Spatiotemporal Interpolation for Sampling Grid Conversion with Application to Scalable Video Coding

by

Marwan Matta

B.Eng.

Department of Electrical Engineering

McGill University

Montreal, Canada

November 1994

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering

ISBN 0-612-05462-4

Abstract

Scan conversion for 2:1 field rate conversion from a SIF transmission format to a CCCIR601 video display format, and scan conversion for scalability are presented. We also address scan conversion from interlaced to progressive formats, also known as deinterlacing. For this purpose we examine adaptive weighted and switched techniques based on spatial and motion-compensated deinterlacing. We introduce a motion-adaptive frame rate conversion based on motion-compensated and temporal interpolation. When motion is relatively low at a pixel neighborhood, temporal interpolation will be carried out, otherwise motion-compensated interpolation will be performed. Indeed when motion is relatively high, temporal frame rate conversion causes repetition of contours and jerkiness. We develop and use a Maximally Flat Digital Filter for generating the samples that will be used in temporal interpolation, and this to improve the quality of this interpolation. We compare motion adaptive frame rate conversion with MPEG-2 conversion, using both visual and PSNR criteria. We also propose a new scheme that achieves multiscale spatial conversion: pattern motion-compensated interpolation. For this we develop a least square solution using Cauchy steepest descent. We further prove the existence and uniqueness of such solution. This method gives better visual quality images than cubic, MPEG-2 and pattern interplation. Finally we analyze the impact that different interpolation algorithms have on MPEG-2 scalable video coding. We present a new down conversion that replaces the MPEG-2 one and removes the aliasing introduced by this method.

Sommaire

La conversion de balayage pour une conversion 2:1 de fréquence de trames, d'un format de transmission SIF à un format d'affichage vidéo CCIR601, et la conversion de balayage pour la "scalability" sont presentées. Nous traitons aussi de la conversion de balayage du format entrelacé à progressive, opération connue aussi sous le nom de désentrelacement. Dans ce but, nous éxaminons des techniques adaptives de pondération et de commutation basées sur l'interpolation compensée par le mouvement et l'interpolation spatialle. Nous introduisons une conversion adaptée au mouvement pour la conversion de la fréquence de trames, et ceci en se basant sur l'interpolation temporelle et l'interpolation compensée par le mouvement. Lorsque le mouvement est relativement faible au voisinage d'un pixel, l'interpolation temporelle sera exécutée, sinon l'interpolation compensée par le mouvement sera réalisée. En effet lorsque le mouvement est relativement élevé, la conversion temporelle de la fréquence de champ cause la répétition des contours et un mouvement saccadé. Nous développons et utilisons un Filtre Numérique Maximallement Plat pour générer les échantillons qui seront utilisés dans l'interpolation temporelle, et ceci afin d'améliorer la qualité de cette interpolation. Nous comparons la conversion, adaptive au mouvement, de la fréquence de champ avec la conversion de MPEG-2, en utilisant les critères visuelles et le PSNR. Nous proposons aussi une nouvelle méthode qui réalise la conversion spatialle à multi-échelle: l'interpolation à motif compensée par le mouvement. Pour ceci, nous développons une solution du moindre carré en utilisant la descente du gradient de Cauchy. Davantage, on prouve l'éxistence et l'unicité d'une telle solution. Cette methode donne de meilleurs résultats visuels que l'interpolation de la cubique, de MPEG-2, et à motif. Finallement, on analyse l'impact qu'ont les différents algotithmes d'interpolation sur le codage vidéo hiérarchique de MPEG-2. Nous présentons un nouveau système de sous-echantillonage qui remplace celui de MPEG-2 et qui supprime les aliasing introduit par cette méthode.

Acknowledgements

I would like to thank my supervisor E. Dubois for his support and guidance throughout this Thesis, and for involving me in a state of the art project.

I would also like to thank N. Baaziz for all her help during my research.

I am also grateful to the Institut National de la Recherche Scientifique, (INRS)-Télécommunications, for providing me with excellent facilities, and to the National Science and Engineering Research Council (NSERC) for its financial assistance during my graduate studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The information revolution of our 20^{th} is a turning point in the history of human kind, as was the industrial revolution in the 19^{th} century. And, as we move towards a global information society with the avenue of the information superhighway, digital video is imposing itself as one of the major communication means with several services, such as video on demand, digital HDTV broadcasting and multimedia data base services.

The need for scan conversion seems to be increasing and this for different applications. Compatibility between different video standards raises the need for scan conversion. While an effort for global standardization has always been made, the variety of standards that exist today makes reliable scan conversion a requirement. This variety can be seen in traditional TV broadcasting, with different picture rates and spatial formats. In North America and Japan a 60 Hz field rate and 525 lines per image was adopted. On the other hand the standard in Europe and the former Soviet Union was a 50 Hz field rate and 625 lines per picture. New services such as Videophone (QCIF format), Videoconference (SIF format), HDTV progressive and interlaced at 50 Hz and 60 Hz, high performance workstations and PCs has increased the number of video formats.

When camera, transmission and display video formats are different, scan conversion is a must in order to make this scheme feasible. This is the case of MPEG-1, where the camera, transmission and display have different video formats. Scan conversion is needed in the down conversion from the CCIR601 camera formats to the SIF transmission one. Further scan conversion is required in the up-conversion of the SIF format to the CCIR601 display formats. Scalability is another application where scan conversion is used.

This thesis will be concerned with scan conversion algorithms which apply to the conversion between camera, transmission and displays of different formats and to scalability.

The main problem that scan conversion faces is due to the nature of the 3D video signal. This nature makes the conditions of the sampling theorem generally not satisfied. This theorem states that if the spatiotemporal spectrum of a time-varying video signal is confined to a certain fundamental region in the 3D frequency domain, exact reconstruction can be achieved by using suitable 3D linear filters. However most video signals do not satisfy this theorem and forcing the signal to be confined to some region by spatiotemporal pre-filtering is undesirable, mainly because there is an unknown relationship between spatiotemporal frequencies in the display and those on the retina of the human observer, due to eye motion and tracking. The existence of motion in the video signals combined with the above argument make spatial interpolation not appropriate for scan conversion. Furthermore, when an interlaced signal is down-sampled to form a progressive signal, frame rate conversion to the original interlaced signal can not be adequately achieved using spatial interpolation, and this due to aliasing. For these reasons motion adaptive and/or compensated processing is required.

Scalability is the ability to extract, from a signal, decoded images at different resolutions. This is done using embedded coding where the signal is encoded by a hierarchy, and a subset of the data representing lower resolution can be extracted. Scan conversion is needed in this scheme at two places:

- down-conversion where the higher resolution is converted to a lower resolution which will form the lower layer bitstream.
- up-conversion where the lower resolution will be interpolated to assist in the coding of the higher resolution.

In scalable video coding, a video sequence is transmitted at different qualities and hence at different costs, depending on the user requirement. Supporting multiple resolutions has important applications in the broadcasting industry, when the top layer is HDTV designed to be compatible with less expensive Standard Definition Television products. The computer industry is also interested in hierarchical multiresolution video coding. One of the application would be "multiplatform decoding" where a video signal is decoded by a range of platforms of different capabilities. Other applications are "multiwindow, multisource decoding" which is a multiparty videoconferencing having several sources displayed simultaneously in different windows with the possibility of independently selecting the size of each source at the decoder.

Scan conversion using motion compensation or adaptive methods has been addressed in the literature. In this research, existing methods were reviewed and some were tested namely weighted and switched based adaptive methods for deinterlacing. A new approach for performing frame rate conversion was suggested based on our conclusion that temporal interpolation performs poorly when there is large motion. This was done using adaptive motion-compensated interpolation and temporal interpolation. Maximally flat digital filters were designed and used for generating the intermediate samples of temporal interpolation. This was done in order to improve the performance of temporal interpolation. A motion detector was used for switching between temporal and motion-compensated interpolation. Further new approaches for multiscale spatial interpolation, based on pattern interpolation and motion-compensated prediction, were proposed. The down-conversion specified by MPEG-2 caused noticeable aliasing which prevented good image reconstruction. We propose a new down-conversion scheme where the simple dropping of every second field will be preceded by deinterlacing followed by vertical filtering.

Adaptive frame rate conversion was evaluated in the context of scan conversion for scalable video coding at the up-conversion stage. This is done in the framework of MPEG-2 [24](Moving Pictures Experts Group). This standard is one of the most widely followed for good quality television coding. The main profile of MPEG-2 was already adopted by the Grand Alliance in the U.S. for HDTV. This standard determines the syntax of the coded bit stream related to the decoding algorithm, but does not fully specify encoding approaches for optimum signal quality, such as motion estimation algorithms, buffer control and quantization coefficients. This standard is based on a general architecture of block based motion compensation in interframe coding, and block based DCT in intraframe coding.

Chapter 2 of this thesis is a background chapter, where we introduce pyramidal coding and MPEG-2 formats. Further, a review of spatial and spatiotemporal interpolation is addressed. Chapter 3 presents adaptive spatiotemporal methods. Adaptive motion-compensated and spatial deinterlacing is one of the issues. Adaptive weighted and switched schemes are presented and compared. Both these methods are compared versus motion-compensated deinterlacing and spatial deinterlacing. The second issue that was addressed was adaptive motion-compensated and temporal frame rate conversion. This method is compared with motion-compensated, temporal and MPEG-2 frame rate conversion. In Chapter 4 we present an oriented spatial interpolation based on block pattern interpolation and where the grid has been made denser via motion compensation. This method is least square based and the solution is an iterative one. A comparison is made between our new method, the methods presented in [37], spatial interpolation and MPEG-2 interpolation scheme. Chapter 5 discusses the impact of scan conversion on MPEG-2 scalable video coding. MPEG-2 pre-processing conversion (down-conversion) is replaced by our proposed one, and the results are compared. Further, MPEG-2 post-processing conversion (up-conversion) is replaced by the adaptive frame rate conversion proposed in chapter 3 and by cubic interpolation. The performance of all these methods is compared with MPEG-2 performance. Finally Chapter 6 presents the summary and conclusion of this work.

Chapter 2

Background

The concept of multiresolution processing is based on the analysis of a signal at a hierarchy of scales. The ability to extract decoded images at different resolutions is known as scalability. A straightforward method to achieve scalability is the simulcast layering algorithm, where the required quality levels are obtained from parallel encoders operating independently and producing separate and simultaneous bit streams. This scheme is wasteful since redundant information is present at different layers. A more attractive and efficient scheme is the class of methods known as embedded coding, in which specific subsets of the binary data can be used to decode lower resolution versions of the video sequence. This class covers two main categories: suband coding and pyramidal coding. We are interested in pyramidal coding since it offers a greater flexibility. In this chapter we present an overview of pyramidal coding, including the approach used in the MPEG-2 standard. MPEG-2 formats at the different layers of the pyramid are discussed as well as the up-conversion used in MPEG-2.

A review of interpolation will also be presented. Interpolation can be achieved either in the spatial dimension or in the spatiotemporal domain. We distinguish linear and nonlinear methods in the spatial dimension. Spatiotemporal methods are divided into fixed, adaptive and motion compensated methods. A review of motion estimation and different methods that use motion compensation will be addressed.

2.1 Scalability

2.1.1 Pyramidal Coding

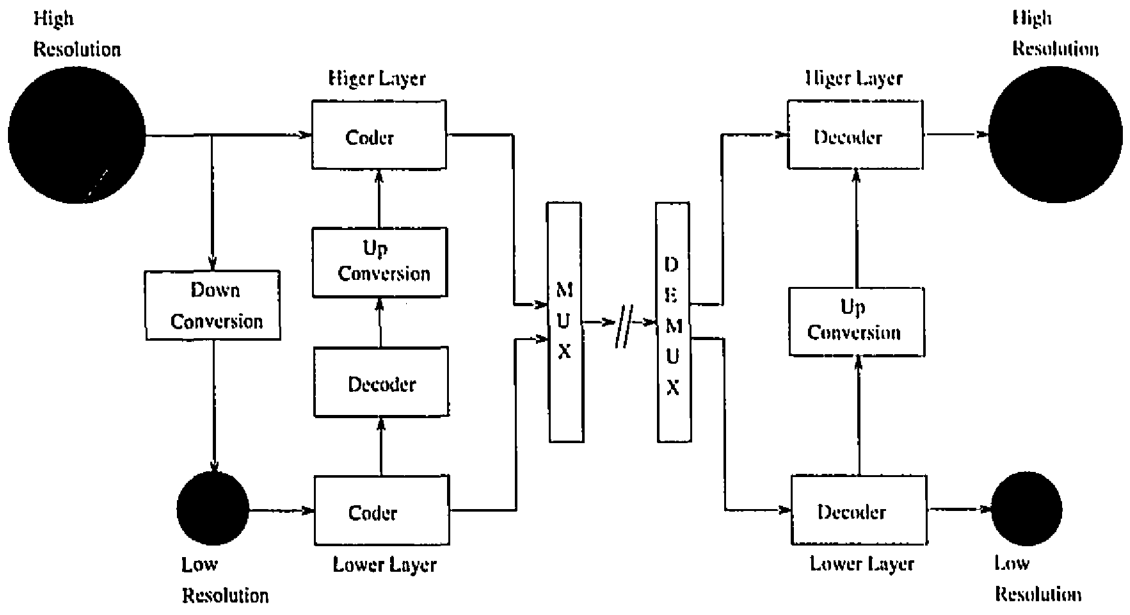


Figure 2.1: Pyramidal Coder and Decoder

Pyramidal coding is a form of embedded coding. In Fig. 2.1 a general two-layer pyramidal structure is presented. The coder structure is to the left of the multiplexer, the decoder part is to the right of the demultiplexer. At the coder structure, the high resolution image will be downsampled to form the lower resolution image of the lower layer. This image will be coded to form the lower layer bitstream. The higher layer coder has two inputs: the high resolution image and the signal from the up-sampled lower layer. These two signals will interact with each other to form the output of the coder: the higher layer bitstream. A classical configuration of this interaction is to code the difference between the two signals [35]. A more advanced scheme proposed by MPEG is to feed the up-sampled decoded lower layer to the upper layer coder, where it can be used in the process of forming a prediction. The upper layer coder chooses between the up-sampled lower layer and the conventional motion compensated prediction from the upper layer. This choice will then be notified to the decoder [23]. Spatial scalability in MPEG-2 is a form of a spatial pyramid in which the high resolution is coded with the help of both the coded low layer and the previously coded frames at the high layer. This MPEG-2 coding scheme will be elaborated on in chapter 5. At the decoder structure the lower layer bitstream is decoded to form the low resolution image. This bitstream is also upconverted and fed to the decoder, along with the higher layer bitstream, to form as an output the high resolution image. For applications using only lower layer quality, decoders will only use the lower layer bitstream.

2.1.2 MPEG-2 Formats

Interlaced and Progressive Formats

MPEG-2 formats can be classified into two main categories: interlaced formats and progressive formats, and as shown in Fig. 2.2, with t as the temporal dimension, and v the vertical one. In interlaced mode each frame is composed of two fields: one with the even lines, the other with the odd ones. Interlaced scanning was introduced when TV was being developed as an ingenious way to solve several problems that occurred while using progressive scanning. Indeed the latter has a temporal flickering problem due to HVS low pass characteristic when using the same frame rate (30 Hz). This flickering is reduced when using interlaced formats. However the former introduces artifacts such as interline flickering. An other form of progressive format is the one that has the same field rate and number of samples as the interlaced one, but on a rectangular grid. This format suffers from a lower vertical resolution. Moving images are three-dimensional signals. Interlacing result in a "quincunx" shape sampling lattice in the vertical-temporal domain as seen in Fig. 2.2(a). Deinterlacing is the

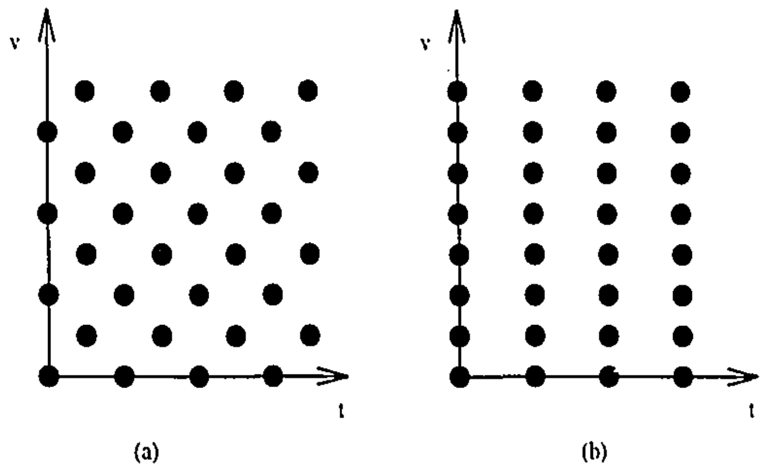


Figure 2.2: (a) Interlaced Format (b) Progressive Format

operation that allows the transformation from an interlaced format to a progressive one.

4:2:0, CCIR and SIF Formats

The original high-resolution sequence of the pyramidal structure is a CCIR 601 format. This is an interlaced format with 240 lines per field and 720 pixels per line, where the fields are displayed at a 60 Hz rate. In this format, the two chrominance components are horizontally downsampled by a factor of 2, so that they have 360 pixels per line. Pre-processing is applied to convert the CCIR 601 to the 4:2:0 format, which is also an interlaced format. The number of pixels per line is reduced from 720 to 704 by removing 16 pixels from the left/right side of the CCIR 601, for both chrominance and luminance. Further, the two chrominance components are vertically downsampled to form 120 lines per field. This down-conversion is performed using a 4-tap vertical filter for the even fields, and a 7-tap vertical filter for the odd fields. The resulting 4:2:0 luminance frame dimension and the 4:2:0 chrominance frame dimension are respectively 704 $\times$ 480 and 352 $\times$ 240. SIF format is the low resolution format at the lower layer in MPEG-2. It is obtained from the 4:2:0 format by dropping the odd fields, and by horizontally downsampling both luminance and chrominance, using the 7-tap filter. This results in a progressive format with 352 pixels per line, 240 line per frame and 30 frame per second. The chrominance components are reduced to 176 pixels per line and 120 lines per frame.

2.2 Spatial Interpolation

Given a sampled signal $I(i,j)$, which is obtained from a continuous signal $I_c(x,y)$ according to the relation: $I(i,j) = I_c(i\Delta x, j\Delta y)$, where i and j are integers, Δx and Δy are reals, as shown in Fig. 2.3. The problem of spatial interpolation is to compute the intensity $\hat{I}(x,y)$ of an unknown sample x by using the existing ones $I(i,j)$ which are represented by dots.

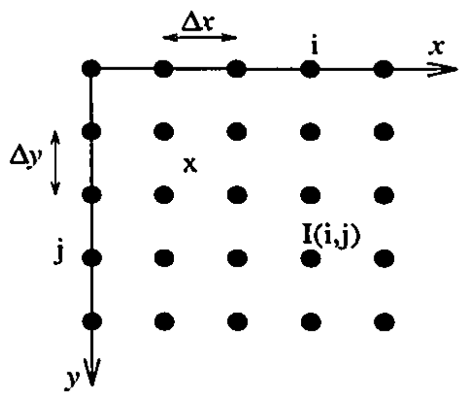


Figure 2.3: General sampling

We distinguish two kinds of spatial interpolation: linear and nonlinear ones. A method will be classified as linear if it obeys the principle of superposition. Methods that use samples from previous, present and/or next fields are called spatiotemporal methods, and will be discussed in a later section.

2.2.1 Fixed Interpolation

Linear polynomial and cubic polynomial are the most common fixed spatial methods. In deinterlacing linear polynomial interpolation is just vertical linear (straight line) interpolation:

$$\hat{I}(i,j) = \frac{I(i,j-1) + I(i,j+1)}{2} \tag{2.1}$$

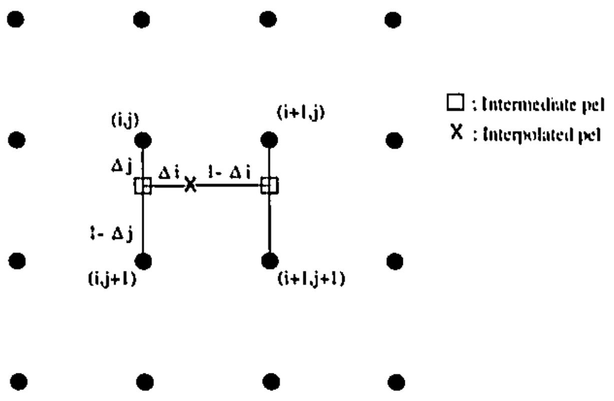


Figure 2.4: Bilinear Interpolation

Bilinear interpolation, which is a general form of linear vertical interpolation, is often used for its simplicity. It is a bidirectional separable filtering which only uses the four surrounding pels. The intensity of pel $(i + \Delta i, j + \Delta j)$ is interpolated by determining first the intermediate interpolated values at $(i, j + \Delta j)$ and $(i + 1, j + \Delta j)$, noting that $0 \leq \Delta i, \Delta j \leq 1$. This is achieved using a weighted sum of two vertical adjacent samples, where the weighting coefficients are inversely proportional to the distance Δj as seen in Fig. 2.4. The intensity for pel $(i, j + \Delta j)$ is determined according to equation (2.2).

$$I(i, j + \Delta j) = (1 - \Delta j) \cdot I(i, j) + \Delta j \cdot I(i, j + 1) \tag{2.2}$$

and similarly for $I(i + 1, j + \Delta j)$. Using once again the weighted sum for these intermediate values gives:

$$\begin{aligned} I(i + \Delta i, j + \Delta j) = \ & (1 - \Delta i)\left[(1 - \Delta j) \cdot I(i, j) + \Delta j \cdot I(i, j + 1)\right] \\ & + \Delta i\left[(1 - \Delta j) \cdot I(i + 1, j) + \Delta j \cdot I(i + 1, j + 1)\right] \end{aligned} \tag{2.3}$$

Interpolation functions have to meet the specifications of coinciding with the sampled data at the interpolation nodes. If the data are equally spaced, cubic splines and linear interpolation function g, of a sampled function f, can be put in the following form:

$$g(x) = \sum_k f(x_k) u\left(\frac{x - x_k}{h}\right) \tag{2.4}$$

where h is the sampling increment, x_k are the interpolation nodes, u is the interpolation kernel. The cubic convolution interpolation kernel is composed of piecewise cubic polynomials defined on the subintervals $(-2, -1)$, $(-1, 0)$, $(0, 1)$ and $(1, 2)$. Outside these intervals the interpolation kernel is zero. Using the necessary constraints gives the cubic convolutional kernel:

$$u(s) = \begin{cases} \frac{3}{2}|s|^3 - \frac{5}{2}|s|^2 + 1 & 0 < |s| < 1 \\ -\frac{1}{2}|s|^3 + \frac{5}{2}|s|^2 - 4|s| + 2 & 1 < |s| < 2 \\ 0 & 2 < |s| \end{cases} \tag{2.5}$$

Note that $u(s)$ is 0 when s is a nonzero integer, and furthermore that $u(s)$ is continuous and has a continuous first derivative. From the boundary conditions we get the cubic convolution interpolation function. When $x_k < x < x_{k+1}$, the cubic convolution function is:

$$\begin{aligned} g(x) = \ & f(x_{k-1})(-s^3 + 2s^2 - s)/2 + f(x_k)(3s^3 - 5s^2 + 2)/2 \\ & + f(x_{k+1})(-3s^3 + 4s^2 + s)/2 + f(x_{k+2})(s^3 - s^2)/2 \end{aligned} \tag{2.6}$$

where $s = (x - x_k)/h$ for $k = 0, 1, 2, \cdots, N$; $f(x_{-1}) = 3f(x_0) - 3f(x_1) + f(x_2)$ and $f(x_{N+1}) = 3f(x_N) - 3f(x_{N-1}) + f(x_{N-2})$. One way of assessing the performance of this interpolation is by observing its frequency response. The filters of Fig. 2.5 are derived for $h = 2$ and with the interpolated samples half way between the interplation node so that $s = .5$.

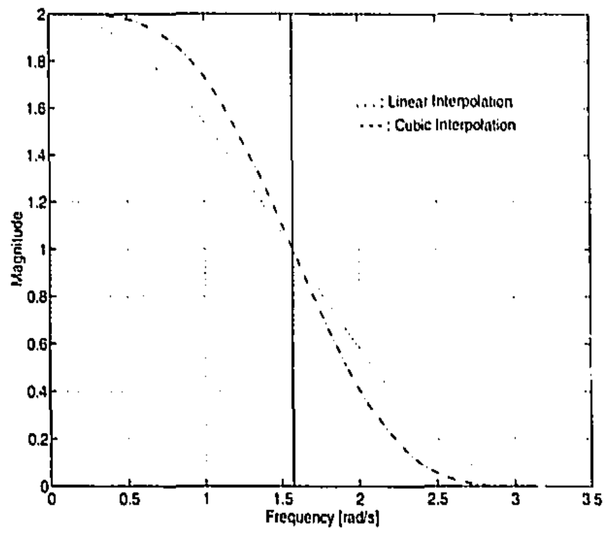


Figure 2.5: Amplitude Spectra of Interpolation Functions

The ideal interpolation filter is shown by the continuous line. Such a filter would pass every frequency component of a band-limited function without change. Deviations from the ideal spectrum in Fig. 2.5 to the left of $\frac{\pi}{2}$ cause a loss of high frequency components which result in the image appearing blurred. On the other hand deviations beyond $\frac{\pi}{2}$ cause aliasing. It can be seen from Fig. 2.5 that the cubic interpolator is a better approximation of the ideal lowpass filter than the two tap linear interpolator. Two dimensional cubic interpolation will be achieved by performing one dimensional cubic interpolation in each dimension. We are interested in the conversion from the sampling grid S_1 to the sampling grid S_2 where the horizontal and vertical ratios of grid 2 over grid 1 are respectively h and v, with h and v not necessarily integers. Let $\hat{I}_2(i_2, j_2)$ be an unknown sample point of grid S_2. In order to determine its value let:

$$x_1 = \frac{i_2}{h} \tag{2.7}$$

$$y_1 = \frac{j_2}{v} \tag{2.8}$$

then if $I_1(x_1, y_1)$ is a point of the grid S_1 we will have $\hat{I}_2(i_2, j_2) = I_1(x_1, y_1)$. Otherwise the value of $I_1(x_1, y_1)$ will be interpolated using a bidirectional cubic interpolation, and $\hat{I}_2(i_2, j_2) = \hat{I}_1(x_1, y_1)$.

Though fixed filters are easy to implement (which is a requirement in video processing for speed and memory constraints) they present the drawback of poorly preserving the vertical resolution and of introducing spatial artifacts such as: staircase effects at diagonal edges and object contours, and loss of spatial resolution.

2.2.2 Nonlinear Interpolation

One remedy to the previous artifacts is through the use of directional filtering. A directional interpolation filter for deinterlacing is presented in [7]. The interpolation of the intensity at sample **x** is carried out in one of the three directions determined by the three pairs shown in Fig. 2.6: $[(i-1, j-1), (i+1, j+1)]$, $[(i, j-1), (i, j+1)]$ and $[(i+1, j-1), (i-1, j+1)]$. These directions represent a three-level quantization of the contour orientation. In order to decide in which direction the interpolation will be performed, three absolute differences are computed:

$$\begin{aligned} D_1 = &\ |I(i-1, j-1) - I(i+1, j+1)| \\ D_2 = &\ |I(i, j-1) - I(i, j+1)| \\ D_3 = &\ |I(i+1, j-1) - I(i-1, j+1)| \end{aligned} \tag{2.9}$$

Then we determine $\Delta = D_{max} - D_{min}$ where $D_{min} = \min(D_1, D_2, D_3)$ and $D_{max} = \max(D_1, D_2, D_3)$. If Δ is less than a threshold T, it is decided that there exists no dominant orientation and the interpolation is done according to equation (2.1); otherwise the interpolation takes the following form:

$$\hat{I}(i,j) = \frac{I(i+id, j-1) + I(i-id, j+1)}{2} \tag{2.10}$$

where

$$id = \arg \min_{ii \in \{-1,0,1\}} |I(i+ii, j-1) - I(i-ii, j+1)| \tag{2.11}$$

T is chosen around 20 for a maximum luminance of 255. Furthermore three levels of orientation quantization is considered as the minimum requirement, and five as a better choice for a good image quality.

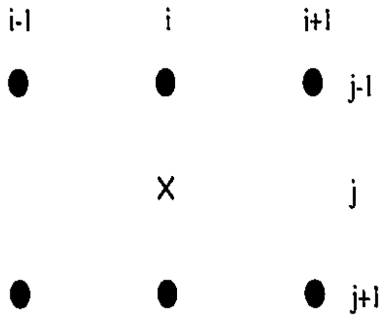


Figure 2.6: Sampled grid

Oriented interpolation is also used in [26]. The orientation $\beta_{\mathbf{x}}$ at pixel $\mathbf{x} = (i,j)$ is determined using steerable filters which are thought to give better results than Sobel operators. The interpolation along the contour orientation is performed according to:

$$\hat{I}(i,j,t) = \frac{I(i+d(\beta_{\mathbf{x}}), j-1, t) + I(i-d(\beta_{\mathbf{x}}), j+1, t)}{2} \tag{2.12}$$

where $d(\beta_{\mathbf{x}})$ is the distance shown in Fig. 2.7 and equal to:

$$d(\beta_{\mathbf{x}}) = \min\left(\left|\frac{.5}{\tan(\beta_{\mathbf{x}})}\right|, d_{max}\right) \cdot \text{sgn}(\tan(\beta_{\mathbf{x}})) \tag{2.13}$$

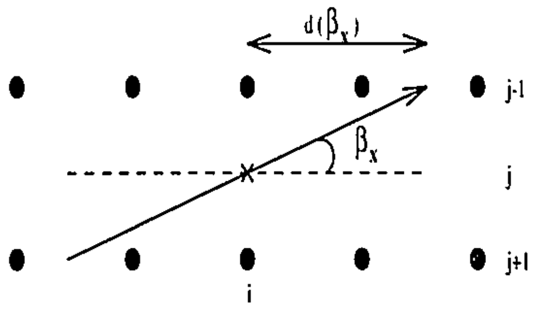


Figure 2.7: Contour Orientation [5]

An other form of oriented interpolation is discussed in [38] and [37]. The orientation in which the interpolation is carried is determined for a whole block of pixels rather than for one single pel. Furthermore the interpolation is performed blockwise. The current processing block is classified in one of the three categories: *constant*, *oriented*, and *irregular*. Constant models represent patterns with a certain uniform intensity; oriented models correspond to blocks with a dominant orientation; and irregular models describe blocks with edges in more than one direction or irregular texture pattern. If the analysis block is constant, deinterlacing of the block is done by vertical pixel replication or bilinear interpolation.

The filter for an oriented model is designed by fitting the known samples in the analysis block into an oriented polynomial of the form:

$$f(x,y) = \sum_{j=0}^{D-1} a_j(ycos\alpha - xsin\alpha)^j \qquad (2.14)$$

where α is the principal orientation of the block along which the intensity variation is negligible. The coefficients a_j are determined by minimizing the squared error

$$E = \sum_{i=0}^{N-1}\left[f(x_i,y_i) - \sum_{j=0}^{D-1} a_j(y_icos\alpha - x_isin\alpha)^j\right]^2 \qquad (2.15)$$

where N is the number of known samples in the analysis block, and $f(x_i, y_i)$ are the intensities of the known samples. If the expansion order D is smaller than the number of degrees of freedom of the block, defined by the number of known samples along its orthogonal direction, a unique least square solution exists and can be represented in matrix form as:

$$\mathbf{a} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{f} \qquad (2.16)$$

where $f_i = f(x_i, y_i)$ and $H_{ij} = (y_i cos\alpha - x_i sin\alpha)^j$.

Once the coefficients are found, we can evaluate the fitting function at the missing odd samples $(\bar{x}_i, \bar{y}_i)$ according to

$$f(\bar{x}_i, \bar{y}_i) = \sum_{j=0}^{D-1} a_j(\bar{y}_i cos\alpha - \bar{x}_i sin\alpha)^j \qquad (2.17)$$

For the irregular block, the same polynomial fitting approach as above is used, but with a non oriented polynomial of the form

$$f(x, y) = \sum_{k=0}^{D_x-1} \sum_{l=0}^{D_y-1} a_{kl} x^k y^l \qquad (2.18)$$

where D_x and D_y are respectively equal to the number of known samples in the horizontal and vertical direction in the analysis block. The least squares solution for the expansion coefficients is identical to that for the oriented model as described by equation (2.16).

A nonlinear spatial interpolation is presented in [18]. For a pixel lying on an edge between two objects, an assumption is made about the possibility of separating these two objects into two regions, each with nearly constant intensity. This separation is accomplished by an edge fitting operation, where the "best edge fit" is the one that minimizes the mean square error between the "two dimensional step-edge function" and the image window of interest. Once the two intensity values (A and B) lying on each of the sides of the "best fit" edge are determined through the latter minimization, a higher resolution grid is superimposed over the determined region of this edge fit. The interpolated values are approximated by A or B, depending on which side of the edge they lie. For those pixels intercepted by the line edge, a weighted average of A and B is the interpolated value. Since these operations are done at a low resolution, each high resolution pixel might receive multiple assignments. Thus the final intensity is then taken as an average of these assignments. This algorithm results in sharper edges than the one obtained through linear interpolation.

Other forms of nonlinear spatial interpolation are discussed in [36]. This interpolation is developed in the framework of recovering at the receiver HDTV sequences, that were downsampled before transmission. The spatial interpolation is done using a ML estimator. The estimator is based on the statistical dependence of the pixels in the immediate neighborhood, and hence uses the surrounding pixels in the estimation procedure. For this, a knowledge of the conditional density $p_{\mathbf{I}|I_o}(\mathbf{i}|i_o)$ is required. **I** is the vector of surrounding pixels used in the estimator, and I_o is the pixel intensity that has to be estimated. The following assumptions are made:
- the differences $\Delta I_k = I_k - I_o$ are statistically independent from the pixel that has to be estimated.
-the differences are statistically independent.
-the marginal probability density of one of the differences is given by:

$$p_{\Delta I_k}(\Delta i_k) = \frac{\beta_k \alpha_k}{2\Gamma\left[\frac{1}{\beta_k}\right]} \exp\left[-|\alpha_k \Delta i_k|^{\beta_k}\right] \tag{2.19}$$

where $\Gamma(x)$ is the Gamma-function. The value of the two parameters α_k and β_k are calculated statistically, i.e using a histogram. The ML estimator for a missing sample is thus the value $\hat{i}_o$ that maximizes the probability density in equation (2.19), and $\hat{i}_o$ is found to satisfy :

$$\sum_{k=1}^{n} \alpha_k |\alpha_k \Delta i_k|^{\beta-1} \operatorname{sgn}(\Delta i_k)\big|_{i_o=\hat{i}_o} = 0 \tag{2.20}$$

where n is the number of pixels used in the estimation, i.e the dimension of **I**. For $\beta = 2$ a linear estimator results. For $\beta = 1$ and for equal α_k we get a median filter. Furthermore an "extended" median filter is designed by decorrelating the random differences through some appropriate linear combination of those random variables.

Codebook based methods are also a form of nonlinear interpolation. This is discussed by Gersho in [13] for a general quantization scheme. The problem is presented in the case where the received sequence is downsampled with ratio 2 : 1. Let **X** denote the 1-D signal $\mathbf{X} = (X_1, X_2, X_3, \cdots)$ and **U** the downsampled sequence $\mathbf{U} = (X_1, X_3, X_5, \cdots, X_n)$ where n is odd. The interpolation consists in estimating the original **X** from **U**. A codebook of the values that **U** can assume is set up using some training sequence. In that case the estimate $\tilde{X}$ of X minimizes:

$$D = E\left[||X - \tilde{X}||^2 | U = c_i\right] \tag{2.21}$$

where c_i is a codevector of the codebook. The solution to this minimization problem is the conditional mean given by: $\tilde{X} = E[X|U = c_i]$. The author shows in this paper that optimal nonliner interpolation can be achieved with negligible complexity, eliminating the need for *ad hoc* linear or nonlinear interpolation techniques.

2.2.3 MPEG-2 Upconversion

MPEG-2 upconversion is also known as spatial prediction. In MPEG-2 upconversion, a lower layer is converted to an interpolated lower layer as shown in Fig. 2.8. MPEG-2 accomplishes the vertical and horizontal resampling using linear polynomial interpolation. When $n = 2$ and $m = 1$ the vertical and horizontal interpolation reduce to a linear filtering with coefficients equal to $\frac{1}{2}$.

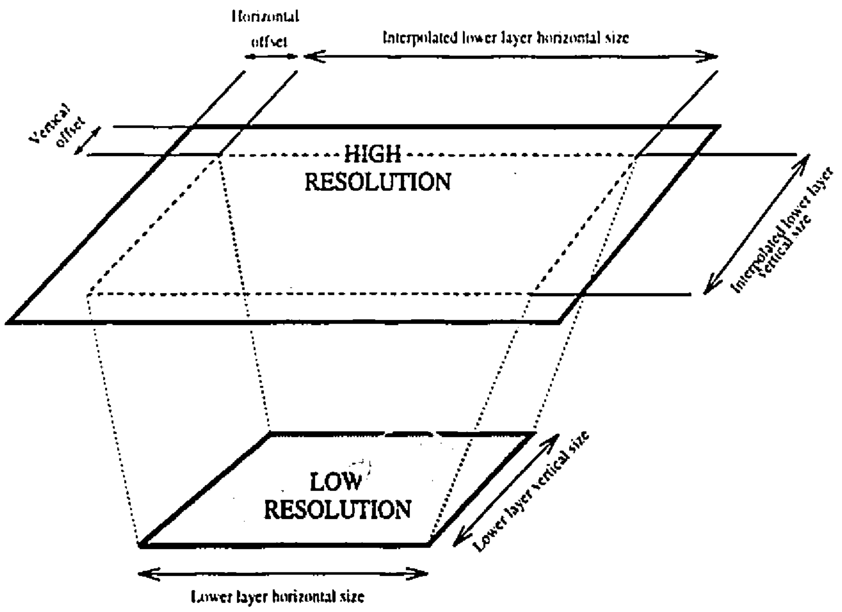

Horizontal offset
Interpolated lower layer horizontal size
Vertical offset
HIGH RESOLUTION
Interpolated lower layer vertical size
LOW RESOLUTION
Lower layer vertical size
Lower layer horizontal size

Figure 2.8: Higher and Lower layer in MPEG-2 scalable system

Let h denote the lower layer horizontal size and v the lower layer vertical size. Furthermore let $h1$ denote the interpolated lower layer horizontal size, and $v1$ the interpolated vertical size. Then h, v, $h1$, $v1$ are related according to:

$$\begin{aligned} h1 &= h\frac{n}{m} \\ v1 &= v\frac{n}{m} \end{aligned} \tag{2.22}$$

where n and m are integer. Furthermore m and n are not fixed, but are chosen according to the application. This offers the possibility of a multiscale conversion. MPEG-2 spatial prediction takes into account the different cases where the lower layer or the upsampled lower layer are interlaced. This is done, whenever needed, as described by [24].

2.3 Temporal and Spatiotemporal Interpolation

Methods that use filtering along the temporal dimension in order to determine the intensity of the missing sample in Fig. 2.3, without any spatial interpolation, are known as temporal methods. When spatial information is used along with temporal filtering, the method will be called a spatiotemporal interpolation.

2.3.1 Fixed Temporal and Spatiotemporal Interpolation

Temporal pixel repetition in equation 2.23, linear temporal filtering in equation 2.24 and linear spatiotemporal filtering in equation 2.25 are the simplest forms of fixed temporal and spatiotemporal filtering.

$$\hat{I}(i,j,t) = I(i,j,t-1) \tag{2.23}$$

$$\hat{I}(i,j,t) = \frac{I(i,j,t-1) + I(i,j,t+1)}{2} \tag{2.24}$$

$$\hat{I}(i,j,t) = \frac{I(i,j-1,t) + I(i,j+1,t) + I(i,j,t-1)}{3} \tag{2.25}$$

Temporal interpolation, which creates the missing lines of each field by taking the average of the lines on the preceding and subsequent fields, gives a very good vertical resolution in the absence of motion, but seriously distorts moving areas as indicated by [39]. This filtering also causes temporal smoothing. Capodiferro [1] used the spatiotemporal filtering and found that though it has a robust performance, it suffers from spatiotemporal blurring effects. That's why he proposes to replace it by median filtering:

$$\hat{I}(i,j,t) = \text{median}\left[I(i,j-1,t), I(i,j+1,j,t), I(i,j,t-1)\right] \tag{2.26}$$

where the median operator extracts the intermediate value between its three arguments. The median filter is found to be very attractive because of its computational simplicity: only logical operations are involved. Furthermore it has the advantage of preserving the edges, which results in sharper images.

All these filters are optimally used in the absence of motion, and should motion be present more complex filters should be introduced, filters which should take into account the motion in the sequence.

2.3.2 Motion Adaptive Interpolation

Since a time-varying sequence generally consists of a fixed background and some moving objects, it is judicious to exploit this characteristic. Motion adaptive processing is a technique where a motion detection is performed and different filtering algorithms are used in moving and stationary parts of the image, as discussed in [10]. A fixed linear temporal interpolator is used in fixed areas. In moving areas this kind of interpolation performs poorly as previously mentioned, and a spatial interpolator is used in these areas. The motion detector generates a motion index M that is based on the interframe difference in a neighborhood of the sample to be interpolated. In order to achieve continuity in the interpolation process, a weighted sum of the two types of interpolation forms the interpolator output. The weights are determined by a normalized version of the motion indicator. Let $I_t(i,j,t)$ and $I_m(i,j,t)$ correspond respectively to the result of the temporal interpolation and spatial interpolation at $\mathbf{x} = (i,j,t)$, then the motion adaptive interpolation is given by:

$$\hat{I}(i,j,t) = \alpha(M)I_m(i,j,t) + (1-\alpha(M))I_t(i,j,t) \tag{2.27}$$

where $\alpha(M)$ relates the motion index to the weighting coefficients.

Motion adaptive interpolation is also treated in [39]. The proposed movement detector measures both the amount of movement and the amount of stationary detail. Movement is measured by passing the input signal through filters designed to remove all stationary components of the signal. Similarly stationary details are measured by filters designed to remove low vertical frequency components. Both measurement outputs are full-wave rectified. If the ratio of the rectified outputs is greater than unity temporal interpolation is performed (stationary areas); ratios greater than two lead to spatial interpolation (moving areas) and ratios between one and two result in a weighted sum of these interpolator outputs.

Other approaches to motion detection are discussed in [28]. A multigradient motion detection is performed based on relative local measurement of spatiotemporal activity. To interpolate the missing sample at $\mathbf{x} = (i, j, t)$, four gradients are needed:

$$\begin{aligned} D_s &= \; |I(i,j-1,t) - I(i,j+1,t)| \\ D_t &= \; |I(i,j,t-1) - I(i,j,t+1)| \\ D_1 &= \; |I(i,j+2,t-1) - I(i,j-2,t+1)| \\ D_2 &= \; |I(i,j-2,t-1) - I(i,j+2,t+1)| \end{aligned} \tag{2.28}$$

From these gradients, a set of relative activities are defined:

$$a_i = \frac{D_i}{\sum D_i} \tag{2.29}$$

with $\sum D_i = D_t + D_s + D_1 + D_2$. The interpolated value is then given by:

$$\hat{I}(i,j,t) = k_1 \cdot I_t(i,j,t) + k_2 \cdot I_s(i,j,t) + k_3 \cdot I_1(i,j,t) + k_4 \cdot I_2(i,j,t) \tag{2.30}$$

where the $I_i(i,j,t)$ are determined by what is called a "hierarchical interpolation". The following constraints are imposed: $\sum k_i = 1$ and $k_i a_i = cte$.

Wang and Mitra[37] also treat motion adaptive interpolation of interlaced sequences. For each processing block, motion detection is performed by comparing the nearby even samples in an analysis block with the corresponding even samples in the previous field. The measure of motion is taken as the absolute difference between the latter samples. If the difference is less than a certain threshold, corresponding samples in previous field are replicated, otherwise spatial pattern interpolation is performed.

Schamel [33] uses motion adaptive interpolation for interlaced to progressive conversion. In stationary parts of the image, a spatiotemporal interpolation is performed. This is done using a diamond-shaped spatial filter applied on the samples of two downsampled adjacent fields. In moving parts the interpolated frame is obtained from one downsampled field using a two-dimensional filter. The author observes that correct motion detection has a great influence on the performance of the whole system. General motion detectors at field n, of interlaced sequences, are based on the difference between the previous and next fields $(n-1)$ and $(n+1)$. However downsampling with interlaced pattern causes aliasing and incorrect decisions of the motion detector may result. For this "three-dimensional motion detection" at field n is used. First, fields $(n-1)$ and $(n+1)$ are interpolated by taking into account the spatial sampling offset of the neighboring fields and using for each field its previous and next field. Then, a frame difference signal is calculated from the interpolated $(n-1)$ and $(n+1)$ fields. The motion detector will hence operate on this difference. As in [10], Schamel is also in favor of the use of "soft" adaptive switching between the two interpolation mode, and this instead of a "hard" switching in order to avoid artifacts caused by the switching to the motion interpolation algorithm when slow motion is detected; a switch which causes a prompt decrease of spatial resolution and hence results in visible blurring. Wang and Mitra [37] also treat motion adaptive interpolation of interlaced sequences.

2.3.3 Motion Compensated Interpolation Methods

Motion estimation

Motion compensation is used in interpolation to improve the quality of moving interpolated areas in the picture. Motion estimation is needed in order to accomplish this interpolation. Spatial motion estimation methods can be classified into two main categories:

- matching based methods,
- spatiotemporal gradient based methods.

All these methods rely on several assumptions: the invariance of luminance along trajectory, local constancy of velocity, and the absence of noise and occlusions. Matching techniques are divided in two categories: block matching and structure matching methods. In block matching, 2-D blocks serve as the elements for matching while in structure matching methods, contour detection is used in order to extract the individual objects of the scene; the extracted objects are than used as the objects for matching. The matching is done through a search in a certain "search window". A cost function is optimized in order to indicate that the search has come to a local optimal solution. The cost function is nothing but a correlation function, like the mean square error or the interframe difference between the block used in the search and the blocks in the "search window" of the neighbor field. Luminance invariance along trajectory is a fundamental assumption in spatiotemporal gradient based methods, and is used to derive the equation of the motion constraint in the continuous 2.31 and discrete 2.32 case:

$$\mathbf{v} \cdot \nabla_{\mathbf{x}} I_c + \nabla_t I_c = 0 \tag{2.31}$$

$$\mathbf{d} \cdot \Delta_{\mathbf{x}} I + \Delta_t I = 0 \tag{2.32}$$

where I_c and I are respectively the continuous and discrete luminance, **d** and **v** are respectively the displacement vector and the motion vector, $\Delta_{\mathbf{x}} I$ and $\Delta_t I$ are a discrete approximation of the spatial and temporal derivatives:

$$\nabla_{\mathbf{x}} I_c \approx \Delta_{\mathbf{x}} I \tag{2.33}$$

$$\frac{\partial I_c}{\partial t} \approx \Delta_t I \tag{2.34}$$

Horn and Schunk [16] determine **v** by the following optimization:

$$\min_{\mathbf{v}} \int\int (e_m^2(\mathbf{v}) + \lambda \cdot e_x^2(\mathbf{v}))\, dx\, dy \tag{2.35}$$

where $e_x^2(\mathbf{v})$ is the smoothing constraints:

$$e_m^2(\mathbf{v}) = \|\nabla_{\mathbf{x}} v_x\|^2 + \|\nabla_{\mathbf{y}} v_y\|^2 \tag{2.36}$$

$$e_x^2(\mathbf{v}) = |\mathbf{v} \cdot \nabla_{\mathbf{x}} I_c + \nabla_t I_c|^2 \tag{2.37}$$

Motion estimation with detection of occlusion areas leads to a better estimated motion vectors than the one obtained by Horn and Schunk motion estimation. This was observed and proved in [6].

Application to standards and frame rate conversion

Ernst and Reuter propose a motion compensated scheme [11], [12], [32] which divides the velocity ranges into separate regions; each region is assigned a fixed filter. This will allow an adaptive interpolation. Furthermore motion compensated interpolation is done using filters which have more than two taps, which requires more than one field store but results in a better passband approximation and a higher stopband. The applications are intended for 50 to 60 Hz field rate conversion and for standards conversion between HDTV and TV.

A motion compensated field rate 2:1 upconversion of interlaced HDMAC system is described in [15]. The motion compensated conversion from 50 Hz to 100 Hz will be done by generating the intermediate fields at 100 Hz by the halfsum of the 50 Hz fields with a 3 tap symmetrical interpolator. However the measured motion vectors are only convenient for operations at 50 Hz and are not sufficiently reliable for operations at 100 Hz. That's why we need to have a fall-back processing for some cases. For this reason the upconverter includes the following three branches:

- a stationary branch which becomes active on the blocks processed in the 80ms mode of the HDMAC system. It allows to double the field rate by a simple frame repetition.
- a motion compensated branch which becomes active on the blocks processed in the 40 ms motion compensated mode of the HDMAC system. It uses a temporal interpolation in the movement direction in order to generate one field between two adjacent fields of the input image.
- a fall-back branch which becomes active on the blocks processed by the 20 ms mode of the HDMAC system. The field rate conversion on this branch is achieved by a vertical intra-field interpolation followed by a temporal averaging.

The motion estimation uses coherency checking in order to get more reliable motion vectors. This is achieved by comparing the selected vector on the current field to those of previous fields, should the difference be relatively large the current vector is disregarded and other vector choices are considered.

Motion compensated field rate conversion by a noninteger ratio is presented in [21]. A block-based motion estimation algorithm is used, where the square of the DFD is minimized using an iterative procedure. The motion compensated interpolation is then carried out to interpolate the pixels values in the missing field. The intensity

$I(\mathbf{x}, t)$ of the missing pixel is given by:

$$\hat{I}(\mathbf{x}, t) = (1-\alpha)I(\mathbf{x} - \alpha\mathbf{D}, t - \alpha T) + \alpha I[\mathbf{x} + (1-\alpha)\mathbf{D}, t + (1-\alpha)T] \qquad (2.38)$$

where $\mathbf{D}$ is the displacement estimate, T denotes the time interval between the existing fields E_1 and E_2, and α ($\alpha < 1$) is a weighting factor depending on the temporal position of the missing field M with respect to E_1 and E_2. Since the displaced pel (at location $\mathbf{x} - \alpha\mathbf{D}$ or $\mathbf{x} + (1-\alpha)\mathbf{D}$) do not necessary fall on the grid, a bi-linear spatial interpolation is used.

A frame rate up-conversion of progressive sequences is considered in [2]. The algorithm is designed for a conversion by a noninteger factor of 1.5 (from 50 to 75 Hz). Two out of three field of the 75 Hz sequence are obtained through repetition of neighboring fields of the 50 Hz sequence. The third one is located half way between two 50 Hz fields, and is interpolated using motion compensation.

The problem of motion compensated upconversion of 24 frames/sec digitized motion pictures to 60 frames/sec digital video signals is addressed in [20]. An intermediate temporal grid of 120 Hz is used. However this grid only intervenes in the design of the interpolation filter: a cut off frequency of 0.1 relative to 120 Hz is required. Furthermore, the motion vector for a pixel $\mathbf{x} = (i, j, t)$ of a 60 Hz frame is estimated on the basis of the 24 Hz signal. The motion compensated pixel value $\hat{I}(i, j, t)$ is then given by:

$$\hat{I}(i, j, t) = \sum h(l) e_{ijt}(l) \qquad (2.39)$$

where l is determined by the order of the temporal filter which is also the number of pixels lying on the linear trajectory that are used in the filtering; $e_{ijt}(l)$ are the values of the pixels along the trajectory that passes through $\mathbf{x}$. If the pixel on the trajectory lie in a 24 Hz frame, its corresponding $c_{ijt}(l)$ is obtained by spatial interpolation, otherwise $c_{ijt}(l)$ is set to zero.

Statistical methods can be used to perform a more robust motion compensated interpolation than *ad hoc* methods [5]. For this K displacement vectors are computed using K different motion estimation methods. For each motion vector (which correspond to a different method) corresponds two pixel value: the forward and backward predicted pixel. Hence K motion vectors give $2K$ pixels which are ordered according to their values in a 2D vector $\mathbf{E_0}(\mathbf{x}, t)$. The interpolated sample value are then given by:

$$\hat{I}(\mathbf{x}, t) = \mathbf{C} \cdot \mathbf{E_0}^T(\mathbf{x}, t) \tag{2.40}$$

where $\mathbf{C}$ is a row matrix of symmetrical weighting coefficients, which sum is equal to one. As the quality of the used motion vectors drops, as the ordered pixels values in $\mathbf{E_0}(\mathbf{x}, t)$ will be distributed around their average. Thus in order to decrease the influence of bad motion vectors on the interpolation, we need to decrease the influence of the extreme ordered pixels. This is done through the statistical coefficients of $\mathbf{C}$.

Application to deinterlacing

Motion compensated deinterlacing is an application of motion compensation. A motion compensated deinterlacing algorithm is presented in [4]. The motion estimated is carried out on two fields of the same parity (even or odd) with a half pixel accuracy. These motion vectors will then be projected on the interpolated field, of opposite parity which lies in the middle. Labeling of covered/uncovered areas during the motion estimation step will be performed. Field repetition will be used in still areas. In moving areas, the compensated mode is used. A vertical intra-field interpolation is used in case of erroneous vector and in the fall back case, which corresponds to uncovered areas or when the vector projections conflict.

Vandendorpe proposes an improved motion-compensated interlace to progressive conversion [34]. The author exploits a generalized form of the Nyquist theorem which states that any signal whose bandwidth is limited to $1/T$ can be recovered from 2 sequences of samples taken at a rate greater than $1/T$ and having different sampling phases. This can be applied to an interlaced sequence where the vertical adjacent samples are sampled at a rate of $1/T$, and which spectrum can fairly be considered bandlimited to $1/T$. The theorem requires two versions of the initial spectrum: the first one will be obtained from the field that is being deinterlaced. To determine the second sequence, the pixels of the following field will be backward projected using motion estimation. If the motion can be considered as translational these pixels will form the second sequence and the theorem can apply to the vertical direction to recover the original analogue signal. This analogue signal will finally be downsampled to form the desired progressive sequence.

In conclusion we find that motion compensated methods give better interpolated image quality than fixed spatial and fixed spatiotemporal interpolation. Indeed, spatial interpolation uses the information contained within one field, while motion compensation uses the information along the trajectory. Furthermore, though fixed spatiotemporal interpolation uses temporal information, the information contained along the trajectory is more accurate with respect to interpolation. In the following chapter we will investigate adaptive (switched or weighted) motion compensated methods and spatial/temporal methods. Spatial methods should tend to perform better when motion is poorly estimated. On the other hand temporal method should give better results when motion is relatively small.

Chapter 3

Adaptive Spatial and Motion Compensated Interpolation Methods

3.1 Motion Compensated Interpolation

3.1.1 Motivation of Motion Compensated Interpolation

To be able to evaluate the advantages that motion compensated filtering offers with respect to spatial filtering, one has to analyze the three dimensional spectrum of a video sequence [14]. For this we will use a simplified model, where we assume global constant velocity. The assumption of a local constant motion is, in general, a valid one over two fields. Let $I_c(x, y, 0)$ describe the intensity of the first frame in the sequence, and let $(v_x, v_y)^T$ describe its velocity then:

$$I_c(x, y, t) = I_c(x - v_x t, y - v_y t, 0) \tag{3.1}$$

The Fourier Transform of $I_c(x, y, t)$ is defined as:

$$\mathcal{I}_c(w_x, w_y, w_t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I_c(x, y, t)e^{(-j(w_x x + w_y y + w_t t))}\,dx dy dt \qquad (3.2)$$

using (3.1) in (3.2) gives:

$$\begin{aligned}\mathcal{I}_c(w_x, w_y, w_t) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I_c(x, y, 0)e^{(-j(w_x x + w_y y))}\,dx dy \int_{-\infty}^{\infty} e^{(-j(w_x v_x + w_y v_y + w_t)t)}\,dt \\ &= \mathcal{I}_{c0}(w_x, w_y)\delta(w_x v_x + w_y v_y + w_t)\end{aligned} \qquad (3.3)$$

We recognize in the first term of equation (3.3) the spatial Fourier Transform of $I_c(x, y, 0)$. The second term is non null if and only if:

$$w_x v_x + w_y v_y + w_t = 0 \qquad (3.4)$$

which is the equation of a plane perpendicular to the vector $(v_x, v_y, 1)^T$: this is the support of the three dimensional spectrum. As an example, consider a constant vertical velocity of 1 pel/s. The intersection of the support with the plane $w_x = 0$ is a line shown in Fig. 3.1(a):

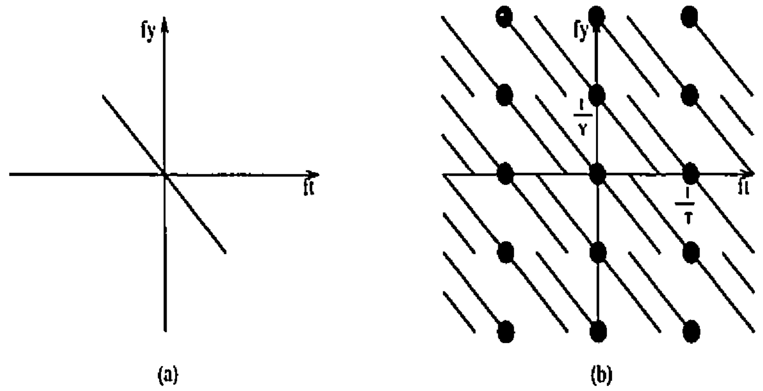


Figure 3.1: (a)Analog Signal (b)Sampled Signal

In order to transmit digital images, sampling of an analogue sequence is required. Assuming an orthogonal lattice we will get replicas as seen in Fig. 3.1(b). If usual low-pass filtering is used at the receiver to extract the analogue pre-sampled signal, low-pass pre-filtering will be required at the transmitter in order to avoid aliasing which appears in Fig. 3.2(a).

This will cause the loss of high spatial frequency components, and hence will reduce

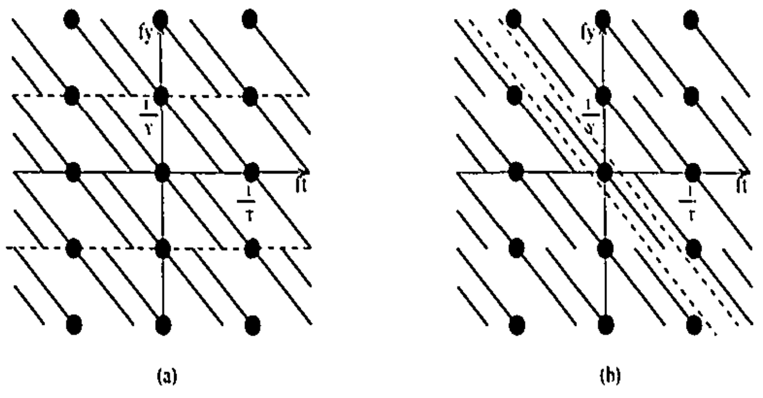


Figure 3.2: (a)Spatial Filtering (b)Motion compensated Filtering

the spatial resolution causing visible blur at low velocity. On the other hand if we use a filter which is oriented in the direction of the velocity, we can avoid aliasing at the reconstruction without the need of pre-filtering, and hence preserve high spatial frequency components as seen in Fig. 3.2(b). In order to achieve such oriented filtering, it is sufficient to filter along the estimated trajectory. It is obvious that the quality of such filtering will depend on the accuracy of the trajectory estimation.

3.1.2 Motion Estimation

In this section we formulate the motion estimation problem [8] , which will lead us to present the outline of the motion estimation algorithm that was used through this thesis.

We define the trajectory of a certain point in a video sequence, as the curve that joins the positions of this point in the three dimensional spatiotemporal space. The ending points of a trajectory are respectively the time when the point becomes visible and the time when the point disappears from the field of view. The mathematical model of a trajectory can be defined by the function $\mathbf{c}(t;\mathbf{x},t_0)$ which gives the spatial position $(x(t),y(t))$, at time t, of a point $\mathbf{x}=(x(t_0),y(t_0))$, at time t_0. An illustration of a trajectory is shown in Fig. 3.3. For a time $t \neq t_0$ we can define a subset $V(t;t_0)$ of the image frame at time t_0, consisting of pixels that are visible over the entire time interval between t and t_0. We define the displacement field at time t_1 as the field $\mathbf{d}(t_1;\mathbf{x},t_0)$ which gives the displacement of each pixel between t_0 and t_1. We only define $\mathbf{d}$ for pixels that in $V(t_1;t_0)$. From this it follows that:

$$\mathbf{d}(t_1;\mathbf{x},t_0)=\begin{cases} \mathbf{x}-\mathbf{c}(t_1;\mathbf{x},t_0) & \text{if } t_1 < t_0; \\ \mathbf{c}(t_1;\mathbf{x},t_0)-\mathbf{x} & \text{if } t_1 > t_0; \end{cases} \quad \mathbf{x}\in V(t_1;t_0) \tag{3.5}$$

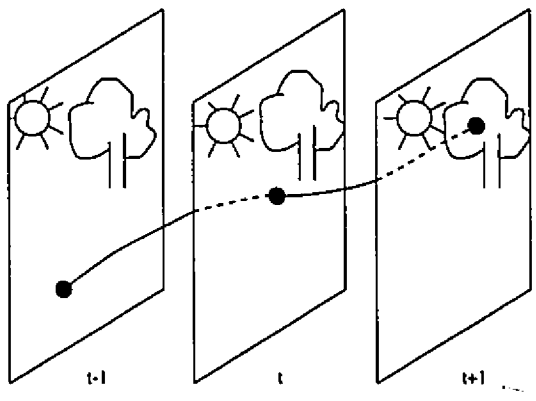


Figure 3.3: Motion trajectory

The velocity field gives the rate of change of position, at a given time, for each pixel in the frame:

$$\mathbf{v}(\mathbf{x},t_0)=\frac{d\mathbf{c}}{dt}(t;\mathbf{x},t_0)\bigg|_{t=t_0} \tag{3.6}$$

The displacement field is obtained from the velocity field by integration. If $t_1 > t_0$ and $x \in \mathbf{V}(t_1; t_0)$ then:

$$\mathbf{d}(t_1; \mathbf{x}, t_0) = \int_{t_0}^{t_1} \mathbf{v}(\mathbf{c}(t; \mathbf{x}, t_0), t)\, dt \tag{3.7}$$

If the motion is uniform over the period of time extending from t_0 to t, i.e $\mathbf{v}(\mathbf{c}(t_1; \mathbf{x}, t_0), t_1) = \mathbf{v}(\mathbf{x}, t_0)$, $\forall t_1 \setminus t_0 \leq t_1 \leq t$, then the displacement is given by $\mathbf{d}(t_1; \mathbf{x}, t_0) = \mathbf{v}(\mathbf{x}, t_0) \cdot (t_1 - t_0)$. Then using equation (3.5) we get:

$$\mathbf{c}(t_1; \mathbf{x}, t_0) = \mathbf{x} + \mathbf{v}(\mathbf{x}, t_0) \cdot (t_1 - t_0), \qquad x \in \mathbf{V}(t_1; t_0) \tag{3.8}$$

if $\mathbf{x}$ is a point at time t and $\mathbf{x}^k$ is a point at time t^k on the same trajectory then under the assumption of constant velocity we have:

$$\mathbf{x}^k = \mathbf{x} + \mathbf{v}(\mathbf{x}, t) \cdot (t_k - t) \tag{3.9}$$

The motion estimator algorithm is a pixel based one [9]. The estimation is done using iterative algorithms on a pyramidal scheme and a sample variance matching error is minimized. Regularization of motion vectors was also carried on. The motion estimator program used was written by Michel Chahine.

3.2 Deinterlacing

In the background chapter several methods that perform deinterlacing have been outlined. However the performance of most of spatially based methods always depends on the presence or absence of motion. On the other hand, the performance of motion compensated methods is limited by the quality of the estimated motion vectors. Next we introduce two schemes that combine the advantages of spatial and motion compensated interpolation. These schemes are based on spatial and temporal differences. The first scheme uses an average of these differences as coefficients which decide the contribution of each of the spatial and motion compensated mode. The other scheme switches between the spatial and motion compensated mode depending on which difference is larger.

3.2.1 Algorithm Description

Motion Compensated Interpolation

As a first step, we estimate the motion vectors at the missing samples that are to be interpolated. The motion estimation has been carried out by using the sample variance over three field: the one we are deinterlacing at time t, the previous one at $t-1$ and the one that follows at $t+1$. The motion vector that results from the motion estimation is **v** in the forward direction and $-$**v** in the backward direction. We then determine the trajectory by projecting the motion vector to the past and next field. In Fig. 3.4 we show the interpolation of point **a**, where the trajectory intercepts the

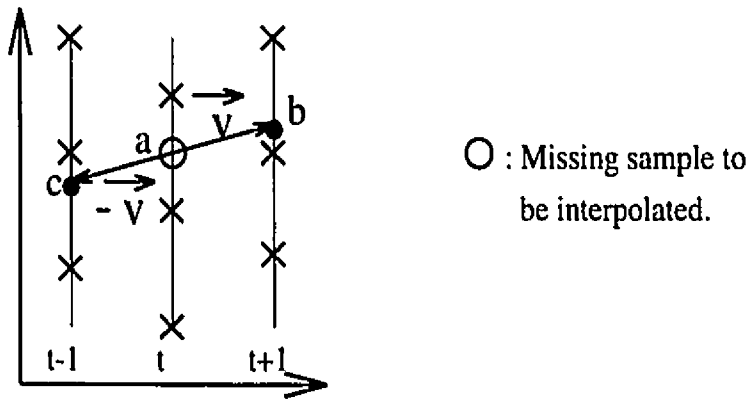


Figure 3.4: Motion Compensated Deinterlacing

previous field at point **c** and the next one at point **b**. It is important to note at this point, that the motion estimation and the trajectory determination were both done under the assumption of constant velocity. Since most of the time, point **b**, and **c** will not fall on the grid, their intensity must be spatially interpolated. For this we have used Keys cubic convolution [19], which is suited for this purpose, both in quality and in the fact that it allows us to determine the intensity at any "real" position. We denote I_a the interpolated intensity at point **a** and I_b the interpolated intensity at point **b**. The final step is the trajectory filtering. This will be done using a two tap FIR filter, where the coefficients assume the values: $[\frac{1}{2} \quad \frac{1}{2}]$, yielding:

$$I_a = \frac{1}{2} \cdot I_b + \frac{1}{2} \cdot I_c \tag{3.10}$$

which can be put in a general form:

$$\hat{I}(x,y,t) = \frac{1}{2} I(x - v_x t, y - v_y t, t-1) + \frac{1}{2} I(x + v_x t, y + v_y t, t+1) \tag{3.11}$$

Spatial Deinterlacing

Spatial deinterlacing was done by one-dimensional vertical filtering. To determine the specifics that the filter should have we will briefly present the problem. Our target is to increase the sampling rate by a factor of two of a sequence $i(n)$ which was originally obtained from a continuous-time signal $i_c(t)$ by $i(n) = i_c(nT)$. We denote the new sequence i_i, such that $i_i(n) = i_c(nT')$ where $T' = \frac{T}{2}$. It is clear that:

$$i_i(n) = i(n/2) = i(nT/2), \qquad n = 0, \pm 2, \pm 4, \cdots \tag{3.12}$$

To achieve this conversion we need a lowpass filter with a gain of 2 and cutoff frequency $\frac{\pi}{2}$, this process is shown in Fig. 3.5. The system to the left is called a sampling rate expander:

$$i_e(n) = \begin{cases} i(n/2), & n = 0, \pm 2, \pm 4, \cdots \\ 0, & \text{otherwise,} \end{cases} \tag{3.13}$$

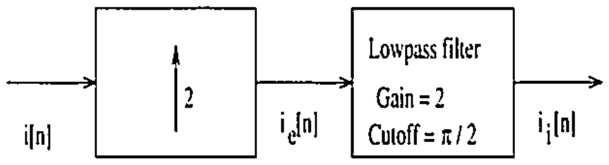


Figure 3.5: Upsampling conversion by a factor of 2

The lowpass filter that we used was a 31 tap symmetric FIR which was designed using N-step Newton method [22]. The filters coefficients are shown in table 3.1. Note that the filter is symmetric so that $h(n) = h(-n)$.

The frequency response of this filter is shown in Fig. 3.6. It can be seen that

Filter's Coefficients			
h[0]=1.0605	h[1]=0.6205	h[2]=-0.5552E-01	h[3]=-0.1678
h4]=0.4161E-01	h[5]=0.6438E-01	h[6]=-0.2491E-01	h[7]=-0.2132E-01
h[8]=0.9952E-02	h[9]=0.3915E-02	h[10]=-0.1065E-02	h[11]=0.8792E-03
h[12]=-0.2588E-02	h[13]=-0.1070E-02	h[14]=0.2509E-02	h[15]=0.2826E-03

Table 3.1: Filter used in the upconversion.

$H(e^{jw})|_{w=0}$ is as desired equal to 2 (which is also the sum of the coefficients). The $3db$ cut-off frequency is at $f_c = 1.5419\ rad$ which is very close to the desired one of $\frac{\pi}{2}$. In order to preserve the original information in the image, we haven't extended the filtering to the existing samples. Thus only the missing ones were subject to vertical filtering.

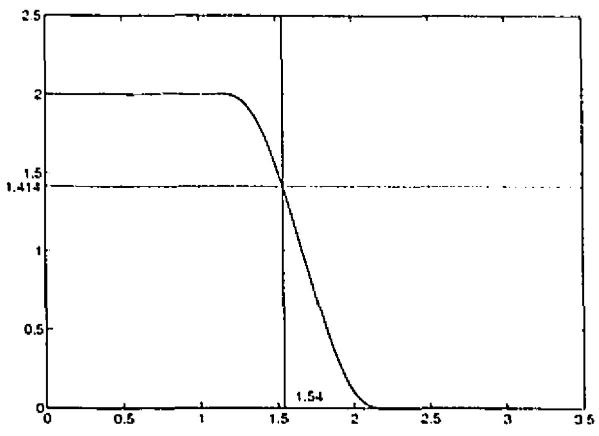


Figure 3.6: Frequency response of N-step Newton method FIR

Weighted sum of spatial and motion compensated interpolation

We define the spatial error $E_s(i,j,t)$ and the motion error $E_m(i,j,t)$ at point $\mathbf{x} = (i,j,t)$ as:

$$E_m(i,j,t) = \sum_{k=i-2}^{i+2} |I(k - v_i, j - v_j, t-1) - I(k+v_i, j+v_j, t+1)| \quad (3.14)$$

$$E_s(i,j,t) = \sum_{k=i-2}^{i+2} |I(k, j-1, t) - I(k, j+1, t)| \quad (3.15)$$

The choice of a window of five pixels in the definition of both errors is done in order to achieve smoothness in the interpolated regions. Had we chosen a window of one pixel, we would have ended up with discontinuities and staircase effects. Let $I_m(i,j,t)$ correspond to the interpolated value obtained in the first paragraph (I_a), and let $I_s(i,j,t)$ be the interpolated value obtained by the filter described in table 3.1. The adaptive method of combining both spatial and motion compensated filtering will be done according to the following weighted sum:

$$\hat{I}(i,j,t) = \frac{E_m(i,j,t)}{E_m(i,j,t) + E_s(i,j,t)} \cdot I_s(i,j,t) + \frac{E_s(i,j,t)}{E_m(i,j,t) + E_s(i,j,t)} \cdot I_m(i,j,t) \quad (3.16)$$

From equation (3.15) we see that whenever $E_m(i,j,t) = 0$ we will have $\hat{I}(i,j,t) = I_m(i,j,t)$, and similarly whenever $E_s(i,j,t) = 0$ then $\hat{I}(i,j,t) = I_s(i,j,t)$, which corresponds to our adaptive goal. Hence the weighted sum will emphasize the spatial interpolation in regions where the motion estimation will have resulted in less reliable motion vectors.

Smaller error decision interpolation

Using the same definitions as in the previous paragraph, we have carried this interpolation, which is easier to implement than the previous one, according to:

$$\hat{I}(i,j,t) = \begin{cases} I_s(i,j,t) & \text{if } E_m(i,j,t) \geq E_s(i,j,t) \\ I_m(i,j,t) & \text{otherwise,} \end{cases} \tag{3.17}$$

3.2.2 Simulations and Results

Next, we compare the previously described methods, i.e. motion compensated deinterlacing, spatial deinterlacing, weighted sum of spatial and motion compensated deinterlacing and smaller error decision deinterlacing. The evaluation will be done using only a visual criterion, due to the absence of original progressive sequences. For this the interpolated sequences will be displayed on the "VIDS", which is a video display having a studio resolution and allow displaying the sequences either field by field or in the palindromic mode.

Courtepointe

The scene of this sequence consists of a piece of fabric which is undergoing a translational horizontal motion from left to right. The fabric is composed of several different patterns. The most critical pattern (which we call stripes pattern) is the one with diagonal intersected lines (in the 2nd, vertically, complete square from the

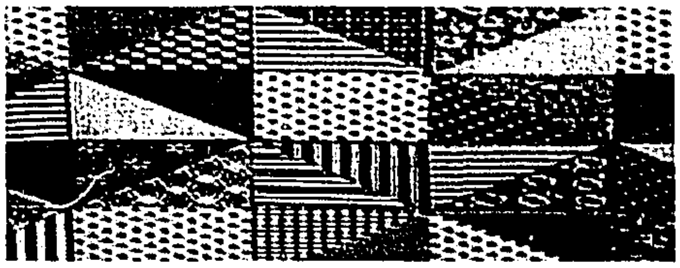


Figure 3.7: Original Courtepointe: field #4

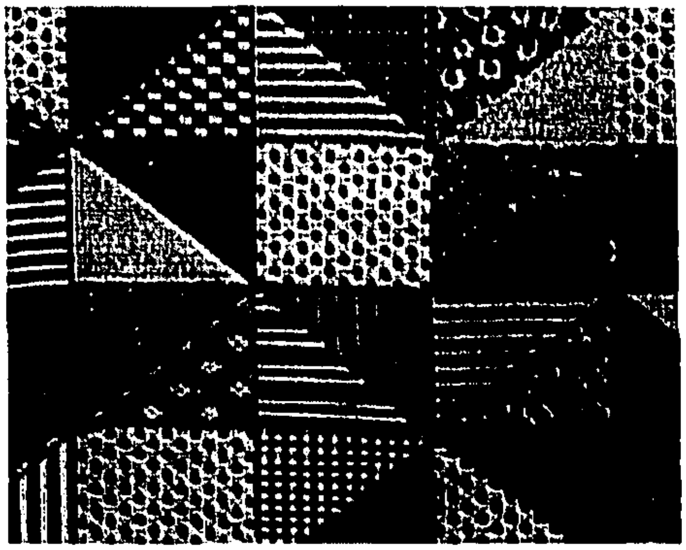


Figure 3.8: Spatial deinterlacing: field #4

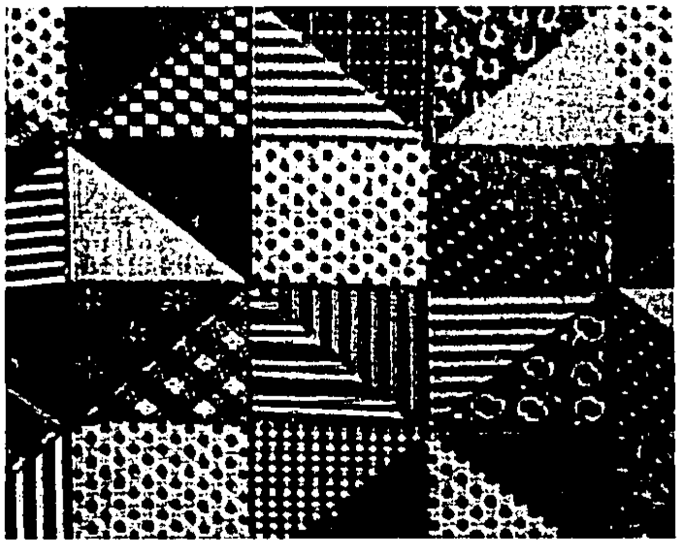


Figure 3.9: Smaller Error Decision Deinterlacing: field #4

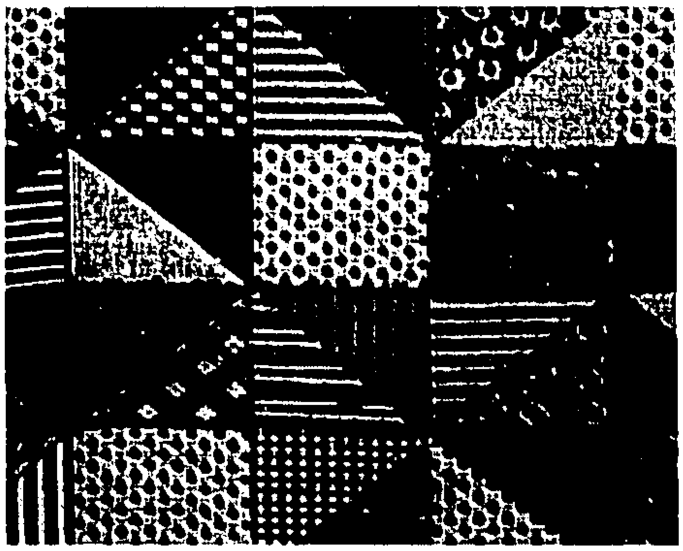


Figure 3.10: Weighted Sum Deinterlacing: field #4

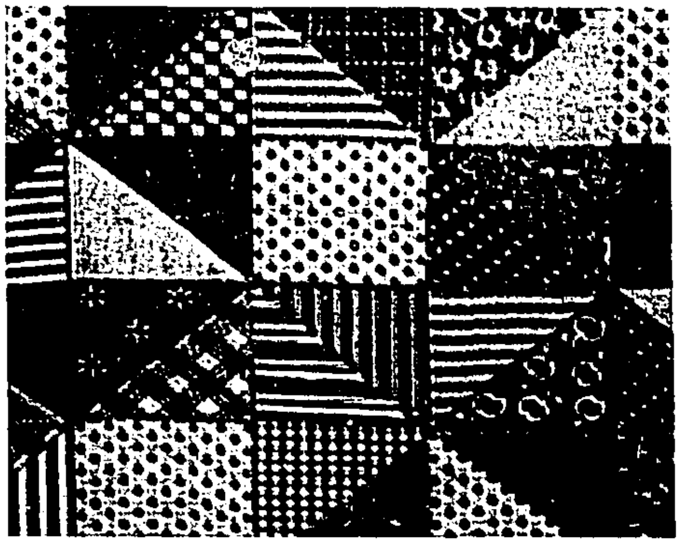


Figure 3.11: Motion Compensated Deinterlacing: field #4

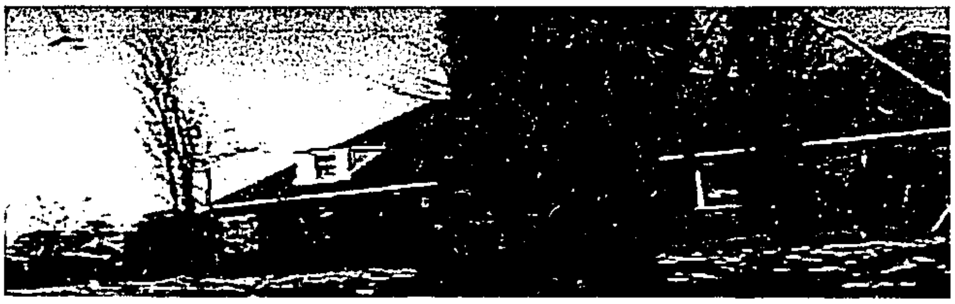


Figure 3.12: Original Flowergarden: field #4

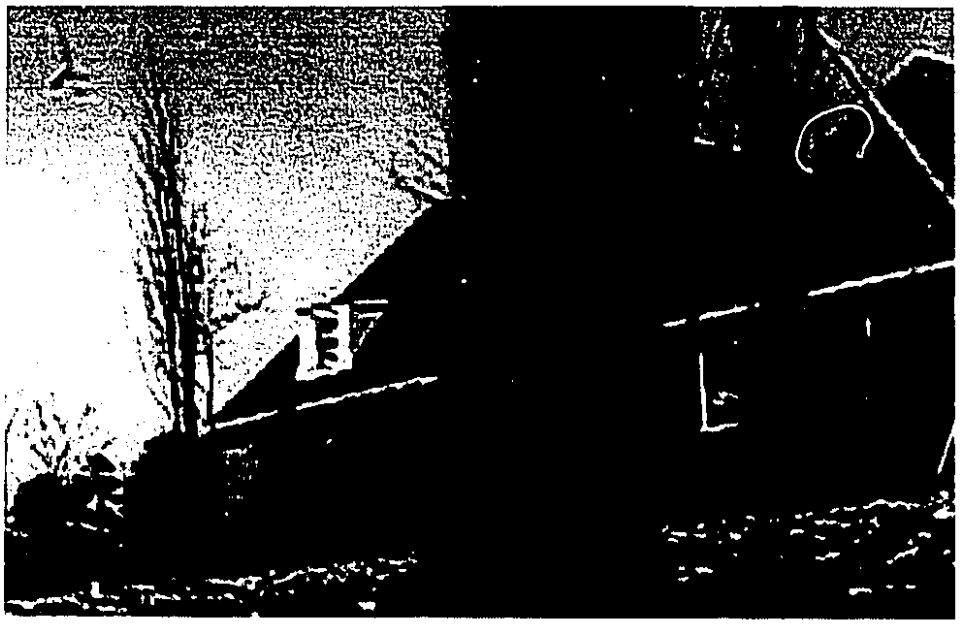


Figure 3.13: Spatial deinterlacing: field #4

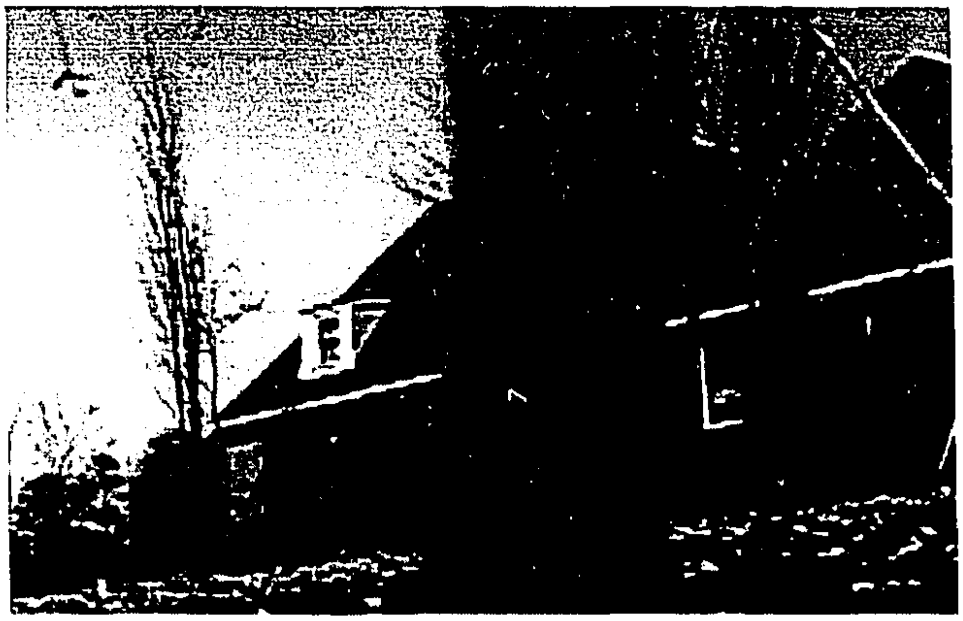


Figure 3.14: Smaller Error Decision Deinterlacing: field #4

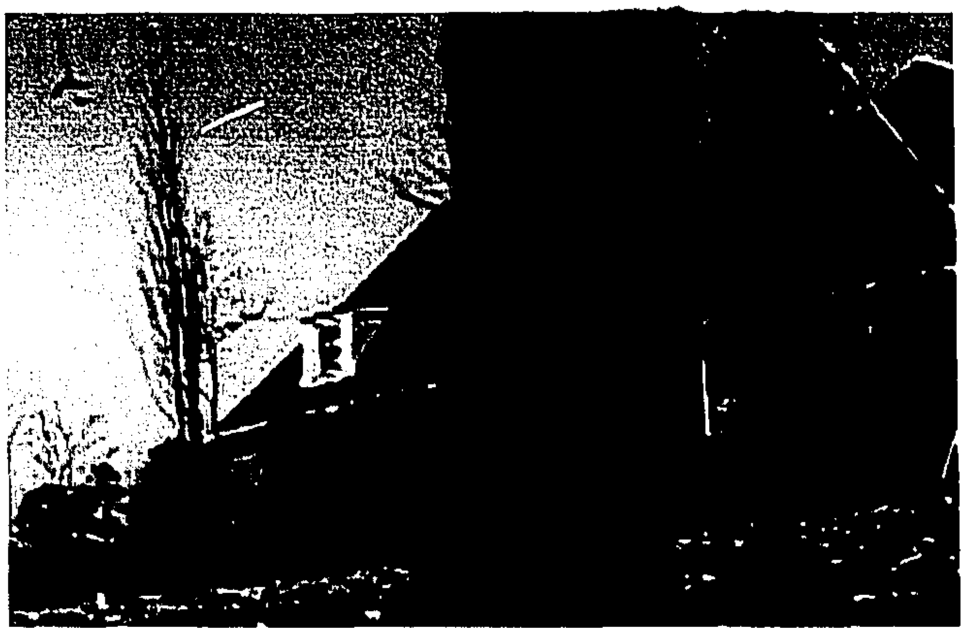


Figure 3.15: Weighted Sum Deinterlacing: field #4

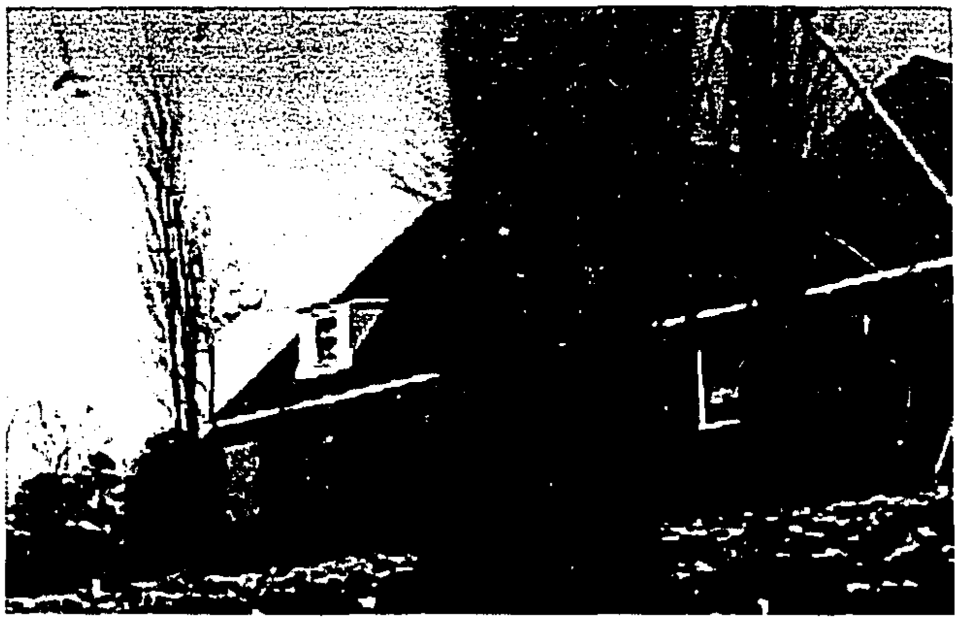


Figure 3.16: Motion Compensated Deinterlacing: field #4

bottom left). We show in Fig. 3.7 the original field #4 of courtepointe. The spatial deinterlacing (Fig. 3.8) does not preserve good resolution and results in a smoothing of edges. Artifacts are visible at diagonal edges which take the form of a staircase, the stripes pattern loses its resolution, and the checkerboard squares adopt irregular sizes. These distortions are hardly visible in the other methods. Smaller error (Fig. 3.9) and motion compensation (Fig. 3.11) are slightly better than the others: the stripes pattern resolution is slightly better than the weighted sum one (Fig. 3.10) and the staircase effect at diagonal edges is reduced. Since the motion is almost uniform, motion was well estimated, and motion based methods performed well.

Flowergarden

The scene of this sequence consists of a series of houses in the background, a flowergarden and a tree in the foreground. The motion of this scene is caused by the horizontal panning of a camera. The tree has a relatively large horizontal motion. The original field #4 is shown in Fig. 3.12. As previously, spatial deinterlacing (Fig. 3.13) does not preserve good resolution. The details of the red tiles are blurred and we observe smoothing of edges at windows. These artifacts are not visible in motion compensated deinterlacing (Fig. 3.16) which preserves contours and resolution of the different object in the scene. However due to the occlusion caused by the motion of the tree, bad motion estimation occurs at the edge of the tree-trunk. This causes the tree-trunk edge to be slightly serrated with some trails. Using both smaller error (Fig. 3.14) and weighted sum (Fig. 3.15) removes part of the error caused by the occlusion because the error term E_m will be less than E_s and hence the value obtained by spatial interpolation at these point will be given a higher weight in the case of weighted sum and will be the value itself in the smaller error decision case, which is slightly better in that case.

3.3 Frame Rate Conversion

Frame rate conversion will be done adaptively using temporal interpolation and motion compensated interpolation. We first present motion compensated and temporal filtering, and than we discuss how they can be combined in order to enhance the interpolation performance.

3.3.1 Algorithm Description

The conversion will be done from a progressive format to an interlaced one with 2:1 temporal upconversion. This is shown in Fig. 3.17.

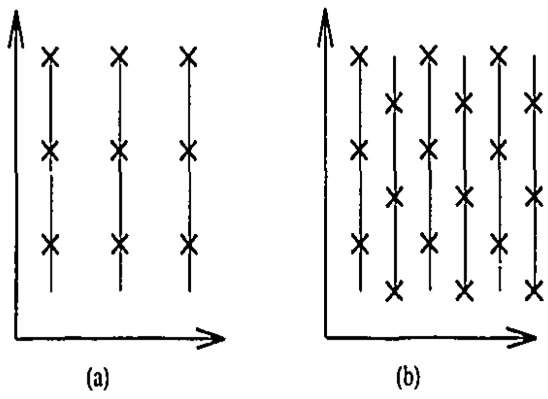


Figure 3.17: (a)Progressive sequence. (b)Upconverted 2:1 interlaced sequence.

Motion compensated conversion

A motion vector will be found for each missing sample in field **B**, by minimizing, over the two fields **A** and **C**, the sample variance at the position of the missing samples in **B**. The determined trajectory passes through **b** and intersect **A** and **C** respectively at **a** and **c**. The intensity at **b** is given by $I_b = (I_a + I_b)/2$, where I_a and I_c have been computed using Keys cubic convolution. In general if $\mathbf{b} = (i, j, t)$, then:

$$I(i, j, t) = I(i - \frac{v_i}{2}, j - \frac{v_j}{2}, t - \frac{1}{2}) + I(i + \frac{v_i}{2}, j + \frac{v_j}{2}, t + \frac{1}{2}) \qquad (3.18)$$

This is illustrated in Fig. 3.18.

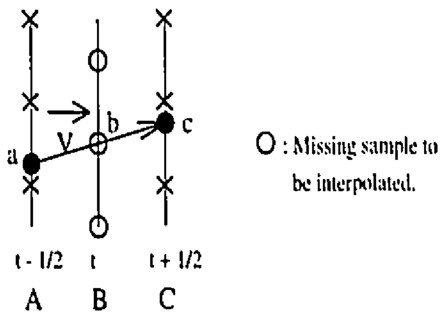
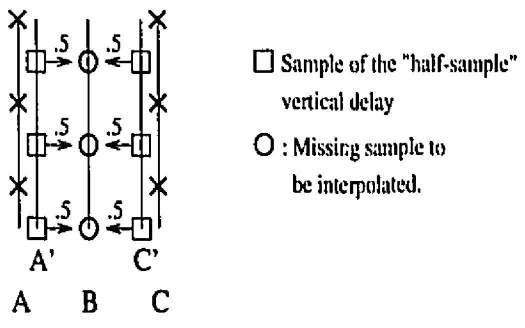


Figure 3.18: Motion Compensated Conversion

Temporal conversion



.5
□ Sample of the "half-sample" vertical delay
O : Missing sample to be interpolated.
A'
C'
A
B
C

Figure 3.19: Temporal Conversion

As a first step we start by creating from both fields **A** and **C** new fields **A'** and **C'**, which will be shifted by half a pixels with respect to the old ones. The new fields will be with "half-sample" vertical delay. Let's consider a discrete-time system with frequency response $H(e^{jw})$ given by:

$$H(e^{jw}) = e^{-jw\Delta}, \qquad |w| < \pi \qquad (3.19)$$

where Δ is a noninteger equal to $\frac{1}{2}$. From [27] we know that such system will have its output $y[n]$ as the values of the bandlimited interpolation halfway between the input sequence values $x[n]$. Hence the ideal filter, we need in order to create the "half-sample" shifted fields, should have the frequency response of the filter in equation (3.19). Note that this filter is a lowpass one, with linear phase $-w/2$. Its cutoff frequency is at $w = \pi$ and its DC magnitude $H(e^{jw})\big|_{w=0} = 1$. Once the new shifted fields are created, we proceed to a temporal filtering with a $[\frac{1}{2} \quad \frac{1}{2}]$ filter, this procedure is illustrated in Fig. 3.19.

Adaptive frame rate conversion

We will use a motion indicator in order to switch between temporal and motion compensated conversion. If we are performing the conversion at point $\mathbf{b} = (i, j, t)$ than we define the motion indicator $\mathbf{M}(i, j, t)$ as:

$$\mathbf{M}(i,j,t) = \frac{1}{9} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} \|\mathbf{v}(k,l,t)\| \tag{3.20}$$

The choice of a window of 3×3, centered at the point we are interpolating, instead of just defining the motion indicator as the magnitude of the motion vector of the former point, was done in order to achieve smoothness and to avoid discontinuities and staircase effects. Let $\mathbf{b} = (i, j, t)$ be the point shown in Fig. 3.18 and 3.19, let $I_m(i, j, t)$ and $I_t(i, j, t)$ denote respectively the motion compensated value at $\mathbf{b}$ and the temporally interpolated one as previously outlined, our scheme of conversion will then have the following form:

$$I(i,j,t) = \begin{cases} I_t(i,j,t) & \text{if } \mathbf{M}(i,j,t) \leq 1 \\ I_m(i,j,t) & \text{if } \mathbf{M}(i,j,t) > 1 \end{cases} \tag{3.21}$$

What equation (3.21) states implicitly, is that whenever the motion indicator, of the displacement between fields **A** and **C**, is less than 1 pel/s (which is equivalent to half a pel/s between **A** and **B**) then temporal filtering is reliable and will be performed in order to avoid the artifacts that motion estimation can produce. This will be discussed later on and will be checked through simulation.

3.3.2 Maximally Flat Digital Filters

The design of the linear phase low-pass filter described in equation (3.19) will be carried out using a symmetrical maximally flat FIR filter with even order (i.e number of coefficients is even). The design of maximally flat FIR filters has been discussed in [25], [31]. Maximally flat FIR filters form a class of filters with flat frequency response in the passband and the stopband. The transfer function of the maximally flat FIR lowpass filter is determined by the order of the filter and the degree of flatness at $w = 0$. We are concerned in the design of a linear phase maximally flat FIR with even order.

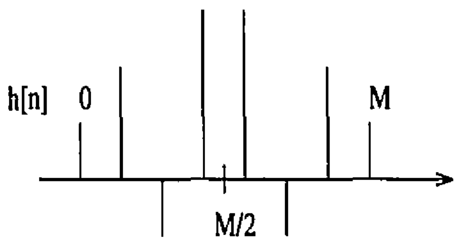


Figure 3.20: General impulse response of a symmetrical even FIR.

The frequency response of linear phase FIR filters with symmetrical impulse response has the form [30]:

$$H(e^{jw}) = e^{-jw(N-1)/2}\left\{\sum_{n=0}^{N/2-1} 2h(n)cos\left[w\left(\frac{N}{2}-n-\frac{1}{2}\right)\right]\right\} \qquad (3.22)$$

where N is the order of the filter and $h(n)$ its impulse response. Letting

$$b(n) = 2h\left(\frac{N}{2}-n\right) \qquad n = 1,2,\cdots,\frac{N}{2} \qquad (3.23)$$

Equation (3.22) becomes:

$$H(e^{jw}) = e^{-jw(N-1)/2}\left\{\sum_{n=1}^{N/2} b(n)cos[w(n-\frac{1}{2}]\right\} \tag{3.24}$$

Denote:

$$H_1(e^{jw}) = \sum_{n=1}^{N/2} b(n)cos[w(n-\frac{1}{2}] \tag{3.25}$$

where $H_1(e^{jw})$ is purely real. The filter of an even order $M = 2N+1$, shown in Fig. 3.20, where M is the number of coefficients of the impulse response $h(n)$, $n = 0,\cdots,2N+1$, is said to be lowpass and maximally flat if equation 3.25 satisfies the following properties:

$$H_1(e^{jw})\Big|_{w=0} = 1 \tag{3.26}$$

$$\frac{d^n H_1(e^{jw})}{d^n w}\Bigg|_{w=0} = 0, \qquad n = 2,4,\cdots,2L \tag{3.27}$$

$$\frac{d^n H_1(e^{jw})}{d^n w}\Bigg|_{w=\pi} = 0, \qquad n = 1,3,\cdots,2(N-L)-1 \tag{3.28}$$

Equations (3.27) and (3.28) state that there are N derivatives that have to be set to zero: L at $w=0$, and $N-L$ at $w=\pi$. L is known as the degree of flatness, in other term it's how flat the filter is at $w = 0$. This will result in an $(N+1)\times(N+1)$ system of linear equation, where the unknowns are $b(n)$. Hence solving this system of equations will result in the impulse response coefficients we are looking for. Note that since the filter is symmetrical, $N+1$ is the number of independent coefficients. It should be also noted $H_1(e^{jw}) = 0$ and all the odd derivative of $H_1(e^{jw})$ are null at $w = \pi$. Furthermore at $w = 0$ all the even derivatives of $H_1(e^{jw})$ are equal to zero. Using this technique, we have derived a set of maximally flat digital filters. The effect of the variation of L on the frequency response is shown in Fig. 3.21, where we see a maximally flat digital filter with $M = 7$ and L varying from 0 to 3. We can see that as L increases as the passband increases, and the attenuation band decreases, i.e. the filter drop sharply at the desired cutoff frequency π.

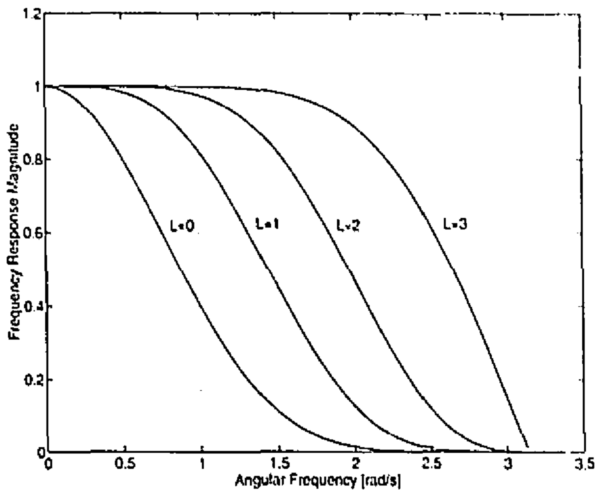


Figure 3.21: Frequency response of a maximally flat FIR with $M = 7$ and $L = 0, \cdots, 3$

In Fig. 3.22 we show a set of maximally flat digital filter where the filters have been designed to have a maximum degree of flatness.

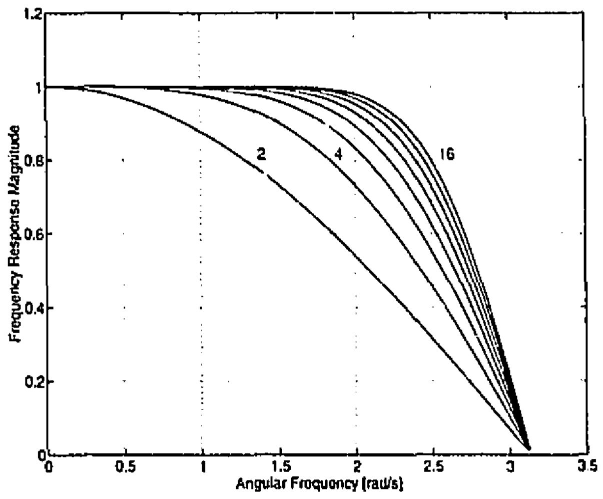


Figure 3.22: Frequency response of a set of maximally flat FIR with $L = N$

The number of coefficients varies from 2 to 16 (i.e $M = 1, \cdots 15$). Since our goal is to design the closest filter to an ideal lowpass one with cutoff frequency at π, we will have to choose the filter with the maximum degree of flatness. The filter of 16 coefficients

($M = 15$) and $L = 7$ in Fig. 3.22 offers a good trade off between complexity and our goal, since 16 coefficients is a feasible size in video processing. The coefficients of this filter are shown in table 3.2:

Filter's Coefficients			
h[0]=-0.000006	h[1]=0.000111	h[2]=-0.000915	h[3]=0.004848
h[4]=-0.018698	h[5]=0.057591	h[6]=-0.159975	h[7]=0.617046
h[8]=0.617046	h[9]=-0.159975	h[10]=0.057591	h[11]=-0.018698
h[12]=0.004848	h[13]=-0.000915	h[14]=0.000111	h[15]=-0.000006

Table 3.2: Maximally Flat Digital Filter.

3.3.3 Simulations and Results

Frame rate conversion was evaluated in the framework of MPEG-2. In MPEG-2, at the lower layer, 4:2:0 formats (704×480) which are interlaced will be downsampled to form the SIF format (352×240) which is a progressive format. This conversion will start by dropping each odd field of the 4:2:0 sequence. Then the left even fields will have their luminance horizontally downsampled by a factor of two. This downsampling will be preceded by a 1-dimensional low-pass filtering (horizontal) in order to attenuate the high frequencies that will cause aliasing when downsampling. This low-pass filtering will be executed using the lowpass filtering version of the filter in table 3.1: the coefficients for this filter are the ones of table 3.1, divided by two; this will result in a DC gain of 1. In order to display SIF formats at the receiver, upconversion to the original CCIR format is needed and this is where frame rate conversion intervenes. Once the missing fields are interpolated using the outlined methods in the previous section, final upconversion to CCIR spatial format will be performed using the filter in table 3.1 (interpolation version) where the luminance will be horizontally upsampled by two. This will result in the 4:2:0 format. To get to the CCIR format vertical up-sampling by two of the chrominance is required and will also be achieved using the previous filter. Next we present the simulation used to assess the performance of our method versus the MPEG-2 one. Both visual and PSNR criteria were considered in the evaluation, although it is important to note that the PSNR criterion by itself can not be considered as the sole means of evaluation or a precise measure and should only be looked at as an indicator. This is because the PSNR is a global measure while the Human Visual System is usually affected by local distortion. Furthermore our concern is moving sequences, where artifacts can only be detected using a visual criterion. That is why the artifacts which appears on the "VIDS" display in the palindromic mode will not be visible in the following images.

Calendar

The first simulation was carried out on the sequence "Calendar". The scene consists of a toy train on a railroad moving from right to left, a ball which is undergoing a rotational motion, a pendulum on the right which also have a rotational motion, and a calendar, in the background, which has a slow vertical motion. Though the simulation was carried on 22 fields, we will only show part of the image due to a space constraint. For this we have chosen the left half part of field number 3, which is of size (352 × 240). In Fig. 3.23 we show the original CCIR image. In order to evaluate the performance of the different methods, we have used the difference error, which is the difference between the original image and the interpolated one. The grey luminance indicates zero error while black and white intensities indicate large error. In Fig. 3.24 and 3.25 we show respectively the difference error of the temporal interpolation and the adaptive one. As can be seen in Fig. 3.24 the temporal interpolation fails to follow the motion of the ball and produce repetition of contours, which results in the white spots at the position of the ball. On the other hand the adaptive method which is using the motion compensation interpolation at these points, succeed in following the motion of the ball and the interpolation is satisfactorily achieved. The same problem occurs on the pendulum where temporal interpolation causes duplication of object which are avoided with motion compensation. In palindromic mode these artifacts were clearly visible: the duplication of objects and contours of the ball and the pendulum resulted in an annoying flickering in the temporal interpolation, while the motion of the pendulum and the ball looked correct in adaptive interpolation. The PSNR of Fig. 3.26 has been computed between the original and the interpolated sequences for 11 consecutive odd field. For odd fields 1 to 13 the PSNR of temporal interpolation is lower than the motion compensated one: this has been checked visually and is shown in the artifacts previously described. For odd field 15 to 21 temporal interpolation gives better PSNR than motion compensation. This is because more regions are having null motion than previously, and in that case temporal interpolation gives better result than motion compensated filtering, which reduces in that case to Keys cubic interpolation. This is why our adaptive method has always the highest PSNR, hence achieving its goal of adaptive interpolation and proving the fact stated in section 3.3.1, that when the motion is less than 1 pel/s temporal filtering is reliable and can avoid artifacts caused by motion estimation.

Next we compare our adaptive interpolation with MPEG-2 one. Fig. 3.27 shows the difference error, and as seen the white spots and contours indicates a large error which corresponds to a PSNR difference of 2 db with respect to the adaptive PSNR Fig. 3.28. When displayed on the "VIDS" field by field, the MPEG-2 interpolated sequence had no large distortions, just some artifacts due to aliasing. That is because

MPEG-2 only considers spatial interpolation and repetition without any temporal adjustment. However when viewed in the palindromic mode the motion was abrupt; this resulted in a jerky motion as opposed to the continuous one that the adaptive method had.

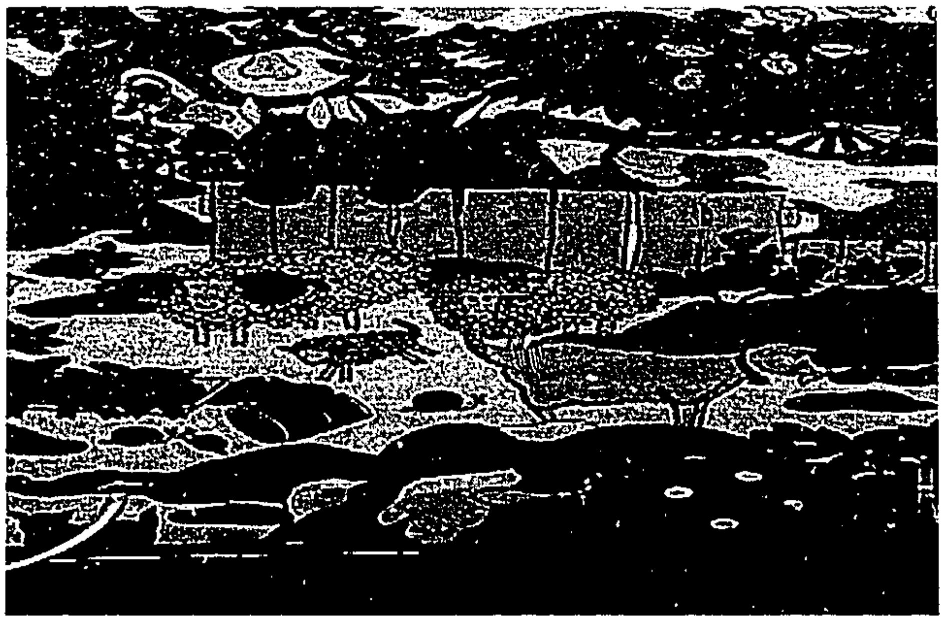


Figure 3.23: Calendar Original: field #3

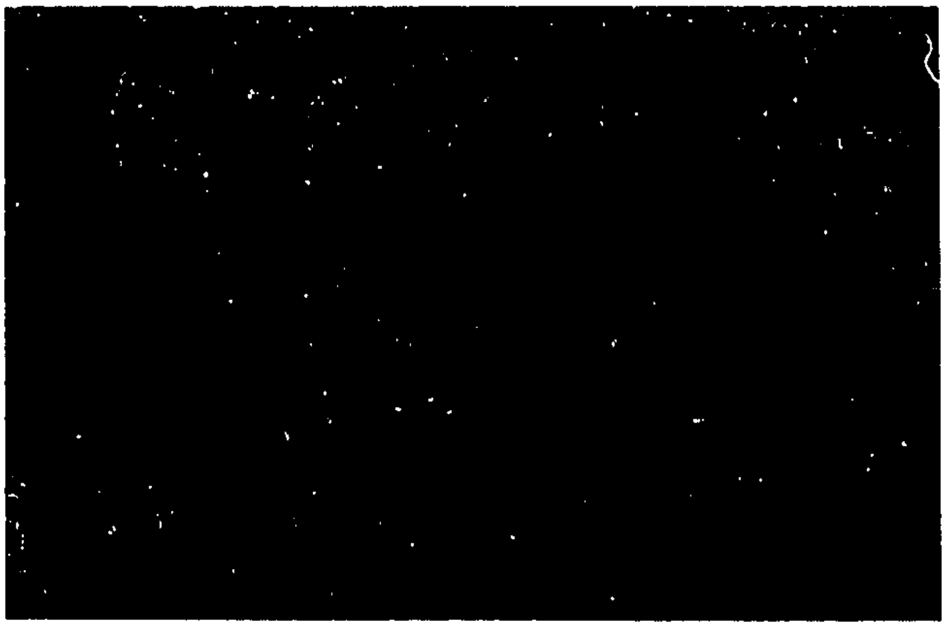


Figure 3.24: Calendar Temporal interpolation error: field #3

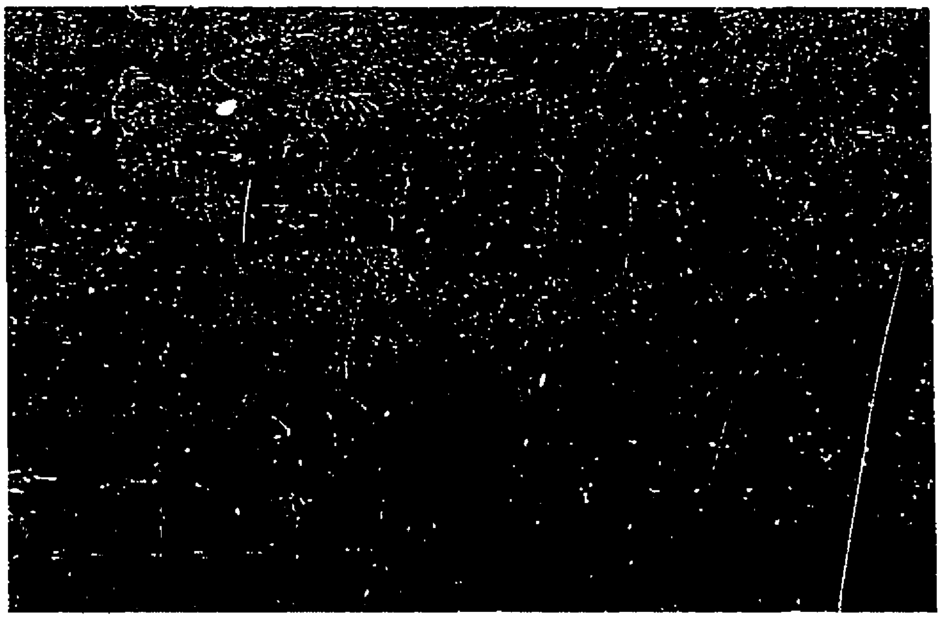


Figure 3.25: Calendar Adaptive interpolation error: field #3

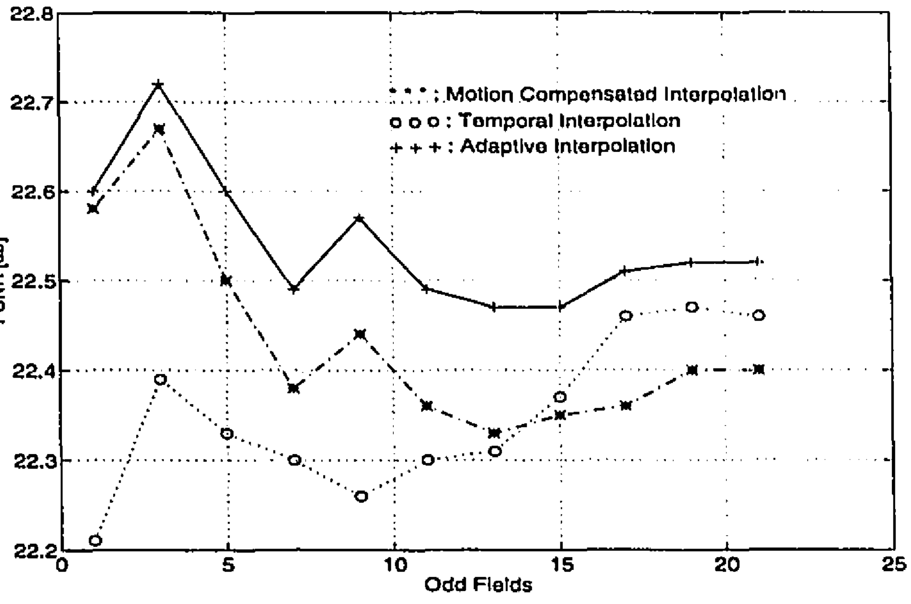


Figure 3.26: CalendarPSNR Comparison of the three methods: Motion Compensated Interpolation, Temporal Interpolation, Adaptive Interpolation

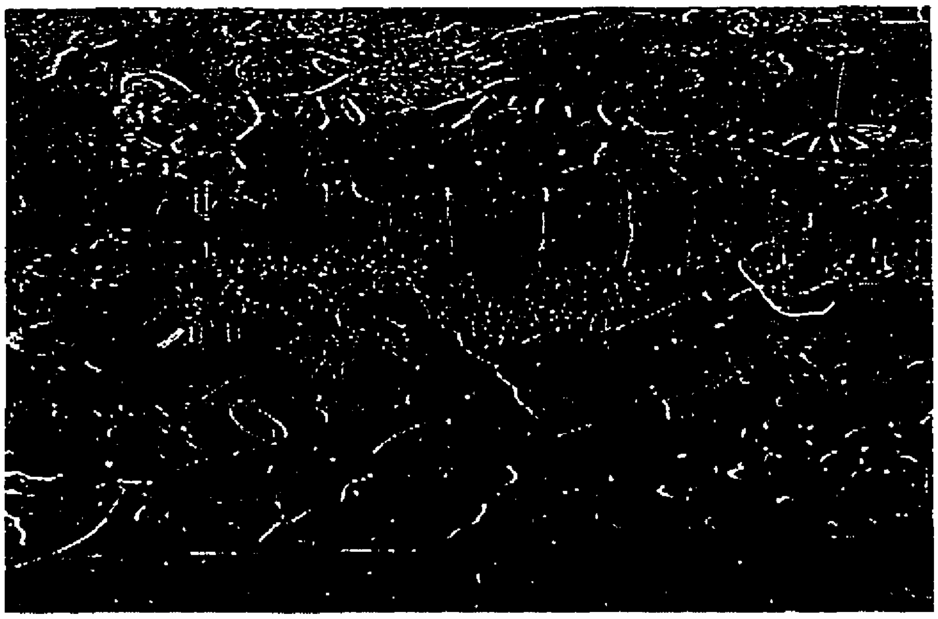


Figure 3.27: Calendar MPEG-2 interpolation error: field #3

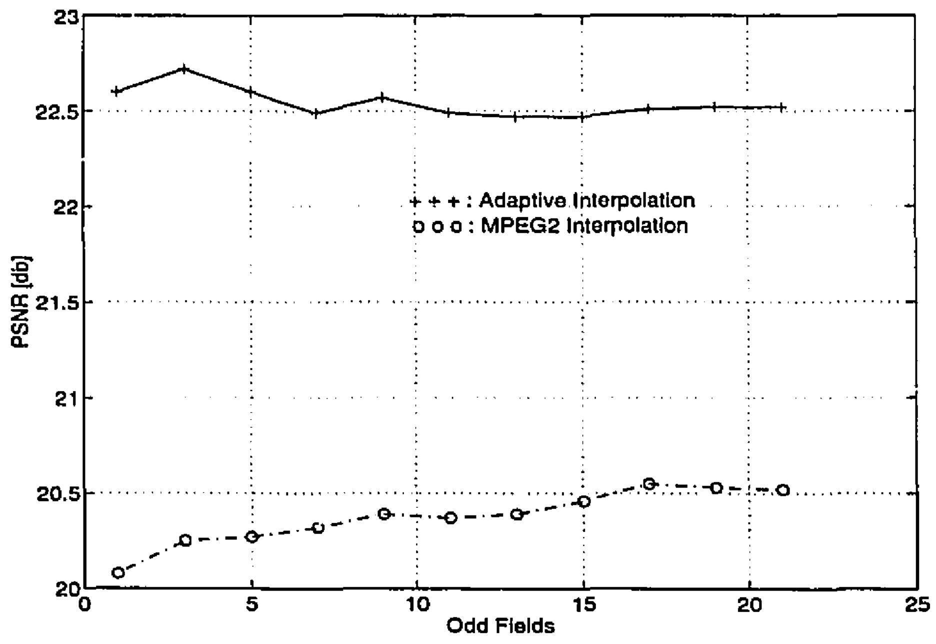


Figure 3.28: Calendar PSNR comparison of our adaptive method and the MPEG-2 one

Flowergarden

In this simulation the SIF sequence is the output of the MPEG-2 decoder at the lower layer, which is a coded-decoded SIF. The scene of this sequence was previously described and we remind that the motion of this scene is caused by the horizontal panning of a camera and that the tree has a relatively large horizontal motion. Though simulations and PSNR were done for the whole image and for a sequence of 20 fields, we will only show the part of the image centered at the tree because of space constraints. We have chosen field #3 and a spatial size of (352×240). In Fig. 3.29 we show the original CCIR image. We will compare next our adaptive and MPEG-2 method. Since the motion is large, the adaptive method will use the motion compensated interpolation at most pixels. Fig. 3.30 is the difference error between the original and the adaptive interpolation images. We can see that the error is relatively small. They are mostly at the tree-trunk edges due to bad estimation of motion at these points caused by occlusion of covered areas. Fig. 3.31 shows the difference error of the original and MPEG-2 interpolation images. The error is so large that the details of the original image are visible. This corresponds to a 4 db difference as shown in Fig. 3.32. This huge error justifies the need for motion compensation. When displayed on the "VIDS" field by field, MPEG-2 interpolated sequence didn't show many artifacts. However in the palindromic mode the motion was discontinuous and jerky while the adaptive interpolation succeeded in rendering the true motion of the original image. The error at the tree-trunk with the adaptive interpolation was visible and resulted in a flickering. This error can be avoided by improving the motion estimator through occluded region treatment.

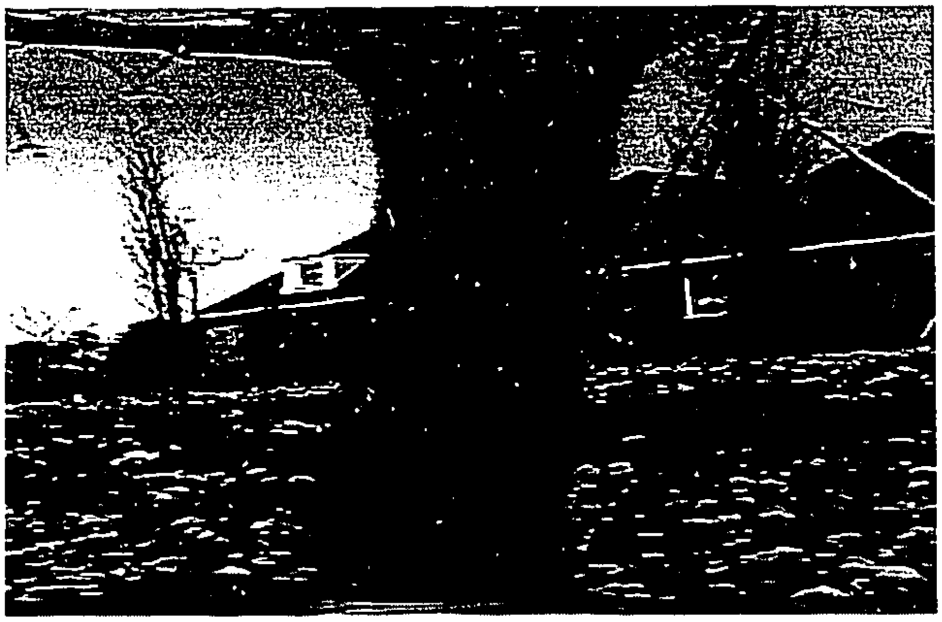


Figure 3.29: Flowergarden Original: field #9

Figure 3.30: Flowergarden Adaptive interpolation error: field #9

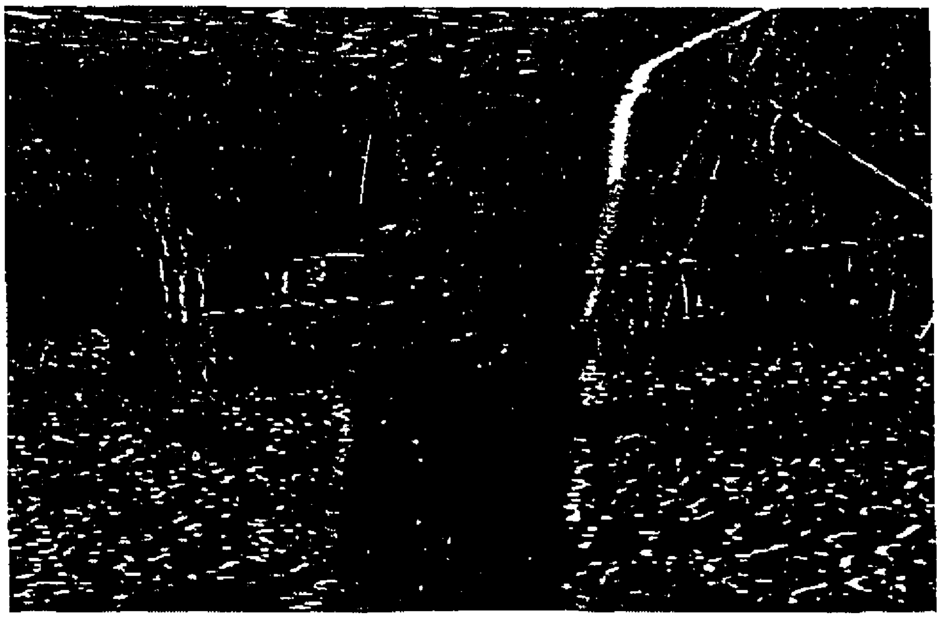


Figure 3.31: Flowergarden MPEG-2 interpolation error: field #9

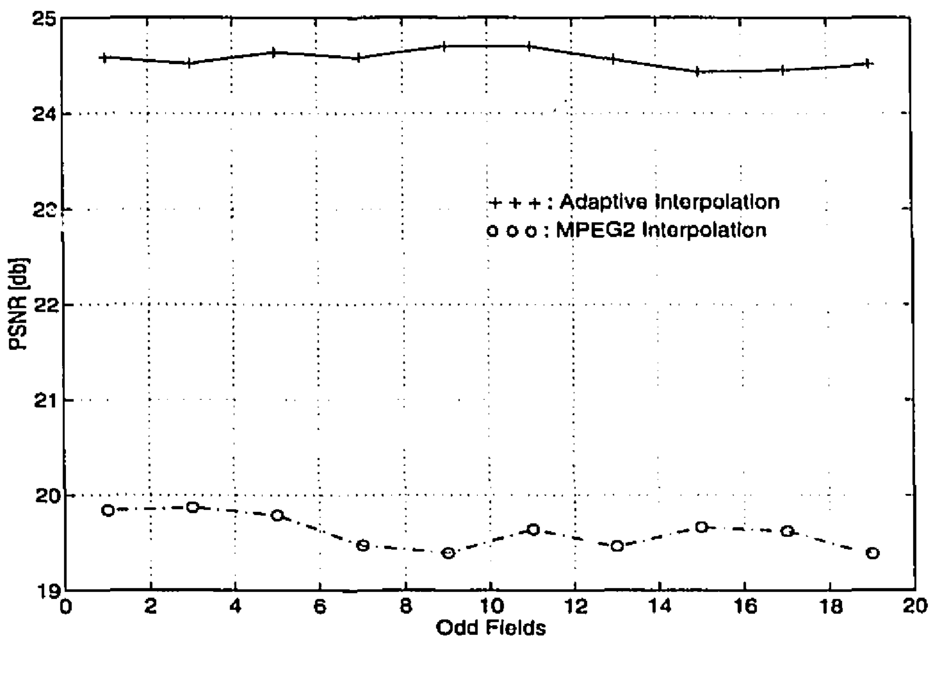


Figure 3.32: Flowergarden PSNR comparison of our adaptive method and the MPEG-2 one

Chapter 4

Multiscale Spatial Interpolation

The previous chapters dealt with interpolation using temporal and spatiotemporal filters, with particular focus on deinterlacing and frame rate conversion. This chapter presents methods in the spatial domain which allow flexible multiscale conversion with integer and non-integer ratios. A modification of the pattern interpolation method of Wang and Mitra (which achieve multiresolution spatial interpolation) is developed. This will be done using a "denser" grid which will be obtained through motion compensation. The solutions will be obtained by a least squares iterative method. Finally, we compare the visual performance of the new method with the method described by Wang and Mitra, Keys cubic convolution interpolation and the MPEG-2 method.

4.1 Pattern Motion Compensated Interpolation

4.1.1 Algorithm Description

An image representation using a set of block pattern models is introduced. The image is divided into blocks of pixels. Each block is classified as either oriented or non oriented. The oriented model characterizes blocks with straight edges or line fragments, the non oriented model includes models which represent image blocks whose intensity variation is not noticeable, blocks containing edges in more than one direction or irregular texture patterns; the classification procedure will be discussed in a later section.

Furthermore an oriented model is an image block for which a principal direction β, $0 \leq \beta \leq \pi$ and a 1-D profile function $\phi(.)$ exists such that:

$$f(x,y) = \phi(y\cos\beta - x\sin\beta) \qquad \text{for } (x,y) \in B \qquad (4.1)$$

The direction β will be obtained using the Sobel operator, as described later in the classification and orientation section. Moreover we define the orientation as a quantization of the directions β. The principal orientation of an oriented model can be obtained by:

$$\alpha = Q(\arctan\frac{f'_y}{f'_x}) \qquad (4.2)$$

where f'_x and f'_y are the gradients along the x and y directions, respectively, and Q represents the quantization level such that the resulting direction falls in one of the orientation zones under consideration. For an arbitrary block, we can find an orientation according to the gradient at that point. A block can be considered to be oriented only if most pixels inside this block assume the same orientation.

We will next increase the number of samples within one block pattern models by using forward and backward prediction using motion estimation. The rationale behind this procedure is to get a more precise fitting function by increasing the accuracy of the a_j of equation (2.15). Suppose we choose the block pattern to have a size of 4×4, then we have 16 known samples which are usually used in [38], [37] interpolation.

Let's consider forward and backward prediction, shown in Fig. 4.1:

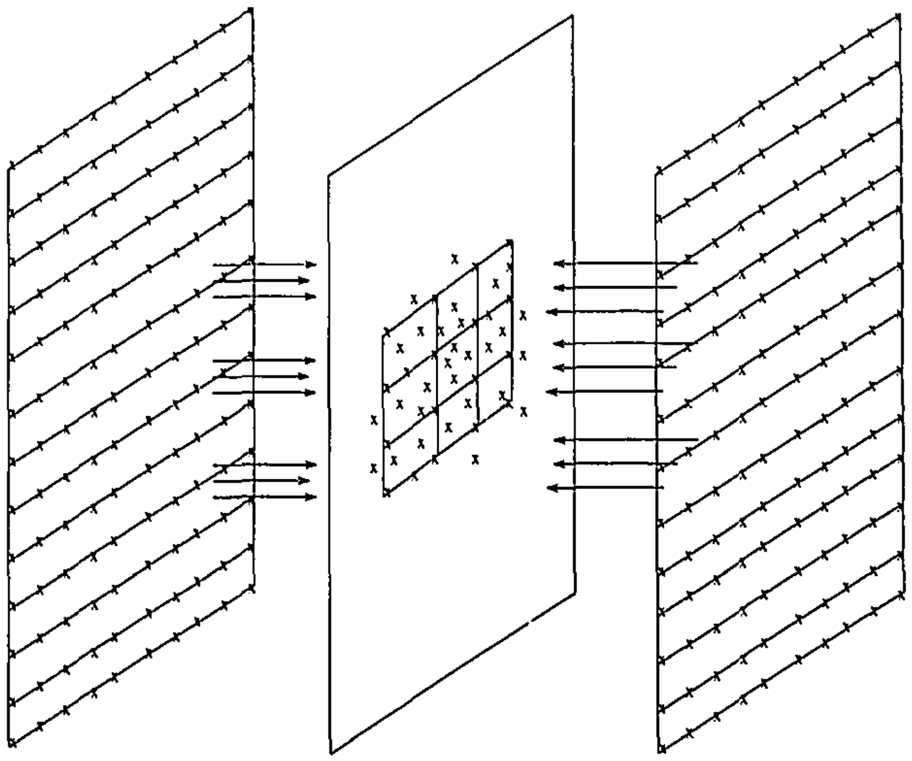


Figure 4.1: Forward and Backward Prediction

As can be seen the number of known samples within the block pattern, that is being interpolated, has increased from a minimum of 16 samples (which corresponds to the case of [38]) to an average of 48 samples.

Next we present an overview of this method. In the first step, we determine the nature of the 4×4 block pattern that is being interpolated: oriented or not oriented. Let $\mathbf{f}$ represent the vector of the original samples of the field of interpolation. Let $\mathbf{f_i}$ be the vector that contains the forward predicted samples which fall in the 4×4 block pattern we are considering, where the subscript i indicates the i^{th} forward field. Let $\mathbf{b_j}$ be the vector that contains the backward predicted samples which fall in the 4×4 block pattern we are considering, where the subscript j indicates the j^{th} backward field. The prediction, backward and forward, is done using motion compensation from the grid of the i^{th} forward and j^{th} backward field onto the current one.

If the 4×4 block pattern is oriented, the value of a sample at point (x, y) is approximated by:

$$\hat{f}(x,y) = \sum_{l=0}^{D-1} a_l (y \cos\alpha - x \sin\alpha)^l \qquad (4.3)$$

Let $\hat{\mathbf{f}}$ be the approximated values of the known samples obtained from equation (4.3). Let $\hat{\mathbf{f}}_{\mathbf{i}}$ be the approximated values of the forward predicted samples obtained from equation (4.3). Let $\hat{\mathbf{b}}_{\mathbf{j}}$ be the approximated values of the backward predicted samples obtained from equation (4.3).

Than we have in matrix form:

$$\hat{\mathbf{F}} = \mathbf{Ha} \qquad (4.4)$$

where

$$\hat{\mathbf{F}} = \begin{bmatrix} \hat{\mathbf{f}} \\ \hat{\mathbf{f}}_{\mathbf{1}} \\ \vdots \\ \hat{\mathbf{f}}_{\mathbf{U}} \\ \hat{\mathbf{b}}_{\mathbf{1}} \\ \vdots \\ \hat{\mathbf{b}}_{\mathbf{V}} \end{bmatrix} \qquad (4.5)$$

where U and V are respectively the number of forward and backward fields used in the interpolation process.

H is the matrix of elements:

$$H_{k,l} = (y_k \cos\alpha - x_k \sin\alpha)^l \qquad (4.6)$$

of dimension $N \times D$, where N is the sum of the dimension of $\mathbf{f}$, $\mathbf{f}_{\mathbf{i}}$, $\mathbf{b}_{\mathbf{j}}$. (x_k, y_k) are the coordinates, with respect to a block axis, of the k^{th} element (sample) of $\hat{\mathbf{F}}$.

Let F be the vector consisting of:

$$\mathbf{F} = \begin{bmatrix} \mathrm{f} \\ \mathrm{f}_1 \\ \vdots \\ \mathrm{f}_U \\ \mathrm{b}_1 \\ \vdots \\ \mathrm{b}_V \end{bmatrix} \tag{4.7}$$

then the best coefficient vector that minimizes the square error defined by

$$E = ||\mathbf{F} - \mathbf{Ha}||^2 \tag{4.8}$$

is

$$\mathbf{a} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{F} \tag{4.9}$$

Multiscale interpolation will be performed similarly to what has been done in the cubic case in the conversion from the S_1 grid to the S_2 one. If the block is oriented the value of its missing samples will be obtained by using equation (4.3) with the value of x and y as given by equation (4.3). If the block is non oriented than this whole procedure is ignored, and the value of the missing sample that is being interpolated will be obtained using Keys Cubic Convolutional interpolation.

4.1.2 Classification and Orientation

As mentioned previously a block will be considered oriented if most pixels assume the same orientation, otherwise the block shall be treated as non-oriented. This classification will be done by counting the number of identical orientation. Then the largest and second largest score will be determined. If the largest score is greater than the threshold t_1 and if the ratio r of the second maximum to the maximum is less than the threshold t_2 then the block will be considered oriented. This is a simple and yet good indicator of the flatness of the histogram when the block size is small. A uniform distribution would have a low maximum score and and a high ratio, whereas a unimodal distribution would have a high maximum score and a low ratio. The ideal situation is when t_1 is chosen equal to the number of pixels in the block, which results in a null t_2. However since this is not the case in practise, we have chosen $t_1 = 10$ and $t_2 = .75$. This means that whenever the block has at least 10 samples with the same orientation, the block will be considered as oriented. Since the human orientation discrimination is limited we have proceeded to a quantization of the orientation. Hence our simulations were carried for both 8 and 12 equally separated oriented zones in the range of 0 to π.

However this quantization introduced some artifacts in the horizontal oriented model, because the human visual system is more sensitive to this direction. If the intensity along the horizontal orientation was not sufficiently constant we had to deal with loss of continuity in the contiguous interpolated oriented blocks. For this we have introduced a test where we determine whether the pixel intensities along the horizontal orientation is sufficiently constant. This was done using the *just noticeable difference*. According to Weber's law the JND is linearly proportional to the background intensity. The background intensity along one horizontal line in a block will be approximated by the average of the maximum intensity f_{max} and the minimum intensity f_{min}, among the pixels of one block which assume the same horizontal orientation. Let $\delta = \text{JND}((f_{max} + f_{min})/2)$. Now if at least one of the four horizontal lines in the 4×4 block had:

$$(f_{max} - f_{min}) < \delta \tag{4.10}$$

then the block was considered non oriented and cubic interpolation was performed. The JND curves were taken from [38]. The orientation at each point was determined by equation (4.2), and the gradients f'_x and f'_y were approximated by the convolution of the Sobel operators with the image [17]. These two operators are defined by the following matrices:

$$\Delta_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{4.11}$$

and

$$\Delta_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.12}$$

The two gradients at a point (i,j) of the grid will be approximated by the convolution of these operators with the image, which resulted in:

$$\begin{aligned} f'_x = \; & I(i+1,j-1) + 2 \cdot I(i+1,j) + I(i+1,j+1) - \\ & I(i-1,j-1) - 2 \cdot I(i-1,j) - I(i-1,j+1) \end{aligned} \tag{4.13}$$

and

$$\begin{aligned} f'_y = \; & I(i-1,j-1) + 2 \cdot I(i,j+1) + I(i+1,j+1) - \\ & I(i-1,j-1) - 2 \cdot I(i,j-1) - I(i+1,j+1) \end{aligned} \tag{4.14}$$

4.1.3 Least Square Iterative Solution

In order to determine the coefficient vector that minimizes the square error of equation (4.8), we have applied the Cauchy steepest descent iterative method. This was done in order to avoid the inversion of matrices that equation (4.9) requires. This inversion would have inherent problems due to singular matrices which would have required singular decomposition. Furthermore the use of the Cauchy steepest descent iterative method, to get the least square solution, was justified by the existence of such solution. Let $h(\mathrm{a}) = \mathbf{Ha} - \mathbf{F}$ and $f(\mathrm{a}) = ||h(\mathrm{a})||^2$, than a^* will be a least square solution of $||h(\mathrm{a})||$, if $f(\mathrm{a}^*) \leq f(\mathrm{a})$ and this $\forall \mathrm{a} \in \mathcal{R}^n$. This is equivalent to saying that $\mathbf{a}^*$ is a global minimum. The minimization will be done according to the steepest descent algorithm:

$$\mathrm{a}^{\mathrm{k+1}} = \mathrm{a}^{\mathrm{k}} - \sigma_k \nabla^T f(\mathrm{a}^{\mathrm{k}}) \tag{4.15}$$

where

$$\sigma_k = \min_{\sigma \geq 0} f(\mathrm{a}^{\mathrm{k}} - \sigma \nabla^T f(\mathrm{a}^{\mathrm{k}})). \tag{4.16}$$

Define $F(\sigma)$ as:

$$\begin{aligned} F(\sigma) &= f(\mathrm{a}^{\mathrm{k}} - \sigma \nabla^T f(\mathrm{a}^{\mathrm{k}})) \\ &= \left[h(\mathrm{a}^{\mathrm{k}} - \sigma \nabla^T f(\mathrm{a}^{\mathrm{k}}))\right]^2. \end{aligned} \tag{4.17}$$

Using first order Taylor expansion for $h(\mathrm{a})$ at $\sigma = 0$ results in:

$$h(\mathrm{a}^{\mathrm{k}} - \sigma \nabla^T f(\mathrm{a}^{\mathrm{k}})) \approx h(\mathrm{a}^{\mathrm{k}}) - \sigma \nabla h(\mathrm{a}^{\mathrm{k}}) \nabla^T f(\mathrm{a}^{\mathrm{k}}). \tag{4.18}$$

Substituting (4.18) in (4.17) gives

$$F(\sigma) = \left[h(\mathrm{a}^{\mathrm{k}}) - \sigma \nabla h(\mathrm{a}^{\mathrm{k}}) \nabla^T f(\mathrm{a}^{\mathrm{k}})\right]^2. \tag{4.19}$$

σ_k of equation (4.16) will be determined by setting $F'(\sigma)$ to zero hence:

$$F'(\sigma_k) = 0 \tag{4.20}$$

$$\Rightarrow \sigma_k = \frac{h(\mathrm{a}^{\mathrm{k}}) \nabla h(\mathrm{a}^{\mathrm{k}}) \nabla^T f(\mathrm{a}^{\mathrm{k}})}{\left(\nabla h(\mathrm{a}^{\mathrm{k}}) \nabla^T f(\mathrm{a}^{\mathrm{k}})\right)^2} \tag{4.21}$$

$$= \frac{(h(\mathrm{a}^{\mathrm{k}}), \nabla h(\mathrm{a}^{\mathrm{k}}) \nabla^T f(\mathrm{a}^{\mathrm{k}}))}{||\nabla h(\mathrm{a}^{\mathrm{k}}) \nabla^T f(\mathrm{a}^{\mathrm{k}})||^2} \tag{4.22}$$

where $(,)$ denotes the scalar product. Furthermore it's trivial that:

$$\nabla^T f(\mathrm{a}^{\mathrm{k}})) = 2h^T(\mathrm{a}^{\mathrm{k}}) \nabla h(\mathrm{a}^{\mathrm{k}}) \tag{4.23}$$

replacing (4.23) in (4.22) yields:

$$\sigma_k = \frac{1}{2} \cdot \frac{||\nabla^T h(\mathbf{a}^k) h(\mathbf{a}^k)||^2}{||\nabla h(\mathbf{a}^k) \nabla^T h(\mathbf{a}^k) h(\mathbf{a}^k)||^2}. \tag{4.24}$$

It's also trivial that $\nabla h(\mathbf{a}^k) = \mathbf{H}$, substituting this result and the expression of $h(\mathbf{a})$ leads to:

$$\sigma_k = \frac{||\mathbf{H}^T(\mathbf{Ha}^k - \mathbf{F})||^2}{||\mathbf{HH}^T(\mathbf{Ha}^k - \mathbf{F})||^2}. \tag{4.25}$$

Finally the steepest descent iteration in equation (4.15) reduces to:

$$\mathbf{a}^{k+1} = \mathbf{a}^k - \frac{||\mathbf{H}^T(\mathbf{Ha}^k - \mathbf{F})||^2}{||\mathbf{HH}^T(\mathbf{Ha}^k - \mathbf{F})||^2}\mathbf{H}^T(\mathbf{Ha}^k - \mathbf{F}). \tag{4.26}$$

As a stopping rule for the iteration in equation (4.26), we have adopted the norm of the coefficient vector difference $||\mathbf{a}^{k+1} - \mathbf{a}^k||$. If

$$||\mathbf{a}^{k+1} - \mathbf{a}^k|| \leq \mathcal{K} \tag{4.27}$$

where $\mathcal{K}$ is a certain threshold, we would assume convergence and stop the iteration at this point. Next we prove what we mentioned earlier: the existence of a global minimum. $\mathbf{a}^*$ is a solution if:

$$\nabla f(\mathbf{a}^*) = 0 \tag{4.28}$$

and since

$$\begin{aligned} f(\mathbf{a}) &= (\mathbf{Ha} - \mathbf{F}, \mathbf{Ha} - \mathbf{F}) \\ &= (\mathbf{a}, \mathbf{H}^T\mathbf{Ha}) - 2 \cdot (\mathbf{Ha}, \mathbf{F}) + (\mathbf{F}, \mathbf{F}) \end{aligned} \tag{4.29}$$

than equation (4.28) is equivalent to:

$$\mathbf{A}^T\mathbf{Aa}^* = \mathbf{A}^T\mathbf{b} \tag{4.30}$$

every $\mathbf{a}^* \in \mathcal{R}^n$ that verifies equation (4.30) is a candidate. Furthermore since $\nabla^2 f(\mathbf{a}^*) = \mathbf{A}^T\mathbf{A}$ is positive semidefinite, than $f(.)$ is convex and every solution of equation (4.30) is a global minimum. So our proof will be completed if we prove that equation (4.30) has always a solution. Equation (4.30) can be written as:

$$\mathbf{A^T A}(\mathrm{a}' - \mathrm{a}'') = \mathbf{A^T b}, \qquad \mathrm{a}' \geq 0, \mathrm{a}'' \geq 0 \tag{4.31}$$

where we have separated $\mathbf{a}^*$ into both its negative and positive components. Also equation (4.31) is equivalent to:

$$\begin{bmatrix} \mathbf{A^T A} & -\mathbf{A^T A} \end{bmatrix} \begin{bmatrix} \mathrm{a}' \\ \mathrm{a}'' \end{bmatrix} = \mathbf{A^T b}, \qquad \begin{bmatrix} \mathrm{a}' \\ \mathrm{a}'' \end{bmatrix} \geq 0 \tag{4.32}$$

by Farkas Lemma equation (4.32) is consistent if and only if:

$$\begin{bmatrix} \mathbf{A^T A} \\ -\mathbf{A^T A} \end{bmatrix} \mathbf{y} \geq 0 \quad \Rightarrow \quad (\mathbf{A^T b})^T \mathbf{y} \geq 0 \tag{4.33}$$

is consistent. Equation (4.33) is equivalent to:

$$\mathbf{A^T A y} = 0 \quad \Rightarrow \quad \mathbf{b^T A y} \geq 0 \tag{4.34}$$

but $\mathbf{A^T A y} = 0 \quad \Rightarrow \quad (\mathbf{Ay})^T \mathbf{Ay} = 0 \quad \Rightarrow \quad \mathbf{Ay} = 0$, hence $\mathbf{b^T A y} = 0$ therefore 4.34 is consistent. Thus since equation (4.33) is equivalent to equation (4.34) it is consistent, and by Farkas lemma (4.32) is also consistent. As a conclusion equation (4.30) is always consistent and a global minimum always exist.

4.2 Simulation

The simulations were carried out for the Courtepointe sequence. We only show the results of the 8th image. In order to assess the improvement that motion compensation brings to the oriented interpolation methods of [37], we have adopted the two following schemes:

- the first scheme, which we call Pattern interpolation, uses for the oriented block Wangs and Mitra's oriented solution. The non-oriented block will be interpolated using cubic interpolation.

- the second scheme, which we call Pattern motion-compensated interpolation, is done according to the description in the previous sections. For simulation purposes, we have used $U = 1$ and $V = 0$.

For both methods, the least square solution is obtained by Cauchy steepest descent. Further, the edge orientation of the 8^{th} image of Courtepointe is shown in Fig. 4.2. These oriented methods were also compared with the cubic and the MPEG-2 linear interpolation. The cubic and the linear interpolation shown respectively in Fig. 4.3 and Fig. 4.4 causes staircase effects at diagonal edges. These effects are eliminated with both oriented interpolation (Fig. 4.5 and 4.6) and clear sharp edges are visible. So, as a first conclusion, oriented intèrpolation results in sharper edges than cubic and linear interpolation. The effect of introducing motion compensation on pattern interpolation, is mostly visible at horizontally oriented pattern. At these patterns, Wang and Mitra's oriented interpolation causes discontinuities that appears between adjacent blocks. These discontinuities were smoothed and removed by the introduction of a denser grid through motion compensation. When compared, the cubic shows better sharpness of contours than the linear. As a general over all ranking we would say that our method gives the best performance followed by Wang and Mitra's, cubic and linear interpolation. The differences are more visible on the VIDS display than on the printed images.

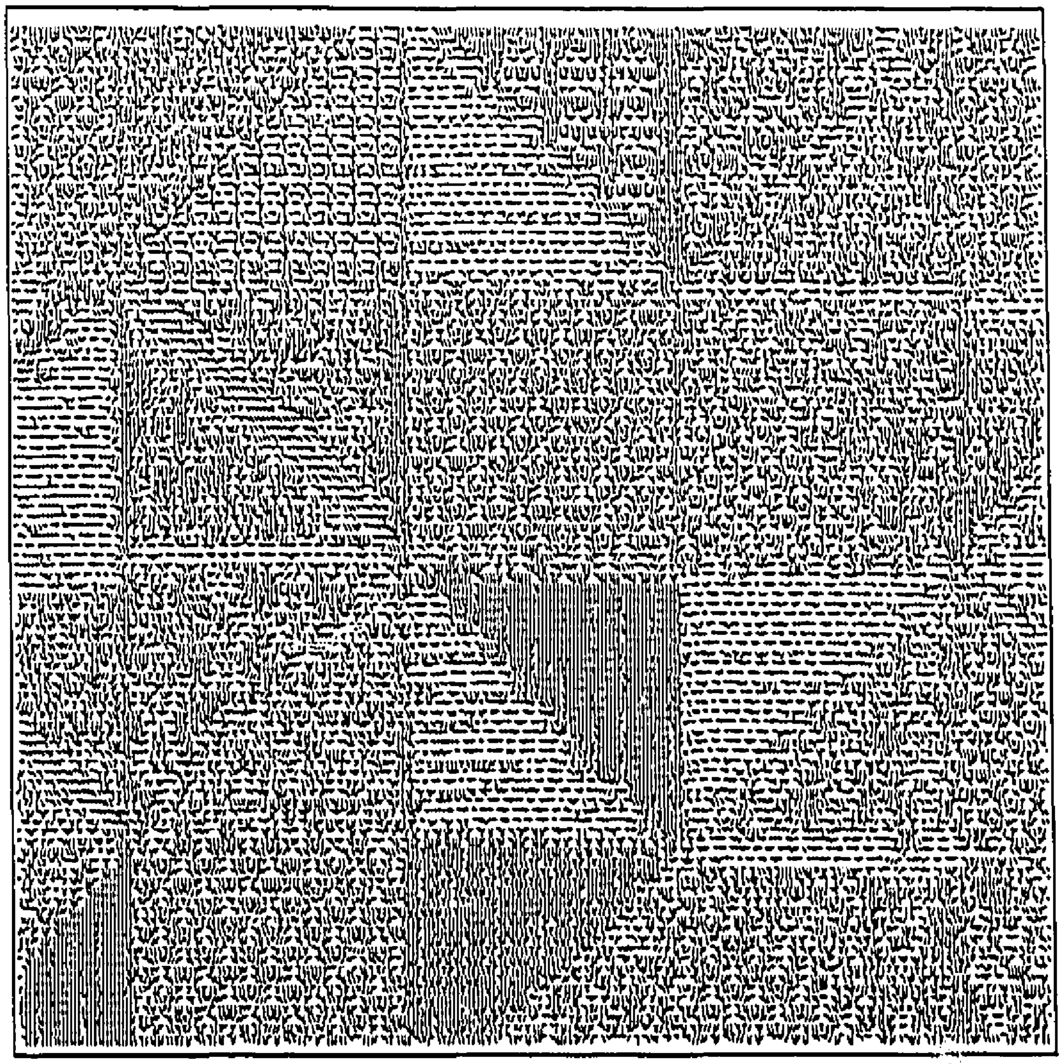


Figure 4.2: Edge Orientation of Courtepointe: field #8

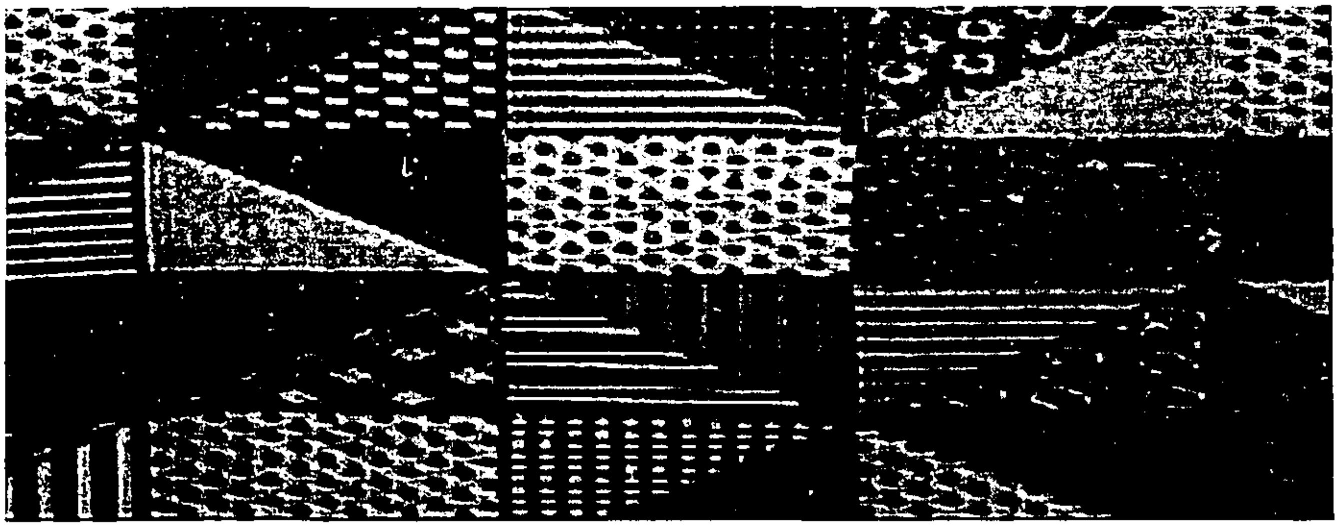


Figure 4.3: Interpolated Courtepointe using Cubic method: field #8

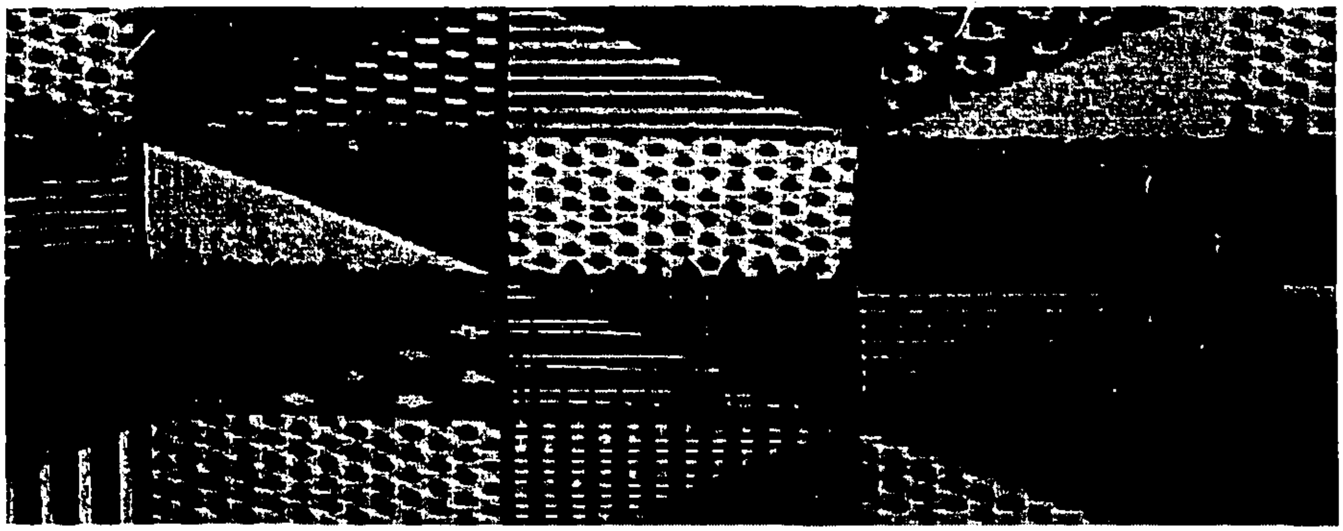


Figure 4.4: Interpolated Courtepointe using MPEG2 method: field #8

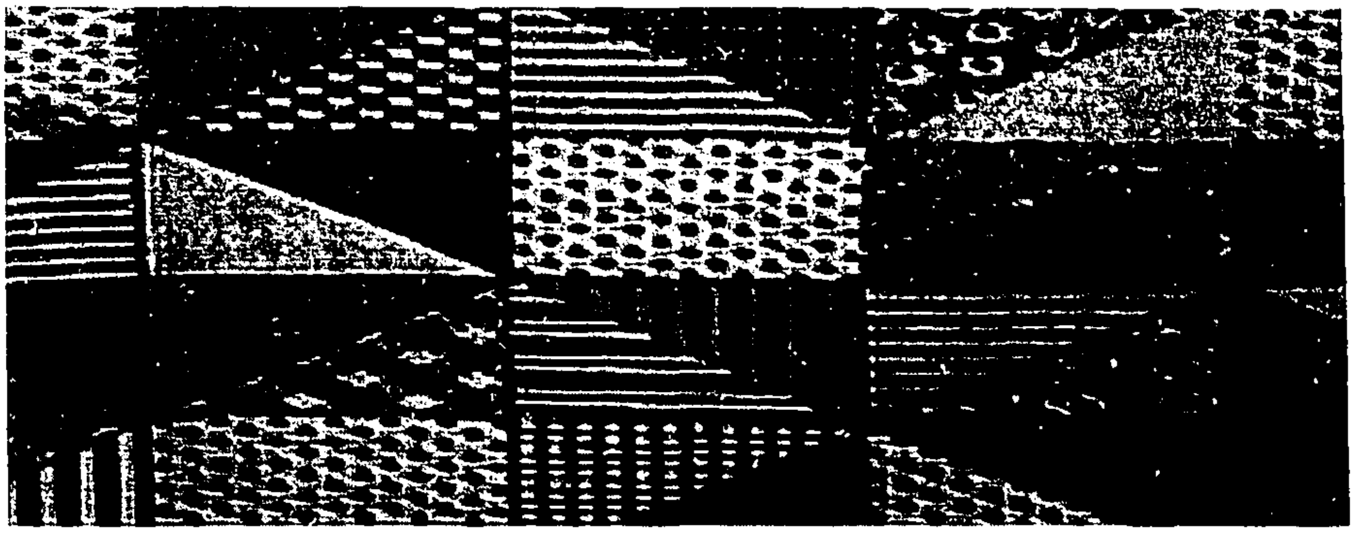


Figure 4.5: Interpolated Courtepointe using Pattern interpolation method: field #8

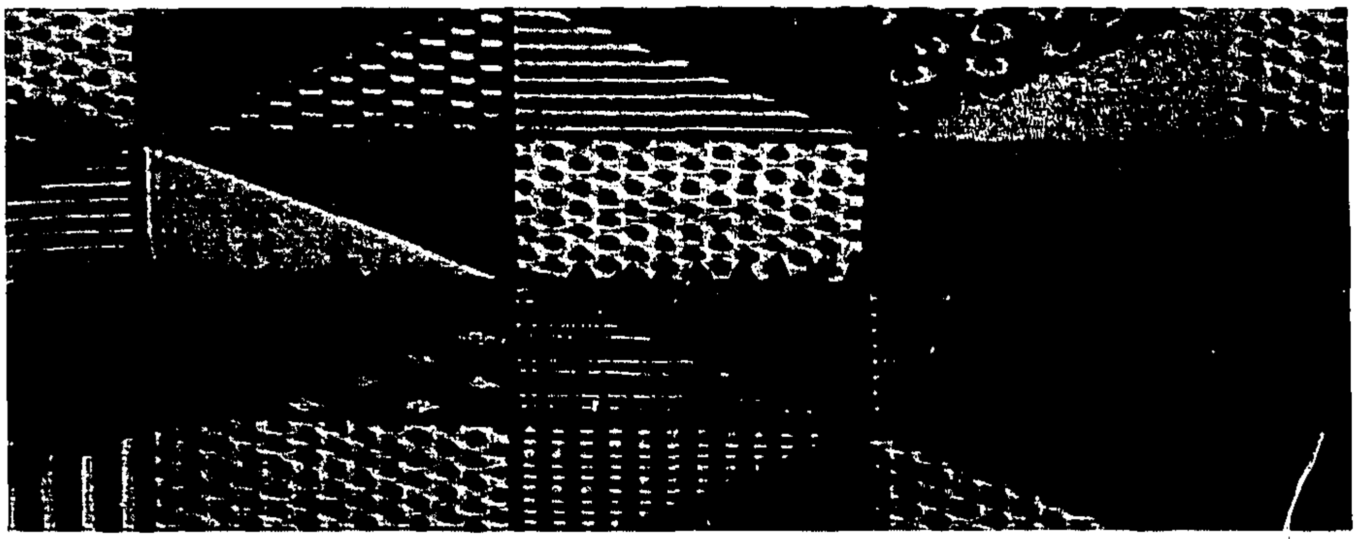


Figure 4.6: Interpolated Courtepointe using Pattern Motion Compensated method: field #8

Chapter 5

Impact of Scan Conversion on MPEG-2 Scalable Coding System

Standards conversion is required at two stages in the MPEG-2 scalable coding system: at the down-conversion where the 4:2:0 sequence will be converted to a SIF format, and at the up-conversion at both high layer coder and decoder. This was discussed in the background chapter and shown by Fig. 2.1. A description of MPEG-2 spatial scalability will next be presented as described in [3], [29], [24]. Following this, we will investigate the impact of replacing the linear interpolation of MPEG-2, by more complex up-conversion methods developed in the previous chapters, namely the cubic interpolation and the adaptive temporal interpolation and motion compensated interpolation. In this context two sequences, Calendar and Flowergarden, will be subject to testing, and the performance of the different methods will be evaluated. Furthermore, the MPEG-2 down-conversion will be upgraded and the performance of the MPEG-2 down-conversion and our proposed down-conversion scheme will be compared, at both low and high layer.

5.1 Up-Conversion

5.1.1 MPEG-2 Spatial Scalability

MPEG-2 high resolution coded pictures are divided into three types: "I", "P" and "B" types. "I" pictures are intra-coded pictures, which means that the coding of these pictures will be done independently of previous and subsequent high resolution pictures. "P" type pictures are called predicted pictures since they are coded relative to a prediction from a previous "I" or "P" type picture. "B" type pictures are bidirectionally interpolated from their closest neighbor "I" or "P" pictures. Two parameters N and M (e.g., $N = 15$, $M = 3$) will specify the coding scheme. The whole sequence is separated into groups of picture (GOPs). A GOP contains N pictures where the 1^{st} is an "I" type picture. The $(M+1)^{st}, (2M+1)^{st}, \cdots$ are "P" type pictures predicted from the previous "I" or "P" picture. The remaining pictures are "B" type pictures. The MPEG-2 spatial scalable coding algorithm depends on the type of the coded picture. If the high resolution image is an "I" type, each 8×8 block can be either Intra-coded or spatially predicted. Spatial prediction is obtained by using as a prediction the up-sampled lower resolution. If the high resolution image is Inter-coded ("P" or "B" type) the coded macroblock will result either from a temporal prediction or a spatiotemporal weighted prediction using a set of pre-defined weights. The temporal prediction comes from a motion compensated macroblock from previously decoded "I" or "P" pictures. Averaging spatial and temporal prediction is often better than either of the two. The selection of averaging weights is performed based on calculation of sum of absolute difference between prediction and reference macroblocks. The weights that produce the minimum absolute difference will be selected. We show in Fig. 5.1 how the upsampled lower layer picture and the temporal predicted picture are combined, on a macroblock basis, to form the high layer picture. Two coefficients are used in the weighting scheme: $w1$ for the odd field lines and $w2$ for the even field lines. The spatial prediction lines will be weighted with wi and added to the temporal prediction lines which are weighted with $1 - wi$ ($i = 1$ (2) if the line belongs to an odd (even) field) . This sum will form the spatiotemporal predicted picture.

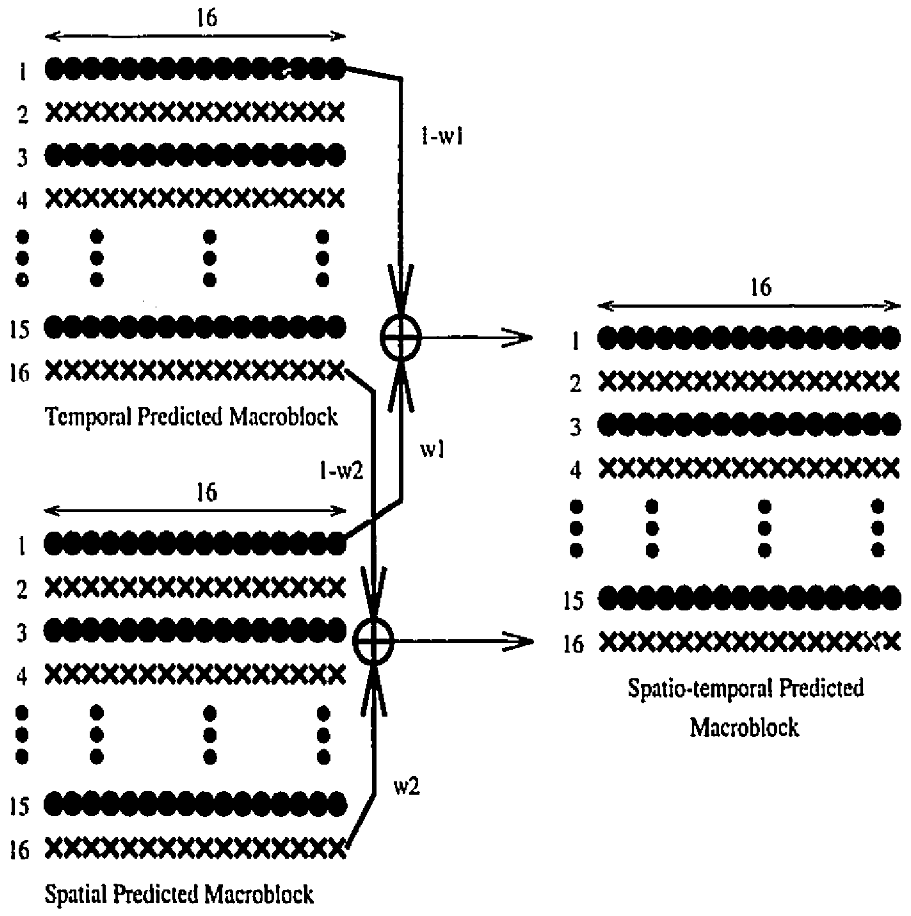

16
1-w1
w1
1-w2
w2
Temporal Predicted Macroblock
Spatial Predicted Macroblock
Spatio-temporal Predicted Macroblock

Figure 5.1: Weighted spatiotemporal prediction

Compression is done using hybrid motion compensated predictive/DCT coding. MPEG-2 has two coding structures: field and frame. In frame structure a macroblock (16×16 block of pixels) is composed of both even and odd field pixels. In field structure the macroblock is only formed from one field. In this scheme the encoder estimates the motion under the assumption that a macroblock (a 16×16 block of pixels) has undergone a translational motion. For the frame structure a motion vector is defined for each 16×8 block of pixels so that each macroblock is associated with two motion vectors. The desired motion vectors are found using a "block matching" algorithm. The motion vectors will then be encoded and transmitted to the decoder. A difference between the predicted and actual frame is formed. The macroblocks of this difference will be divided into four 8×8 blocks for the luminance components. These blocks will be compressed using the DCT transform.

5.1.2 Impact of Up-Conversion

To assess the impact of interpolation on scalable coding systems several up-conversion scheme were used in the up-sampling of the lower resolution to form the spatial prediction. A first method considers this operation as purely spatial up-sampling operation, as recommended in the MPEG-2 standard. In that framework we used the cubic interpolation, discussed in the previous chapter, as an alternative to the linear interpolation proposed by MPEG-2. A second approach considers this operation as a spatiotemporal process combining a progressive-to-interlace conversion with a spatial upsampling. The frame rate conversion has been performed, on the SIF format, using adaptive temporal interpolation and motion-compensated interpolation as described in chapter 2; the 2:1 horizontal upsampling was done using the filter of table 3.1. Two criteria were used to evaluate the impact of these methods on the high resolution coding: the PSNR and the percentage of compatibility. The PSNR value was computed between the decoded higher resolution and the original picture. The percentage of compatibility indicates by how much the spatial prediction has contributed in the spatiotemporal predicted picture formation. The following tests were made with $M = 3$ and $N = 15$ resulting with the following coded 7 frame sequence: I B B P B B P.

The first set of simulations was carried out for the Flowergarden sequence. The enhancement layer is coded at a bit rate of 4Mbit/sec with the cooperation of a coded base layer at 2Mbit/sec, so that we have a total of 6Mbit/sec that are used in the generation of the high resolution. The compatibility percentages of the coding of the higher layer are shown in table (5.1), with different upconversion methods. These percentages are an average over a sequence of 7 frames (i.e 1 Intra, 2 Predicted, 4 Bidirectional). We first notice that the percentage of compatibility of the "B" type

Up-Conversion	Intra	Predicted	Bidirectional
MPEG-2	91.00%	75.80%	9.825%
Cubic	91.20%	75.85%	9.000%
Adaptive	92.30%	77.35%	9.125%

Table 5.1: Percentage of compatibility for Flowergarden with a lower resolution at 2Mb/s and a higher resolution at 4Mb/s.

images is much lower than the "I" and "P" percentages. This is due to the fact that whenever a macroblock, [illegible] a "B" type picture, is bidirectionally interpolated, the spatial prediction is disregarded. The only potential contribution of the lower layer will be for those macroblocks which have been unidirectionally interpolated (using forward or backward temporal prediction). In a "B" picture type, most of the macroblocks will be bidirectionally interpolated, which explains the low percentage indicating a low contribution of the base layer. When comparing the relative impact that the different interpolations have on the percentage of compatibility, we see that the cubic interpolation has a higher percentage than the MPEG-2 linear interpolation, and this for both Intra and Predicted pictures. Further, the adaptive has a clear higher percentage than both cubic and MPEG-2, for both "I" and "P" type pictures. A high percentage in an "I" picture means that more macroblocks were taken from the upsampled low-resolution, which should decrease the coding cost and time: the macroblocks in question won't be coded as part of the high layer bitstream but would be extracted at the receiver by upsampling the low layer resolution. A higher percentage in a "P" picture will mean that the number of macroblocks obtained through spatiotemporal weighting, Fig. 5.1, has increased, and knowing that the quality of a spatiotemporal macroblock is better than either one of the two, implies that we will increase the quality of the 4Mbit/sec pictures without any extra transmission cost. Indeed the bits that will bring the improvement will be extracted from the low resolution, sent in a different channel, after up-sampling. In "B" type pictures MPEG-2 maintains its higher percentage over the cubic and the adaptive, and this due to the low potential participation of spatial prediction (previously mentioned). We further notice that the use of motion compensation (in the adaptive method) has a visible influence on the increase of the percentage: while this increase is minor when we limit ourselves to spatial interpolation, we see that this increase, in motion compensated method, is six time larger (on average) in the case of Intra type pictures, and 30 times larger (on average) in the case of Predicted type pictures. This result can be expected since spatial interpolation fails to render the motion component in the scene, and hence the decision to give a higher weight for the spatial prediction will be favoured when forming the spatiotemporal weighted macroblock in the adaptive case.

Results of Fig. 5.2 show and compare the PSNR values of the decoded enhancement layer when the linear interpolation (proposed in MPEG-2 standard) is replaced by a cubic interpolation or by an adaptive motion-compensated temporal interpolation. Both new methods perform better than the standard one, as can be seen by the PSNR curves. The use of motion compensation in the adaptive method clearly boosts the PSNR gain over the MPEG-2 linear interpolation and the cubic interpolation, and thus is superior by an average of 0.38 db over the linear and by an average of 0.27 db over the cubic. The cubic is also superior over the linear by an average of 0.11 db. This result was expected since as seen in the previous chapter cubic interpolation perform better than linear one, and motion compensated based interpolation gives better results than spatial interpolation. So for Flowergarden both quality and coding cost has been improved by using more advanced up-conversion method.

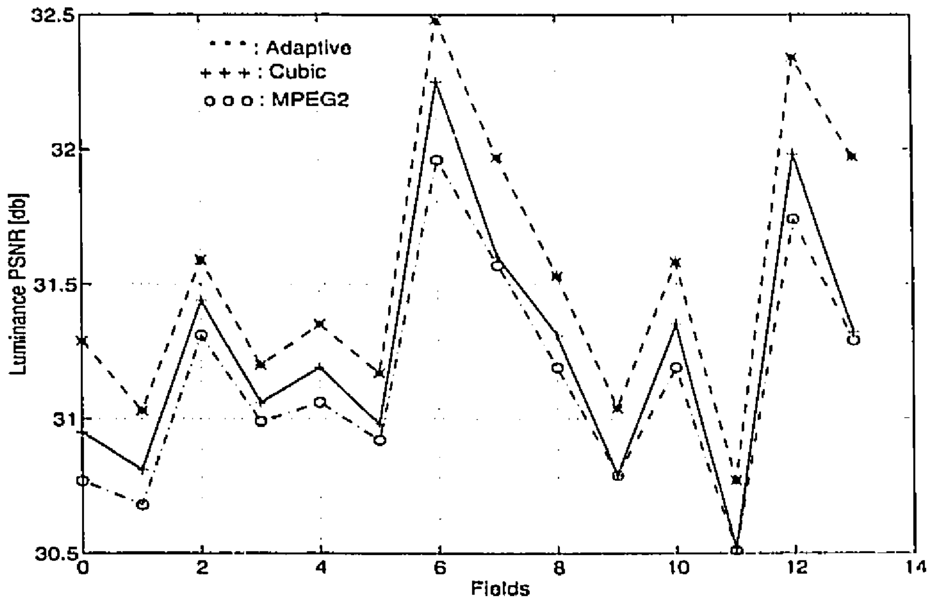


Figure 5.2: PSNR results for a decoded Flowergarden sequence using different interpolation methods

The next set of simulations was performed for the Calendar sequence. The enhancement layer was coded at a bit rate of 4 Mbit/sec and the base layer at 2 Mbit/sec, resulting in a total of 6 Mbit/sec. The compatibility percentages are shown in table (5.2), with different upconversion methods. These percentage were also obtained by averaging over a sequence of 7 frames.

Up-Conversion	Intra	Predicted	Bidirectional
MPEG-2	90.20%	83.80%	9.875%
Cubic	90.50%	85.25%	9.175%
Adaptive	89.40%	86.60%	8.575%

Table 5.2: Percentage of compatibility for Calendar with a lower resolution at 2Mb/s and a higher resolution at 4Mb/s.

The same remarks, that were made in the Flowergarden case with regards to the percentage of compatibility of the "B" type picture, still hold for Calendar "B" type pictures. Though the MPEG-2 shows a slightly higher percentage in the "I" pictures over the adaptive (which migh be caused by the fact that we are only using one "I" as an average), the Cubic and Adaptive have a significant higher percentages than MPEG-2 in the "P" type, reaching 3.3% difference in the Adaptive case and only 1.4% in the Cubic up-conversion. This means that maintaining the same coding cost we have improved the picture quality. Once again the use of motion compensation clearly influences the decision to give a higher weight for the spatial prediction in the spatiotemporal macroblock, thus resulting in a higher increase than if we limit ourselves to purely spatial up-conversion (i.e. cubic).
Results of Fig. 5.3 show the PSNR values of the decoded higher layer bit stream with different up-conversion methods: linear (MPEG-2), cubic, and adaptive motion-compensated and temporal interpolation. Both new methods perform better than the linear one, as shown by the PSNR curves. This better performance is more evident

in the adaptive interpolation with an average of 0.6 db higher than MPEG-2. This improvement was clearly visible when displayed on the VIDS in the palindromic mode. While the coding, with MPEG-2 up-conversion, resulted in severe flickering in several parts of the images, namely at the pigs in the center, at the fence, and at the red ellipse in the center, using the adaptive-upconversion in the coding scheme was able to almost totally remove the flickering. Furthermore, the resulting sequence looked more stable and smoother. The use of the cubic interpolation in the up-conversion brought a minor reduction to this flickering. So one might say that the PSNR curves are very expressive of the visual improvement that was brought.

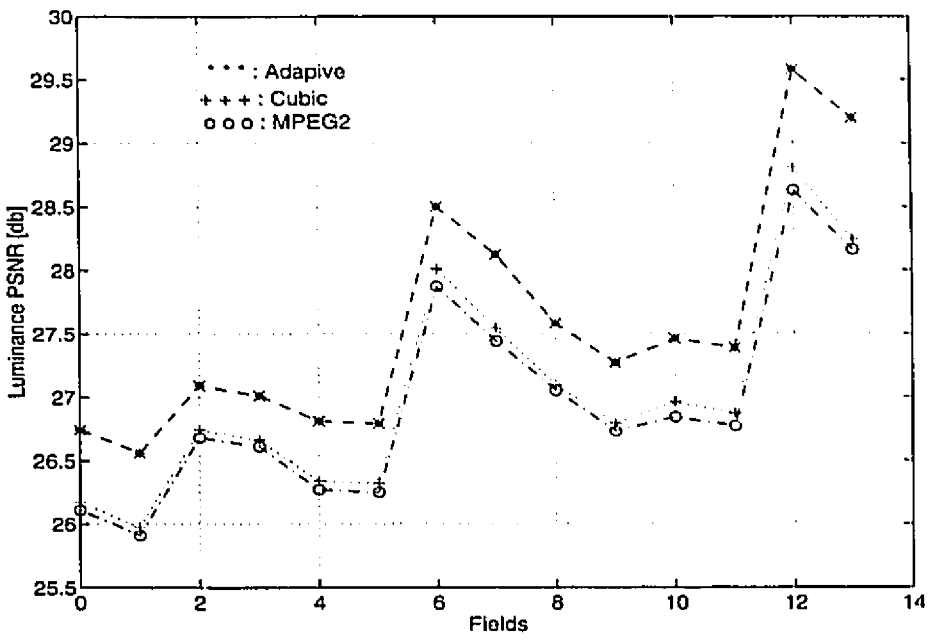


Figure 5.3: PSNR results for a decoded Calendar sequence using different interpolation methods

Other tests were carried on a coded Flowergarden and Calendar but at a lower layer rate of 1.5Mbit/sec, and a high layer rate of 4.5 Mbit/sec. The new up-conversion methods resulted in a better PSNR and higher percentage of compatibility in the "P" type, with the adaptive having always the higher edge. However the improvement that these new methods brought was less significant than the one with 2Mbit/sec lower layer rate and 4Mbit/sec higher layer rate. Indeed when the bit rate of the lower layer decreases, so does the quality of the lower resolution sequence, and thus rendering smaller the gain that the lower layer might bring in the coding of the high resolution.

5.2 Down-Conversion

5.2.1 New proposed Down-Conversion

Down conversion is required when passing from a high resolution (4:2:0) to a low resolution (SIF). In MPEG-2 the 2:1 horizontal decimation (for both luminance and chrominance) is done using a 7-tap filter in order to remove the high frequency components and hence reduce the horizontal aliasing. However the interlace to progressive conversions is simply achieved by dropping every second field of the original sequence. Thus vertical aliasing is expected to result in a reduced quality, when converting back to the original 4:2:0 format (and subsequently to the CCIR601 format). The conversion to the 4:2:0 format will be needed at two stages: when displaying the decoded SIF (low resolution) sequence at the receiver, and in the coding of the original 4:2:0 (high resolution) where the conversion of the SIF format to the 4:2:0 is required when forming the spatiotemporal prediction. The distortion at this last stage will have its impact on the quality of the decoded high resolution pictures at the receiver. In order to avoid the artifacts introduced by the vertical aliasing, we propose a more sophisticated method, shown in Fig. 5.4, where the original 4:2:0 sequence has been deinterlaced, then vertically downsampled by a factor of two, using the filter in table 3.1. This same filter was also chosen to replace the 7-tap filter used in horizontal downsampling, for both MPEG-2 and new down-conversion methods. This filter was selected since it offered better performance in terms of frequency selection and small amount of aliasing effects. Furthermore it was used in both methods (ours and MPEG-2) in order to give a fair evaluation of what deinterlacing followed by filtering will bring.

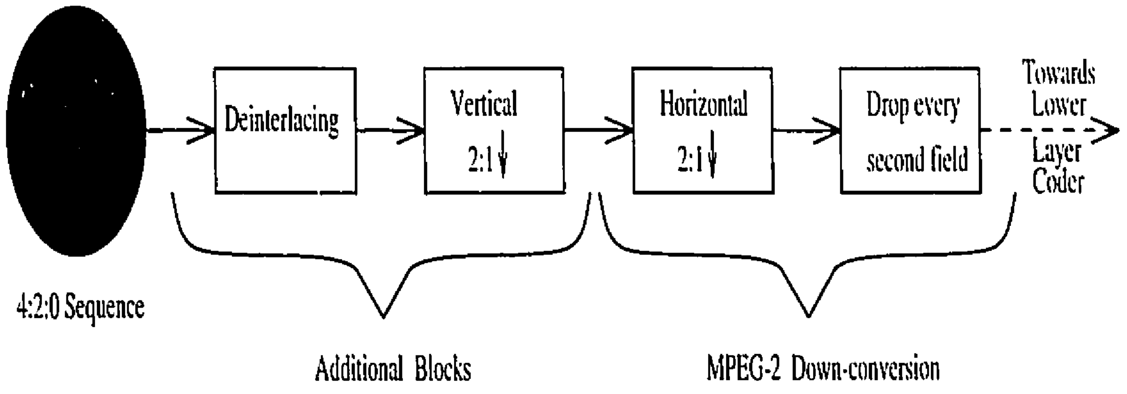


Figure 5.4: Proposed Down-Conversion

5.2.2 Impact of Down-Conversion

The following tests were carried on the sequence Calendar. Since the motion in most areas of the picture is almost null, and in order to avoid problems resulting from bad motion estimation (which are most likely to occur considering the large details in the sequence), we have assumed a zero motion and the deinterlacing, of the new down-conversion method, was simply achieved by filling the missing lines with the existing one in the immediately next field. The new proposed down-conversion scheme was first tested without introducing any coding. Thus the original images were down-converted, using the new and old method, then up-converted using a frame repetition scheme where the odd fields were obtained by shifting the preceding even fields by half a pixel, with the previously proposed maximally flat digital filter. The horizontal up-conversion was done using the 31-tap filter mentioned in section 3.2.1. The PSNR between the recovered sequence and the original one, for both old and new down-conversion method, is reported in Fig. 5.5 and this for 14 fields.

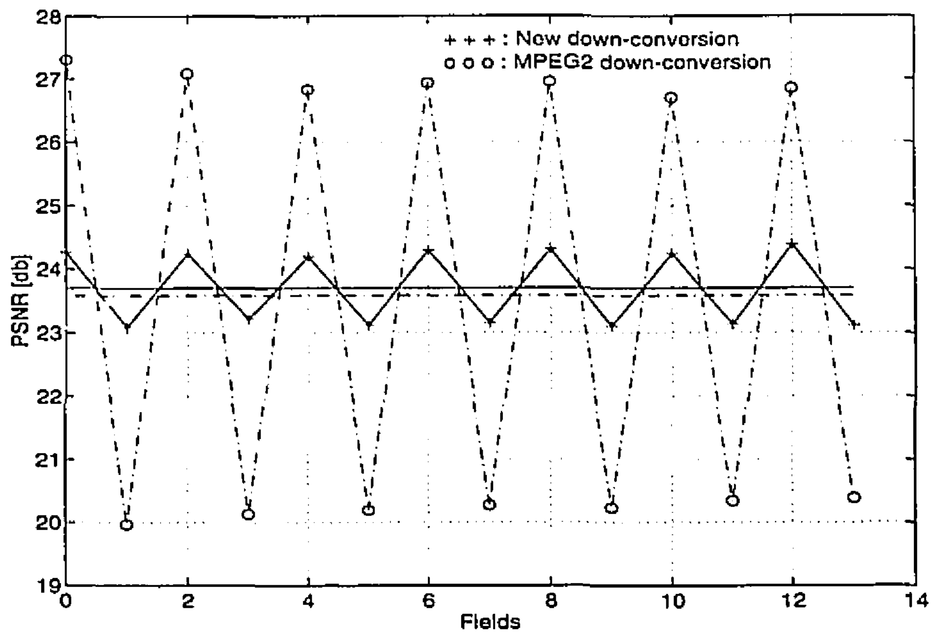


Figure 5.5: PSNR results for reconstructed Calendar (SIF) sequence using different down-conversion methods

Furthermore the average, for each method, is indicated by the straight lines, with the continuous line for the new method and the dashed line for the MPEG-2 one. We can see that for all even fields the MPEG-2 method resulted in a higher PSNR, by an average of 2.67 db. This is expected since deinterlacing followed by filtering will cause smoothing of the image, hence resulting in a lower PSNR. On the other hand the PSNR of the odd fields in the new method is higher, by an average of

2.92 db. than the corresponding MPEG-2 one, which means that the new method was capable of removing a good part of the aliasing. The overall average of the new method is also superior to the MPEG-2 one. We also notice a larger fluctuation in the PSNR image when using the MPEG-2 method; this fluctuation is considerably reduced when using the new down-conversion scheme. The visual assessment of the results was made through the palindromic mode of the VIDS. A significant decrease of the vertical aliasing was clearly visible. In the MPEG-2 method, flickering caused by aliasing, was present at contours, the numbers of the calendar, the wheel of the train, and mostly in the scene of the calendar, above the numbers, where very significant flickering distorted the scene. In the new method the flickering was significantly removed at all the above places. Thus the new down conversion method gave a very good quality, and the decrease in resolution due to smoothing in the even fields was not noticeable when compared with the MPEG-2 sequence.

The second test consisted of coding the down-converted sequences, followed by an up-conversion, identical to the one described before, and this in order to evaluate the impact on the lower resolution at the receiver when using new down-conversion method and neglecting the impairments of the channel (assuming ideal transmission). The coding was done at a bit rate of 1.5 Mbit/sec. The PSNR between the recovered sequence and the original one, for both old and new down-conversion method, is shown in Fig. 5.6 and this for 14 fields. As previously, the overall averages were also plotted. The same pattern of results is noticed: larger PSNR in the MPEG-2 method for even fields due to the smoothing introduced by deinterlacing and filtering, larger PSNR in the new scheme for odd fields caused by the removal of aliasing, new method with overall average higher by .28 db, lower PSNR fluctuation in the new method. In the palindromic mode a considerable reduction of aliasing was observed in the new methods, when compared with the MPEG-2 one. However this reduction was less significant than the one obtained without coding and this was due to masking by coding impairments.

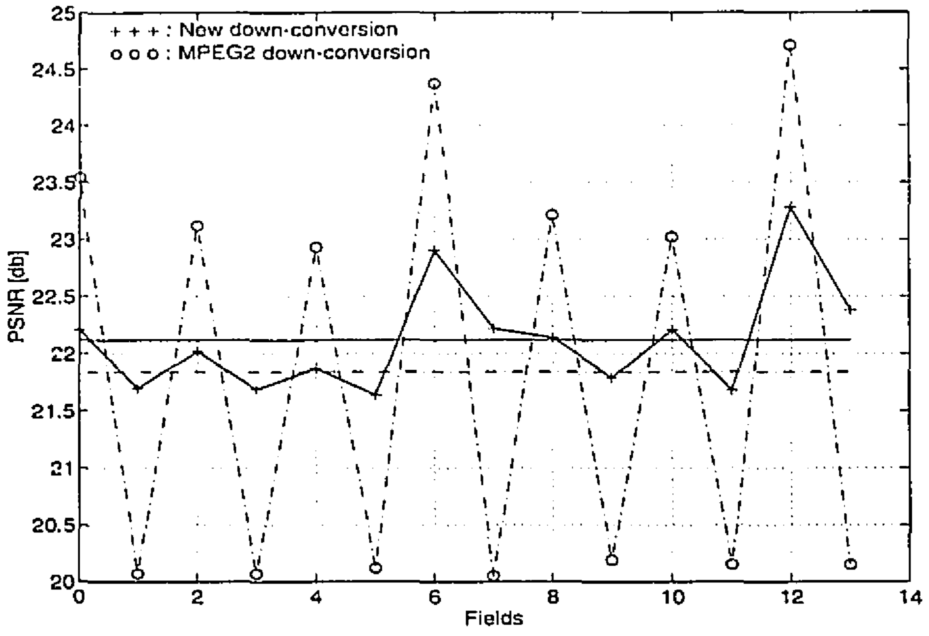


Figure 5.6: PSNR results for reconstructed coded Calendar (SIF at 1.5 Mbit/sec) sequence using different down-conversion methods

Next we investigate the effect that down-conversion schemes might have on the coding of the high layer. Deinterlacing followed by filtering will introduce smoothing as previously mentioned. Thus the PSNR of the spatially predicted macroblocks will be reduced, resulting in a lower spatiotemporal PSNR. However the new down-conversion method will also reduce aliasing, which will compensate for the reduction of quality caused by smoothing. Indeed, as Fig. 5.7 shows, though the MPEG-2 down-conversion gives better PSNR, the difference is minor and we can conclude that the quality of the 4.5 Mbit/sec high resolution has not been altered by the smoothing that our new down-conversion method causes.

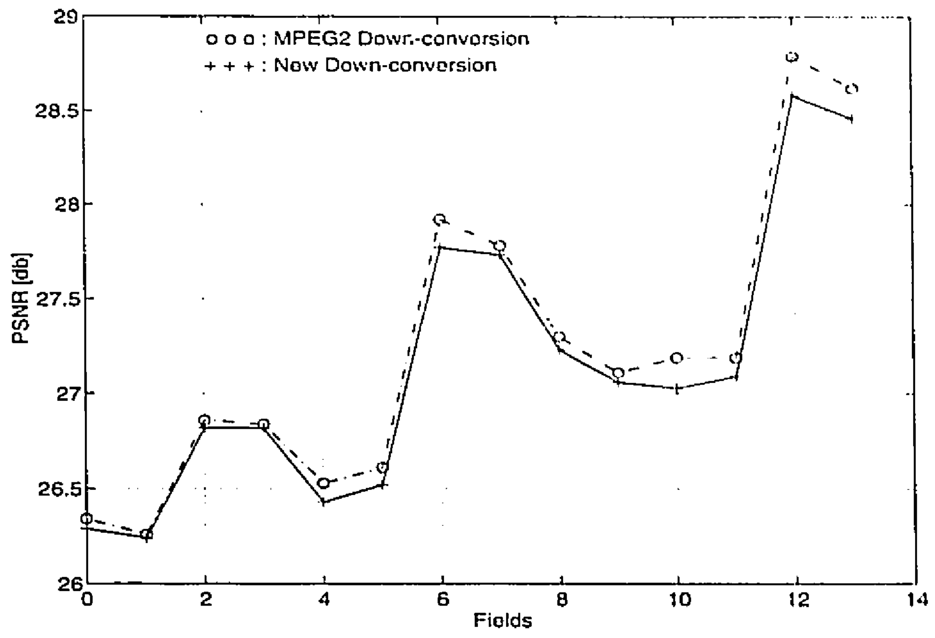


Figure 5.7: PSNR results for the high resolution decoded Calendar sequence using different down-conversion methods

Chapter 6

Conclusion

In this thesis we have presented scan conversion algorithms which apply to the conversion between camera, transmission and displays of different formats and to scalability. Scan conversion from interlaced to progressive formats, also known as deinterlacing, was done using adaptive spatiotemporal interpolation. For this, spatial and motion-compensated interpolation were adaptively weighted and switched (smaller error decision). The performance of these adaptive schemes were visually compared with spatial deinterlacing and motion-compensated deinterlacing. Scan conversion for 2:1 field rate conversion, from a progressive format to an interlaced one, was also achieved using adaptively temporal and motion-compensated interpolation. This frame conversion was evaluated in the conversion from a SIF transmission format to a CCIR601 display format. Multiscale spatial interpolation was also addressed. This was achieved using pattern motion-compensated interpolation. Visual performance of the latter method was compared with MPEG-2 spatial interpolation, cubic interpolation and pattern interpolation. Finally we have studied the impact that scan conversion had on MPEG-2 scalable coding, and this at both pre- and post-processing.

We have introduced a new method to perform frame rate conversion. This method was based on our conclusion that temporal interpolation performs poorly for large motion. Frame rate conversion was done using adaptively motion-compensated interpolation and temporal interpolation. At pixels in a neighborhood of low velocity, temporal interpolation was performed. Motion-compensated interpolation was done at pixels in a neighborhood of high velocity. A motion detector based on a neighborhood velocity averaging was used for switching between temporal and motion-compensated interpolation. Maximally flat digital filters were designed and used for generating the intermediate samples of temporal interpolation. This was done in order to improve the performance of temporal interpolation.
We have also introduced a new method, based on [37], that achieves multiscale spatial interpolation: pattern motion-compensated interpolation. For this, pattern interpolation was applied to a denser grid which was obtained through motion-compensated prediction from a determined number of previous and forward picture fields. We have developed a least square solution using Cauchy steepest descent. We have further proved the existence and uniqueness of such solution.
A new method was also proposed for the pre-processing of MPEG-2. The simple dropping of every second field will be preceded by deinterlacing followed by vertical resampling.

In the deinterlacing problem, spatial deinterlacing introduced artifacts such as loss of resolution and smoothing of edges. These distortions are removed with motion-compensated deinterlacing. However wrong motion vectors due to bad estimation or occlusions effects undermine the motion-compensated interpolation and results in edge serration. Using both weighted sum and switched methods removes part of the error caused by occlusions and bad motion estimation. Indeed in the case of motion estimation error, the error term E_m will be less than E_s and hence the value obtained by spatial interpolation is given a higher weight in the case of weighted sum and is the value itself in the smaller error decision case.

Temporal frame rate conversion fails to follow the motion of objects and therefore causes duplication of objects and contours which results in an annoying flickering. This is avoided by using motion-compensated interpolation. However in regions having null motion, temporal interpolation gives better results than motion-compensated interpolation, due mainly to motion errors which have a larger impact when motion is almost null. Moreover adaptive frame rate conversion gave the highest PSNR when compared with temporal conversion, motion-compensated conversion and MPEG-2 conversion. Furthermore using maximally flat digital filters in the temporal interpolation gave better results in terms of reduction of aliasing and preserving resolution than the one obtained with a two tap linear filter. Using a maximally flat filter of 16 coefficients offered a good trade off between complexity and image resolution.

Pattern motion-compensated interpolation gave better visual results than cubic interpolation and MPEG-2 method. This better performance corresponds to higher resolution in diagonal edges and removal of staircase effect. Furthermore this new method gave better results than the method in [37] especially at horizontal edges.

In the pre-processing of MPEG-2, replacing the simple dropping of every second frame by deinterlacing followed by vertical resampling turned out to give a large decrease of vertical aliasing. Furthermore the flickering due to that aliasing was significantly removed. Thus the new down-conversion method gave a very good quality, and the possible decrease in resolution, due to smoothing in the even fields was not noticeable when compared with the MPEG-2 sequence. At the coding, replacing MPEG-2 spatial linear up-conversion with our adaptive frame rate conversion resulted in a higher

PSNR and a better image quality.

Most of the problems faced in the realization of this thesis were due to bad motion estimation and to the long time that the motion estimator program required. With the development of the VLSI technology, chips that realize fast and good motion estimator, at a reasonable price, are going to be available which will make the methods proposed in this thesis suitable for hardware implementation. However good quality motion estimates is a must, and robust motion estimators are to be developed. Pattern motion-compensated interpolation seems promising. Faster and better fitting algorithms will allow the use of numerous fields in the forward and backward prediction, which will result in a much denser grid. However this also requires adequate motion estimation, and algorithms that go beyond the linear motion estimation model can be used. In the new down-conversion scheme, the outcome highly depended on the quality of the deinterlacing. Better deinterlacing schemes are needed and if motion-compensated deinterlacing is to be used, more robust motion estimators should be examined.

Bibliography

[1] L. Capodiferro, "Interlaced to progressive conversion by median filtering," in *Procd. 3-rd Int. Workshop on Signal Processing of HDTV*, Sept. 1989.

[2] R. Castagno, P. Haavisto, G. Ramponi, and M. Balanza, "A motion compensated algorithm for frame rate up conversion of progressive image sequences," *International Workshop on HDTV'93*, Ottawa, Canada, vol. II, Oct. 1993.

[3] T. Chiang and D. Anastassiou, "Hierarchical coding of digital television," *IEEE Communications Magazine*, vol. 32, pp. 38–45, May 1994.

[4] B. Chupeau and P. Salomon, "Motion compensated deinterlacing for studio applications," *International Workshop on HDTV'93*, Ottawa, Canada, vol. II, Oct. 1993.

[5] G. de Haan, P. Biezen, H. Huijgen, and O. Ojo, "Graceful degradation in motion compensated field-rate conversion," *International Workshop on HDTV'93*, Ottawa, Canada, vol. II, Oct. 1993.

[6] R. Depommier and E. Dubois, "Motion estimation with detection of occlusion areas," *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 269–272, Mar. 1992.

[7] T. Doyle and M.Looymans, "Progressive scan conversion using edge information," in *Procd. 3-rd Int. Workshop on Signal Processing of HDTV*, Aug. 1989.

[8] E. Dubois, "Motion-compensated filtering of time-varying images," *Multidimensional Systems and Signal Processing*, vol. 3, pp. 211–239, 1992.

[9] E. Dubois and J. Konrad, "Motion estimation and motion compensated filtering of video signals," *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 95–98, Apr. 1993.

[10] E.Dubois, "The sampling and reconstruction of time-varying imagery," *Proc. IEEE*, vol. 73, pp. 502–522, Apr. 1985.

[11] M. Ernst and T. Reuter, "Adaptive filtering for improved standards conversion," in *Procd. 2-nd Int. Workshop on Signal Processing of HDTV*, Mar. 1988.

[12] M. Ernst and T. Reuter, "A universal standards converter for HDTV applications," in *Procd. 3-rd Int. Workshop on Signal Processing of HDTV*, Aug. 1989.

[13] A. Gersho, "Optimal nonlinear interpolative vector quantization," *IEEE Trans. on Comm.*, vol. 38, pp. 1285–1287, Sept. 1990.

[14] B. Girod and R. Thomas, "Motion compensated field interpolation from interlaced and non-interlaced grids," *Proceedings of SPIE conference on Image coding*, M. Kunt and T.S. Huang (ed.), vol. 594, pp. 186–193, 1985.

[15] M. Haghiri and P. Guishard, "A motion compensated field rate conversion algorithm," in *Procd. 3-rd Int. Workshop on Signal Processing of HDTV*, Aug. 1989.

[16] B. Horn and B. Schunck, "Determining Optical Flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[17] A. K. Jain, *Fundamentals of digital image processing*. Prentice-Hall, 1989.

[18] K. Jensen and D. Anastassiou, "Spatial resolution enhancement of images using nonlinear interpolation," *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 2045–2048, Apr. 1990.

[19] R. G. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-29, pp. 1153–1160, Dec. 1981.

[20] R. Lagendijk and M. Sezan, "Motion compensated frame rate conversion of motion pictures," *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 453–456, Mar. 1992.

[21] M. Lamnabhi and J. Lhuilier, "Motion compensated time rate conversion of video sequences," in *Procd. 2-nd Int. Workshop on Signal Processing of HDTV*, Mar. 1988.

[22] G. Lampropoulos and M. Fahmy, "A new technique for the design of two-dimensional FIR and IIR filters," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. ASSP-33, pp. 268–280, Feb. 1985.

[23] G. Morisson and I. Parke, "A spatially layered hierarchical approach to video coding," *Signal Processing: Image Communication*, vol. 5, pp. 445–462, Dec. 1993.

[24] MPEG-2, "Coding of moving pictures and associated audio," *Draft International Standard: ISO/IEC 13818-2*, Mar. 1994.

[25] N. Y. N. Aikawa and M.Sato, "Design method of maximally flat FIR filter in consideration of even and odd order," *IEEE Int. Symp. Circuits and systems*, Singapore, pp. 276–279, 1991.

[26] A. Nguyen, "L'interpolation et la conversion de la structure d' échantillonage des images entrelacées en mouvement," Master's thesis, INRS-Télécommunications, 1991.

[27] A. Oppenheimen and R. Schafer, *Discrete-time signal processing*. Prentice-Hall, 1989.

[28] D. Pele, P.Siohan, and B. Choquet, "Field rate conversion by motion estimation/compensation," in *Procd. 2-nd Int. Workshop on Signal Processing of HDTV*, Mar. 1988.

[29] A. Puri and A.Wong, "Spatial domain resolution scalable video coding," in *SPIE Visual Communications and Image Processing Conference*, Boston, Mass., Nov. 1993.

[30] L. Rabiner and B. Gold, *Theory and application of digital signal processing.* Prentice-Hall, 1975.

[31] L. Rajagopal and S. D. Roy, "Optimal design of maximally flat FIR filters with arbitrary magnitude specifications," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-37, pp. 512–518, Apr. 1989.

[32] T. Reuter, "Standards conversion using motion compensation," *Signal Processing*, no. 16, pp. 73-82, 1989.

[33] G. Shamel, "Pre- and postfiltering of HDTV signals for sampling rate reduction and display up-conversion," *IEEE Trans. Circuits Syst.*, vol. CAS-34, pp. 1432–1439, Nov. 1987.

[34] L. Vandendorpe, L. Cuvelier, B. Maison, and P. Delogne, "Improved motion-compensated conversion between interlaced TV and HDTV formats," *International Workshop on HDTV'93,* Ottawa, Canada, vol. I, Oct. 1993.

[35] M. Vetterli and M. Uz, "Multiresolution coding techniques for digital television: a review (invited paper)," *Multidimensional Systems and Signal Processing*, vol. 3, pp. 161–187, 1992.

[36] D. Vleescuver and I. Bruyland, "Non-linear interpolators in compatible HDTV image coding," in *Procd. 2-nd Int. Workshop on Signal Processing of HDTV*, Mar. 1988.

[37] Y. Wang and S. K. Mitra, "Motion/pattern adaptive interpolation of interlaced video sequences," *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 2829–2832, May 1991.

[38] Y. Wang and S. K. Mitra, "Image representation using block pattern models and its image processing applications," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 15, pp. 321–336, Apr. 1993.

[39] M. Weston, "Fixed, adaptive, and motion compensated interpolation of interlaced TV pictures," in *Procd. 2-nd Int. Workshop on Signal Processing of HDTV*, Mar. 1988.