Signal Processing: *Image Communication* 2 (1990) 365-374
Elsevier

TIME-RECURSIVE DEINTERLACING FOR IDTV AND PYRAMID CODING*

Feng-Ming WANG, Dimitris ANASTASSIOU

Department of Electrical Engineering and Center for Telecommunications Research, Columbia University, New York, NY 10027-6699, U.S.A.

Arun N. NETRAVALI

AT&T Bell Laboratories, Murray Hill, NJ 07974, U.S.A.

Received 20 March 1990

Abstract. Most improved definition television (IDTV) receivers use progressive scanning to reduce artifacts associated with interlacing (e.g. interline flicker, line crawl, etc). We propose some novel techniques of motion compensated interpolation of the missing lines of interlaced monochrome and color sequences, reducing the artifacts associated with interlacing, and effectively increasing the vertical resolution of the image sequences. Time-recursive motion compensated prediction is introduced, in which all previously displayed history (not just the previous field) is used to predict the missing pixel values. The next future field is also used for the same purpose, by a look-ahead scheme. Motion estimation is done using a quadtree-based segmented block-matching technique with half pixel accuracy. To avoid artifacts and obtain full resolution in still regions, like background, motion adaptation is used as well. We also discuss how to apply this algorithm to pyramid coding to achieve better compression rate and compatibility of progressive with interlaced format standards.

Keywords. Motion compensation, deinterlacing, block matching, HDTV, IDTV, pyramid coding.

1. Introduction

Interlaced scanning is an efficient method of bandwidth compression that was appropriate when TV frame memories were expensive. However, in addition to the loss of vertical resolution, interlacing results in many well-known artifacts. With the cheap frame memories becoming widely available, IDTV receivers display non-interlaced video by appropriate processing at the receiver without changing the scanning structure of the current television system, such as NTSC. Proper conversion from interlaced to progressive format reduces line flicker and improves the vertical resolution of the displayed images. Such techniques [6, 12] are also important in high definition television (HDTV), since many of the HDTV proposals either involve transmission of interlaced video, or high spatial frequency information at a reduced temporal rate.

Figure 1 shows the three-dimensional (3-D) domain of a sampled image sequence $x(m, n, t)$, in which the missing lines are in dotted form. In Fig. 2, the vertical-temporal grid (for a constant value of n) is shown, with circles on the scan lines of an interlaced sequence and crosses on the missing lines.

Interlaced scanning introduces aliasing, unless the moving image is properly pre-filtered with a low-pass filter. The 3-D frequency responses of three possible such filters are shown in Fig. 3, as a direct consequence of the shape of the quincunx sampling grid in Fig. 2. Different reconstruction methods are appropriate for each form of prefilter-

* Work supported in part by the National Science Foundation under grant ECD8811111 A0, by the NSF Presidential Young Investigator award ECS-84-51499, and by the New York State Center for Advanced Technology under project 89-A10.

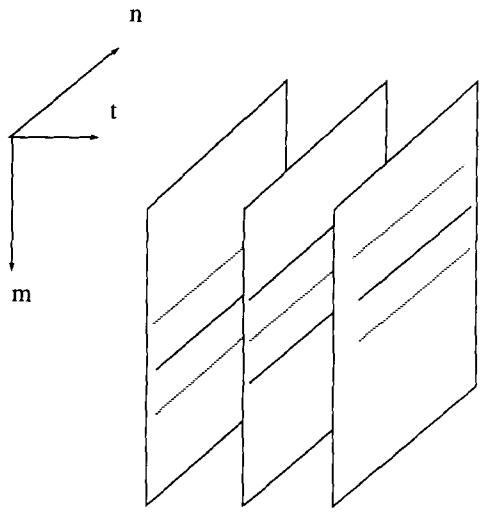

Fig. 1. 3-D sampled sequence.

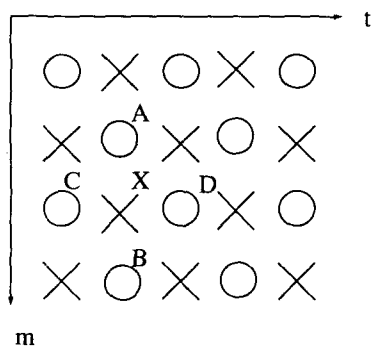

Fig. 2. Vertical-temporal grid.

ing. In actual image capturing devices, temporal filtering is due to camera time integration and is performed independent of spatial filtering, resulting in separable prefiltering. If vertical filtering is not strong enough to fully cover the area between two successive field lines, there is some information missing between the field lines, which can be estimated using appropriate models for the image sequence source. From psychophysical experiments, we have found that temporal filtering is blurring moving objects, due to eye tracking, and that the apparent quality of interlaced video is best if it comes from progressive video by dropping half the lines with motion adaptive prefiltering. It is also possible to avoid prefiltering altogether, with quite acceptable results. In that case, deinterlacing only consists of predicting the missing lines because each existing line remains as it is.

In Section 2, we describe deinterlacing as a resolution enhancement technique using nonlinear interpolation based on a source model. In Section 3, we summarize most used deinterlacing techniques. In Section 4, the technique of Time-Recursive Motion Compensated Prediction is introduced, in which only one frame memory is needed, but the effect of all previous displayed history is used to predict the missing lines. The deinterlacing algorithm is described in Section 5 in detail. In

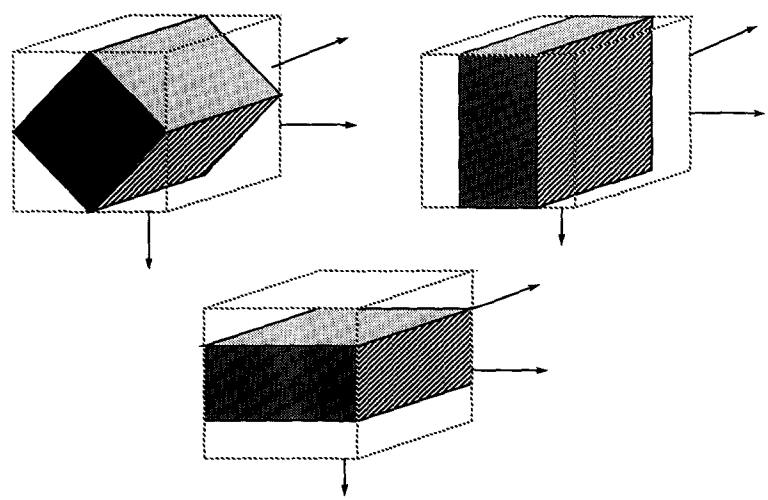

Fig. 3. Three possible antialiasing pre-interlacing filters.

Section 6, we discuss how color information is used in our algorithm. We show that chrominance signals slightly but systematically enhance the correctness of motion prediction and improve quality. We outline a systolic array based implementation of the scheme in Section 7. In Section 8, the application of this scheme to pyramid coding is introduced. Pyramid coding using deinterlacing is useful to achieve compatibility of possibly progressive HDTV with current interlaced TV. The 'helper' signal [7] of pyramid coding can be coded in different ways to achieve lower bit rate. In Section 9 we present the conclusions together with a comparison of various techniques using numerical distortion measurements.

2. Resolution enhancement based on a source model

We assume that, preceding interlaced scanning, separable vertical-temporal filtering was done as in the dotted part (external cube) of Fig. 3, i.e., as if the pictures were progressively scanned with a frame rate equal to the interlaced field rate. Equivalently, such interlaced sequences are produced by properly dropping half the lines from a progressively scanned sequence. Our experiments show that the resulting aliasing, which gives rise to various interlacing artifacts, is reduced by nonlinear interpolation of the missing lines. This resolution enhancement is a consequence of the fact that the sampling theorem can be 'defeated' (e.g., some high frequency information between the solid and dotted volumes of Fig. 3 can be predicted) if there is extra knowledge of the signal source. The source model used here assumes that the video scene contains a number of rigid objects moving in a translational manner. The same concepts can be used to achieve purely spatial or temporal resolution enhancement. Nonlinear interpolation reduces the energy of the enhancement signal, defined as the difference between the deinterlaced and the actual signals. Therefore, a two-channel hierarchical pyramid based [3] coding scheme will benefit from nonlinear interpolation because the 'helper' signal of the second channel can be coded more efficiently (see Section 8).

Some simple temporal models assume that the image scene contains rigid moving objects and that the components of their displacement vectors from frame to frame are not necessarily integer multiples of the spatial sampling period. This implies that the same object is sampled in many frames. Considering the overlap of all these sampling grids on a specific object, it is obvious that the 'virtual' spatial sampling rate for the object is much denser than the one in any individual frame. Assuming, for the moment, that the image sequence is sampled without any antialiasing prefiltering, the above fact implies that one can derive, at least in theory, a high-definition image sequence from the samples of the many frames in which it was captured. If there is spatial antialiasing prefiltering, then the blurred frame can be reconstructed from the samples of one frame only, and no additional information is conveyed from the other frames; if there is temporal antialiasing prefiltering, then moving objects will also be blurred. According to perceptual psychology, the human eyes tend to track moving objects, so that, if ideal temporal prefiltering is applied, blurred moving objects will reduce image quality. This argument shows that it is desirable to apply weaker antialiasing prefiltering schemes, in video capture, than theoretically needed, even for progressively scanned scenes, or to make the amount of such prefiltering dependent on motion, in a pixel-by-pixel adaptive manner. In the case of traditional interlaced scanning, as explained above, there are missing lines and we can predict them from previous fields.

3. Traditional deinterlacing schemes

Deinterlacing is an interpolation problem. There are various fundamental differences between the problem of predicting pixel values for interpolation purposes and the same problem for coding purposes. Unlike coding, where inaccurate predictions only increase the bit rate slightly, such

inaccuracies can be catastrophic in interpolation, where the result is displayed as it is. Furthermore, in the case of coding, the predicted values must be accessible to the decoder, so they must be functions of the reconstructed (hence distorted) values of previously encoded pixels. In the case of interpolation, predictions can be functions of the actual pixel values of both previous and following frames, provided that some small delay, due to lookahead, can be tolerated. Finally, unlike coding, in which the displacement vector field must be describable in a simple manner because it must also be coded and transmitted (unless estimated from reconstructed data known to the decoder), the displacement vector field can be arbitrary.

There are various simple ways to predict the pixel X of a missing line (see Fig. 2), and various local, motion adaptive or motion compensated techniques have been proposed. In Fig. 2, A, B, C and D are the pixels above, below, behind and in front of pixel X. Intraframe techniques can use line doubling, i.e. $X = A$, linear interpolation, i.e. $X = \frac{1}{2}(A + B)$, or more complicated nonlinear spatial interpolation techniques [4, 8]. Such techniques cannot predict information which is lost from the subsampled current field, but appears in neighboring fields. For example, Fig. 4 shows the result of linear interpolation of a face picture, in which the upper and lower teeth were captured in neighboring field lines, resulting in the appearance of elongated teeth.

Fig. 4. Results of linear interpolation.

Interframe interpolation can be as simple as $X = C$ or $X = \frac{1}{2}(C + D)$. This is fine for stationary objects, but it obviously results in severe artifacts, like multiple objects in a frame, if there is motion. Predicting X by $\frac{1}{4}(A + B + C + D)$ is not a satisfactory compromise. The nonlinear median filter $X = \text{Med}(A, B, C)$ usually results in acceptable visual appearance, and is used (with some extra features) in IDTV receivers [5], because it takes the value of C, appropriately clipped to stay within the range defined by the values of A and B. If C is outside the range defined by A and B, then it is likely, but not guaranteed, that there is motion at that point, and clipping its value at least protects from grossly false predictions. If C is inside that range, then it is assumed that there is no motion, so C is chosen for the interpolated value. Even though artifacts like those of Fig. 4 could still appear, there is some overall obvious improvement over interlaced displays.

If there is motion, then some form of motion compensation must be used. Schemes based on symmetrical matching search of a pair of one-dimensional horizontal windows, centered around the pixel X (see Fig. 2), sliding over the four closest horizontal lines at A, B, C and D are simple to implement, but fail to predict vertical motion accurately. Furthermore, all techniques based on symmetrical matching search of a pair of 2-D blocks centered around the pixel to be predicted assume constant velocity and cannot provide for acceleration. This does not result in serious visual artifacts in purely temporal interpolation when a whole frame is predicted based on the two neighboring frames. We found, however, that it is unacceptable in the problem of deinterlacing, due to the accelerating motion of the objects in the scene.

4. Time-recursive motion compensated prediction

Recursive filters are known to have lower implementation complexity for a similar filtering

effect, compared to nonrecursive ones. Even first order linear recursive filters have infinite impulse response and produce an output depending on the whole history of inputs. A generalization of this fact is used in our proposed technique, which is based on motion compensation to use the whole history of the previous fields for prediction of the missing lines of the current field. The following nonlinear time-recursive relation is used:

$$y(t) = F[y(t-1), x(t)], \tag{1}$$

in which $x(t)$ is the input field at time t, $y(t)$ is the output frame at time t, and F is a function to be defined. Let us also define by $\hat{x}(t)$ the frame at time t, coming from linear spatial interpolation of $x(t)$. Note that the field $x(t)$ is included in half of the lines of the frame $y(t)$, and of the frame $\hat{x}(t)$ as well. The initial condition $y(0)$ is chosen equal to $\hat{x}(0)$.

The frame $y(t)$ in (1) is found by motion compensation. This can be done in various ways. We have used motion estimation by quadtree-based segmented block matching of blocks of the known field $x(t)$ on the frame $y(t-1)$, allowing displacement vectors with half pixel accuracy. The initial size of blocks is 16×8, which corresponds to a 16×16 square block with half the lines missing. In this way, if a missing line happens to be also missing from the previous field, it will still be taken from the higher resoluion signal $y(t)$ which contains information from the previous fields.

A future field, e.g. $x(t+1)$, can also be used in evaluating the frame $y(t)$. In that case, (1) with lookahead becomes

$$y(t) = F[y(t-1), x(t), x(t+1)], \tag{2}$$

in which each block from the field $x(t)$ is matched to either the previous interpolated frame $y(t-1)$ or the next linearly interpolated frame $\tilde{x}(t+1)$, depending on which of the two results in smaller error. Providing for look-ahead results in minimal delay, equal to the field-refresh period. It is important whenever there are sudden changes of scene, in which case there can be no proper match between $x(t)$ and $y(t-1)$.

In order to avoid unattenuated propagation of noise, the following relation gives better results than (2):

$$y(t) = (1-c)\hat{x}(t) + cF[y(t-1), x(t), x(t+1)], \tag{3}$$

where c is between 0 and 1, usually close to 1. In our experiments, we found a near optimum value of $c = 0.9$.

We also use motion adaptation to maintain the sharpness and quality of the motionless part of each frame. Corresponding blocks in $x(t)$ and $x(t-2)$ are compared; if they are well matched, $y(t)$ is generated by $y(t-1)$ only:

$$y(t) = \begin{cases} y(t-1) & \text{if no motion;} \\ \text{as in (3),} & \text{otherwise.} \end{cases}$$

This modification keeps background sharp and noiseless. Otherwise, flickering temporal patterns with period 2 can appear even in totally motionless areas.

5. Description of algorithms

As described above, block matching is performed to interpolate the missing lines of each field. The current field (at time t) is separated into rectangular blocks of size 16×8. Due to the missing lines, the number of lines in each block is twice the number of pixels horizontally. Accordingly, the block matching procedure must be modified. We use the traditional distortion functions:

$$\sum |x(m, n, t) - y(m - d_m, n - d_n, t-1)|$$

or

$$\sum |x(m, n, t) - \hat{x}(m - d_m, n - d_n, t+1)|,$$

and the summations are over all locations (m, n) of the field $x(t)$ that belong to the block. Therefore, half of the lines of the blocks in $y(t)$ and $\hat{x}(t)$ are not involved in the distortion evaluation. In the above equations, d_m and d_n are the coordinates of the displacement vectors. For each block, interpolation is done for either the past or the future,

depending on which of the two results in the smaller distortion.

Some of the blocks, however, may contain a piece of a moving object and a piece of the background, or they may contain pieces of more than one moving object. The resulting interpolated values will be incorrect for part of the block. In order to avoid this quality degradation, after determining the optimum block matching, corresponding to the displacement vector (d_m, d_n), we evaluate the maximum absolute deviation over all pixels of the 16×8 block:

$$\max\{|x(m, n, t) - y(m - d_m, n - d_n, t - 1)|\}$$

or

$$\max\{|x(m, n, t) - \hat{x}(m - d_m, n - d_n, t + 1)|\},$$

depending on whether optimum matching was found with the past or future frame, respectively. If the maximum deviation exceeds a certain threshold, then the block is separated into four subblocks of size 8×4, and block matching is repeated for each of the subblocks. The same procedure is repeated for one more level: if the maximum deviation exceeds the threshold, then each block is again separated into four subblocks of size 4×2. This quadtree-based segmented block matching is a reasonable compromise between complexity and quality.

We observed that allowing the components of the displacement vectors to have half pixel accuracy has slightly but consistently improved the accuracy of predicting the missing lines. Therefore, after evaluating the optimum displacement vectors, we are fine-tuning the values of its components by local search of maximum half pixel displacement difference, using bilinearly interpolated values.

Finally, as an extra optional postprocessing step to protect from any inaccurate predictions, we can clip the predicted value of each pixel of the missing lines, so that it is in some range defined by the pixels below and above it. This is done by substituting the value, say X, of each of the interpolated pixels by $\text{Med}(X, F, G)$, where $F = \max(A, B) + Q$, $G = \min(A, B) - Q$, Q is a predetermined positive constant, and the values of A and B were defined in Fig. 2. The value of Q must depend on the degree of sophistication of the motion estimation technique. It must be high for accurate estimation, and low if there is risk of false interpolation. A small value of Q, however, can deterioriate the quality of the deinterlaced frames. In our experiments we did not observe any quality improvement from using this step. We believe, however, that such a stage is desirable, unless motion estimation is guaranteed to be accurate.

6. Motion prediction enhancement using color signal

The above process can also be applied to color signals and we can use color information to make better motion predictions. Instead of using luminance only, we use both luminance and chrominance for motion rediction.

The algorithm described in Section 4 is performed on Y, U and V signals at the same time, and one difference value (Y_{err}, U_{err} or V_{err}) is generated for any 16×8 block of each signal separately. We define the total difference value (T_{err}) of one block as

$$T_{\text{err}} = Y_{\text{err}} + d * (U_{\text{err}} + V_{\text{err}}),$$

in which d is a positive value; the smallest T_{err} represents the best match for the block.

Since motion compensation is performed on both luminance and chrominance, more calculations are needed. However, those additional calculations can be performed in parallel and the quality of the deinterlaced picture is enhanced because we make a better motion prediction. The hardware complexity is slightly augmented.

In order to numerically measure the effects of this algorithm, a progressive color sequence, 'garden', was interlaced by dropping half of the lines. After the deinterlacing algorithm is applied to the 'garden' sequence, the MSE between the original pictures and the generated ones was calculated. The average MSE of all the frames in the sequence

is decreasing as d increases. The decreasing tendency is slight but systematic, which proves that color information does slightly help motion prediction. The best value of d for the 'garden' sequence was found to be 0.8, but slightly smaller values are expected for typical sequences.

7. Implementation considerations

In this section, we outline a systolic array based implementation, based on the three-step approach [10]. We focus on the quadtree-based segmented block matching, which is computationally expensive. The hardware implementation of traditional block matching has been tackled in [9], without any reference to quadtree-based segmentation.

If $x(i, j)$ are the pixel values in the current block of size $N \times N$, and y is the reference data, then we have

$$S(d_m, d_n) = \sum_{m=1}^{N} \sum_{n=1}^{N} |x(m, n) - y(m+d_m, n+d_n)|,$$

$$-d_{\max} \leq d_m, d_n \leq d_{\max},$$

where $S(d_m, d_n)$ is the distortion measure between x and y, corresponding to displacement (d_m, d_n), and $d_{\max}$ is the maximum allowed displacement in each dimension.

Each pixel difference evaluation requires four parameters, resulting in a four-dimensional (m, n, d_m, d_n) problem. To map it to a dependence graph [10] easily, it can be decomposed into a pair of a three-dimensional and a one-dimensional problems by first fixing d_n, i.e.,

$$S_1(d_n) = \min_{d_m} S(d_m, d_n), \tag{4}$$

$$\hat{d}_m(d_n) = d_m|_{S=S_1}, \tag{5}$$

$$S_{\min} = \min_{d_n} S_1(d_n). \tag{6}$$

The motion vector of the current block is given by

$$(\hat{d}_m(d_n), d_n)|_{S=S_{\min}}. \tag{7}$$

There are two more considerations for quadtree-based segmentation:

(1) The block size is adaptive based on the values of the minimum absolute difference between the current and the reference block.

(2) The data on the current block come from a field instead of a frame, i.e., half of the lines in the block are missing.

The systolic array structure in [9] can be modified to fit our requirements. Instead of using 4×4 arrays, we can use 4×2 arrays with some delays to solve the second problem. With four 4×2 arrays and few adders and comparators, a 8×4 quadtree-based segmented block matching system can be built.

Each set of the 4×2 blocks generate a minimum distortion measurement which is stored and added to the results of the other three blocks. If the minimum total difference of the four blocks is greater than a threshold, the distinct results are used. The same process can be expanded to 16×8 blocks with higher complexity.

Motion adaptation is another important component of the scheme. It is based on difference calculations, so we can modify (4)-(7) to accommodate these calculations, allowing the structure above to be used for this purpose.

8. Applications to pyramid coding

Pyramid coding [3] has been found to be a useful tool for multiresolution data compression. The traditional way to generate pyramids is to use linear interpolation. However, the enhancement signal, generated by a traditional scheme, has high energy which makes high compression rate hard to achieve.

Interlacing is a form of subsampling by 2 and the deinterlacing process described here is one kind of nonlinear interpolation. If we substitute the linear interpolation part in a traditional pyramid scheme with the deinterlacing algorithm described above, the energy of the 'helper' signal is reduced significantly. In Fig. 5, down-sampling

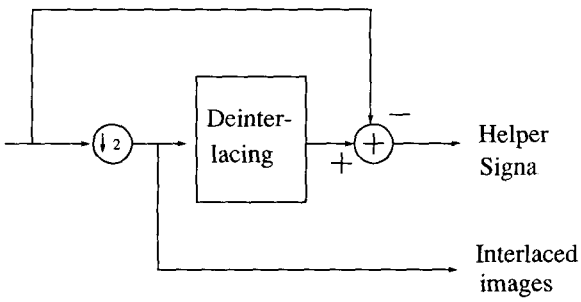

Fig. 5. Two channel pyramid coding with critical sampling and deinterlacing.

is performed at the temporal-vertical domain (interlacing) and no prefilter is used because of the reasons we mentioned in Section 4, thus achieving critical sampling. Therefore, the number of samples for the helper signal in Fig. 5 is only half of the one used in the general pyramid scheme. Because of both critical sampling and energy reduction of the helper signal (compared with linear deinterlacing), it can be coded efficiently with low bit rate. Further, the two channels appear to be 'more independent' of each other, thus justifying the above observations.

If we cascade other stages which perform purely spatial [8] or purely temporal nonlinear interpolation, the pyramid scheme can generate multi-resolution signals which could make HDTV or IDTV compatible with traditional interlaced TV. This scheme produces coded transmission signals compatible with TV formats at various resolutions.

Since the enhancement signals have different statistics compared to normal pictures, we expect that different coding techniques may achieve lower bit rate than DCT (Discrete Cosine Transform) coding for them. Adaptive arithmetic coding [2, 11] is another method that is suitable for coding low-energy signals, like the helper signal in Fig. 5.

Adaptive arithmetic coding separates a coding problem into two parts: first, finding a statistical model for the entity to be modeled, and second, performing the coding based on this model with optimum performance. The symbols in the helper signal are coded sequentially based on an estimate of their probability, which is calculated using a particular template. Before coding, the pixel values of helper signals are linearly quantized with adjustable quantization stepsizes. We control the bit rate by choosing the size of the quantization stepsizes.

The pixels around the current pixel X decide the state of pixel X. Based on previous statistics and the state, the probability and entropy of pixel X is evaluated, and the minimum cost (in bits) for coding X is equal to $-\log_2(\text{prob})$, where prob is the probability that X assumes the value that it eventually assumed, under the current state. The total cost for the whole signal is the bit rate that we can achieve. According to our simulation, based on several video sequences, the helper signal can be coded with less than 0.3 bits/pixel using arithmetic coding without noticeable degradation.

9. Simulation results and conclusions

In order to numerically measure the effects of the various algorithms, we have used various sequences of progressive video images and then interlaced them by dropping half the lines. As a particular example, we have used the standard progressive 'Miss America' sequence, which was also interlaced by omitting half the lines from each frame. After applying different interpolation algorithms to these interlaced sequences, the outputs were compared with the original sequence in terms of their MSEs evaluated by comparing the interpolated values of all the pixels in the missing lines, with their known actual values. This sequence was used for demonstration purposes and generation of tables with error measurements.

Figure 6 shows typical MSE curves from a number of successive frames belonging to that sequence. Shown are the MSEs when the interpolated value for each pixel is equal to (see Fig. 2)

(1) that of the previous field [C];
(2) linear spatial interpolation $[\frac{1}{2}(A+B)]$;
(3) median filter [Med(A, B, C)];
(4) nonrecursive (i.e., using $\hat{x}(t-1)$ instead of $y(t-1)$) motion compensated interpolation, without look-ahead and without half pixel accuracy;

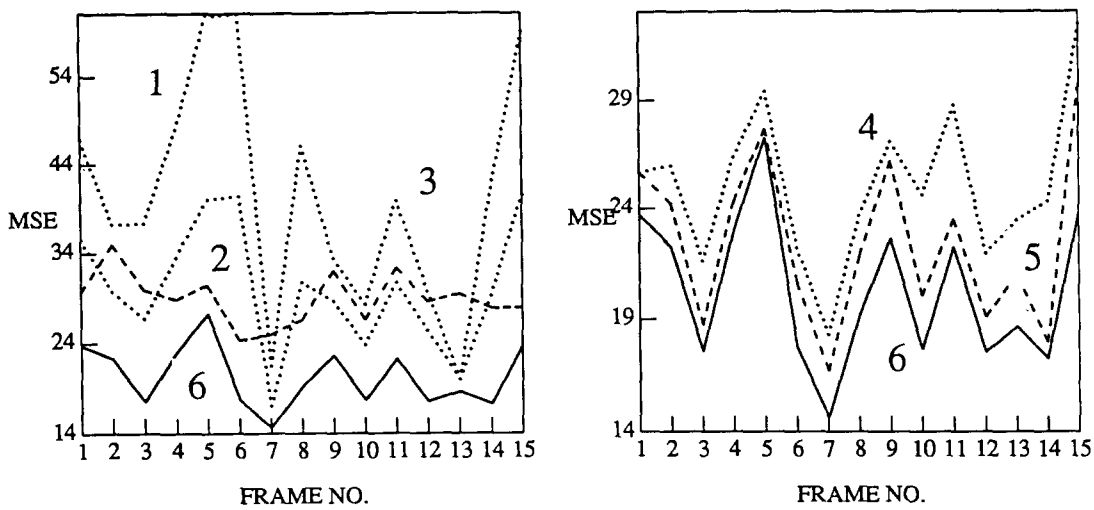

Fig. 6. MSE curves of different deinterlacing methods.

(5) time-recursive motion compensated interpolation, without look-ahead and without half pixel accuracy;
(6) time-recursive compensated interpolation, with look-ahead, half pixel accuracy and motion adaptation.

The MSE of these algorithms, as shown at Fig. 6 is being reduced, as the index of the algorithm used increases.

The proposed algorithms have reasonably low complexity for real time VLSI implementation in the near future, since they include simple straightforward operations and block matching motion estimation. Furthermore, using time-recursive prediction, the frame memory requirements are not severe, because one past frame conveys information from all previously displayed history. Applications include Improved Definition TV Receivers and interpolation for multiresolution multiple channel video coding.

It is worth observing that the resolution of the output frame $y(t)$ in (1), (2) and (3) can be chosen, in both spatial dimensions, to be twice as large as that of the actual frame, expecting that even more high spatial frequency information can be recursively extracted. This will be true if the technique proposed in this paper is combined with additional, purely spatial, nonlinear interpolation techniques [8]. Such approach would require four times more frame memory, but would facilitate the process of block matching with half pixel accuracy.

Finally, the following algorithmic modifications can be used for further work:

(1) Separate transmission of motion vectors and flags determining whether interpolation is forward or backward. This 'digitally assisted deinterlacing' achieves higher quality and greatly reduces the complexity of the receiver, but requires modification of the transmitter signal (e.g., using the vertical blanking interval).
(2) Generalized time-recursive deinterlacing, e.g., from downsampled video in linequincunx structures.
(3) Second-order time-recursive deinterlacing, using the previous 'co-sited' already deinterlaced field in addition to the previous and next fields.

References

[1] D. Anastassiou, "Generalized three-dimensional pyramid coding for HDTV using nonlinear interpolation", *Picture Coding Symposium*, Cambridge, MA, March 1990.

[2] D. Anastassiou, W. B. Pennebaker and J. L. Mitchell, "Gray-scale image coding for freeze-frame videoconferencing", *IEEE Trans. Comm.*, Vol. COM-34, No. 4, April 1986.

[3] P.J. Burt and E.H. Adelson, "The Laplacian pyramid as a compact image code", *IEEE Trans. Comm.*, Vol. COM-31, No. 4, April 1983.

[4] D. De Vleeschauwer and I. Bruyland, "Nonlinear interpolators in compatible HDTV image coding", in L. Chiariglione, ed., *Signal Processing of HDTV*, North-Holland, Amsterdam, 1988.

[5] T. Doyle, "Interlaced to sequential conversion for EDTV applications", in: L. Chiariglione, ed., *Signal Processing of HDTV*, North-Holland, Amsterdam, 1988.

[6] B. Girod and R. Thoma, "Motion-compensating field interpolation from interlaced and non-interlaced grid", *SPIE Image Coding*, Vol. 594, 1985.

[7] M. Isnardi, J.S. Fuhrer, T.R. Smith, J.L. Koslov, B.J. Roeder and W.F. Wedam, "Encoding for compatibility and recoverability in the ACTV system", *IEEE Trans. Broadcasting*, Vol. BC-33, No. 4, December 1987.

[8] D.K. Jensen and D. Anastassiou, "Spatial resolution enhancement of images using nonlinear interpolation", *Proceedings 1990 IEEE Internat. Conf., Acoust. Speech Signal Process.*, Albuquerque, NM, April 1990.

[9] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms", *IEEE Trans. Circuits and Systems*, Vol. 36, No. 10, October 1989.

[10] S.Y. Kung, *VLSI Array Processor*, Prentice-Hall, Engelwood Cliffs, NJ, 1987.

[11] W.B. Pennebaker, J.L. Mitchell, G.G. Langdon and R.B. Arps, "An overview of the basic principles of the Q-Coder adaptive binary arithmetic coder", *IBM J. Res. Develop.*, Vol. 32, November 1988, p. 717.

[12] F.M. Wang, D. Anastassiou and A.N. Netravali, "Time-recursive motion compensation deinterlacing", *Proceedings 3rd International Worksvp on HDTV*, Torino Italy, August 1989.