A Fuzzy Motion Adaptive Algorithm for Interlaced-to-Progressive Conversion

P. Brox **I. Baturone** **S. Sánchez-Solano**
Instituto de Microelectrónica de Sevilla - CNM – CSIC
Avda. Reina Mercedes S/N. Edificio CICA
41012 Sevilla (SPAIN)
e-mail: brox@imse.cnm.es

Abstract

Interlaced-to-progressive algorithms are currently required by video format conversion systems in order to display a progressive scanning used in modern visualization equipments. De-interlacing algorithms use interpolation techniques to calculate missing pixels in transmitted fields. A motion adaptive algorithm which employs fuzzy logic to adapt the interpolation strategy to the presence of motion in the images is proposed in this paper. The performance of this new approach is evaluated by extensive simulation of different video sequences.

Keywords: Motion adaptive, de-interlacing, fuzzy inference systems.

1 Introduction

Motion detection is crucial to many fundamental tasks in image processing (such as de-interlacing [1] or picture rate-up conversion [2]) which resort to the interpolation of image sequence data to increase the vertical resolution of the image (de-interlacing) or the number of pictures which compose the video sequence (rate-up conversion). Motion adaptive interpolation techniques provide efficient solutions for this kind of problems because they allow to apply different interpolation algorithms in the static and dynamic parts of the images. Obviously, their performance relies strongly upon the quality of motion detection schemes.

Motion detectors basically evaluate the difference between pixels in consecutive pictures to make a decision. However, due to noise and vertical details, this value may not be a good measurement. To increase the robustness of motion detectors, several proposals have been described in the literature. Some examples are the use of a low-pass filter to reduce fluctuations of the values near edges, or the linear combination of several detector outputs [2].

Fuzzy logic has also been applied to detect motion in format conversion systems which take advantage of its interpolation capability to obtain new data in areas where the decision is not trivial. Techniques described in [3] and [4] propose fuzzy motion adaptive algorithms for de-interlacing.

Interlaced format was introduced to halve the required bandwidth in current TV systems (NTSC, PAL). It consists in transmitting fields with the half of the lines instead of the whole frames. At the receiver side, a de-interlacing (or interlaced-to-progressive conversion) algorithm reconstructs the missing lines applying interpolation techniques. Figure 1 illustrates this process. The advent of HDTV systems, high

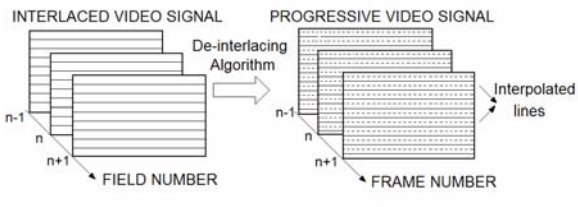

Figure 1: De-interlacing task

quality monitors, displays and projectors has increased the need for de-interlacing algorithms in the last few years.

Linear de-interlacing techniques, which always perform the same kind of interpolation between pixels, have been widely applied. Among them, temporal algorithms (such as field insertion) exploit the correlation in the time domain achieving good results in static areas but introducing very annoying effects in moving areas of the image. Spatial interpolation algorithms (such as line doubling or line average) present as main advantage their low implementation cost, since memory is not required to store previous fields. However, they introduce blurring and stairs-case effect in vertical details and edges. Theoretically, a linear combination of both techniques should provide the best results. To provide it, motion adaptive interpolation techniques were introduced.

A novel motion adaptive interpolation algorithm is proposed in this paper. It uses a fuzzy inference system to decide the most convenient interpolation according to the presence of motion. The paper is organized as follows. Section 2 includes the description of the algorithm and its application to de-interlace video sequences. The performance of the new algorithm, compared with others of similar complexity in terms of memory requirements, is evaluated in Section 3. Finally some conclusions are given in Section 4.

2 Fuzzy Motion Adaptive Algorithm

The idea of motion adaptive algorithms for de-interlacing was originally described in [5]. It basically consists in using two different interpolators, one for static areas and another one for moving areas. The main novelty of our proposal is to use three instead of two kinds of interpolators depending on the level of motion in the picture. This is carried out by using a fuzzy logic-based inference system to apply the following heuristic knowledge in order to evaluate the missing pixels in a field:

1) If motion is small then the best option is to use information from previous fields performing a temporal filtering (I_T) as interpolation method.

2) If motion is large then the best option is to use information from the current field performing a spatial filtering (I_S) as interpolation method.

3) In other cases, the motion is medium and a linear combination of the spatial and temporal filtering will be the best option.

These rules, summarized in Table 1, allow improving the results of conventional motion detectors providing smooth transactions between the three interpolators. To achieve it, the fuzzy sets illustrated in Figure 2 are used to represent the concepts "SMALL", "MEDIUM", and "LARGE", instead of threshold values.

Table 1: Fuzzy rule set of the proposed algorithm

if	*motion (x,y,t)*	*then*	*I(x,y,t)*
	SMALL		$c_1 = I_T(x,y,t)$
	LARGE		$c_2 = I_S(x,y,t)$
	MEDIUM		$c_3 = \gamma\, I_T(x,y,t) + \lambda\, I_S(x,y,t)$

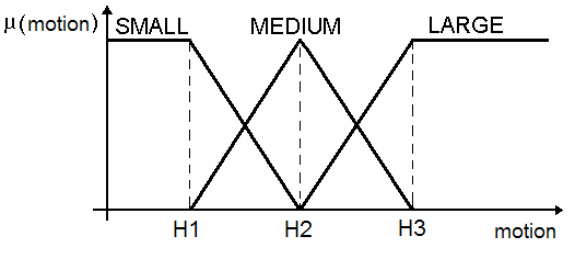

Figure 2: Membership functions for the fuzzy motion adaptive system

The input of the system is the bi-dimensional convolution of the difference of luminances, H_{ij}, given by the following expression:

$$motion(x,y,t) = \sum_{i=1}^{3}\left(\sum_{j=1}^{3} H_{ij} C_{ij}\right) \quad (1)$$

$$H = \begin{bmatrix} H(x-1,y-1,t-1) & H(x-1,y,t) & H(x-1,y+1,t-1) \\ H(x,y-1,t-1) & H(x,y,t) & H(x,y+1,t-1) \\ H(x+1,y-1,t-1) & H(x+1,y,t) & H(x+1,y+1,t-1) \end{bmatrix}$$

$$C1 = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad H(x,y,t) = \frac{|I(x,y,t-1) - I(x,y,t+1)|}{2}$$

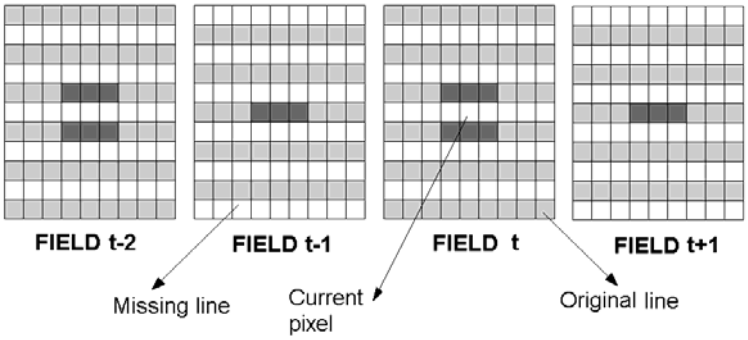

Figure 3: Pixels in dark grey are taking part in bi-dimensional convolution

where x and y are the spatial coordinates of the processed pixel in a frame, and t determines the order of the field in the sequence.

Bi-dimensional convolution is very suitable to measure motion since it considers the spatial and temporal neighborhood of the current pixel. Besides, it provides high flexibility because convolution weights, C_{ij}, allow giving more priority to nearest pixels in the neighborhood. Expression (1) shows one of the bi-dimensional convolution windows which have been used. Considering this expression, pixels (in dark grey) shown in Figure 3 are taking part in the bi-dimensional convolution. A study using convolution windows with different sizes is presented in Section 3.

The luminance component of the interpolated pixel is calculated applying the Fuzzy Mean defuzzication method as follows:

$$I(x,y,t) = \frac{\sum_{i=1}^{3} \alpha_i(x,y,t) \cdot c_i(x,y,t)}{\sum_{i=1}^{3} \alpha_i(x,y,t)} \qquad (2)$$

where α_i are the activation degrees of each rule.

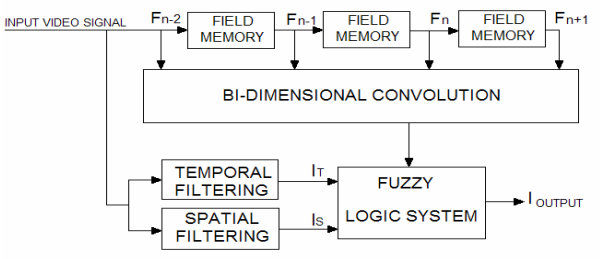

Figure 4: Block diagram of the fuzzy motion adaptive algorithm

Substituting the consequents, c_i (see Table 1), and applying that $\alpha_1+\alpha_2+\alpha_3$ is always equal to 1, the above expression can be given as:

$$I(x,y,t) = I_T(x,y,t) \cdot (\alpha_1(x,y,t) + \gamma \cdot \alpha_3(x,y,t)) + I_S(x,y,t) \cdot (\alpha_2(x,y,t) + \lambda \cdot \alpha_3(x,y,t)) \qquad (3)$$

Figure 4 shows the block diagram of the fuzzy motion adaptive algorithm. According to (3), the algorithm applies a temporal filtering if the bi-dimensional convolution of the difference of luminances is really small (α_1 is equal to 1 and the rest of α_i are 0). It performs a spatial filtering if the motion level is really large (α_2 takes the value 1 and the others α_i are 0). Otherwise two rules are activated and a non-linear combination between two of the three consequents is applied.

From the description of the fuzzy system, different threshold values H1, H2 and H3 are used in the descriptions of membership functions for "SMALL", "MEDIUM" and "LARGE" (see Figure 2). Regarding the consequents of the fuzzy rule set, the performance of the fuzzy motion adaptive algorithm also depends on the parameters γ and λ, which determine the third interpolator function as a linear combination of the intra-field (I_S) and inter-field (I_T) method. Despite there is no restriction to determinate these five values, some of them achieve better results than the other ones. In order to estimate these values, a set of input/output training patterns from progressive video sequences is used to minimize an error function between the original values (obtained from the progressive video sequences)

Table 2: Description of the fuzzy system rule set with Xfuzzy3.0

if	*motion (x,y,t) and I_S(x,y,t) and I_T(x,y,t)*	*then*	*I(x,y,t)*
	SMALL and DUMMY and DUMMY		$c_1 = I_T(x,y,t)$
	LARGE and DUMMY and DUMMY		$c_2 = I_S(x,y,t)$
	MEDIUM and DUMMY and DUMMY		$c_3 = \gamma I_T(x,y,t) + \lambda I_S(x,y,t)$

and the interpolated ones. This is carried out performing a supervised learning algorithm.

In order to realize it, the development environment Xfuzzy 3.0 is used [6]. It is a whole environment for designing fuzzy sets that is composed of a set of CAD tools covering the different stages of description, verification, simplification and synthesis of inference systems based on fuzzy logic. Xfuzzy 3.0 is free software and it can be downloaded from the web page: http://www.imse.cnm.es/Xfuzzy.

Xfuzzy 3.0 integrates a CAD tool, *xfsl,* to tune fuzzy systems described in XFL (the specification language in Xfuzzy) [7]. Considering that rule consequents could be described as linear functions of there input variables (I_T, I_S and *motion*), the fuzzy system has been performed within Xfuzzy as a first-order Takagi-Sugeno system. To achieve it, the rule set in Table 1 has been translated into the equivalent one shown in Table 2, where the membership function called "DUMMY" returns a value of one independently of the input value. This is necessary to include the input variables I_T and I_S in the antecedents of the rule set. Figure 5 illustrates the graphical user interface of the CAD tool *xfedit* within Xfuzzy 3.0 which eases the descriptions of the rule set.

xfsl allows the user different learning algorithms as well as tuning only specific parameters of the system and a selecting criterion to stop the process. In particular, the well-known Marquardt-Levenberg algorithm is chosen and the parameters H1, H2, H3, λ and γ are enabled to participate in the tuning process. Figure 6 shows the evolution of three error functions along the learning process after eleven

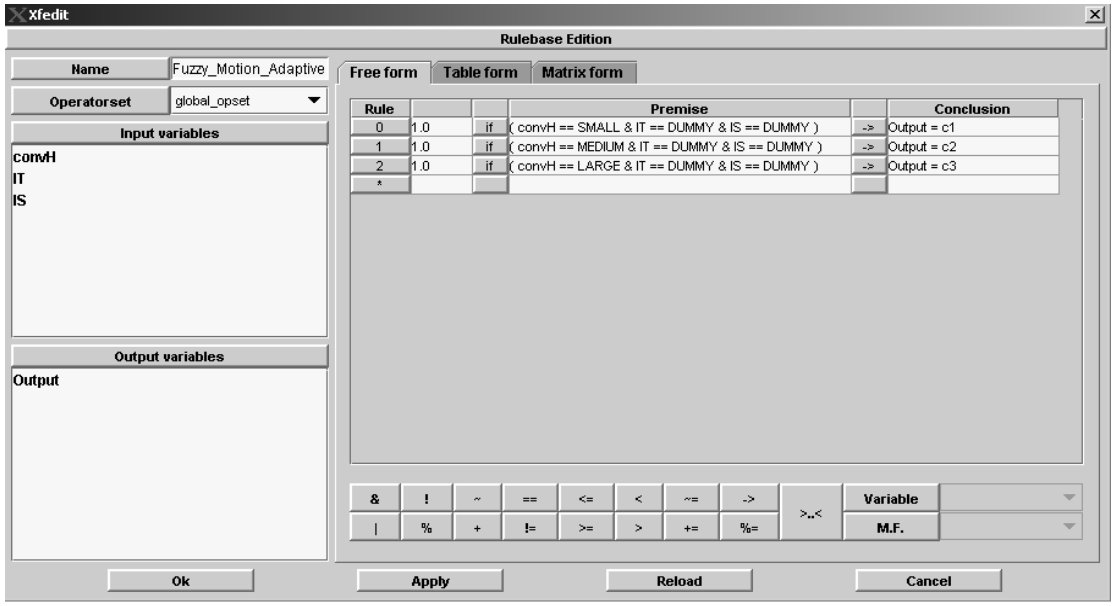

Figure 5: Description of the fuzzy rule set using *xfedit* within Xfuzzy 3.0

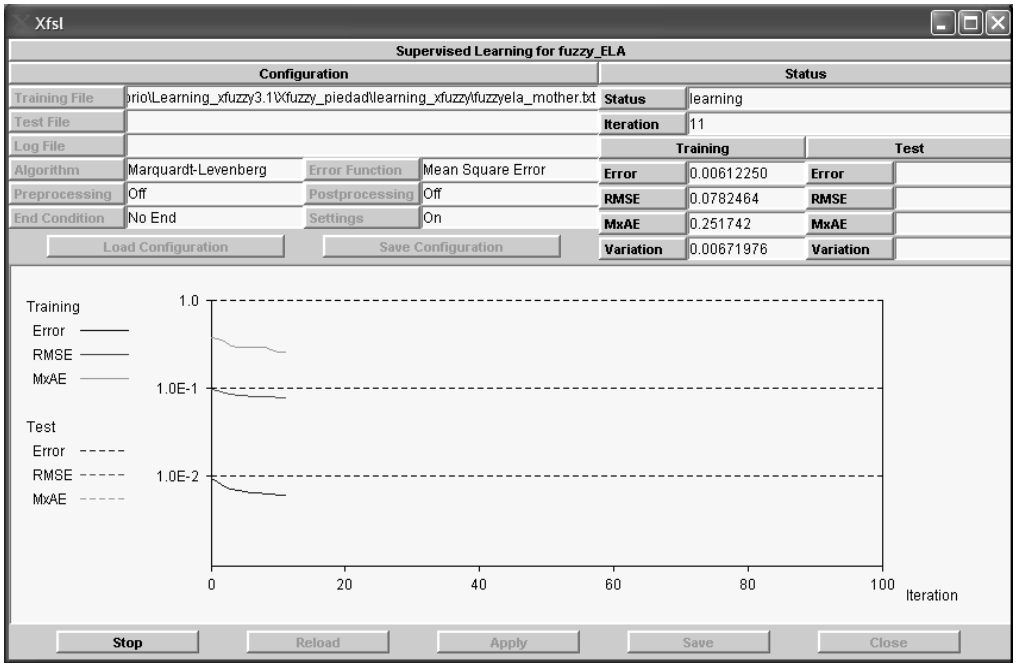

Figure 6: Evolution of the tuning process

iterations.

Comparing with other fuzzy motion adaptive algorithms, our proposal reduces considerably the computational complexity of the method in [3] (it will be proved in the Section 3). The bi-dimensional convolution of the difference matrix was firstly introduced in [4] to compute the set of fuzzy inference of rules. In our case, this operator is used as input of the fuzzy system to distinguish the different levels of motion in the image.

3 Simulation results

The simulation results presented in this section allow comparing our proposal with other de-interlacing algorithms. In order to employ an objective performance measurement, original progressive video sequences whose even/odd lines were previously eliminated have been de-interlaced and an error function has been employed to evaluate the behavior of different de-interlacing algorithms. The proposed algorithm has been compared with line doubling, line average, field insertion, VT filtering using two [8] and three fields [9] and the two fuzzy motion adaptive algorithms proposed in [3] and [4]. The three motion adaptive algorithms use the same interpolators: line average as spatial filtering and field insertion as temporal filtering (as it was explained in Section 2, our approach uses a linear combination of both techniques as third interpolator).

Different sizes of the convolution windows have been considered for our method: 3x3, 5x3 and 5x5. The weights of the new matrices 5x3 and 5x5 are shown in expressions (4) and (5), respectively. They have been selected empirically, given more priority to closest pixels in the neighborhood. Obviously, when the window size is bigger, more pixels are

$$C2 = \frac{1}{32}\begin{pmatrix} 1 & 1 & 1 \\ 2 & 3 & 2 \\ 3 & 6 & 3 \\ 2 & 3 & 2 \\ 1 & 1 & 1 \end{pmatrix} \qquad (4)$$

$$C3 = \frac{1}{42}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 2 & 1 \\ 1 & 3 & 6 & 3 & 1 \\ 1 & 2 & 3 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad (5)$$

Table 2: Average PSNR (values in dBs) when de-interlacing several video sequences

Video Sequence	Missa	Paris	Trevor	Salesman	News	Mother	Carphone
Format	**CIF (288x352)**				**QCIF (144x176)**		
Line Doubling	36.44	23.61	31.05	29.75	25.18	31.81	28.25
Line Average	40.47	26.67	35.04	33.53	29.25	35.94	32.61
Field Insertion	38.36	29.86	34.36	36.17	33.13	36.14	30.34
VT 2 fields	40.25	30.73	36.61	36.54	35.46	39.61	34.08
VT 3 fields	40.52	31.37	37.16	36.95	35.67	40.89	34.54
Fuzzy Motion Adaptive [3]	40.01	33.12	35.38	37.62	34.73	39.49	32.27
Fuzzy Motion Adaptive [4]	40.18	35.28	36.69	38.29	37.51	41.87	34.78
Proposed 1 (3x3)	40.51	35.78	37.49	38.44	38.68	41.93	34.83
Proposed 2 (5x3)	40.58	35.93	37.61	38.49	38.91	42.02	34.92
Proposed 3 (5x5)	40.63	35.99	37.68	38.55	38.97	42.05	34.97

considered to evaluate motion with the corresponding increase in the computational cost.

Table 2 shows the average PSNR values obtained when de-interlacing the fields of several standard video sequences. Three rows, corresponding to the three bi-dimensional windows used for our proposal, are included in the table. The PSNR results show that the new algorithm performs better than all the other algorithms*. The inclusion of a third consequent increases the robustness of the motion detector obtaining higher PSNR values than the other two fuzzy motion adaptive algorithms. This characteristic can be also corroborated with the de-interlaced images showed in Figure 7 and Figure 8 (especially in marked areas with white circles). Regarding the three options for our proposal, the method which works with a bigger bi-dimensional convolution window achieves the best results.

Nevertheless, not only quality performance but also computational time and cost should be evaluated. This is the reason why all these algorithms have been programmed in Matlab and executed on the same PC (a 2.0 GHz Pentium 4 processor running the MS-Window XP operating system). The total CPU time and the computational time ratio according to the fastest algorithm are shown in Table 3. Moreover, the complexity is also evaluated in terms of memory requirements as it is shown in Table 4. Comparing with the VT (3 fields) algorithm (widely used in TV industry), our proposal requires an extra-field memory since four fields are used to evaluate the amount of motion. However, it increases considerably the detection of motion.

4 Conclusions

A novel motion adaptive algorithm for de-interlacing has been presented in this paper. It employs a fuzzy system which model heuristic knowledge to classify different areas in the field according to the presence of motion. Different interpolations are applied depending on this classification. Field insertion is performed in static areas, line average in moving areas, and a combination of both in the rest of the image. As a result, the proposed method provides better solutions eliminating the blurring and the

* In spite of there are other perceptual measures, the majority of research community in image processing use the PSNR to estimate the quality of the reconstructed image.

Table 3: Computation time required by de-interlacing algorithms

	Line Doub.	Line Aver.	Field Insert.	VT 2fields	VT 3fields	Motion adap.[3]	Motion adap.[4]	Proposed Algorithm 3x3	5x3	5x5
CPU time (s)	2.03	2.05	3.28	10.62	14.65	143.03	29.21	30.95	31.71	32.14
Ratio	1	1.01	1.61	5.23	7.21	70.42	14.37	15.25	15.62	15.82

annoying stairs-step effect of the de-interlaced images. Besides, it is achieved at the expense of a low increment in the complexity.

Acknowledgements

The authors wish to express their gratitude to Dr. J. Gutiérrez-Ríos for his encouragements and advices. This work has been partially funded by the projects TEC2005-04359/MIC from the Spanish Ministry of Education and Science and TIC2006-635 from the Andalusian Regional Government. The first author is also supported by the Spanish Ministry of Education under the program F.P.U. for Phd. Students.

References

[1] G. De Haan and E.B. Bellers. De-interlacing: An overview. Proc. of the IEEE, vol.86, pp.1839-1857, Sept.1988

[2] G. De Haan. Video processing. University Press, Eindhoven, 2004.

[3] D. Van de Ville, B. Rogge, W. Philips and I. Lemahieu. De-interlacing using fuzzy-based motion detection. *Proc. 3rd Int. Conf. on Knowledge-Based Intelligent Information Engineering Systems*, pp.263-267, Adelaide, Australia, Aug. 1999

[4] J. Gutiérrez-Ríos, F. Fernández-Hernández, J. C. Crespo and G. Treviño. Motion adaptive fuzzy video de-interlacing method based on convolution techniques. *Proc. of Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Perugia, Italy, July 2004

[5] A. M. Bock. Motion adaptive standards conversion between formats of similar field rates. Signal Processing: Image Communication, pp.275-280, vol.6 no.3, June 1994

[6] F.J. Moreno-Velo, I. Baturone, S. Sánchez-Solano, A. Barriga. Rapid design of complex fuzzy systems with XFUZZY, *Proc. IEEE Int. Conf. on Fuzzy Systems*, pp.342-347St. Louis, USA, May 2003

[7] F.J. Moreno-Velo, I. Baturone, R. Senhadji and S. Sánchez-Solano. Tuning complex fuzzy systems by supervised learning algorithms, *Proc. IEEE Int. Conf. on Fuzzy Systems*, pp.226-231, St. Louis, USA, May 2003

[8] Genesis Microchip, Inc., Preliminary data sheet of Genesis gmVLD8, 8 bit digital videoline doubler, version 1.0, June 1996

[9] M. Weston. Interpolating lines of video signals. US-patent 4, 789-893, Dec. 1998

Table 4: Storage devices

	Line Doub.	Line Aver.	Field Insert.	VT 2fields	VT 3fields	Motion adap.[3]	Motion adap.[4]	Proposed Algorithm 3x3	5x3	5x5
Field Memories	0	0	1	1	2	3	3	3	3	3
Line Delays	0	1	0	1	2	0	0	0	0	2
Pixel Delays	0	0	0	0	0	4	4	2	4	4

Figure 7: (a) Progressive frame of "Carphone" sequence. (b) The corresponding interlaced field. De-interlaced image applying: (c) line doubling, (d) line average, (e) field insertion, and (f) VT filtering 2 fields

Figure 8: De-interlaced image applying: (a) VT filtering 3 fields, (b) fuzzy motion adaptive in [3] and (c) in [4], proposal with 5x5 (d), 5x3 (e) and (f) 3x3 window