Title	Generalized motion and edge adaptive interpolation de-interlacing algorithm
Author(s)	Chung, RHY; Wong, KYK; Chin, FYL; Chow, KP; Yuk, SC
Citation	Wseas Transactions On Computers, 2006, v. 5 n. 11, p. 2544-2551
Issued Date	2006
URL	http://hdl.handle.net/10722/88909
Rights	Creative Commons: Attribution 3.0 Hong Kong License

Generalized Motion and Edge Adaptive Interpolation De-interlacing Algorithm

Ronald H.Y. Chung[1], Kwan-Yee K. Wong[2], Francis Y.L. Chin[3], K.P. Chow[4] and S.C. Yuk[5]
Department of Computer Science
The University of Hong Kong
Pokfulam Rd., Hong Kong Special Administrative Region
Hong Kong
hychung@cs.hku.hk[1], kykwong@cs.hku.hk[2], chin@cs.hku.hk[3], chow@cs.hku.hk[4], scyuk@cs.hku.hk[5]

Abstract: - This paper presents a generalized motion and edge adaptive de-interlacing framework, which offers a structured way to develop de-interlacing algorithm. The framework encompasses many typical de-interlacing algorithms, ranging from simple interpolation based algorithms, to more complex edge dependent and motion adaptive algorithms. Based on this framework, we develop a new de-interlacing algorithm which is efficient and artifacts-free. The proposed algorithm was evaluated by five video sequences, namely, "Akiyo", "Mother and Daughter", "Silent", "Foreman" and "Stefan". Experimental results confirm that the proposed algorithm performs, both objectively and subjectively, much better than other similar algorithms. These promising results indicate that the proposed framework has good potential for realizing even better de-interlacing algorithms.

Key-Words: - De-interlacing Methods, Motion Adaptive Interpolation, Edge Dependent Interpolation.

1 Introduction

Interlaced scanning technique has been exclusively adopted in television (TV) systems since the invention of TV over 70 years ago. It has been widely accepted as a practical technique with reasonable tradeoff among three factors: bandwidth, flicker, and resolution. The present-day technologies in communication and computing, however, are efficient and powerful enough to handle video sequence in the progressive scanning manner. As a result, recent advances in High Definition TV (HDTV) and Personal Computers (PCs) call for progressive scanning. To ensure interoperability between the interlaced scanning format in TV and the progressive scanning format in HDVT and PCs, the need for conversion between the two scanning format is increasing. This process of interlace-to-progressive scanning conversion is called de-interlacing.

An intuitive and trivial way for de-interlacing is to interleave the two consecutive fields back into a progressive frame. Since a time difference exists between the two fields, visual artifacts, such as the most appealing *line crawling* effect at moving edges as shown in Fig. 1, can severely degrade the visual quality of the reconstructed progressive frame. Over the last decade, many de-interlacing algorithms with different computational requirements and corresponding performances have been proposed to improve the visual quality of the de-interlaced progress frame.

De-interlacing algorithms in the literature can be broadly divided into three categories: spatial methods [1-2], motion adaptive methods [3-4], and motion compensation based methods [5-6]. Spatial methods are usually the simplest and the most efficient methods among the three categories of algorithms, which are favorable for hardware implementation. Essentially, spatial methods employ interpolation techniques, and exploit the correlation between vertically neighboring samples in a field when interpolating pixels. The simplest form of these algorithms is *line doubling* (or *line repetition*), which simply replicates the odd field to the even field in reconstructing the progressive frame. In a sense, this is equivalent to upsampling from only the odd field and hence it suffers from aliasing problem. As a result, it also introduces another visual artifact, *jagged edge*, although it can completely remove line crawling artifact. To deal with the aliasing problem, edge dependent interpolation technique [7] can be employed to interpolate the missing pixels from neighboring scan lines, such that the interpolated values are most visually aligned to edge orientations. However, this is applicable only when the edge orientations can be correctly estimated. The computational complexity, unfortunately, usually increases with the correctness of the estimation.

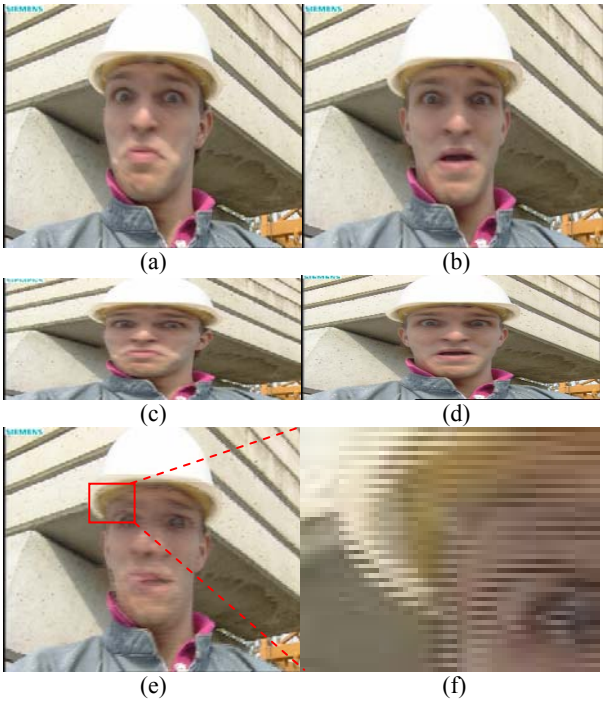


Fig 1. (a) Progressive frame n, (b) Progressive frame $n + 1$, (c) Odd field of progressive frame n, (d) Even field of progressive frame $n + 1$, (e) Reconstructed progressive frame by interleaving the odd and even fields, (f) Enlarged portion in (e) showing line crawling effect.

Motion adaptive methods, on the other hand, make the interpolation adaptive to motion as static regions can never suffer from the line crawling effect. They are considered to be superior to spatial methods in the sense that they preserve vertical resolution by interleaving the odd field and even field for static regions, while they sacrifice vertical resolution by interpolation only for moving regions. However, motion adaptive algorithms suffer from the *switching artifact*, when inaccurate motion detection leads to incorrect decision in switching between the interleaving and interpolation modes.

Motion compensation based algorithms are now being considered as the most advanced de-interlacing algorithms. They employ the concept of motion compensation in video compression to compensate the inter-field motions between the odd and even fields. This requires very accurate motion estimation techniques to generate dense motion field in order to avoid artifacts inherent in motion compensation. This is again a highly computational intensive process, which does not seem to be economical for hardware implementation. As such, various kinds of artifacts can appear in the motion compensated field image due to incorrectly estimated motion field. To rectify this, post-processing such as spatial and temporal filtering are usually required to suppress those artifacts, which further increase the computation burden.

Among these three classes of algorithms, spatial de-interlacing algorithms appear to be the most efficient ones, with inferior visual quality though. Motion compensation based algorithms, on the other hand, appears to be the most sophisticated ones while their computational demand drives them away from hardware implementation, especially when the demand of high resolution videos offsets the technological advancements in computing power and resources. Motion adaptive methods appear to be the most appropriate category of algorithms for de-interlacing as it offers reasonable visual quality with manageable computational requirements. As such, we revisit the problem of de-interlacing following the motion adaptive interpolation approach. Specifically, we first formulate a generalized framework for motion adaptive interpolation de-interlacing methods, and seek ways to suppress the switching artifact that arises from inaccurate motion detection. We derive a new motion and edge adaptive interpolation de-interlacing method based on the framework, utilizing only simple motion and edge detection techniques together with a novel interpolation coefficients adaptation scheme. The proposed algorithm has been tested with five standard test sequences and experimental results confirm that it gives the best objective performance, peak-signal-

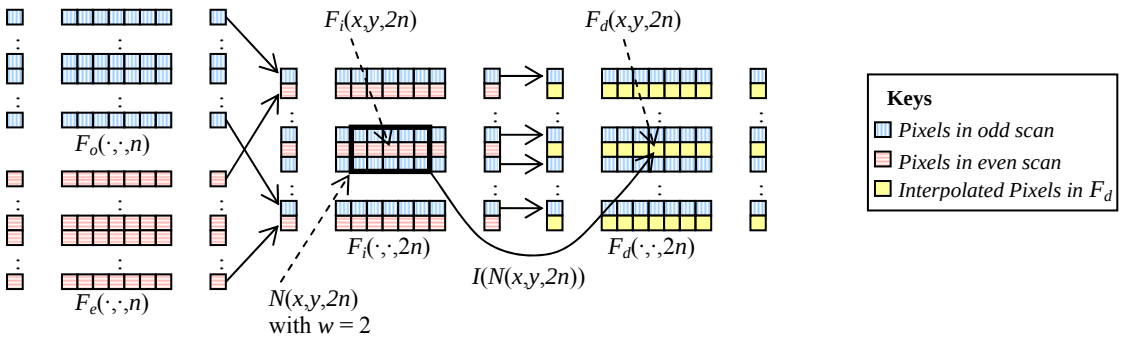


Fig. 2. Relationship between $F_o(\cdot,\cdot,n)$, $F_e(\cdot,\cdot,n)$ and $F_i(\cdot,\cdot,2n)$. $F_d(x,y,2n)$ is reconstructed by interpolating the pixels within $N(x, y, 2n)$ (with $w = 2$ in this example).

to-noise ratio (PSNR) for all the test sequences, when compared with three other similar algorithms. The reconstructed progressive frames (de-interlaced frames) obtained from the proposed algorithm also appear to be artifacts-free with visually best performance.

This paper is organized as follows. Section 2 first presents the generalized framework for motion adaptive de-interlacing methods, followed by Section 3 which details our proposed de-interlacing algorithm. Section 4 provides the experimental results, discussions on the data gathered and the performance comparison of different algorithms. Finally, Section 5 concludes the whole paper.

2 Generalized Framework for Motion and Edge Adaptive Interpolation De-Interlacing Methods

2.1 De-interlacing Problem Statement

Let $F_p(x, y, 2n)$ and $F_p(x, y, 2n + 1)$ be the luminance of the pixel at the spatial coordinate (x, y) in the $2n$-*th* and $(2n + 1)$-*th* frames of a progressive video sequence, respectively. In TV systems, a sequence of progressive frames will first be decomposed into a sequence of alternating odd and even fields, F_o and F_e, respectively, defined as follows:

$$F_o(x, y, n) = F_p(x, 2y, 2n), \quad (1)$$
$$F_e(x, y, n) = F_p(x, 2y + 1, 2n + 1), \quad (2)$$

for $0 \le x < W$ and $0 \le y < \lfloor H/2 \rfloor$, where W and H denote the width and height of the progressive frame, respectively.

Given a flow of field images, an interlaced frame $F_i(x, y, 2n)$ which interleaves the odd and even fields is thus defined as:

$$F_i(x,y,2n) = \begin{cases} F_o(x, \frac{y}{2}, n) & \text{if } y \bmod 2 = 0 \\ F_e(x, \frac{(y-1)}{2}, n) & \text{if } y \bmod 2 \neq 0 \end{cases}. \quad (3)$$

As illustrated above, a sequence of field images is essentially a flow of vertically decimated progressive images with twice the temporal sampling rate of F_i.

With these understandings, the de-interlacing problem can then be formulated as finding some ways to reconstruct a progressive frame $F_d(x, y, 2n)$, from $F_i(x, y, 2n)$, such that it is as close to $F_p(x, y, 2n)$, both subjectively and objectively, as possible.

Although the de-interlacing problem formulated here considers only the luminance component of an image, it is straightforward to extend the same concept in handling images with chrominance components.

2.2 Proposed Motion and Edge Adaptive Interpolation De-interlacing Framework

Motion adaptive interpolation can generally be considered as the problem of interpolating even field samples in $F_d(x, y, 2n)$ from $F_i(x, y, 2n)$ while keeping the odd field samples unaltered. This follows from (1) and (3) which shows that $F_i(x, y, 2n) = F_p(x, y, 2n)$ whenever y is divisible by two. As such, the way for motion adaptive interpolation methods to construct $F_d(x, y, 2n)$ can be generalized as:

$$F_d(x,y,2n) = \begin{cases} F_i(x,y,2n) & \text{if } y \bmod 2 = 0 \\ I(N(x,y,2n)) & \text{if } y \bmod 2 \neq 0 \end{cases}, \quad (4)$$

where $N(x, y, 2n)$ denotes the set of neighboring pixels to the current pixel at spatial coordinates (x, y) in $F_i(\cdot,\cdot,2n)$, and $I(\cdot)$ is the interpolation function that interpolates the missing even scan line pixels in F_d from $N(x, y, 2n)$.

To get rid of severe blurring effect, we propose to limit the number of neighboring pixels to be considered in $N(x, y, 2n)$. In particular, we define it as:

$$N(x,y,2n) = \{F_i(x',y',2n) : |x'-x| \le w, |y'-y| \le 1\}. \quad (5)$$

In a sense, $N(x, y, 2n)$ consists of the luminance values of the pixels that is within a window of size $(2w + 1) \times 3$, centered at (x, y). It limits the neighborhood of the interpolated pixel to the pixels within the current scan line and immediate neighboring scan lines as depicted in Fig. 2.

Suppose we further define $N_{upper}(x, y, 2n)$, $N_{current}(x, y, 2n)$ and $N_{lower}(x, y, 2n)$ as:

$$\begin{aligned} N_{upper}(x,y,2n) &= \{F_i(x', y-1, 2n) : |x'-x| \le w\} \\ &= \left\{F_o\left(x', \frac{y-1}{2}, n\right) : |x'-x| \le w\right\}, \end{aligned} \quad (6)$$

$$\begin{aligned} N_{lower}(x,y,2n) &= \{F_i(x', y+1, 2n) : |x'-x| \le w\} \\ &= \left\{F_o\left(x', \frac{y+1}{2}, n\right) : |x'-x| \le w\right\}, \end{aligned} \quad (7)$$

$$N_{current}(x,y,2n) = \{F_i(x',y,2n) : |x'-x| \le w\} = \left\{F_e(x',\frac{y-1}{2},n) : |x'-x| \le w\right\}. \quad (8)$$

Hence, $N(x, y, 2n) = N_{upper}(x, y, 2n) \cup N_{current}(x, y, 2n) \cup N_{lower}(x, y, 2n)$, which means that $N(x, y, 2n)$ can be separated into three different sets of neighboring pixels. Two sets of which come from the upper and lower scan lines from the odd field, while the remaining comes from the current scan line which maps into the even field. With this formulation, we can then define the interpolation function I in such a way that interpolating pixels from odd fields and even fields are first separately filtered within their set of pixels, motion intensity dependent interpolation can then be applied to these filtered pixels to obtain the interpolated pixels for filling up the even field pixel in $F_d(x, y, 2n)$.

We suggest the interpolation function to be defined like this:

$$I(N(x,y,2n)) = \alpha_u G_u(N_{upper}(x,y,2n)) + (1-\alpha_u-\alpha_l)G_c(N_{current}(x,y,2n)) + \alpha_l G_l(N_{lower}(x,y,2n)), \quad (9)$$

where $G_u(\cdot)$, $G_c(\cdot)$and $G_l(\cdot)$ are the filtering functions for the pixels in the upper, current and lower scan lines respectively; and α_u, α_l are the interpolation coefficients that can vary according to the motion intensity estimated at the current pixel (x, y). Note that $G_u(\cdot)$, $G_c(\cdot)$ and $G_l(\cdot)$ can be customized in such a way that they are edge dependent to realize edge dependent interpolation scheme.

With this framework, it is possible to derive a number of de-interlacing algorithms with different characteristics, as will be described in the following sub-section.

2.3 Mapping of Typical Algorithms into the Proposed Framework

It can be shown that, most of the typical de-interlacing algorithms can actually be mapped successfully into the proposed framework.

2.3.1 Line Doubling Algorithm (LDA)

Line doubling algorithm (LDA) can be realized with the following settings according to the proposed framework:

$\alpha_u = 1$, $\alpha_l = 0$, $w = 0$ s.t. $N_{upper}(x, y, 2n) = F_i(x, y - 1, 2n)$; G_u is an all pass filter s.t. $G_u(N_{upper}(x, y, 2n)) = F_i(x, y - 1, 2n)$

2.3.2 Line Averaging Algorithm (LAA)

Line Averaging Algorithm (LAA), which is also known as simple interpolation algorithm, interpolates the missing pixels in the even scan line from the pixels in immediate upper and lower odd scan lines. This can be realized by setting $\alpha_u = \alpha_l = 0.5$, $w = 0$ s.t. $N_{upper}(x, y, 2n) = F_i(x, y - 1, 2n)$ and $N_{lower}(x, y, 2n) = F_i(x, y + 1, 2n)$; G_u and G_l are all pass filter s.t. $G_u(N_{upper}(x, y, 2n)) = F_i(x, y - 1, 2n)$ and $G_l(N_{lower}(x, y, 2n)) = F_i(x, y + 1, 2n)$. From these settings,

$$I(N(x,y,2n)) = \frac{F_i(x,y-1,2n) + F_i(x,y+1,2n)}{2}, \quad (10)$$

which is equivalent to the averaging operation in LAA.

2.3.3 Motion Detection based Interpolation (MDI)

By making α_u, α_l, and α_c adaptive to motion detection result, motion adaptive interpolation de-interlacing can be realized. For instance, when motion intensity is large, the interpolation function I should acts like line averaging filter to get rid of the line crawling artifact, whereas when motion intensity is low, the interpolation function I should preserve as much detail as possible in $F_i(x, y, 2n)$ for higher vertical resolution. A simple way to do this is to formulate α_u, α_l, and α_c as follows:

$$\alpha_u = \alpha_l = \begin{cases} 0.5 & if\ MD(x,y,2n) \ge Threshold_{motion} \\ 0 & otherwise \end{cases}, \quad (11)$$

where $MD(x, y, 2n)$ is a scalar obtained from the motion detector at pixel (x, y) in frame $2n$. The value of this scalar increases with motion intensity, and $Threshold_{motion}$ is the threshold for differentiating high intensity motions from lower ones.

With these interpolation coefficients defined, the interpolation function will switch between averaging operation and interleaving function according to motion intensity.

When working together with the following settings: $w = 0$ s.t. $N_{upper}(x, y, 2n) = F_i(x, y - 1, 2n)$, $N_{lower}(x, y, 2n) = F_i(x, y + 1, 2n)$, $N_{current}(x, y, 2n) = F_i(x, y, 2n)$; G_u, G_c and G_l are all pass filter s.t. $G_u(N_{upper}(x, y, 2n)) = F_i(x, y - 1, 2n)$, $G_c(N_{current}(x, y, 2n)) = F_i(x, y, 2n)$, $G_l(N_{lower}(x, y, 2n)) = F_i(x, y + 1, 2n)$, a motion detection based interpolation de-interlacing algorithm can be realized.

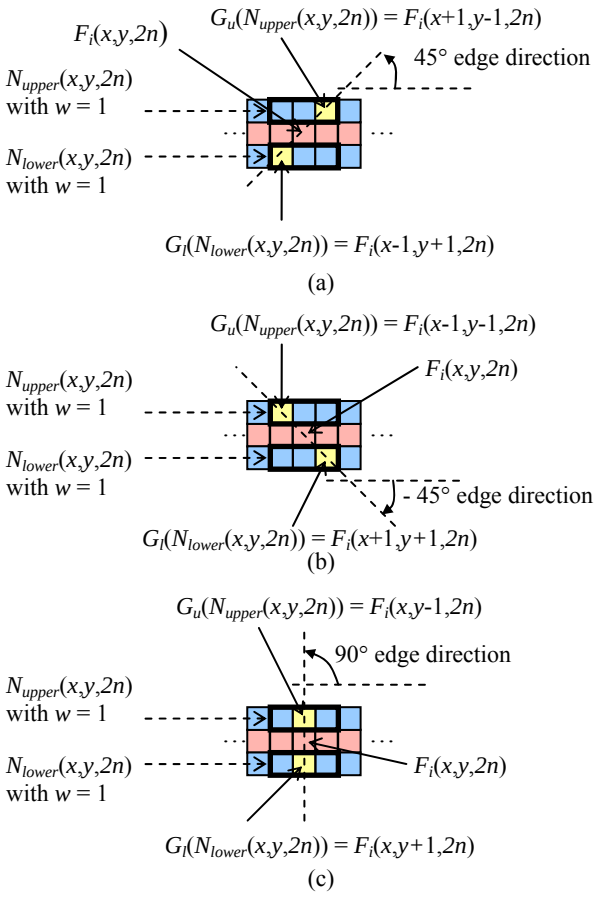


Fig. 3. Example settings for edge dependent interpolations (a) 45° directional interpolation, (b) -45° directional interpolation, and (c) 90° directional interpolation.

2.3.4 Edge Dependant Interpolation (EDI)

Edge dependant interpolation de-interlacing algorithms are essentially directional interpolation method to preserve edge directions. These algorithms estimate the direction of the edge of the interpolated pixel, and perform interpolation along the edge direction to reduce jagged edge artifact. Under the proposed framework, this can be realized by setting G_u and G_l to be selective filters as depicted in Fig. 3.

As illustrated above, many typical de-interlacing algorithms can be mapped into our framework according to different parameter settings. There are actually numerous ways to play around with different settings to come up with various kinds of de-interlacing algorithms, indicating the generality of the proposed framework. In the next section, we will propose a new de-interlacing algorithm.

3 Proposed De-interlacing Algorithm

The new de-interlacing algorithm proposed in this section addresses the problem of switching artifact in motion detection based interpolation algorithm. We believe that the origin of switching artifact comes from incorrect decisions made in switching between the interpolation and interleaving modes, especially when such decision making is merely based on thresholding operation as stated in (11). Although adaptive or multilevel thresholding techniques might help to reduce the number of undesirable artifacts, erroneous detections of motion are not completely avoidable. Hence, instead of relying on motion detector for making binary decision in mode switching, we propose to adapt the interpolation coefficients according to the motion intensity to enable smooth transition between the interpolation and interleaving modes. To do this, we first state the requirements for coefficients adaptation.

3.1 Coefficients Adaptation Requirements

Assume a motion detector return a scalar $MD(x, y, 2n)$, with values falling within $[0, \infty)$, which represents the motion intensity for a pixel at spatial position (x, y) in the $2n$-th interlaced frame, the interpolation coefficients α_u and α_l should satisfy the following constraints:

(I) $\lim_{MD(x,y,2n)\to 0} \alpha_u = 0, \lim_{MD(x,y,2n)\to 0} \alpha_l = 0$

(II) $\lim_{MD(x,y,2n)\to \infty} \alpha_u = \frac{1}{2}, \lim_{MD(x,y,2n)\to \infty} \alpha_l = \frac{1}{2}$

Constraint (I) ensures that the interpolation coefficients will approach to the values that correspond to the interleaving operation for static regions, while constraint (II) ensures that the coefficients will result in interpolation operation for those regions with fast motion.

3.2 Proposed Interpolation Coefficients

We suggest to use the following coefficients for the proposed de-interlacing algorithm

$$\alpha_u = \alpha_l = \frac{MD^2(x,y,2n)}{2MD^2(x,y,2n) + T^2}. \quad (12)$$

In this way, the two constraints presented in Section 3.1 can be satisfied. The parameter T is a configurable parameter that controls the sensitivity of the coefficients to the motion intensity. The relation of T to α_u, α_l can be best illustrated in Fig. 4.

As shown in Fig. 4, the coefficients α_u and α_l increase with motion intensity. The parameter T controls the rate of increase of α_u and α_l, where a larger T indicates a smaller sensitivity of α_u and α_l to the increase in the motion intensity. With α_u and

α_l defined this way, the interpolation coefficient for the current scan line becomes:

$$1-\alpha_u-\alpha_l = 1-\frac{2MD^2(x,y,2n)}{2MD^2(x,y,2n)+T^2} = \frac{T^2}{2MD^2(x,y,2n)+T^2} \tag{13}$$

which indicates that the contribution from the current scan line will decreases with the motion intensity.

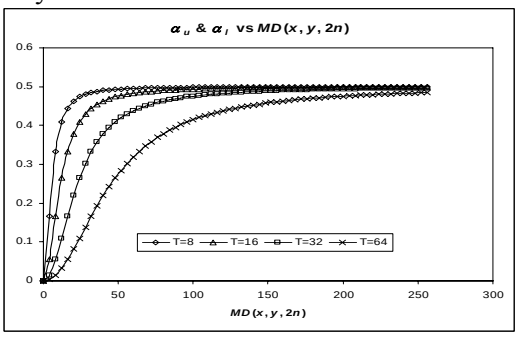


Fig. 4. α_u, α_l vs $MD(x, y, 2n)$ for different values of T.

By this formulation of interpolation coefficients, the algorithm blends the interpolation with interleaving results according to the motion intensity instead of abrupt switching between the two modes of operation. This can help to remove the switching artifact that is sensitive to human vision systems.

3.3 Motion Detection

To illustrate the robustness of the proposed framework and to ensure efficient operations, we employ the simplest form of motion detector in our algorithm. Specifically, the motion intensity is defined as the mean-absolute-difference (MAD) over a 3x3 window from the previous interlaced frame, which is defined as:

$$MAD(x,y,2n) = \frac{1}{9}\sum_{i=-1}^{1}\sum_{j=-1}^{1}\left|\begin{matrix}F_i(x+i,y+j,2n)\\ -F_i(x+i,y+j,2(n-1))\end{matrix}\right|. \tag{14}$$

In order to make the motion detector less sensitive to noises and errors, we smooth the motion intensity along the temporal domain and thus the actual motion intensity function is defined as follows:

$$MD(x,y,2n) = \begin{cases} MAD(x,y,2n) & if \quad MAD(x,y,2n) \geq MD(x,y,2(n-1)) \\ \dfrac{MAD(x,y,2n)+MD(x,y,2(n-1))}{2} & otherwise \end{cases} \tag{15}$$

Defined in this way, the motion intensity function will be responsive to sudden increase in motion, while smoothing out erroneous detection for slow movement which MAD sometimes fails to catch.

3.4 Edge Dependent Interpolation

We also incorporate the concept of edge dependent interpolation in our proposed algorithm. We adapt the filtering functions G_u and G_l based on the three edge orientations (-45°, 45°, 90°) as we have mentioned in Section 2.3.4. We employ the method in [7] for edge orientation estimation.

To this end, we have our new de-interlacing algorithm completely defined.

4 Experimental Results

We evaluated the performance of our algorithm on five video sequences, namely "Akiyo", "Mother and Daughter", "Silent", "Foreman" and "Stefan". These sequences are chosen because they represent different classes of motions, which give a complete evaluation of the algorithm under different scenarios. Fig. 5 shows one representative frame for each sequence. "Akiyo" is a sequence with almost completely static background and very slow head and shoulder motions. From this sequence, we can evaluate how well our algorithms preserve the details in static background. "Mother and Daughter" and "Silent" are also sequences with static background, but with faster movements in foreground objects. In particular, "Silent" sequence contains fast hands and fingers movements, which can trigger switching artifacts. Finally, "Foreman" and "Stefan" are sequences with large foreground and camera panning motions. As such, jagged edge artifact can easily appear in de-interlaced frame as the motion adaptive interpolation filter tends to reduce the vertical resolution, which induces aliasing problem. Each test sequence consists of 300 progressive frames, and we extracted odd and even fields in alternating frames and interleaved the two fields to produce 150 interlaced frames. By doing so, the quality of the de-interlaced frames can be evaluated by the objective measure, Peak-Signal-to-Noise Ratio (PSNR), where the corresponding progressive frames can serve as the ground truth for PSNR calculations.

The performance of our algorithm was evaluated against three other algorithms, namely Line Doubling Algorithm (LDA), Line Averaging Algorithm (LAA), and Motion Detection Based Interpolation (MDI). To make a fair comparison and

(a) (b) (c) (d) (e)

Fig. 5. Representative frame in each sequence (a) Akiyo, (b) Mother and Daughter, (c) Silent, (d) Foreman, (e) Stefan.

to illustrate the effectiveness of our coefficients adaptation scheme, we also incorporated the same edge dependent interpolation (EDI) to both LAA and MDI. In addition to that, we employed the same motion detector in both the MDI and our algorithm, which is defined in (15). The parameters T and $Threshold_{motion}$ for the proposed algorithm and MDI are both set to 32.

Table 1 summarizes the average PSNR improvements of each algorithm, over the corresponding interlaced sequence, for each test sequence. It can be seen from the table that the newly proposed algorithm achieved the best PSNR improvements. The ranking for all the algorithms according to the PSNR improvements is consistent among all the test sequences, with MDI being the second best algorithm, while LAA got the worst performance. For sequences with static background such as "Akiyo", "Mother and Daughter", and "Silent", we can see that LAA and LDA did not have PSNR improvement at all over the corresponding interlaced sequences, indicating that they were not objectively better after the de-interlacing operations. In particular, it is not surprising that LDA had the worst performance because it did not take edge orientation into account for de-interlacing. As for our proposed algorithm and MDI, their better performance can be justified by that fact that they preserved as much details as possible in static area, while performing the necessary interpolation operations only for those moving regions.

For sequences "Foreman" and "Stefan", in which fast foreground and camera panning motions dominate, LAA and LDA did show positive average PSNR improvements because virtually all pixels in the even field need interpolation, which is inline with the strategy of LAA and LDA. It is interesting to note that LAA performed better than MDI for the foreman sequence, indicating MDI might suffer from switching artifact, which we will describe later. Our proposed algorithm and MDI still performed well for these two sequences, indicating that the incorporation of motion information for de-interlacing can help boosting up the video quality.

Table 1: Average PSNR Improvement of each algorithm for the five test sequences, over the corresponding interlaced sequences.

Seq. / Algorithm	Proposed	MDI	LAA	LDA
Akiyo	0.581	0.078	-3.989	-9.921
Mother and Daughter	3.270	1.709	-2.386	-7.423
Silent	6.478	5.359	-0.802	-4.679
Foreman	5.550	4.612	4.817	0.827
Stefan	6.001	5.799	5.552	2.165

Fig. 6(c) to 6(f) show the de-interlaced results of a frame in the "Foreman" sequence for each algorithms. The progressive frame and the interlaced frame, shown in Fig.6 (a) and 6(b) respectively, are also included for subjective evaluation. From Fig. 6(b), it shows that there is slow camera panning motion in this frame as indicated by the small movements in the background, while there are small movements in the facial and head regions and fast movements of fingers. LDA suffered severely from jagged edge artifact as depicted in Fig 6(c), while LAA performed significantly better due to the edge dependent interpolation scheme as illustrated in Fig. 6(d). LAA did not suffer from switching artifacts, and generate quite visually pleasant de-interlaced frames. However, it cannot preserve the details in static regions, notably in the text overlay regions in the top-left corner of the image. MDI, on the other hand, does not suffer much from jagged edge artifact, and the characters "SIEMENS" in the top-left corner of the image is clearly visible, indicating its ability to switch between interpolation and interleaving mode. However, it suffered from switching artifact, which is noticeable in the eyes, mouth and fingers regions. The de-interlaced frame from the proposed algorithm appears to be the best, in the sense that it correctly preserves the static text overlay regions, and suppresses unwanted switching artifacts for moving regions. This shows that the proposed coefficients adaptation scheme presented in Section 3.2 works well enough to enable smooth transition from interpolation to interleaving mode, and vice versa.

5 Conclusion

In this paper, a generalized motion and edge adaptive interpolation de-interlacing framework is presented. The framework is general in the sense that many typical de-interlacing algorithms can be successfully mapped into the framework, while it enables a structured way for algorithmic tuning. Based on this framework, a new motion and edge adaptive de-interlacing algorithm has also been proposed. Although the new algorithm only employs the simplest form of motion detection and edge orientation estimation methods, it enables smooth transition from interpolation to interleaving mode based on a novel interpolation coefficients adaptation scheme. Experimental results show that the proposed algorithm has the best objective performance as indicated in average PSNR improvements, while offering the visually best de-interlaced frames with no noticeable artifact when compared with similar algorithms. Besides, due to its simplicity, this algorithm is computationally efficient, which is a plus for hardware implementation. Future directions will be focused on improving the interpolation coefficients adaptation scheme, and incorporating more sophisticated motion detection and edge orientation methods to achieve a better de-interlace algorithm.

References:

[1] M. Byun, M.K. Park, and M.G. Kang, "EDI-based deinterlacing using edge patterns," in *Proc. ICIP05*, Genoa, Italy, Sep. 2005, pp. 1018-1021.

[2] S. H. Hong, R. H. Park, S. Yang, and J.Y. Kim, "Edge-preserving spatial deinterlacing for still images using block-based region classification," in *2006 Digest of Technical Papers Int. Conf. Consumer Electronics*, Las Vegas, NV, USA, Jan. 2006, pp. 85-86.

[3] S.F. Lin, Y.L. Chang, and L.G. Chen, "Motion adaptive interpolation with horizontal motion detection for deinterlacing," in *IEEE Trans. on Consumer Elec.*, vol. 49, no. 4, Nov. 2003, pp. 1256-1265.

[4] S.C. Tai, C.S. Yu, and F.J. Chang, "A motion and edge adaptive deinterlacing algorithm," in *Proc. ICME2004*, Taipei, Taiwan, Jun. 2004, pp. 659-662.

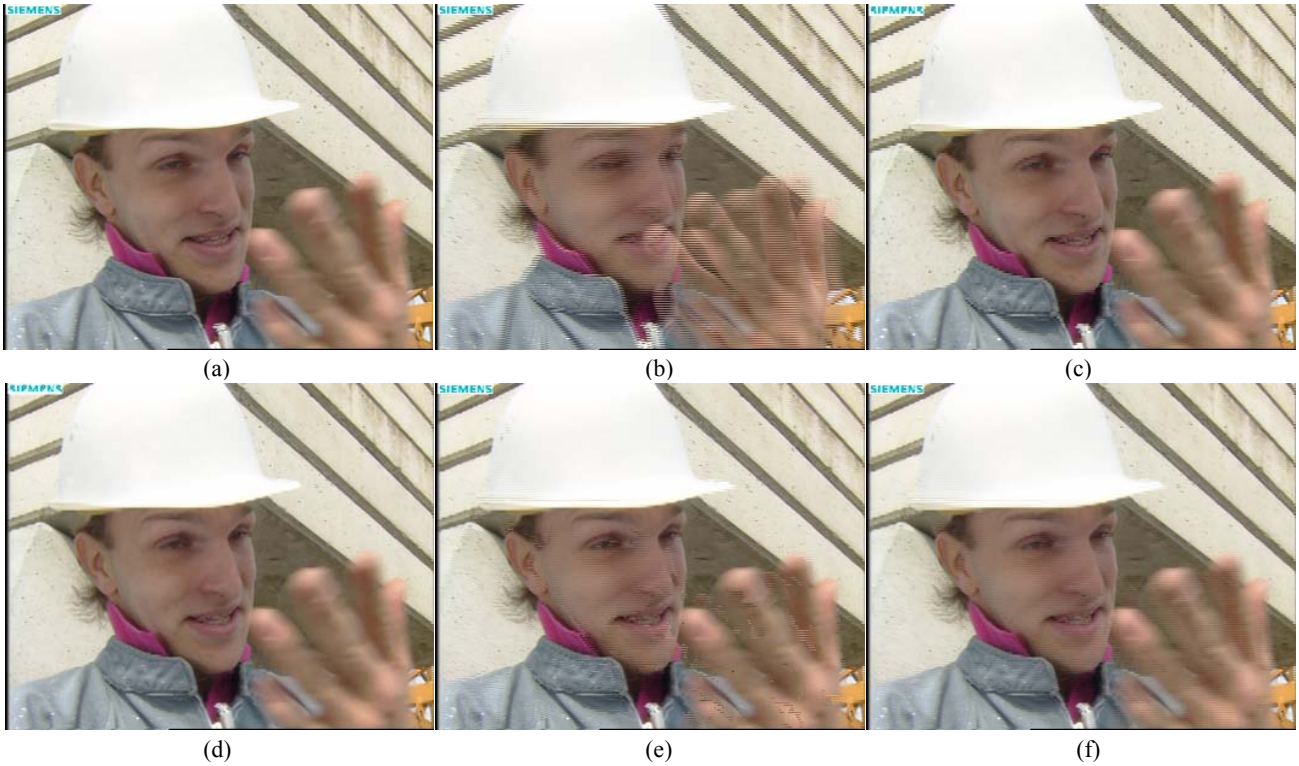
[5] A.M. Tourapis, O.C. Au, and M.L. Liou, "Advanced de-interlacing techniques with the use of zonal based algorithms," in *Proc. VCIP-2001*, San Jose, CA, Jan. 2001, pp. 948-958.

[6] X. Gao, J. Gu, and Jie Li, "De-interlacing algorithms based on motion compensation," in *IEEE Trans. on Consumer Elec.*, vol. 51, no. 2, May 2005, pp. 589-599.

[7] T. Doyle and M. Looymans, "Progressive scan conversion using edge information," in *Signal Processing of HDTV II*, L. Chiariglione, Ed. Amsterdam, The Netherlands: Elsevier, 1990, pp. 711-721.

Acknowledgment

This research was jointly sponsored by Multivision Intelligence Surveillance Limited and the Innovation and Technology Commission of the Government of the Hong Kong Special Administrative Region, under the Grant UIM/167.



(a) (b) (c)

(d) (e) (f)

Fig. 6. De-interlaced frames comparison: (a) Original progressive frame, (b) Interlaced frame, (c) De-interlaced frame by LDA, (d) De-interlaced frame by LAA, (e) De-interlaced frame by MDI, (f) De-interlaced frame by proposed algorithm.