEMPORAL DE-INTERLACING ALGORITHMS

They consider previous and/or subsequent frames to exploit temporal correlation. So, they require storage for more than one field in the implementation. The most basic algorithms are field repetition, bilinear field, VT median filter, and 3-D edge oriented.

#### 3.1 Field Repetition

Missing lines can be interpolated by copying lines from previous frame at the same position. Mathematically can be defined as:

$$F_o(x, y, n) = \begin{cases} F_i(x, y, n) & \text{mod}(y, 2) = \text{mod}(n, 2) \\ F_i(x, y, n-1) & \text{otherwise} \end{cases}$$

At high frame rates, the temporal correlation between two adjacent fields may be very high. So, the field repetition can perform well with minor noticeable blurring. But, at lower temporal rates, the blurring effect becomes more pronounced (result is in Figure 3b for frames shown in a).

#### 3.2 Bilinear Field Interpolation (Averaging)

It uses the average of the previous and future lines to interpolate the current missing line. Mathematically:

$$F_o(x, y, n) = \begin{cases} F_i(x, y, n) & \text{mod}(y, 2) = \text{mod}(n, 2) \\ \frac{F_i(x, y, n+1) + F_i(x, y, n-1)}{2} & \text{otherwise} \end{cases}$$

Result in Figure 3c gives “ghostly” image with badly blurred edges especially in moving objects and is unacceptable compared with the field repetition. But, it works very well in stationary regions.

#### 3.3 Vertical Temporal (VT) Median Filter

The missed lines are interpolated by applying median filter on the spatial neighbors in the vertical direction and the same position in the previous field. Mathematically:

$$F_o(x, y, n) = \begin{cases} F_i(x, y, n) & \text{mod}(y, 2) = \text{mod}(n, 2) \\ \text{median}(F_i(x-1, y, n), F_i(x+1, y, n), F_i(x, y, n-1)) & \text{otherwise} \end{cases}$$

Median filter belongs to the class of *order statistical filters*. It adapts itself to both of the stationary and moving regions as well. In a stationary region, the value of pixel C is between the values of A and B. Thus, pixel C will be chosen by the median operation resulting in temporal filtering. In a moving region, the value of pixel C is different from this of A or B, meaning that, pixels A and B have a higher correlation with each other than with pixel C. The median filter will then select either pixel A or pixel B resulting in spatial filtering. A number of variations on this median filter have been suggested such as the 7-tap filter, robust to motion, where the following operation is implemented [14, 15],

$$A = F_i(x-1, y, n), B = F_i(x+1, y, n), C = F_i(x, y, n-1),$$

$$D = F_i(x, y, n+1), E = \frac{A+B}{2}, \text{ and } F = \frac{C+D}{2}.$$

$$F_o(x, y, n) = \begin{cases} F_i(x, y, n) & d(y, 2) = \text{mod}(n, 2) \\ \text{median}(A, B, C, D, E, F) & \text{otherwise} \end{cases}$$

In the presence of motion, linear interpolation of pixel E is better than line repetition of either pixel A or B. Pixel E is given twice as much weight in the median filter. Results are shown in Figures 3d & 3e.

### 3.4 3-D Edge-Orientated De-interlacer

If intra-field interpolation is necessary because of motion, then edge orientation dependent algorithms can at least preserve low frequency spatial information through inspection of the diagonal (temporal-horizontal) edge gradient. Oh et al. [16] suggested linear interpolation between the pixels of minimum gradient, where the gradient is tested in six orientations computed as follows:

$$U_a = |A-F|, U_b = |B-E|, U_c = |C-D|$$

$$U_i = |I-N|, U_j = |J-M|, U_k = |K-L|$$

where,

$$A = f(x-1, y-1, n), B = f(x-1, y, n), C = f(x-1, y+1, n), D = f(x+1, y-1, n),$$

$$E = f(x+1, y, n), F = f(x+1, y+1, n), G = f(x-3, y, n), H = f(x+3, y, n),$$

$$I = f(x, y-1, n-1), J = f(x, y, n-1), K = f(x, y+1, n-1), L = f(x, y-1, n+1),$$

$$M = f(x, y, n+1), N = f(x, y+1, n+1).$$

Then, the pixels are interpolated by using:

$$F_{edge}(x, y, n) = \begin{cases} X_a, U_a = \text{MIN} \left\{ U_a, U_b, U_c, U_i, U_j, U_k \right\} \\ X_b, U_b = \text{MIN} \left\{ U_a, U_b, U_c, U_i, U_j, U_k \right\} \\ X_c, U_c = \text{MIN} \left\{ U_a, U_b, U_c, U_i, U_j, U_k \right\} \\ X_i, U_i = \text{MIN} \left\{ U_a, U_b, U_c, U_i, U_j, U_k \right\} \\ X_j, U_j = \text{MIN} \left\{ U_a, U_b, U_c, U_i, U_j, U_k \right\} \\ X_k, U_k = \text{MIN} \left\{ U_a, U_b, U_c, U_i, U_j, U_k \right\} \end{cases}$$

$$X_a = \frac{A+F}{2}, X_b = \frac{B+E}{2}, X_c = \frac{C+D}{2}$$

$$X_i = \frac{I+N}{2}, X_j = \frac{J+M}{2}, X_k = \frac{K+L}{2}.$$

The result of this algorithm is shown in Figure 3f.

## 4. ADAPTIVE MOTION DE-INTERLACER

In regions with heavy changes in the temporal domain, pixels on the same spatial location in two adjacent fields may correspond to different objects. Thus, temporal interpolation produces image with error, and spatial interpolation will be better in this situation. To overcome this problem, a motion-adaptive method can be used [17]. The choice of the de-interlace methods depends on the output of motion detector. If motion is detected, intra-field method is used; otherwise, inter-field interpolation is adapted. Motion detection for each pixel can be based on the temporal variation in a small neighborhood surrounding the pixel. The essential stage of the motion adaptive method is the motion detection process. The next section covers some of the issues with detecting motion in interlaced sequences and describes the different algorithms that are implemented in this study.

#### 4.1 Motion Detectors

They are used to detect changes in a video sequence from one field to the next. Pixels with significant change are moving, while the others are static. This task is straightforward for a progressive sequence. A simple frame difference can be used to detect changes and a threshold can be applied to indicate the presence or absence of motion. Illustration of this idea is shown in Figure 4a. For interlaced sequences, frame differencing is impossible since consecutive fields do not have the same pixel locations. The following sections discuss some of the proposed solutions for motion detection in interlaced sequences.

**4.1.1 Two-Field Detector:** Converting the fields into frames, motion detection can be performed as in the progressive case and any spatial de-interlacing method can be used for this purpose. A theoretical approach to this problem is presented by Li et al. [18]. Since the difference between an even and an odd fields is a phase shift, they suggested applying a phase correction filter to one field so that it may be more appropriately compared with the opposite polarity field for motion detection. They used the following 6-tap filter for phase correction of the current frame:

$$F_o[x, y, n] = \begin{cases} F_i[x, y, n] & \text{mod}(y, 2) = \text{mod}(n, 2) \\ 3 \cdot F_i[x, y, n \pm 5] - 21 \cdot F_i[x, y, n \pm 3] + 147 \cdot F_i[x, y, n \pm 1] & \text{o.w.} \end{cases}$$

The pixels above and below the reconstructed pixel do not necessarily have any correlation to the actual value of that pixel. Thus, the interpolation may be done incorrectly and the absolute difference will not be an appropriate measure of motion. This leads to a significant number of false-detections, and results in a lower quality. Moreover, the algorithm looks forward and backward till the fifth frame in each direction.

**4.1.2 Three-Field Detector:** It is only compares pixels on the corresponding lines without interpolation:  $\text{motion\_detection} = |f(x, y, n-1) - f(x, y, n+1)|$

It is unable to detect fast moving (Figure 4b) [19, 20].

**4.1.3 Four-Field Detector:** The output of this detector is based on three absolute differences rather than only one. So, it is better in detecting fast motion (Fig. 4c):

$$\begin{aligned} A &= |F_i(x, y, n-1) - F_i(x, y, n+1)| \\ B &= |F_i(x-1, y, n) - F_i(x-1, y, n-2)| \\ C &= |F_i(x+1, y, n) - F_i(x+1, y, n-2)| \\ \text{motion\_detected} &= \max(A, B, C) \end{aligned}$$

**4.1.4 Five-Field Detector:** Grand Alliance [21, 22] developed a detector based on the computation of five differences to reduce the number of missed detection:

$$\begin{aligned} D1 &= |F_i(x, y, n-1) - F_i(x, y, n+1)| \\ D2 &= |F_i(x-1, y, n) - F_i(x-1, y, n-2)| \\ D3 &= |F_i(x+1, y, n) - F_i(x+1, y, n-2)| \\ D4 &= |F_i(x-1, y, n) - F_i(x-1, y, n+2)| \\ D5 &= |F_i(x+1, y, n) - F_i(x+1, y, n+2)| \end{aligned}$$

The two pixel differences between the fields  $(n, n-2)$  and  $(n, n+2)$  are averaged, allowing additional improvement over the previous methods:

$$\text{motion\_detected} = \max(D1, \frac{D2+D3}{2}, \frac{D4+D5}{2})$$

However, this detector looks for motion in both the forward and backward directions. Thus, it is not immediately clear whether the motion adaptive de-interlacing algorithm should use the previous field or the next field for temporal interpolation. In this case, the median of the pixel in the previous frame, the pixel in the next frame, and the linear interpolation of the adjacent pixels are calculated as [23]:

$$X = \text{med}\left(\frac{A+B}{2}, C, D\right)$$

$A = f(x-1, y, n), B = f(x+1, y, n), C = f(x, y, n-1), \text{ and } D = f(x, y, n+1)$ . It is sufficient in reducing missed detections; some errors occasionally occur, but these are rare and require very fast, periodic motion (Figure 4d).

## 4.2 The Proposed Motion-Based De-interlacer

In this paper, we present a de-interlacing algorithm based motion [24, 25]. The algorithm is optimal in dealing with only three successive fields. Meaning, saving memory, low computational requirements, and fast response are obtained. Based on the motion detector used, the current frame is divided into three segments: static, slow-, and fast-motion. Then, each segment is processed individually. Using the pixel labels:

$$\begin{aligned} B &= f(x-1, y, n), E = f(x+1, y, n), P = f(x, y, n-1), N = f(x, y, n+1), \\ I_b &= f(x-2, y, n-1), J_b = f(x, y-1, n-1), K_b = f(x+2, y, n-1), \\ L_b &= f(x, y+1, n-1), I_n = f(x-2, y, n+1), J_n = f(x, y-1, n+1), \\ K_n &= f(x+2, y, n+1), L_n = f(x, y+1, n+1). \end{aligned}$$

In static segment, just comparing adjacent temporal pixels in the previous and following fields would have relatively close values and have a much different actual value in current pixel. This generally would occur if the current frame contained an image undergoing large motions. Thus, to make sure that the luminance values of vertical neighboring pixels (B & E) are relatively close to the luminance values of the temporary neighboring pixels P and N, the following condition is proposed, to determine if pixels are static or not:

$$\begin{aligned} |lum(P) - lum(N)| &\leq h_1, \text{ and} \\ \left| \frac{lum(P) + lum(N)}{2} - \frac{lum(B) + lum(E)}{2} \right| &\leq h_1 \end{aligned}$$

Then, the static segment is interpolated using the line repetition method. The following condition is proposed to locate the pixels of the low motion:

$$h_1 < (|I_b - I_n| + |J_b - J_n| + |K_b - K_n| + |L_b - L_n|) / 4 \leq h_2.$$

As well, the average of each color-coded pixel pairs are used as an input to 5-tap median filter. According to Weber law stating that the eye is more sensitive to small luminance differences in dark areas rather than in bright ones. The pixel pair that has the lowest luminance sum divided by absolute difference will double as the fifth and final input to the median filter. Finally, the remaining pixels, which are the high motion pixels are likely to have little in common with previous or current fields and are thus interpolated purely in the spatial domain using 5-tap ELA. The results are shown in Figure 5.

## 5. OBJECTIVE QUALITY MEASUREMENTS

For  $MXN$  color progressive and de-interlaced frames  $f(x, y)$  and  $g(x, y)$  respectively, the RMSE, SNR, and PSNR used in objective quality assessments are defined as:

$$RMSE = \sqrt{\frac{1}{MN} \sum_{x=0, y=0}^{M-1, N-1} E^2(x, y)},$$

$$SNR = 10 \log_{10} \left( \frac{1}{(MN) \cdot MSE} \sum_{x=0, y=0}^{M-1, N-1} g^2(x, y) \right),$$

$$PSNR = 10 \log_{10} \left( \frac{255^2}{MSE} \right)$$

where  $E$  is the *difference between the two frames*, MSE is the *mean square error*. In image and video processing, the *peak signal to noise ratio* is a modified version of the SNR and it is widely used [26]. The larger SNR or PSNR is the better the quality of the processed frame. The results for all the algorithms are given in Table 1.

## 6. DISCUSSION AND CONCLUSION

We tested five of the most spatial algorithms used in industry on a number of consecutive frames. The results show how the 5\_tap ELA algorithm is superior to others. Five of the basic temporal algorithms are also tested. The results show the bilinear field interpolation is better than field repetition. The 7-tap VT median filter is better than the 3-tap one. The VT median filters do enhancement to the edges. Several of motion detectors used for de\_interlaced fields are tested as well. The result is better in case of 4-and 5-field cases. Finally, our proposed algorithm based motion detection is also tested. It gives optimistic result However, we think it needs more investigations and modifications; this is what we are working on and will be published elsewhere.

## 7. REFERENCES

1. Adobe Dynamic Media Group, "A Digital Video Primer," <http://www.adobe.com/motion/events/pdfs/dvprimer.pdf>, 2002.
2. Microsoft, "Broadcast-Enabled Computer Hardware Requirements", WinHEC'97, Broadcast Technologies White Paper, pp. 11-12, 1997.
3. J. S. Lim, and B. A. Heng, "Application of Deinterlacing for the Enhancement of Surveillance Video," M.Sc. Thesis, MIT, June 2001.
4. J. Deame, "Motion Compensated De-interlacing: The Key of the Digital Video Transition," SMPTE 141<sup>st</sup> Technical Conference in NY, November 19-22, 1999.
5. F. J. Chang, and S. C. Tai, "A Motion and Edge Adaptive De-interlacing Algorithm," M.Sc. Thesis, National Cheng Kung University, Taiwan, June 2003.
6. G. de Haan, and E. B. Bellers, "De-interlacing—an Overview", Proceedings of IEEE, Vol. 86, No. 9, pp. 1839-1857, September 1998.
7. G. de Haan, and E. B. Bellers, "Deinterlacing of Video Data," International Conference on Consumer Electronics, Vol. 43, No. 3, pp. 819-825, Aug. 1997.
8. G. de Hann, and E.B.Bellers, "Advanced Motion Estimation and Motion Compensated De-interlacing," Soc. of Motion Picture and TV Eng. Journal, pp. 777-786, Nov. 1997.
9. A. Skarabot, G. Ramponi, and D. Toffoli, "Image Sequence Processing for Videowall Visualization," in Proc. IST/SPIE 12<sup>th</sup> Annual Int. Symp. Elec. Imaging 2000, San Jose, California, January 23-28, 2000.
10. A. J. Patti, M. I. Sezan, and A. M. Tekalp, "Robust Methods for High-Quality Stills from Interlaced Video in the Presence of Dominant Motion," IEEE Transaction on Circuits and Systems for Video Technology, Vol. 7, No.2, pp. 328-341, April 1997.
11. C. J. Kuo, C. Liao, C. C. Lin, "Adaptive Interpolation Technique for Scanning Rate Conversion," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 6, No. 3, pp. 317-321, June 1996.
12. D. M. Martinez, and J. S. Lim, "Spatial Interpolation of Interlaced Television Pictures," IEEE International Conference on Acoustics, Speech, and Signal Processing, Volume 3, pp. 1886-1889, 1989.
13. T. Doyle and M. Looymans, "Progressive Scan Conversion Using Edge Information," Signal Processing of HDTV, II, L. Chiariglione, Ed., Elsevier Science Publishers, pp. 711-721, 1990.
14. 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. on Consumer Elec., Vol. 46, pp. 999-1006, Nov. 2000.
15. J. Salo, Y. Nuevo, and V. Hameenaho, "Improving TV Picture Quality With Linear-Median Type Operations," IEEE Transactions on Consumer Electronics, Vol. 34, No. 2, pp. 373-379, Aug. 1988.
16. H. S. Oh, Y. Kim, Y. Y. Jung, A. W. Morales, and S. J. Ko, "Spatio-Temporal Edge Based Median Filtering for De-interlacing," Digest of the International Conference on Consumer Electronics, 2000.
17. T. S. Gunawan, S. S. Tandjung, and C. M. Nang, "An Efficient Parallelization Scheme of Motion Estimation Algorithms on Myrinet-Connected Workstations," 6<sup>th</sup> Int. Conf. on Control, Automation, Robotic and Vision 5-8 Dec. 2000, Marina Mandarin, Singapore.
18. R. Li, B. Zheng, M. L. Liou, "Reliable Motion Detection /Compensation and Its Applications to De-interlacing," IEEE Trans. on Circuits and Systems for Video Technology, Vol.10, No.1, pp. 23:29, Feb. 2000.
19. T. Koivunen, "Motion Detection of an Interlaced Video Signal," IEEE Transactions on Consumer Electronics, Vol. 40, No 3, pp. 753-760, August 1994.
20. C. Lee, S. Chang, and C. Jen, "Motion Detection and Motion Adaptive Pro-Scan Conversion," IEEE Int. Symposium on Circuits and Systems pp. 666-669, 1991.
21. B. Bhatt, et al., "Grand Alliance HDTV Multi-Format Scan Converter," IEEE Trans. on Consumer Electronics, Vol. 41, No. 4, pp. 1020:1031, Nov.1995.
22. The Grand Alliance, "The U.S. HDTV Standard," IEEE Spectrum, Vol. 32, No. 4, pp. 36-45, April 1995.
23. Y. Y. Jung, B. T. Choi, Y. J. Park and S. J. Ko, "An Effective De-interlacing Technique Using Motion Compensated Interpolation," IEEE Trans. on Cons. Elec., Vol. 46, No. 3, pp. 460-464, Aug. 2000.
24. R. Simonetti, S. Carrato, G. Ramponi, and A. P. Fillisan, "De-interlacing of HDTV Images for Multimedia Applications," Signal Processing of HDTV, IV, Elsevier Science Publishers, pp. 765-772, 1993.
25. El-S. A. El-Badawy, N. M. Elfaramawy, and H. M. Rahman, "Adaptive Motion De-interlacing Technique," M. Sc. Thesis, Alexandria Univ., Alexandria, Egypt, 2006.
26. Y. Q. Shi and H. Sun, "Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards," CRC Press LLC, 2000.

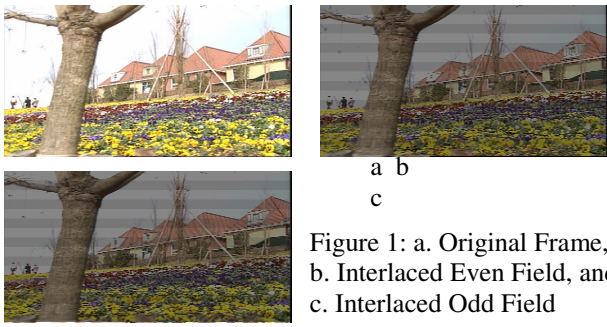


Figure 1: a. Original Frame, b. Interlaced Even Field, and c. Interlaced Odd Field



Figure 3: Temporal Interpolation for the Three Successive Frames in a. Using: b. Interpolated Field, c. Bilinear Field, d. 3-Tap VT Median Filter, e. 7-Tap VT Median Filter, and f. 3D Oriented Edge.

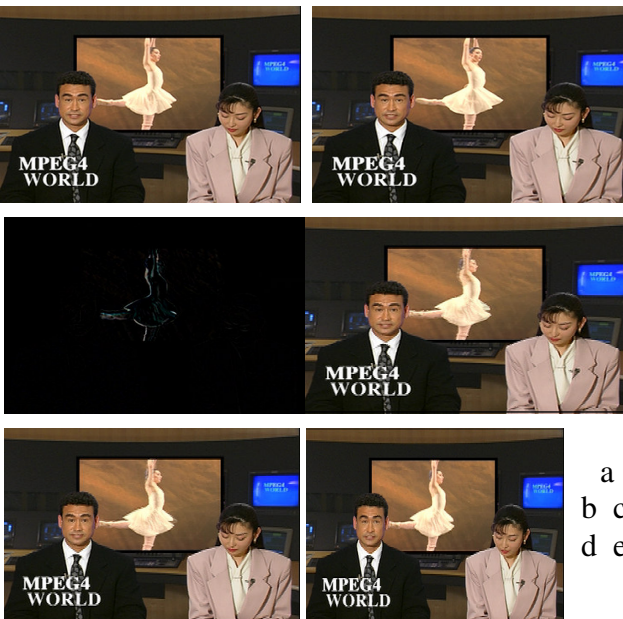


Figure 4: Motion Detectors. a. 4<sup>th</sup> and 5<sup>th</sup> Progressive Frames, b. Difference of 1<sup>st</sup> and 2<sup>nd</sup> Progressive Frames, c. Interpolation Using 3-Field Motion Detector, d. Interpolation Using 4-Field Motion Detector, and e. Interpolation Using 5-Field Motion Detector.

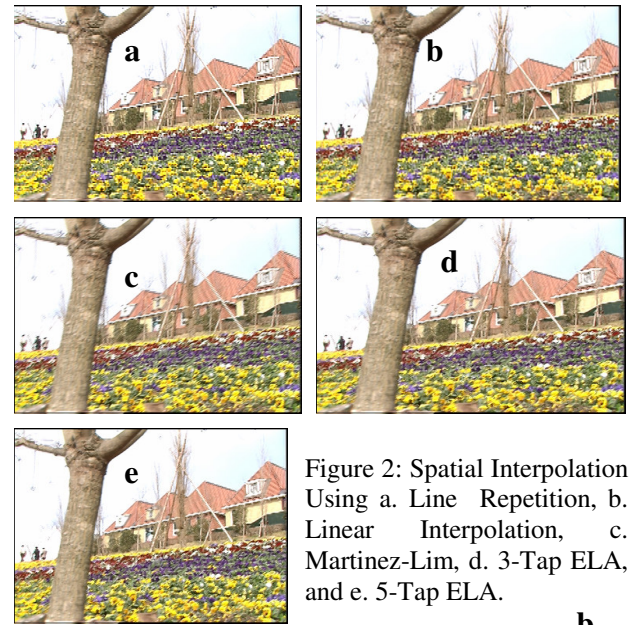


Figure 2: Spatial Interpolation Using a. Line Repetition, b. Linear Interpolation, c. Martinez-Lim, d. 3-Tap ELA, and e. 5-Tap ELA.

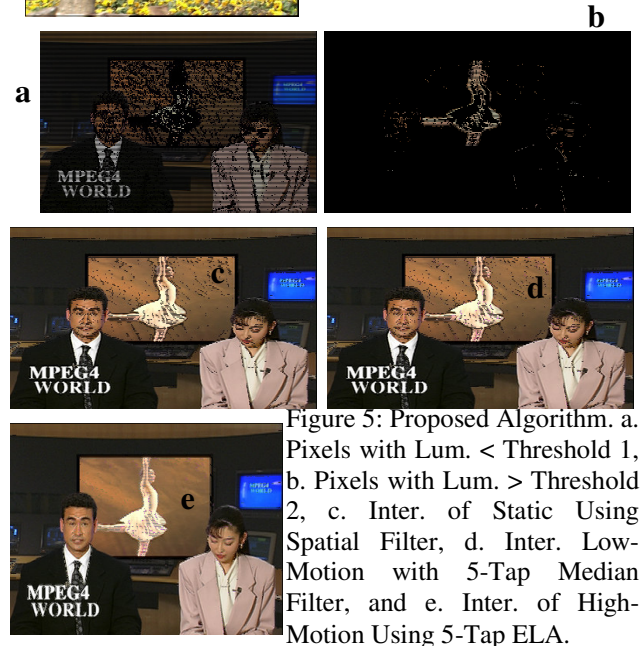


Figure 5: Proposed Algorithm. a. Pixels with Lum. < Threshold 1, b. Pixels with Lum. > Threshold 2, c. Inter. of Static Using Spatial Filter, d. Inter. Low-Motion with 5-Tap Median Filter, and e. Inter. of High-Motion Using 5-Tap ELA.

<i>De-interlaced Algorithm</i>	<i>RMSE</i>	<i>PSNR (dB)</i>	<i>SNR (dB)</i>
<i>Line repetition</i>	5.79	32.88	8.77
<i>Linear interpolation</i>	5.2	33.82	9.69
<i>Martinez &amp; Lim</i>	5.64	33.1	8.98
<i>3-tap ELA</i>	5.55	33.25	9.1
<i>5-tap ELA</i>	5.04	34.09	9.98
<i>Field repetition</i>	1.7	43.51	19.33
<i>Bilinear interpolation</i>	0.75	50.64	26.45
<i>3D edge orientation</i>	3.29	37.78	13.57
<i>3-tap VT median</i>	2.93	38.79	14.59
<i>7-tap VT median</i>	2.91	38.89	15.2
<i>3-field Motion Detection</i>	3.68	36.82	12.58
<i>4-field Motion Detection</i>	0.29	58.85	34.64
<i>5-field Motion Detection</i>	0.28	58.95	34.76
<i>Adapted based motion</i>	2.7	40.22	16.78

Table 1: Objective Quality Assessment for All the Algorithms Discussed In This Paper