INTERPOLATION AND MOTION COMPENSATION OF INTERLACED VIDEO

by

David Hargreaves

B.A.Sc. Simon Fraser University, 1992

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE
in the School
of
Engineering Science

SIMON FRASER UNIVERSITY
July, 1994

APPROVAL

Name: David Alan Edward Hargreaves

Degree: Master of Applied Science

Title of thesis : Interpolation and Motion Compensation of Interlaced Video

Examining Committee: Dr. John Jones
Associate Professor of Engineering Science, SFU
Chair

Dr. Jacques Vaisey
Assistant Professor of Engineering Science, SFU
Senior Supervisor

Dr. Paul Ho
Associate Professor of Engineering Science, SFU
Internal Supervisor

Dr. John Dill
Professor of Engineering Science, SFU
Examiner

Date Approved: July 20, 1994

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

"Interpolation and Motion Compensation of Interlaced Video"

Author:

(signature)

(name)

July 13, 1994
(date)

Abstract

This thesis investigates the associated problems of motion estimation, interpolation and motion compensation when applied to interlaced video sequences. Motion estimation has many different applications including interpolation, motion compensation, and object recognition. Interpolation is important for the purposes of converting interlaced video to progressive sequences, the "pause" function on video tape recorders, and, as will be shown in this thesis, motion compensation. Motion compensation, on the other hand, is used in video codecs (such as those based on MPEG and H.261) to remove temporal redundancy. This technique is critical if the codec is to operate at a low rate.

In this thesis, a number of different motion estimation methods and de-interlacing techniques (both spatial and motion compensated) are considered, and applied to different motion compensation algorithms for interlaced sequences.

Our investigation of motion estimation and interpolation methods focuses on Bayesian techniques. Using statistical models for the motion field, the Maximum A Posteriori Probability (MAP) estimate of the motion field is obtained by applying simulated annealing. Modifications to this technique are included for use with interlaced sequences. These modifications have proven to provide excellent reconstruction of the missing lines in interlaced video.

Using the interpolation results from the spatial and Bayesian techniques, various methods of motion compensating interlaced sequences are considered that use information from two previous fields to improve performance. The new techniques provide significant improvement over current methods. However, despite the improved interpolation obtained by the Bayesian methods, the performance of the motion compensation methods using spatial interpolation techniques is very similar to that of the Bayesian methods. This significant result indicates that good interlaced motion

compensation algorithms may not need complex motion estimation methods.

Acknowledgments

I would like to thank my senior supervisor, Dr. J. Vaisey, for his guidance, support and reassuring comments/suggestions throughout this work. He always seemed available for any question (important or otherwise).

In addition, I would like to thank my ex-fellow student, Marlo Gothe, for his help and endless supply of answers (even after he had left)!

I am grateful to Canadian Cable Labs, the SFU Graduate Fellowship Committee, and the School of Engineering Science for their financial support through research grants, fellowships and teaching assistantships respectively.

Finally I would like to thank my parents, Rod and Pat, my brother, Brian, and of course, my fiancée, Lora, for their support during my studies.

Contents

Abstract iii

Acknowledgments v

List of Tables x

List of Figures xiv

List of Symbols xv

1 Introduction 1

1.1 Interlaced Video 1

1.1.1 Notation 2

1.2 Motion Compensation 4

1.2.1 Block-Based Motion Compensation 5

1.3 Motion Compensation of Interlaced Video 6

1.4 Interlaced-to-Progressive Conversion 7

1.5 Test Video Sequences 7

1.6 Performance Measures 8

1.7 Contributions of this Thesis 10

2 Spatial Interpolation of Interlaced Sequences 12

2.1 Linear, Quadratic and Cubic Interpolators 13

2.1.1 Linear 13

2.1.2 Quadratic 14

2.1.3 Cubic 14

2.2 Up-sampling 14
2.3 Least Mean Squared Interpolator 15
2.4 Line Shift Interpolation 16
2.5 Experimental Results 17

3 Motion Compensation of Interlaced Sequences 20
3.1 Interlaced Frame-Based Motion Compensation 20
3.2 Single Field-Based Motion Compensation 21
3.2.1 Different Parity Field 21
3.2.2 Different Parity Field–Interpolated 22
3.2.3 Same Parity Field 22
3.3 Multiple Field-Based Motion Compensation 22
3.4 Hybrid Motion Compensation 23
3.5 Results 23
3.5.1 Previous Frame Performance 23
3.5.2 Comparision of Motion Compensation Methods 24
3.5.3 Effect of Interpolation on MC Methods 25

4 Bayesian Formulation of Motion Estimation 28
4.1 Gibbs Distribution and Markov Random Fields 28
4.1.1 Gibbs Distribution 28
4.1.2 Markov Random Fields 31
4.2 Maximum A Posteriori Probability (MAP) Estimation 33
4.3 Models 34
4.3.1 Structural Model 35
4.3.2 Observation Model 35
4.3.3 Displacement Field Model 36
4.4 A Posteriori Probability 37
4.5 Stochastic Solution to Motion Estimation 38
4.5.1 Gibbs Sampler 38
4.5.2 Simulated Annealing 40
4.6 Motion Discontinuities 42
4.6.1 Displacement Field Model with Motion Discontinuities 42

4.6.2 Line Field Model 43
4.6.3 Gibbs Sampler for Model with Motion Discontinuities 46
4.7 Results 47
4.7.1 Results for the *comp* sequence 47
4.7.2 Results for *pongi* sequence 49
4.7.3 Computational Cost 54

5 Interlaced Bayesian Motion Estimation 56
5.1 Estimating Motion in Interlaced Sequences 56
5.2 Structural Model Fault 56
5.3 Improved Interpolation 58
5.4 Pixel Prediction Along Linear Trajectories 58
5.5 Variable Previous Frame Motion Estimation 59
5.5.1 Modified Variable Previous Frame Motion Estimation 63
5.6 Dual Frame Motion Estimation 63
5.7 Results 64
5.7.1 Interpolation Results 64
5.7.2 Motion Compensation Results 75

6 Reduced Complexity Motion Estimation 85
6.1 Determining Optical Flow 85
6.1.1 Results 87
6.2 Gauss-Newton Minimization of the MAP Criterion 88
6.2.1 Results 91
6.3 Survey of Other Motion Estimation Methods 91
6.3.1 Block-based Motion Estimation 93
6.3.2 Pixel-recursive Motion Estimation 93
6.3.3 Hierarchical Motion Estimation 94

7 Conclusions 95
7.1 Motion Estimation 95
7.2 Interpolation of Interlaced Video 96
7.3 Motion Compensation of Interlaced Video 96
7.4 Future Work 97

References 98

List of Tables

4.1 Computational Cost of Various Motion Estimation Methods 55

5.1 Terminology for various motion estimation methods used in the results presented in this chapter. 65

5.2 Parameters for methods 2 through 5 for the results presented in this chapter. 65

List of Figures

1.1 Spatial filtering characteristics of the human visual system (Netravali and Haskell 1988) 3
1.2 Three fields of an interlaced video sequence 3
1.3 Block-based motion compensation 5
1.4 First Frames of Simulation Test Video Sequences *pongi*, *missa*, *comp* and *caltrain* 9

2.1 Interpolation of Interlaced Image Columns 12
2.2 Bandlimited White Noise Model 16
2.3 Line shift interpolation 17
2.4 Performance of Spatial Interpolation Techniques on the *pongi* sequence 18

3.1 Frame formed from combination of the first two fields of the interlaced *pongi* sequence. 21
3.2 Performance of Motion Compensation Algorithm using only Ω_{n-1} on the *pongi* sequence 24
3.3 Performance of Various Motion Compensation Algorithms on the *pongi* sequence 25
3.4 Performance of Two Field Interlaced Motion Compensation using Spatial Interpolation Techniques on the *pongi* sequence 26
3.5 Performance of Hybrid Interlaced Motion Compensation using Spatial Interpolation Techniques on the *pongi* sequence (seven steps) 27

4.1 Neighborhoods of order 1 through 3. Note that system of order k includes the sites contained in order $k-1$ 30

4.2 The First Order Neighborhood system (a) along with associated cliques: (b) one-element and (c),(d) two-element 31
4.3 The Second Order Neighborhood system (a) along with associated cliques: (b) one-element, (c),(d),(e),(f) two-element, (g),(h),(i),(j) three-element, (k) four-element 32
4.4 The Motion Estimation Process 34
4.5 Block diagram of Gibbs Sampler algorithm 39
4.6 Block diagram of simulated annealing algorithm 41
4.7 (a) First order Neighborhood system $\mathcal{N}^1_{(\mathbf{d},l)}$ for vector field $\mathbf{d}_t$ with line field l_t and the associated (b) horizontal and (c) vertical cliques. 43
4.8 Neighborhood system $\mathcal{N}_l$ for line field l_t of (a) horizontal and (b) vertical line element, and the associated (c),(d) four-element and (e),(f) two-element cliques. 44
4.9 Potentials associated with various line element configurations (up to a rotation) for the (a) four-element and (b) two-element cliques. 45
4.10 Actual motion field for moving block area of the second frame of the *comp* sequence. 48
4.11 Motion field for moving block area of the second frame of the *comp* sequence. No line field is used. $\lambda_{\mathbf{d}} = 0.3, \lambda_g = 0.1$. Exponential annealing schedule. $T_0 = 5.0, a = 0.9, 100$ iterations. 50
4.12 Motion field for moving block area of the second frame of the *comp* sequence. Line field after 39 iterations. $\lambda_{\mathbf{d}} = 0.3, \lambda_g = 0.1, \lambda_l = 0.5$. Exponential annealing schedule. $T_0 = 5.0, a = 0.9, 100$ iterations. For line field $T_l = 50T$. 51
4.13 Motion field for shoulder and upper arm area of the second frame of the *pongi* sequence. No line field is used. $\lambda_{\mathbf{d}} = 0.07, \lambda_g = 0.0025$. Exponential annealing schedule. $T_0 = 5.0, a = 0.9, 100$ iterations. 52
4.14 Motion field for shoulder and upper arm area of the second frame of the *pongi* sequence. Line field after 39 interations. $\lambda_{\mathbf{d}} = 0.3, \lambda_g = 0.0025, \lambda_l = 0.5$. Exponential annealing schedule. $T_0 = 5.0, a = 0.9, 100$ iterations. For line field $T_l = 50T$. 53

5.1 Small section of progressive frame 57

5.2 Small section of vertically, spatially interpolated interlaced frame corresponding to the progressive frame in the previous figure 58
5.3 Weighting Function 60
5.4 Three previous frames used in variable motion estimation 62
5.5 Example of two different current frames 62
5.6 Diagram of the y-t plane of an interlaced sequence showing different displacement vectors. 64
5.7 Performance of motion-estimated interpolation methods on the *pongi* sequence. 67
5.8 Actual second frame of the *pongi* sequence. 67
5.9 Reconstructed second frame of the *pongi* sequence using method 1. ... 68
5.10 Reconstructed second frame of the *pongi* sequence using method 2. ... 68
5.11 Reconstructed second frame of the *pongi* sequence using method 3. ... 69
5.12 Reconstructed second frame of the *pongi* sequence using method 4. ... 69
5.13 Reconstructed second frame of the *pongi* sequence using method 5. ... 70
5.14 Performance of motion-estimated interpolation methods on the *caltrain* sequence. 72
5.15 Actual second frame of the *caltrain* sequence. 72
5.16 Reconstructed second frame of the *caltrain* sequence using method 1. . 73
5.17 Reconstructed second frame of the *caltrain* sequence using method 4. . 73
5.18 Error frame for the reconstruction of the second frame of the *caltrain* sequence using method 1. 74
5.19 Error frame for the reconstruction of the second frame of the *caltrain* sequence using method 4. 74
5.20 Performance of motion-estimated interpolation methods 76
5.21 Performance of spatial method 1 and Bayesian method 4 using two field motion compensation for the *pongi* sequence 77
5.22 Performance of spatial method 1 and Bayesian method 4 using two field motion compensation for the *caltrain* sequence 77
5.23 Performance of various Bayesian interpolation methods using two field motion compensation for the *pongi* sequence 78
5.24 Performance of various interpolation methods using three field motion compensation for the *pongi* sequence 79

5.25 Performance of methods 1 and 4 using three field motion compensation for the *caltrain* sequence 80
5.26 Performance of various interpolation methods using hybrid motion compensation for the *pongi* sequence 81
5.27 Performance of methods 1 and 4 using hybrid motion compensation for the *caltrain* sequence 81
5.28 Fractional motion compensation 82
5.29 Performance of various interpolation methods using fractional pixel motion compensation for the *pongi* sequence 83
5.30 Performance of various interpolation methods using fractional pixel motion compensation for the *caltrain* sequence 84

6.1 Sample location for estimation of the partial derivatives of the brightness 88
6.2 Motion Field for the moving block area of the second frame of the modified *comp* sequence generated using the Horn & Schunck algorithm. 100 iterations. 89
6.3 Motion Field for the moving block area of the second frame of the *comp* sequence generated using the Horn & Schunck algorithm. 100 iterations. 90
6.4 Motion Field for the moving block area of the second frame of the *comp* sequence generated using the Gauss-Newton minimization of the MAP criterion. $\lambda_{\mathbf{d}} = 0.3, \lambda_g = 0.0025$, 100 iterations. 92

List of Symbols

Υ_n	current frame of a progressive sequence
Υ_{n-1}	previous frame of a progressive sequence
Ω_n	current field of an interlaced sequence
Ω_{n-1}	previous field of an interlaced sequence (opposite parity from current field)
Ω_{n-2}	second previous field of an interlaced sequence (same parity as current field)
Ψ_n	current frame of an interlaced sequence composed of Ω_n and Ω_{n-1}
Ψ_{n-1}	previous frame of an interlaced sequence composed of Ω_{n-1} and Ω_{n-2}
$\Lambda_{\mathbf{d}}$	discrete sampling lattice for displacement vectors
Λ_g	discrete sampling lattice for pixel amplitudes
$\mathcal{N}$	neighborhood system
$\mathcal{N}_j$	j^{th} order neighborhood system
s_i	i^{th} site on a lattice
$\eta(s_i)$	neighborhood of i^{th} site
c	clique
$\mathcal{C}$	set of cliques
$V(\mathcal{X}, c)$	potential function – dependent only on the samples of $\mathcal{X}$ that are in c
$\mathcal{X}$	sample from random process X
δ	vertical sampling period of an interlaced field
Δ	offset from origin of interpolation point
a	constant in simulated annealing schedule
α	constant in a potential for single-element line clique
g	observed image (data)
$\tilde{g}$	interpolated observed image at positions not on sampling grid
G	image random field

n	noise (assumed Gaussian)
$\mathbf{\Gamma}$	Markov random field
Γ_i	i^{th} site in a Markov random field
$\mathbf{d}$	displacement field (motion field)
$\hat{\mathbf{d}}$	estimate of the displacement field
$\hat{\mathbf{d}}^*$	optimal estimate of the displacement field
l	displacement discontinuity field (line field)
$\hat{l}$	estimate of the displacement discontinuity field
$\hat{l}^*$	optimal estimate of the displacement discontinuity field
$M_{\mathbf{d}}$	number of vectors in one displacement field
t_-	temporal position of previous frame/field
t_+	temporal position of current frame/field
$U, U_g, U_{\mathbf{d}}, U_l$	energy functions in Gibbs probability distributions
$U^i_{\mathbf{d}}$, U^i_l	local energies used in the Gibbs sampler
$V, V_{\mathbf{d}}, V_l$	potential functions
φ	temperature schedule
$\mathbf{x}$	spatial position in an image (frame/field)
$Z, Z_{\mathbf{d}}, Z_l$	normalizing constants in Gibbs probability distributions
r	displaced pixel difference
$\tilde{r}$	estimation of the displaced pixel difference using $\tilde{g}$
T_0	initial temperature in simulated annealing process
T_n	temperature at n^{th} iteration in simulated annealing process
T_f	final temperature in simulated annealing process
n_τ	index of site at which replacement is made at time τ in the Gibbs sampler

Chapter 1

Introduction

Currently, the cable and telecommunications industry is undergoing a shift in emphasis away from analog methods to new digital video transmission such as those found in High Definition Television (HDTV) and in CCIR 601, which is the video standard for studio quality digital transmission. In order for digital video transmission to be used over typical channels such as that used by the NTSC standard (equivalent to 10Mbits/sec), compression must be performed. This compression is the practice of eliminating redundancy from the original data stream and/or introducing distortion in order to reduce the required data rate.

One of the most effective and hence most common tools used in source coding for digital video is called *motion compensation* (MC). Motion compensation allows the elimination of some of the frame-to-frame redundancy by estimating the motion between the two frames. This motion information can then transmitted along with the resulting error (lossless coding). While this topic has been studied extensively for progressive (non-interlaced) video sequences, there are very few methods specifically designed to improve motion compensation for interlaced video.

1.1 Interlaced Video

When television programs or movies are displayed on a screen, it is in the form of individual pictures or frames which are rapidly displayed in succession. This method takes advantage of the ability of the human visual system (HVS) to temporally interpolate between frames (Netravali and Haskell 1988). This ability is dependent on the

ambient lighting level and the overall brightness of the displayed frames–the brighter the display, the more chance of visible flicker.

In movies, a technique known as *progressive* video is used where a series of full frames are displayed rapidly after one another. To reduce the visible flicker, the projector normally flashes each frame onto the screen two or three times. This is sufficient for dark movie theaters; however, television cannot do this since it would require considerable storage to redisplay frames. Therefore, a new technique is needed such as *interlacing.*

Interlaced video is an efficient technique that attempts to take advantage of the spatial-frequency dependent characteristics of the HVS to eliminate flicker. As shown in Figure 1.1, the HVS performs sharper filtering on higher spatial frequencies than lower spatial frequencies. In television, 2-to-1 interlace is used. This technique displays lower spatial frequencies at a higher rate than the higher spatial frequencies by scanning and displaying first the odd lines of the frame and then the even lines as shown in Figure 1.2. Each of these sets of lines is known as a *field.* Since individual lines are only updated every second field, the high spatial frequencies contained on individual lines are not displayed as rapidly as the low spatial frequencies represented by groups of lines.

One effect that results from the interlaced technique occurs when a single frame is viewed. Since the odd and even fields are scanned at different times, the edge of moving objects can appear serrated since in one of the fields the object will be in a slightly different position than in the other field. This effect can be seen when using the "pause" feature on a video tape recorder.

1.1.1 Notation

Three sets of notation will be used in this thesis to refer to video sequences:

1. Υ will be used to designate the frames of progressive sequences.
2. Ω will be used for individual fields of an interlaced sequence.
3. Ψ will be used to refer to frames of an interlaced sequence. Note that two fields of an interlaced sequence are combined to form a frame.

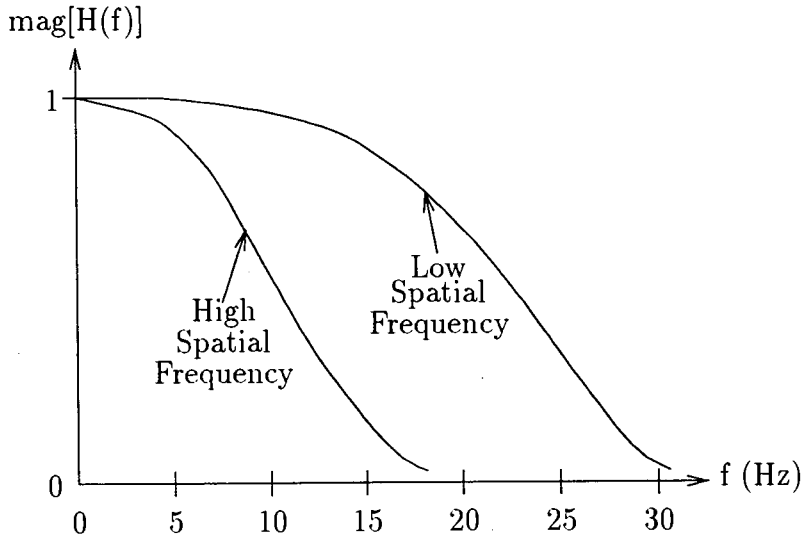


Figure 1.1: Spatial filtering characteristics of the human visual system (Netravali and Haskell 1988)

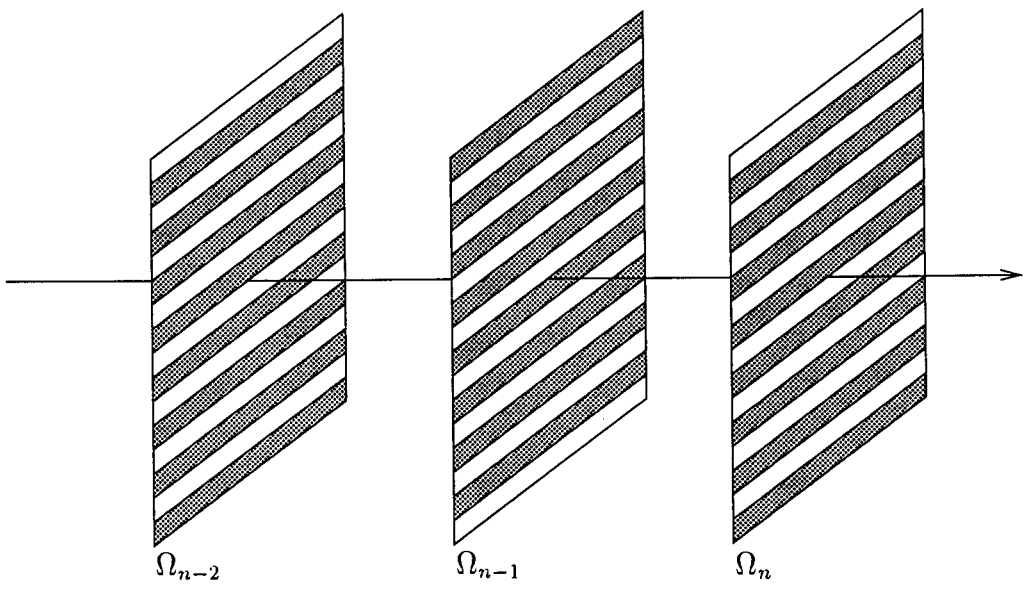


Figure 1.2: Three fields of an interlaced video sequence

In each case subscripts will be used to indicated the location in time of a given frame or field. For example, Υ_n, Ω_n, and Ψ_n all refer to the n^{th} or current frame/field in the respective sequence, and Υ_{n-1}, Ω_{n-1}, and Ψ_{n-1} all refer to the $(n-1)^{th}$ or previous frame/field in the respective sequence. In addition, individual pixels in a frame/field will be referred to by the notation $\Upsilon_n(x,y)$, $\Psi_n(x,y)$ or $\Omega_n(x,y)$ where the pixel has the spatial location (x,y).

In interlaced sequences, the current field, Ω_n, has different parity from the previous field, Ω_{n-1}, and has the same parity as the second previous field, Ω_{n-2}. The current frame in an interlaced sequence, Ψ_n, is composed of the two fields, Ω_n and Ω_{n-1}, combined together to form a full frame.

1.2 Motion Compensation

Motion compensation is a technique that uses the redundancy between frames in a video sequence to compress the data. The idea is to predict the next frame by estimating the motion between frames. Once a motion estimate has been made, the algorithm only transmits the (usually quantized) difference between two frames, which is contained in the motion information and the estimation error. If the motion estimation is good, a high rate of compression can be achieved (Hsing 1987).

In this technique the picture is divided into two parts – the *motion vectors*, which estimate the motion in the picture, and the *residual*, which is the error between the actual image and its estimate. In lossless coding, both the exact motion vector and a version of the residual must be transmitted. However, if the motion estimation is good, only the motion vector need be transmitted.

There exist numerous different methods for estimating the motion between frames. Two major categories of motion compensation are block-based motion compensation and pixel-recursive motion compensation. Block-based motion compensation is the most common of the methods since it is generally more computationally realistic than other methods. Therefore, block-based motion compensation is used in MPEG, which is the current video compression standard (LeGall 1991), and we will also use block-based motion compensation as the initial method in this research. A number of the other common techniques of motion estimation are surveyed in Chapter 6.

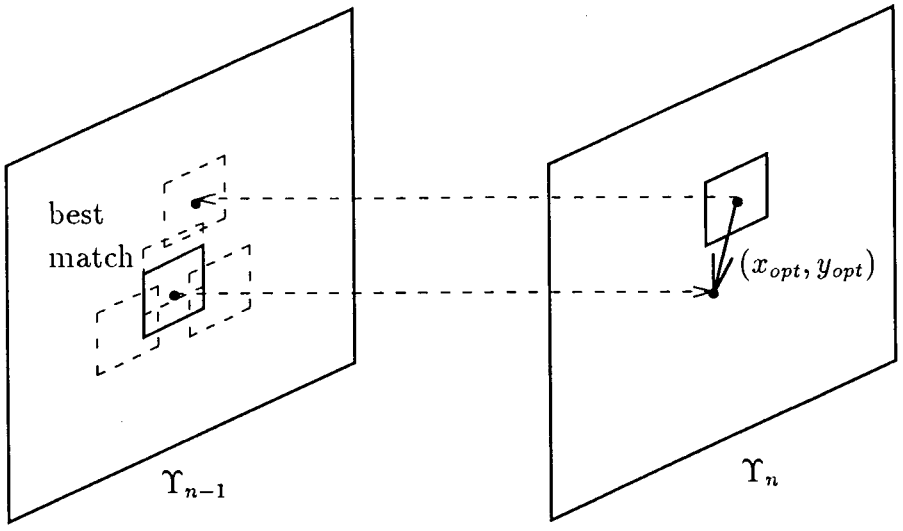


Figure 1.3: Block-based motion compensation

1.2.1 Block-Based Motion Compensation

Block-based motion compensation algorithms divide the current frame (progressive sequence) into blocks. Then, for each block in the current frame, a search of the previous frame or frames is made to find the best matching block (the block which has the smallest error measure with respect to the current block). Figure 1.3 shows two successive frames in a sequence – the current frame (Υ_n) and the previous frame (Υ_{n-1}). In the current frame, the present block is shown and in the previous frame the block in the same location as the current block is shown (dotted line) along with two possible matching blocks (dotted lines) and the best (in terms of some error measure) matching block (solid line). In this thesis the error measure that is used is the mean squared error (MSE), which will be described later in this chapter.

Many video sequences are made up largely of stationary objects (such as the background in a teleconferencing scene). Therefore, the majority of the motion vectors are often zero or close to zero. In order to minimize the number of bits required to code the motion vector, it is necessary that we pick not only the lowest error matching block but also the motion vector with the smallest magnitude. This means that if there are two blocks, both with the lowest error measure, then we should pick the match as the one with the lowest motion vector magnitude. The reason for this choice of motion vectors is that we want to decrease the entropy of the motion vectors to

improve compression after entropy coding (Gersho and Gray 1992) and, since the motion vectors are most commonly zero, choosing the smallest motion vector reduces the entropy.

Once the best match has been found, the motion vector ((x_{opt}, y_{opt}) in Figure 1.3) is calculated and transmitted for each block. If lossless transmission is required, then the error between the current block and its best matching block must also be transmitted. This error is known as the *residual.* In general the error is quantized and entropy coded before transmission, and therefore, it is desireable to reduce the energy of the error as much as possible (find the best match) (Gersho and Gray 1992). A lower energy corresponds to lower entropy and hence improved compression. The energy of the error will be formally defined later in this chapter.

The two main parameters in block-based motion compensation are the block size, b, and the search area size, p. The block size is the length of one side of the block, while the search size is the length from the middle of the current block to the edge of the area searched in the previous frame(s). Generally, the selection of these parameters involves a trade-off between computational complexity and performance.

1.3 Motion Compensation of Interlaced Video

In progressive video sequences, each frame contains both the odd and the even lines. Alternatively, in interlaced video sequences, every frame contains only the odd or even lines depending on its parity. For progressive sequences the best match is normally found in the previous frame, since the displacement of the motion is normally smallest between successive frames. However, in interlaced sequences, the previous field (different parity) does not contain the same lines as the current field, and therefore it is sometimes advantageous to use the second previous field (same parity). MPEG-2 does not take full advantage of this relationship since it uses either frame-based (Ψ_n is coded) or simple field-based (Ω_n is coded) methods. It does not use both previous fields during the search process. This thesis will address this problem along with the related problems of motion estimation for interlaced sequences and interpolation of interlaced sequences.

1.4 Interlaced-to-Progressive Conversion

Another important process is conversion from interlaced sequences to progressive sequences. This involves recovering the lines that are missing from the interlaced sequence. In previous work, simple schemes have been used such as simply combining two successive, different-parity fields to form a frame, or just repeating every line of a field to form the frame. Methods such as these give rise to annoying artifacts such as jagged object edges and blurring.

A significant section of this thesis will investigate interlaced-to-progressive conversion. Techniques that will be investigated include spatial filtering techniques; however, in order to obtain better results than spatial filtering, it is necessary to use motion estimates in the conversion process, and therefore, techniques of motion estimation for use in interlaced-to-progressive conversion will also be discussed.

1.5 Test Video Sequences

The four sequences used in the comparison of methods in this thesis are the "Ping-Pong" sequence (referred to as *pongi*), the "Miss America" sequence (referred to as *missa*, an artificial sequence known as the "Composite" sequence (referred to as *comp*) and the "Calendar-Train" sequence (referred to as *caltrain*). All three sequences are originally progressive sequences with frames of size 360 × 240 pixels, 360 × 288 pixels, 360 × 240 pixels, and 512 × 400 pixels. The *caltrain* sequence, however, has been cropped to a size of 360 × 240 pixels (from the bottom right corner) to reduce computations. The four sequences are gray-level sequences with pixel amplitudes quantized to 8 bits. In order to create interlaced sequences, even and odd lines were removed from odd and even frames respectively. This allowed comparison with the actual progressive frames for the purpose of performance measurement.

pongi sequence

The *pongi* sequence shows a high motion ping-pong game whose scene pans right and then left. Included in the background is a poster containing high detail, which includes horizontal lines that are a single pixel wide. The first frame of the *pongi*

sequence is shown in Figure 1.4. In addition, the dark background is composed of a random texture.

missa sequence

The *missa* sequence shows a speaking person's head and shoulders on a fixed background. It is a low motion sequence indicative of the type of video found in applications such as video conferencing. The first frame of the *missa* sequence is shown in Figure 1.4.

comp sequence

The *comp* sequence is composed of the first frame of the *pongi* sequence with a 50×50 pixel block from the *missa* sequence superimposed with its upper left corner initially at $(150, 150)$. The block is then moved by 4 pixels right and 4 pixels down to obtain the each following frame of the sequence. This sequence therefore has known motion and is used in evaluation of the motion estimation algorithms. The first frame of the *comp* sequence is shown in Figure 1.4.

caltrain sequence

The *caltrain* sequence contains high motion consisting of a toy train pushing a ball, a spinning mobile and a moving calendar. The background is wall-paper with homogeneous regions depicting various objects. The first frame of the *caltrain* sequence is shown in Figure 1.4.

1.6 Performance Measures

The performance of motion compensation/estimation can be measured objectively and/or subjectively. Objective measurements are the easiest to make, but objective performance measures do not imply the same subjective performance rating. In subjective measurements, the human visual system (HVS) is used to perceive impairments. As a result, objective and subjective measures of the same sequences may have different or even reversed performance ratings. In this thesis, strictly objective measures will be used, since these clearly relate the performance of different compression

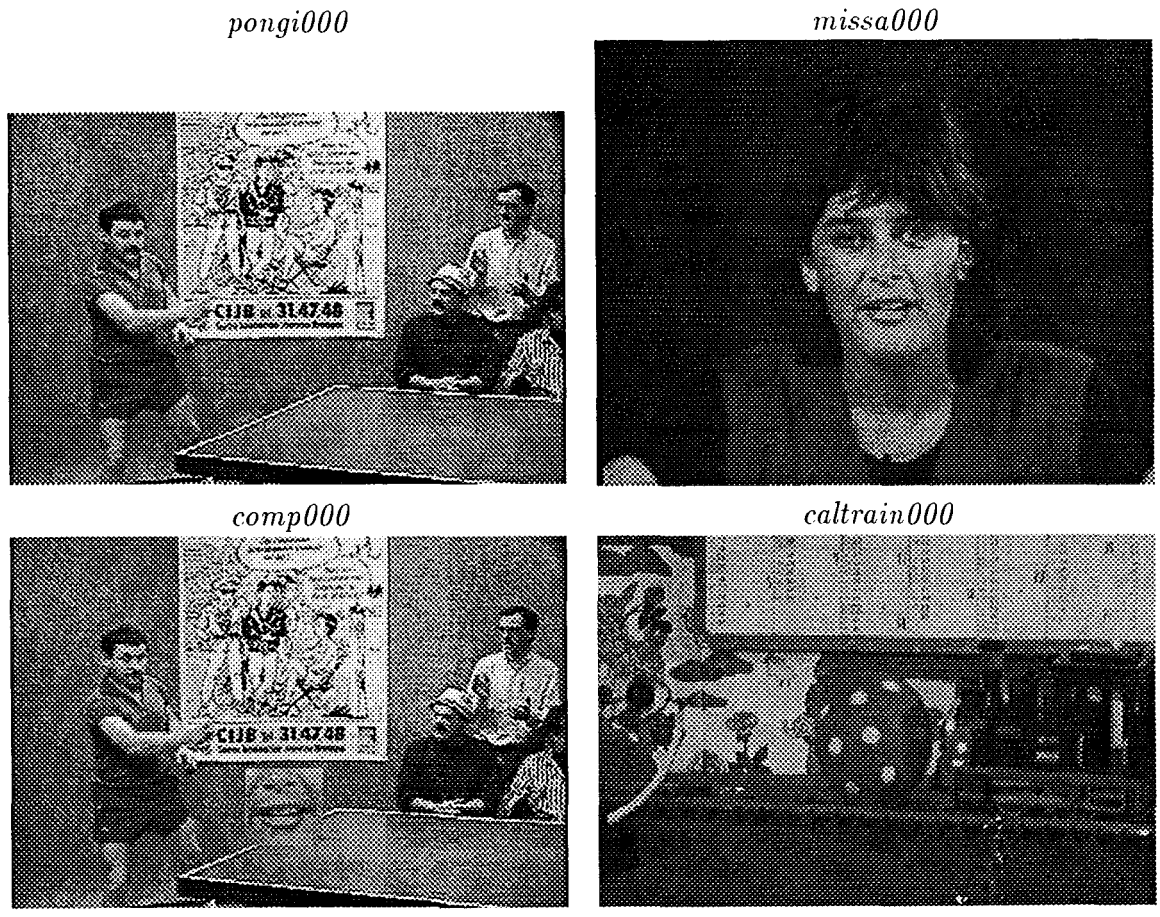


Figure 1.4: First Frames of Simulation Test Video Sequences *pongi*, *missa*, *comp* and *caltrain*

schemes and allow direct comparison with published results.

One of the most common objective performance measures used in image and video compression research is the *mean squared error* (MSE). This parameter is defined as

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}[\Upsilon_{orig}(i,j) - \Upsilon_{recon}(i,j)]^2 \tag{1.1}$$

where $\Upsilon_{orig}(i,j)$ is the pixel amplitude at location (i,j) of the original image Υ_{orig}, $\Upsilon_{recon}(i,j)$ is the pixel amplitude at location (i,j) of the reconstructed image Υ_{recon}, and M and N are the width and height of the images respectively.

MSE: Motion Compensation Performance Measure

In the case of motion compensation (MC), we are using our motion estimate to predict the next frame and therefore the mean squared error is a measure of how much the energy of the signal is reduced by removing the predictable information. A predictor is said to be *optimal* within some class of predictors if it minimizes the chosen error measure over all predictors in the given class (Gersho and Gray 1992). Therefore a motion compensation algorithm that results in a lower MSE than other MC algorithms is considered to perform better than other algorithms. For this reason the MSE is the chosen measure for comparison of MC methods in this thesis.

MSE: Frame Reconstruction Performance Measure

The MSE is unable to indicate where in the frame the error occurred and therefore does not correspond to HVS perceivable impairments directly. Therefore, in the case of frame reconstruction (eg. interlaced-to-progressive conversion), it can only be considered a reasonable estimate of image quality. A rule of thumb states that the HVS can perceive a changes in image quality of 1 dB.

Peak-Signal-to-Noise Ratio

The performance is also commonly measured in terms of the peak-signal-to-noise ratio (PSNR) which is related to the MSE by

$$PSNR = 10 \log_{10} \left(\frac{256^2}{MSE} \right) \tag{1.2}$$

where the numerator represents the square of the peak input pixel amplitude. Often the PSNR is perferred over the MSE since it is independent of the quantization applied to the frame or image.

1.7 Contributions of this Thesis

This thesis has contributed to the areas of motion estimation in interlaced video, reconstruction of interlaced video, and motion compensation of interlaced video. The major contributions can be summarized as follows:

1. A number of spatial interpolation methods have been implemented and compared for different video sequences (Chapter 2).

2. A novel, improved method of motion compensating interlaced video has been developed and implemented (Chapter 3).

3. Using Bayesian motion estimation (Chapter 4) and a number of new modifications for interlaced sequences, a better motion estimaton method has been presented which leads to better frame reconstruction (Chapter 5).

4. Two computationally efficient motion estimation methods have been implemented and applied to the reconstruction and motion compensation of interlaced video (Chapter 6).

Chapter 2

Spatial Interpolation of Interlaced Sequences

Spatial interpolation of an interlaced sequence is important, since it attempts to regenerate the missing information contained in the unknown fields using only the current field. This recovered information is useful in improving motion compensation, in accurately estimating the motion for the entire frame and in improving the perceptual quality when a single frame of an interlaced sequence is viewed alone.

The simplest methods of spatial interpolation involve considering the columns of a field as a one-dimensional signal and then applying a 1-D filter to interpolate the missing values. This process is shown in Figure 2.1 where $y(\Delta)$ is the value to be interpolated and Δ is the distance from the origin. The estimate of $y(\Delta)$ will be designated $\hat{y}(\Delta)$.

The interpolation process of a single pixel of an interlaced frame can be represented

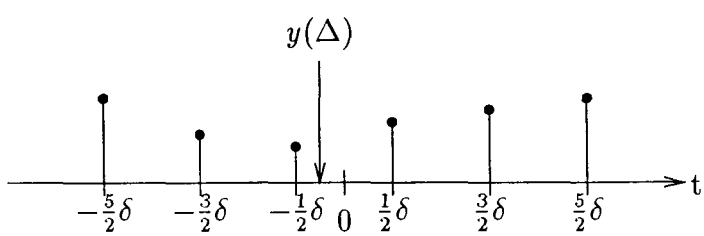


Figure 2.1: Interpolation of Interlaced Image Columns

as

$$\hat{y}(\Delta) = \mathbf{b}(\Delta)^T \begin{bmatrix} \vdots \\ y(-\frac{3}{2}\delta) \\ y(-\frac{1}{2}\delta) \\ y(\frac{1}{2}\delta) \\ y(\frac{3}{2}\delta) \\ \vdots \end{bmatrix} \tag{2.1}$$

where $\hat{y}(\Delta)$ is the interpolated value, Δ is distance from the origin in Figure 2.1, $\mathbf{b}(\boldsymbol{\Delta})$ is the filter as a function of Δ. The filter, $\mathbf{b}(\boldsymbol{\Delta})$, is multiplied by the sequence to be interpolated (samples of $y(t)$ at $t = \pm\frac{1}{2}\delta, \pm\frac{3}{2}\delta, \pm\frac{5}{2}\delta, \ldots$) where δ is the sampling period as shown in Figure 2.1. Since, in the case of interlaced-to-progressive conversion, we are interested in interpolating only the value exactly half-way between two samples of $y(t)$, $\Delta = 0$. For simplicity, the notation $\mathbf{b}$ will be used to represent $\mathbf{b}(0)$. The remainder of this Chapter will discuss different spatial methods for performing this interpolation.

2.1 Linear, Quadratic and Cubic Interpolators

The linear, quadratic and cubic interpolators (Konrad 1989) are outlined below for interpolation of interlaced sequences ($\Delta = 0$).

2.1.1 Linear

The linear interpolator is given by

$$\hat{y}(\Delta) = \begin{bmatrix} \frac{1}{2} - \frac{\Delta}{\delta} & \frac{1}{2} + \frac{\Delta}{\delta} \end{bmatrix} \begin{bmatrix} y(-\frac{1}{2}\delta) \\ y(\frac{1}{2}\delta) \end{bmatrix} \tag{2.2}$$

which, for the case where $\Delta = 0$, simplifies to

$$\hat{y}(0) = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} y(-\frac{1}{2}\delta) \\ y(\frac{1}{2}\delta) \end{bmatrix} \tag{2.3}$$

2.1.2 Quadratic

For the case where $\Delta = 0$, the quadratic interpolator is given by

$$\hat{y}(0) = \begin{bmatrix} -\frac{1}{8} & \frac{3}{4} & \frac{3}{8} \end{bmatrix} \begin{bmatrix} y(-\frac{3}{2}\delta) \\ y(-\frac{1}{2}\delta) \\ y(\frac{1}{2}\delta) \end{bmatrix} \tag{2.4}$$

2.1.3 Cubic

The cubic interpolator for the case where $\Delta = 0$ is given by

$$\hat{y}(0) = \begin{bmatrix} -\frac{3}{48} & \frac{9}{16} & \frac{9}{16} & -\frac{3}{48} \end{bmatrix} \begin{bmatrix} y(-\frac{3}{2}\delta) \\ y(-\frac{1}{2}\delta) \\ y(\frac{1}{2}\delta) \\ y(\frac{3}{2}\delta) \end{bmatrix} \tag{2.5}$$

Keys (1981) has also developed a method known as cubic convolution interpolation which reduces to Equation 2.5 for $\Delta = 0$.

2.2 Up-sampling

By dividing an interlaced frame into columns, interpolation can be thought of as a simple up-sampling process. The perfect interpolation filter then is an ideal low pass filter, assuming that there is no aliasing present in the vertical direction of the sampled image. This assumption is invalid since most cameras introduce vertical aliasing; however, for the purposes of this method, we assume that the aliasing introduced is negligible. The impulse response of this filter is then

$$h_{ideal}(n) = \frac{\sin(n\frac{\pi}{2})}{n\frac{\pi}{2}} \tag{2.6}$$

In order to reduce the impulse response of the filter to a finite length, two different windows, $w(n)$, were used, which resulted in a filter

$$h(n) = h_{ideal}(n)w(n) \tag{2.7}$$

The first of these windows was rectangular (Oppenheim and Schafer 1989):

$$w(n) = \begin{cases} 1 & \text{if } 0 \leq n \leq M \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

To improve the transition region, a Hanning window was also used (Oppenheim and Schafer 1989):

$$w(n) = \begin{cases} \frac{1}{2} - \frac{1}{2}\cos(2\pi\frac{n}{M}) & \text{if } 0 \leq n \leq M \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

2.3 Least Mean Squared Interpolator

This method, used in Horvat, Bird, and Goulding (1992), attempts to minimize the mean squared error of the estimate, $\hat{y}(\Delta)$, where the mean squared error is given by

$$\epsilon^2 = E[(y(\Delta) - \hat{y}(\Delta))^2] \tag{2.10}$$

Using (2.1) and taking the gradient of ϵ^2 with respect to $\mathbf{b}$ and setting it equal to zero results in the following optimal filter design

$$\mathbf{b} = (E[\mathbf{y}\mathbf{y}^T])^{-1} E[y(\Delta)\mathbf{y}] \tag{2.11}$$

For general k, where the filter length is $2k$,

$$E[y(\Delta)\mathbf{y}] = \mathbf{r}_y = \begin{bmatrix} \mathcal{R}(\Delta + (k-1)\delta + \frac{1}{2}\delta) \\ \mathcal{R}(\Delta + (k-2)\delta + \frac{1}{2}\delta) \\ \vdots \\ \mathcal{R}(\Delta + (k-k)\delta + \frac{1}{2}\delta) \\ \mathcal{R}(\Delta + (k-k)\delta - \frac{1}{2}\delta) \\ \vdots \\ \mathcal{R}(\Delta + (k-2)\delta - \frac{1}{2}\delta) \\ \mathcal{R}(\Delta + (k-1)\delta - \frac{1}{2}\delta) \end{bmatrix} \tag{2.12}$$

and

$$E[\mathbf{y}\mathbf{y}^T] = \mathbf{R}_y = \begin{bmatrix} \mathcal{R}(0) & \mathcal{R}(-\delta) & \dots & \mathcal{R}((-2k+1)\delta) \\ \mathcal{R}(\delta) & \mathcal{R}(0) & \dots & \mathcal{R}((-2k+2)\delta) \\ \vdots & \vdots & \vdots & \vdots \\ \mathcal{R}((2k-1)\delta) & \mathcal{R}((2k-2)\delta) & \dots & \mathcal{R}(0) \end{bmatrix} \tag{2.13}$$

where $\mathcal{R}(\tau)$ is the autocorrelation function of $y(t)$. Then

$$\mathbf{b} = \mathbf{R}_y^{-1}\mathbf{r}_y \tag{2.14}$$

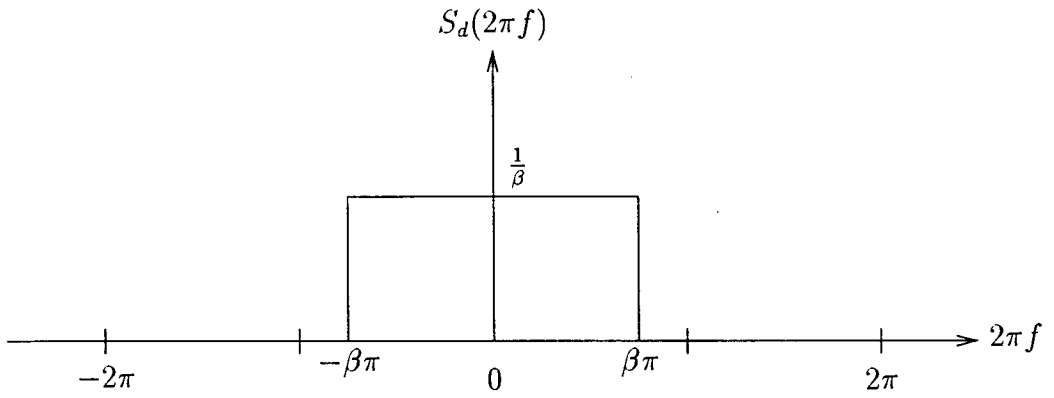


Figure 2.2: Bandlimited White Noise Model

To simplify the filter design, we assume that the columns of each frame are filtered white noise processes with flat spectral densities (Figure 2.2). Then the autocorrelation function is given by

$$\mathcal{R}(\tau) = \frac{\sin(\beta\pi\tau)}{\beta\pi\tau} \tag{2.15}$$

where β is the oversampling factor as shown in Figure 2.2. Using this model, the filter, $\mathbf{b}$ can easily be designed. Horvat, Bird, and Goulding (1992) provide an analysis of this interpolation method.

2.4 Line Shift Interpolation

Vertical spatial interpolation does not work well when non-vertical edges exist in the image. In this case it is better to use non-vertical interpolation such as in the method outlined in (Martinez and Lim 1989). This method is based on the line shift model, where the function

$$s(x, y) = s(x - v(y - y_0), y_0) \tag{2.16}$$

describes the relationship between adjacent lines of a frame. In this equation, $s(x, y)$ is the pixel intensity at location (x, y), $v(y)$ is the horizontal velocity as a function of the separation between lines, and (x_0, y_0) is the center of the small region in which this model holds.

The algorithm involves two steps:

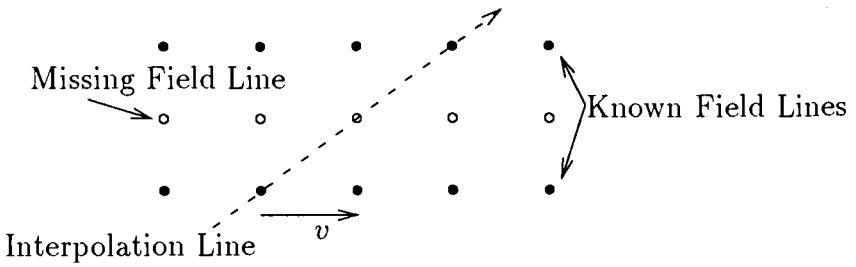


Figure 2.3: Line shift interpolation

1. The velocity, v, is estimated from the known pixel values surrounding the point of interest.

2. The velocity estimate is then used to determine the interpolation line along which an average is made of the points on the line above and below the point of interest.

This process is shown in Figure 2.3, where the velocity estimate is $+1$ pixel.

2.5 Experimental Results

In order to test the performance of the various methods of spatial interpolation described in this chapter, simulations were run on the *pongi* interlaced sequence and the reconstructed images were compared with the actual progressive sequence. The results are shown in Figure 2.4, which shows the average mean squared error for each of the methods over the first thirty fields of the *pongi* sequence. The up-sampling method with the rectangular window has been ignored, since its results are significantly worse than the other methods for comparable filter lengths. Note that the "zig-zag" characteristic of the plots results from the fact that there is more background high-frequency detail lost in the odd fields of the sequence than the even.

The best performing method on the *pongi* sequence in terms of the MSE measure is the linear interpolation technique. This is because there is a lot of high frequency detail which dramatically reduces the correlation between pixels as the distance between the pixels increases. Therefore, the shorter filter used in the linear interpolation technique is more effective.

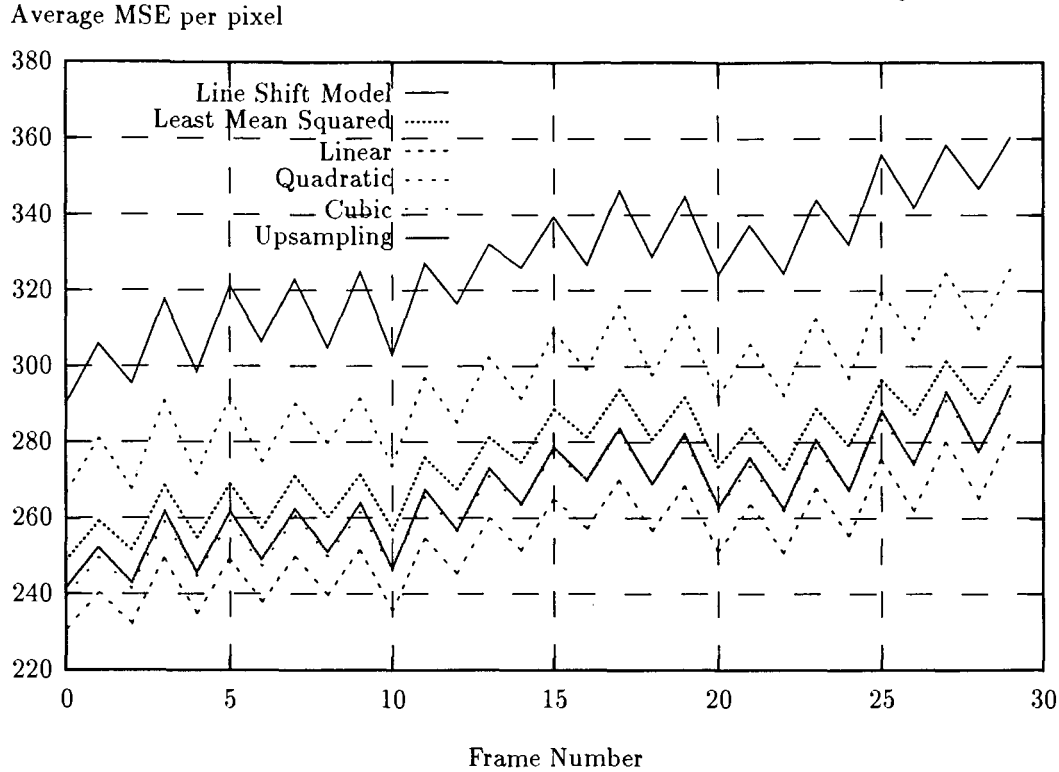


Figure 2.4: Performance of Spatial Interpolation Techniques on the *pongi* sequence

Results for the *missa* sequence are not shown since all of the methods perform well due to the low spatial detail this sequence. While the difference between methods is very small, the best performing technique varies depending on the frame: in odd frames, the linear interpolation technique outperforms the other methods, while in the even frames, the cubic interpolation technique is more effective. Clearly this results from the fact that the *missa* sequence contains less high frequency detail and therefore there is more correlation between pixels.

It is important to note that in the *pongi* sequence, the linear interpolation technique is only 1dB better than the worst (in terms of MSE) technique shown, the line shift method, while in the case of the *missa* sequence, the difference between best and worst is much smaller. Subjectively, the line shift method is sometimes better than the other methods, since in many cases it results in better recovery along object edges.

Overall, none of these methods are satisfactory, since they all introduce blurring and other annoying artifacts into the frames that are easily visible. For this reason, it is necessary to examine other techniques such as motion compensated interpolation

to obtain better results.

Chapter 3

Motion Compensation of Interlaced Sequences

While there exists a great deal of literature on motion compensation of progressive sequences, the problem of how best to motion compensate interlaced sequences has not been addressed in great detail. This chapter presents some of the simple methods that have been used in the past along with some new, better-performing methods. Throughout this chapter, block-based motion compensation is used to test the various algorithms described.

3.1 Interlaced Frame-Based Motion Compensation

One common method of performing motion compensation on interlaced sequences is to simply push each pair of successive fields together forming a progressive sequence. (This is also commonly done to provide the pause feature on a video tape recorder.) Therefore, if Ω_n contains the odd lines, the current frame is given by

$$\Psi_n(x,y) = \begin{cases} \Omega_n(x, \frac{y+1}{2}) & \text{for odd } y \\ \Omega_{n-1}(x, \frac{y}{2}) & \text{for even } y \end{cases} \tag{3.1}$$

This combining of fields results in a progressive sequence with half the number of frames as fields in the interlaced sequence. Normal motion compensation techniques

(such as block-based motion compensation, which was discussed in Chapter 1) can then be performed on the sequence.

This method of combining fields works well when there is no motion in the sequence, however, when motion is introduced, the edges of objects can become serrated causing poor performance as shown in Figure 3.1.

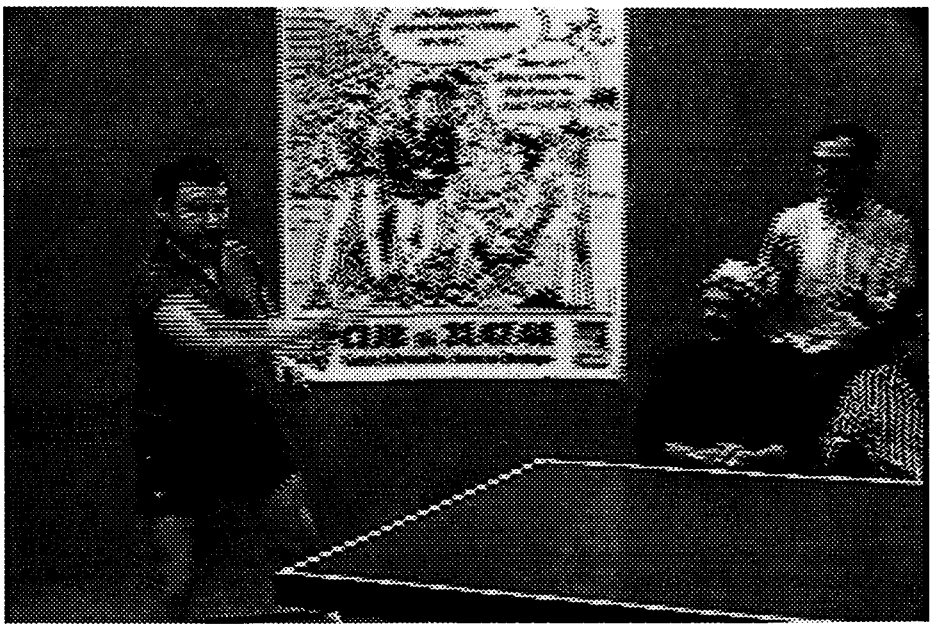


Figure 3.1: Frame formed from combination of the first two fields of the interlaced *pongi* sequence.

3.2 Single Field-Based Motion Compensation

Another simple method of performing motion compensation on interlaced sequences is to use block-matching techniques between the fields of of the sequence.

3.2.1 Different Parity Field

The first approach to single field motion compensation is to use the previous field, Ω_{n-1}, in the same way that progressive motion compensation is done. This introduces problems because the parity of Ω_{n-1} is different from that of the current field, Ω_n.

The major effect of this parity difference is that regions with no motion in Ω_n do not have good matches in Ω_{n-1}. For this reason, performance of the motion compensation process degrades considerably.

3.2.2 Different Parity Field–Interpolated

To reduce the degradation due to the poor matches found when Ω_{n-1} is used, the missing lines of Ω_{n-1} can be interpolated to form a field of the same parity as Ω_n which is known as $\Omega_{n-1}^{(r)}$. The performance of this method depends heavily on the interpolation process. For simple spatial interpolation techniques such as those discussed in Chapter 2, the performance of this method is better than without interpolation but is still poor.

3.2.3 Same Parity Field

Significant improvements in the performance of motion compensation of interlaced sequences can be achieved if the second previous frame, Ω_{n-2}, is used to find the matches. This improvement over Ω_{n-1} occurs because Ω_{n-2} contains the same parity lines as Ω_n and therefore matches for stationary parts of the sequence are good. Where this method fails is in the regions of the sequence containing motion. If this motion has a y-component that is even, then a good match will still be found; however, when the y-component is odd or large, good matches are generally not found and the performance degrades accordingly.

3.3 Multiple Field-Based Motion Compensation

Previous work on progressive motion compensation has shown that better performance can be achieved by searching multiple temporal frames for the best match (Gothe and Vaisey 1993). Since, in interlaced sequences there is different information missing from Ω_{n-1} and Ω_{n-2}, better performance can be expected if both frames are used in the motion compensation process. In addition, further improvement is likely if $\Omega_{n-1}^{(r)}$ is used instead of Ω_{n-1}.

Searching both frames for the best block match introduces an additional component into the motion vector that informs the decoder which of the previous frames to

use; however, this additional component is usually outweighed by the improvement in the matches.

3.4 Hybrid Motion Compensation

Since in many cases the interpolated field, $\Omega_{n-1}^{(r)}$, produces errors due to the imperfect interpolation, it is advantageous to combine the information contained in the two fields, Ω_{n-2} and $\Omega_{n-1}^{(r)}$. One method is to form linear combinations of these two fields such that combination frames given by

$$\Omega_c(\zeta) = (1-\zeta)\Omega_{n-1}^{(r)} + \zeta\Omega_{n-2} \tag{3.2}$$

where $\zeta \in [0,1]$. Two methods can be used to find the best match:

1. The domain of ζ can be discretized and limited in size allowing computationally realistic searches to be made for the best match.

2. A prediction algorithm can be used to obtain a best estimate of ζ which gives the best match.

3.5 Results

To test the methods described in this chapter, simulations were run using block-based motion compensation with an exhaustive search technique. The results presented in this section are indicative of the performance relationship between the various methods.

3.5.1 Previous Frame Performance

Figure 3.2 shows the average MSE per pixel for the two cases where only Ω_{n-1} is used. Clearly, using $\Omega_{n-1}^{(r)}$ (linear interpolation method) achieves better performance by approximately ten percent on average. For this reason, all other experiments involving the previous field used $\Omega_{n-1}^{(r)}$ rather than Ω_{n-1}.

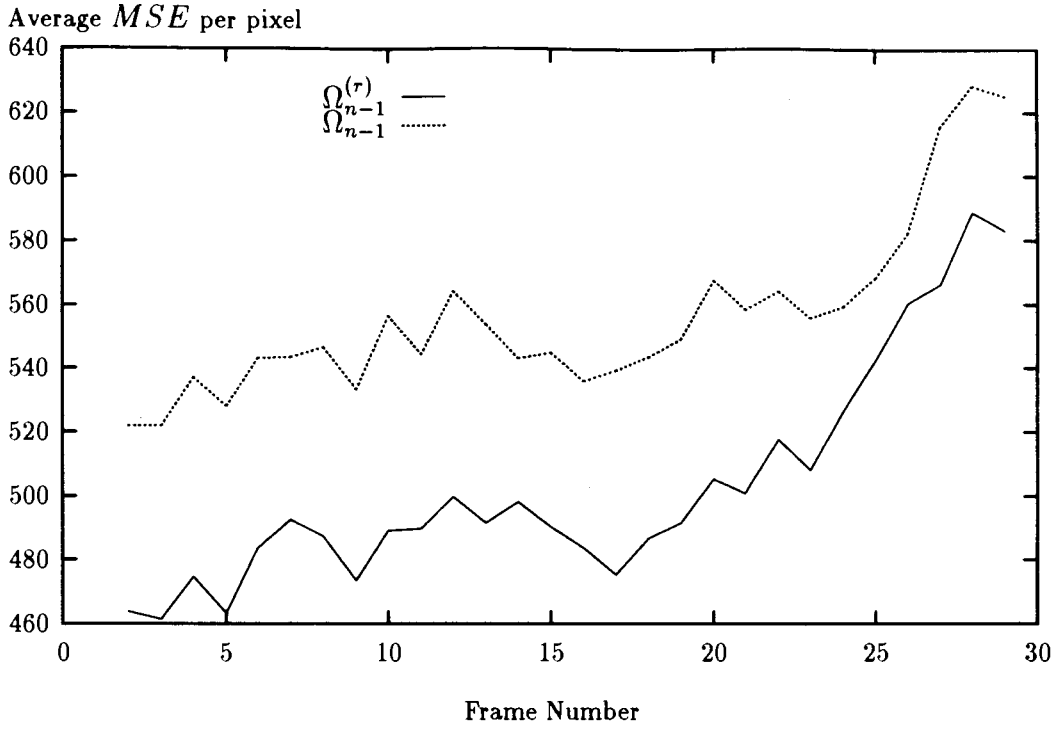


Figure 3.2: Performance of Motion Compensation Algorithm using only Ω_{n-1} on the *pongi* sequence

3.5.2 Comparision of Motion Compensation Methods

Figure 3.3 shows how the methods described in this chapter compare with each other. The method using only the previous frame (either $\Omega_{n-1}^{(r)}$ or Ω_{n-1}) is not shown, since it is significantly worse than those shown. Using only Ω_{n-2} is the worst of the four methods followed by the frame-based method (described in section 3.1). Using both $\Omega_{n-1}^{(r)}$ and Ω_{n-2} gives a 15-20% improvement on average over the frame-based method. The hybrid method with seven discrete values of ζ (every combination is exhaustively searched in the search area) gives a further 7-10% improvement.

Also shown in Figure 3.3 are the performance curves for multiple field MC and hybrid MC using the actual field, $\Omega_{n-1}^{(a)}$, rather than an interpolated one. These are the lower bounds for the performance of these methods. Both methods using $\Omega_{n-1}^{(a)}$ are better than the other methods by a significant margin indicating that better interpolation methods will lead to better performance of the motion compensation algorithms.

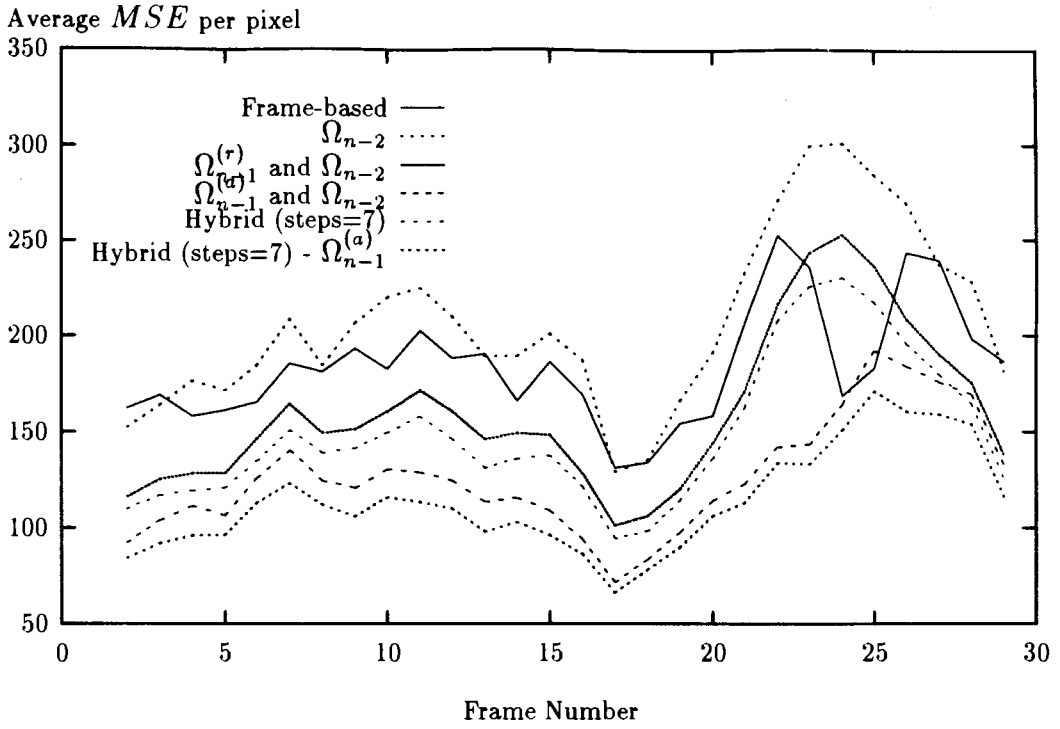


Figure 3.3: Performance of Various Motion Compensation Algorithms on the *pongi* sequence

3.5.3 Effect of Interpolation on MC Methods

In order to discover the effect of various interpolation methods on the MC methods, simulations were run using the spatial interpolation techniques described in Chapter 2. Both the multiple field MC and hybrid MC were tested.

Figure 3.4 shows the average MSE per pixel for the multiple field MC algorithm for a number of different spatial interpolation methods along with the curve when the actual missing lines, $\Omega^{(a)}_{n-1}$, are used (lower bound of method). It is clear from this figure that no method is significantly better than any of the others although in general using the linear interpolator does give the best results. There is, however, a significant difference between the spatial interpolators and the actual field indicating the potential for improvement with better interpolation methods.

In Figure 3.5, the average MSE is shown for the different spatial interpolation methods for the hybrid MC algorithm using 7 discrete values for ζ. There is little difference between the interpolation methods with the linear interpolator being the best by a small margin. As with the previous figure, there is significant potential

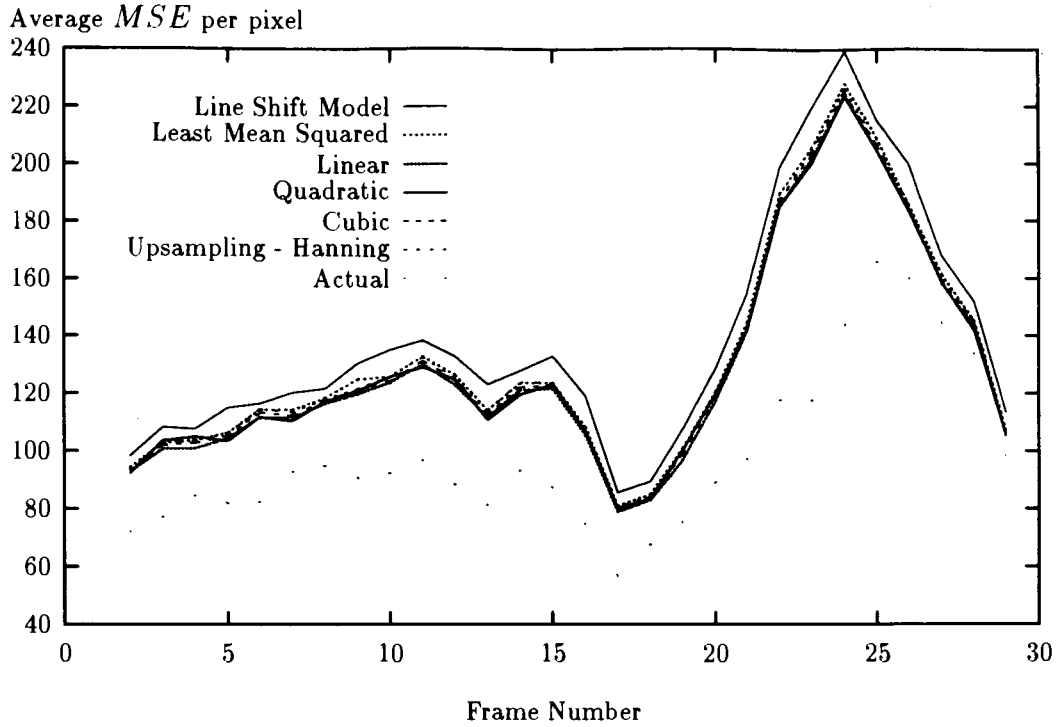


Figure 3.4: Performance of Two Field Interlaced Motion Compensation using Spatial Interpolation Techniques on the *pongi* sequence

improvement by using better interpolation methods.

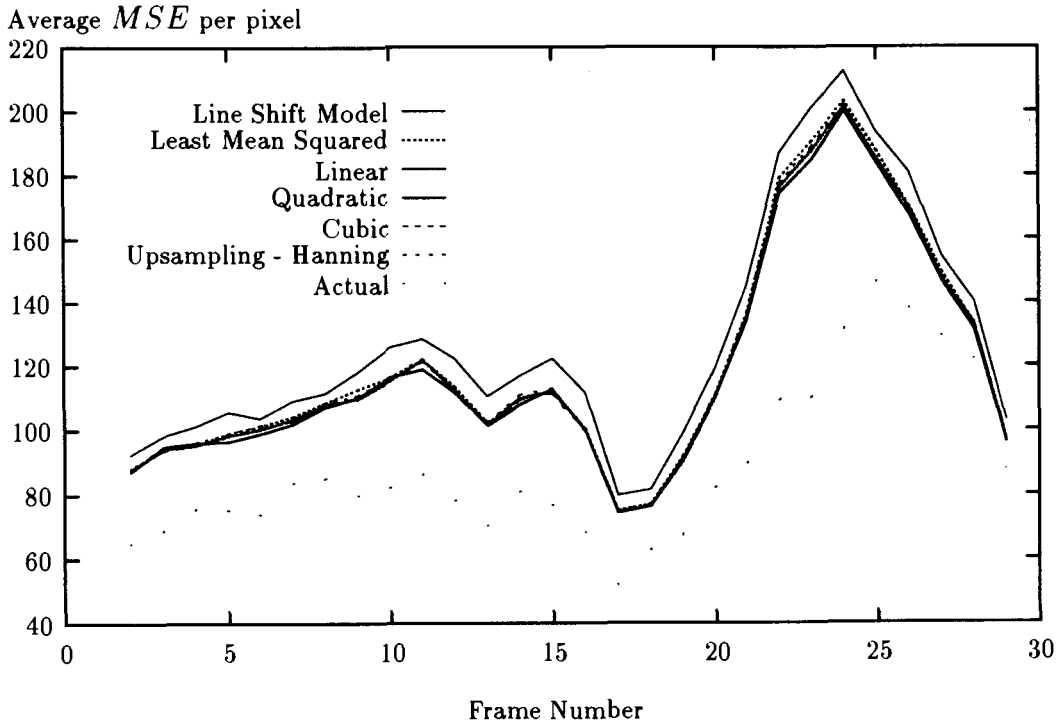


Figure 3.5: Performance of Hybrid Interlaced Motion Compensation using Spatial Interpolation Techniques on the *pongi* sequence (seven steps)

Chapter 4

Bayesian Formulation of Motion Estimation

An excellent approximation of the motion in the video sequence is required in order for hybrid motion compensation to work well. Motion approximations can be obtained in many different ways including via block-based algorithms. The technique that seems to provide one of the best motion estimations is called "Bayesian Formulation of Motion Estimation" and can be found in (Konrad 1989). This chapter provides a brief outline of the development of the method along with some results. The notation in this section is consistent with that of Konrad.

4.1 Gibbs Distribution and Markov Random Fields

In this technique of motion estimation, the motion fields are assumed to be Markov random fields described by a Gibbs distribution. This section describes the basics of Gibbs distributions and Markov random fields. A more complete description of this material can be found in (Geman and Geman 1984) and (Konrad 1989).

4.1.1 Gibbs Distribution

The Gibbs distribution requires the definition of a number of concepts including a *neighborhood system*, *clique*, and *potential function*.

Neighborhood Systems

A neighborhood is a group of sites (locations in a field) around a central site. The group of sites is called the *neighborhood* of the central site. Note that the central site is not contained in its own neighborhood. In general, neighborhoods are used to introduce some dependence between the central site and its neighbors. For example, a desirable quality of motion fields is that they be smooth. This quality can be introduced by favouring neighborhoods in which the sites are all similar in value to the central site. The following definition formalizes the term neighborhood.

If we assume that a lattice, Λ, is a collection of M sites designated s_i, where $i = 1 \ldots M$, and that $\mathcal{X}$ is a sample field from random field X defined over Λ and chosen from a state-space $\mathcal{S}$, then $\mathcal{N}^k$ is a neighborhood system if and only if

$$\mathcal{N}^k = \{\eta^k(s_i) : s_i \in \Lambda, \eta^k(s_i) \subset \Lambda, 1 \leq i \leq M\},$$

where the k^{th} order "neighborhood", $\eta^k(s_i)$, of site $s_i \in \Lambda$ satisfies

1. $s_i \notin \eta^k(s_i)$
2. if $s_j \in \eta^k(s_i)$, then $s_i \in \eta^k(s_j)$ for any $s_i \in \Lambda$

The k^{th} order neighborhood is formed by taking the $(k-1)^{th}$ order neighborhood and adding the closest set of sites which are not part of the $(k-1)^{th}$ order neighborhood. For example, the second order neighborhood is constructed by adding the diagonal sites to the first order neighborhood.

In general, the notation $\eta^k(s_i)$ will be shortened to $\eta(s_i)$ when it is not important which neighborhood system we are using. For the purposes of this motion estimation technique, only spatial (two-dimensional) neighborhood systems will be used and therefore a site may also be referred to by its coordinates, (m, n). The notation $\eta(m, n)$ will then be used to refer to the neighborhood of site (m, n). Figure 4.1 shows the neighborhoods of order 1 through 3. Note that every higher order neighborhood includes all of the sites in neighborhoods of lower order.

For example, a second order neighborhood system is the set of all second order neighborhoods. A second order neighborhood of site at (m, n) contains the sites at $(m \pm 1, n)$, $(m, n \pm 1)$ and $(m \pm 1, n \pm 1)$. Site (m, n) is **not** contained in its own neighborhood (condition 1 above). Also note that since site $(m + 2, n)$ is not in the

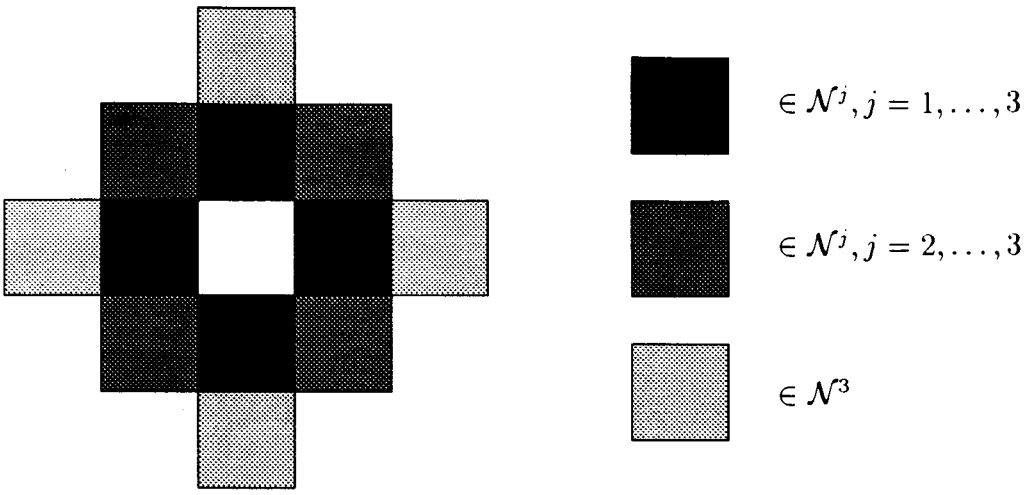


Figure 4.1: Neighborhoods of order 1 through 3. Note that system of order k includes the sites contained in order $k-1$

second order neighborhood of site (m,n), then site (m,n) is also not in the second order neighborhood of site $(m+2,n)$ (condition 2 above).

While the details of their use will become clear later in this chapter, at this point it is sufficient to understand that neighborhoods introduce a dependence between adjacent sites (in each other's neighborhood) causing a "smooth" motion field to be favoured.

In this thesis, only the first, $\mathcal{N}^1$, and second, $\mathcal{N}^2$, order systems were used, since higher order systems increase the computational complexity significantly with limited performance improvement (Konrad 1989). The neighborhood systems must also be redefined at the image boundaries to allow inclusion of the edge sites.

Cliques

Included in a neighborhood system is a set of cliques. A clique, c, is defined with respect to the neighborhood system such that either

1. two sites in a *clique*, c, are neighbors i.e., $(m,n),(p,q) \in c$ and $(m,n) \neq (p,q) \Rightarrow (m,n) \in \eta(p,q)$

2. or, c consists of a single site.

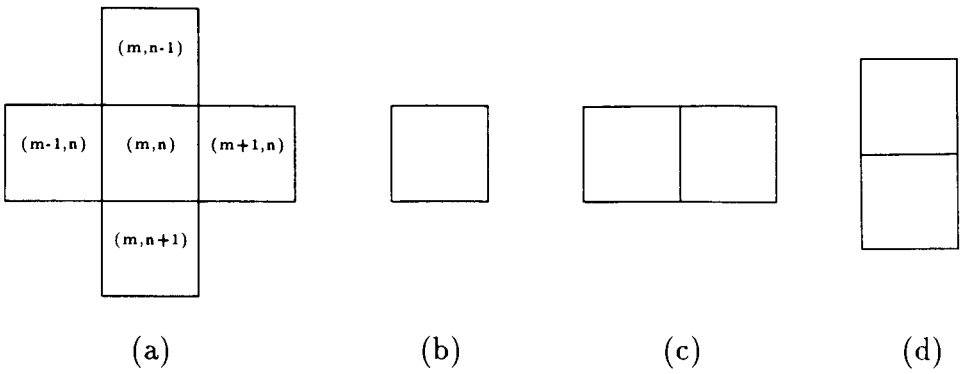


Figure 4.2: The First Order Neighborhood system (a) along with associated cliques: (b) one-element and (c),(d) two-element

The set of cliques is represented by $\mathcal{C}$. Figures 4.2 and 4.3 show the cliques for the first and second order neighborhood systems respectively. Each higher order set of cliques contains all the cliques from the lower order systems.

Definition of a Gibbs Distribution

With the above terminology, a *Gibbs distribution* with respect to the lattice, Λ, and the neighborhood system, $\mathcal{N}$, can be defined as a probability measure on the state space, $\mathcal{S}$, such that

$$\pi(\mathcal{X}) = \frac{1}{Z} e^{U(\mathcal{X})/\beta} \tag{4.1}$$

where β and Z are constants, and U is the *energy function.* U is of the form

$$U(\mathcal{X}) = \sum_{c \in \mathcal{C}} V(\mathcal{X}, c) \tag{4.2}$$

where $V(\mathcal{X}, c)$ is the *potential function* defined over the whole field for clique, c, and is only dependent on those samples of $\mathcal{X}$ that are in clique c. Z is called the *partition function*, which normalizes the expression to make $\pi(\mathcal{X})$ a probability measure.

4.1.2 Markov Random Fields

In this motion estimation method, the motion vector field is assumed to be a Markov random field. A Markov random field (MRF) has the following properties: [1]

[1]The notation X_i and $\mathcal{X}_i$ refers to the i^{th} site in the field

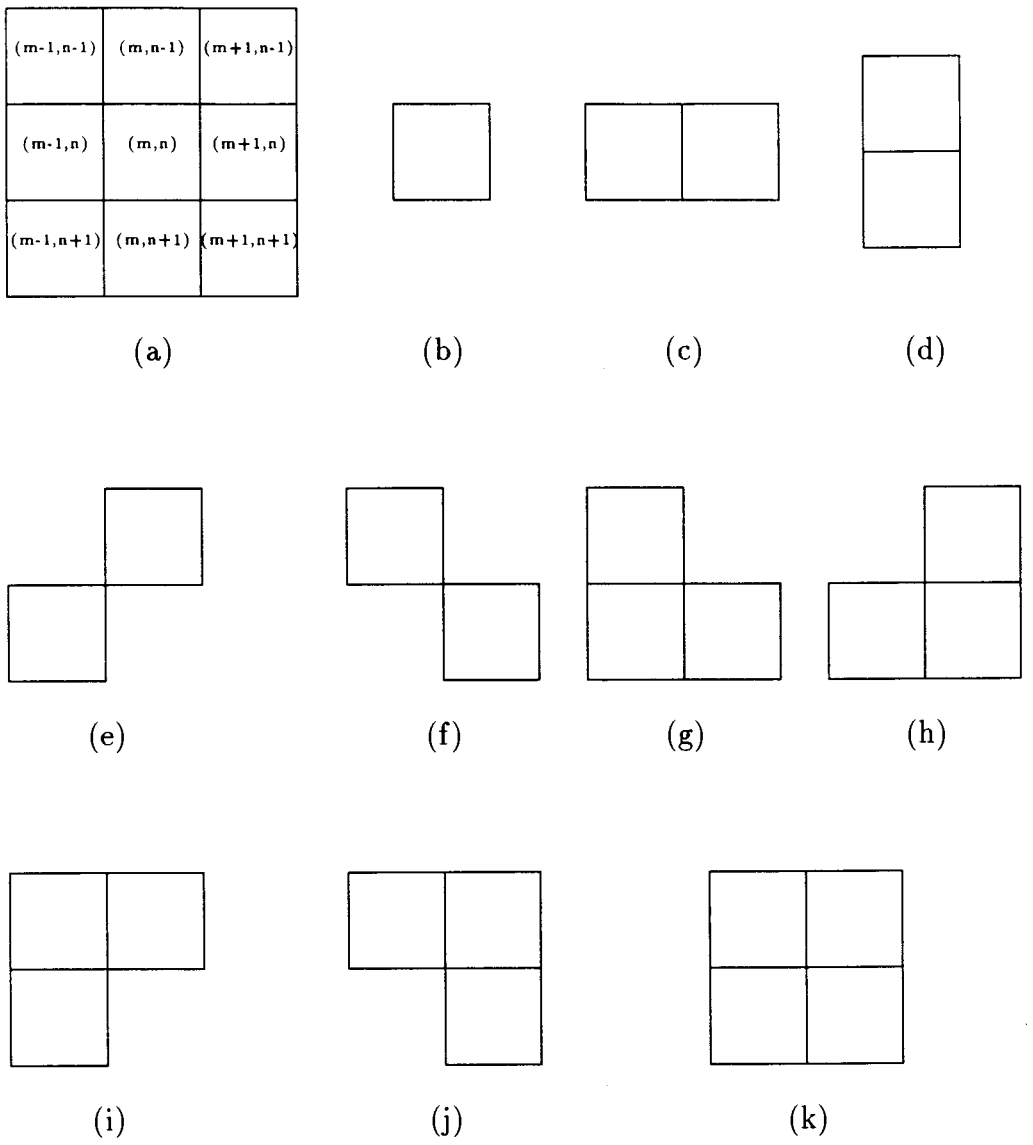


Figure 4.3: The Second Order Neighborhood system (a) along with associated cliques: (b) one-element, (c),(d),(e),(f) two-element, (g),(h),(i),(j) three-element, (k) four-element

1. $P(X = \mathcal{X}) > 0, \ \forall \mathcal{X} \in \mathcal{S}$

2. $P(X_i = \mathcal{X}_i \mid X_j = \mathcal{X}_j, \forall j \neq i) = P(X_i = \mathcal{X}_i \mid X_j = \mathcal{X}_j, \forall j \in \eta(i)), \ \forall i, \mathcal{X} \in \mathcal{S}$

where P is a probability measure. In other words, the probability of site i having a certain state (value), is only dependent on the sites in the neighborhood of site i and the probability of each state is greater than zero.

In order to characterize the MRF, the Hammersley-Clifford theorem (Besag 1972; Konrad 1989) can be applied. This theorem states that if X is a MRF on lattice Λ with respect to neighborhood system $\mathcal{N}$, then the probability distribution of its sample realizations (possible configurations) is a Gibbs distribution as given in Equation 4.1. The characterization of a MRF is then dependent on the potential functions, V.

4.2 Maximum A Posteriori Probability (MAP) Estimation

The purpose of this method is to estimate the motion field for a video frame given the current frame, g_{t_+}, and the previous frame, g_{t_-}, which are considered to be samples of the random fields G_{t_+} and G_{t_-} respectively (see Figure 4.4). This is done by attempting to find the "most likely" displacement field, $\hat{\mathbf{d}}_t^\star \in \mathcal{S}_\mathbf{d}$, given the observations g_{t_+} and g_{t_-}. Note that $\hat{\mathbf{d}}_t^\star$ is the best estimate of the actual displacement field, $\mathbf{d}_t$ (one vector of $\mathbf{d}_t$ is shown in Figure 4.4), while $\hat{\mathbf{d}}_t$ represents any given displacement field (sample of the random displacement vector field process, $\mathbf{D}_t$), and $\mathcal{S}_\mathbf{d}$ is the set of all possible displacement vectors, i.e., the displacement vector state space.[2] Also, note that a displacment vector field between two frames can be represented by two-dimensional vectors located at each site on the sampling lattice in the video sequence. Displacement fields are, in general, different from block-based motion estimation where there is a displacement vector for each block rather than each point on the sampling lattice. The MAP requirement then, is

$$P(\mathbf{D}_t = \hat{\mathbf{d}}_t^\star \mid G_{t_-} = g_{t_-}, G_{t_+} = g_{t_+}) \geq P(\mathbf{D}_t = \hat{\mathbf{d}}_t \mid G_{t_-} = g_{t_-}, G_{t_+} = g_{t_+}) \ \forall \hat{\mathbf{d}}_t \in \mathcal{S}_\mathbf{d} \tag{4.3}$$

[2]The subscript t on the displacement vector notation indicates that the motion vector is estimated for some time instance, t

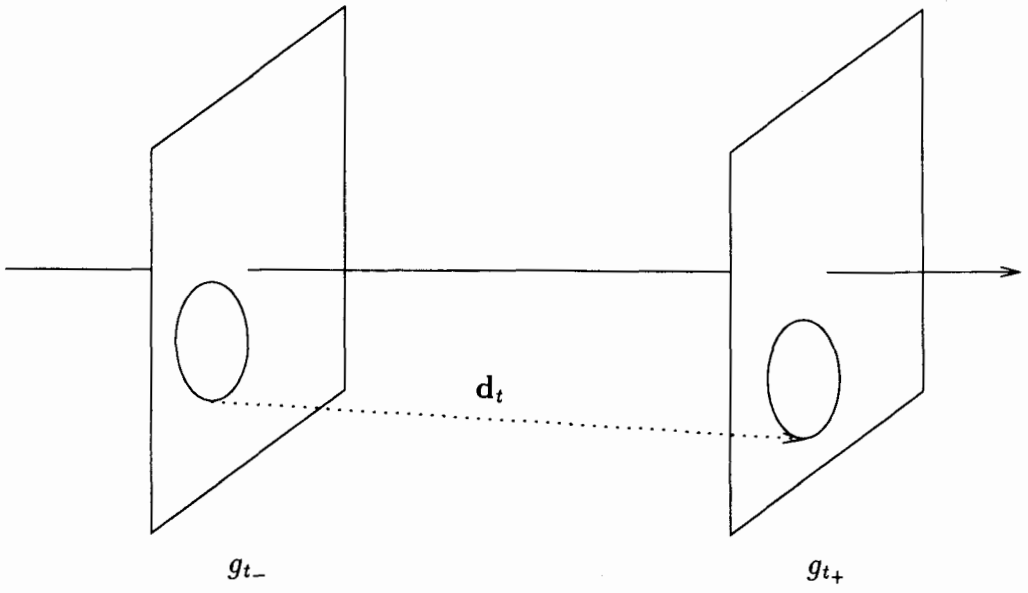


Figure 4.4: The Motion Estimation Process

By applying Bayes rule, the posterior distribution can be obtained as

$$P(\mathbf{D}_t = \hat{\mathbf{d}}_t \mid G_{t-} = g_{t-}, G_{t+} = g_{t+}) = \frac{P(G_{t+} = g_{t+} \mid \mathbf{D}_t = \hat{\mathbf{d}}_t, G_{t-} = g_{t-}) \cdot P(\mathbf{D}_t = \hat{\mathbf{d}}_t \mid G_{t-} = g_{t-})}{P(G_{t+} = g_{t+} \mid G_{t-} = g_{t-})} \tag{4.4}$$

Since the denominator in Equation 4.4 is independent of the displacement vector process, $\mathbf{D}_t$, it can be ignored and the resulting MAP estimate is the solution to

$$\max_{\hat{\mathbf{d}}_t}[P(G_{t+} = g_{t+} \mid \mathbf{D}_t = \hat{\mathbf{d}}_t, G_{t-} = g_{t-}) \cdot P(\mathbf{D}_t = \hat{\mathbf{d}}_t \mid G_{t-} = g_{t-})] \tag{4.5}$$

In order to evaluate Equation 4.5, the explicit forms of the conditional probability distributions must be known. These are outlined in the next section.

4.3 Models

There are three models that must be considered to obtain explicit forms of the conditional probability distributions in Equation 4.5. These are called the structural model, the observation model and the displacement field model (Konrad 1989).

4.3.1 Structural Model

For any motion estimation algorithm, it is necessary to assume some relationship between the motion vectors and the pixel intensity values. The normal assumption that is made is that there are no illumination effects present or, equivalently, that the pixel intensity is constant along the path of any motion vector. Therefore, the following expression holds

$$u(\mathbf{x} - \mathbf{d}(\mathbf{x}, t), t_-) = u(\mathbf{x}, t_+) \tag{4.6}$$

where $u(\mathbf{x}, t)$ is the true underlying continuous image, $\mathbf{x}$ is some spatial location in the current frame, t_- is the temporal location of the previous frame and t_+ is the temporal location of the current frame.

4.3.2 Observation Model

This model takes into account all the distortion that is added between the true underlying image, u, and the observed image, g. Sources of noise during this process include image sensor noise, quantization noise and distortion due to temporal and spatial aliasing (Konrad 1989). For the purpose of this method, let the following relationship hold:

$$g(\mathbf{x}, t) = u(\mathbf{x}, t) + \mathrm{n}(\mathbf{x}, t) \tag{4.7}$$

where $\mathrm{n}(\mathbf{x}, t)$ is a Gaussian random variable. This relationship has been shown to be reasonably valid by (Konrad 1989). Then, using the structural model (Equation 4.6), the following relationship can be derived:

$$g(\mathbf{x}, t_+) - g(\mathbf{x} - \mathbf{d}(\mathbf{x}, t), t_-) = \mathrm{n}(\mathbf{x}, t_+) - \mathrm{n}(\mathbf{x} - \mathbf{d}(\mathbf{x}, t), t_-) \tag{4.8}$$

where the left side of the equation is called the *displaced pel difference* (DPD) and is denoted by $r(\mathbf{d}(\mathbf{x}, t), \mathbf{x}, t)$. The right side of the equation is called the *displaced noise difference*. Since the right side of Equation 4.8 is simply a sum of Gaussian random variables, it is itself Gaussian and therefore the conditional probability (from Equation 4.5) can be expressed in the form [3]

$$P(G_{t_+} = g_{t_+} \mid \mathbf{D}_t = \hat{\mathbf{d}}_t, G_{t_-} = g_{t_-}) = (2\pi\sigma^2)^{-M_\mathbf{d}} \cdot e^{-U_g(g_{t_+} | \hat{\mathbf{d}}_t, g_{t_-})/2\sigma^2} \tag{4.9}$$

[3]Note that the displaced noise difference is assumed to be independent for each site.

where σ^2 is the variance of the displaced noise difference, $M_{\mathbf{d}}$ is the number of vectors in a single displacement vector field, and the energy, U_g, is defined as

$$U_g(g_{t_+} \mid \hat{\mathbf{d}}_t, g_{t_-}) = \sum_{i=1}^{M_{\mathbf{d}}} [g(\mathbf{x}_i, t_+) - g(\mathbf{x}_i - \hat{\mathbf{d}}(\mathbf{x}_i, t), t_-)] \tag{4.10}$$

where $g(\mathbf{x}, t)$ is the pixel intensity value at location $\mathbf{x}$ at time t. This intensity value may be simply the sample value or, in the case where the value is not observed directly (eg. missing lines of an interlaced field), it may be an interpolated value.

4.3.3 Displacement Field Model

In this method, the proposed model for the displacement field is a vector Markov random field (VMRF). This differs from the scalar Markov random field (MRF) described in section 4.1.2 only by the definition of a state. In the case of a scalar MRF, each site is from $\mathcal{R}$, while for a two-dimensional VMRF, such as the displacement field, each site is taken from $\mathcal{R}^2$. Due to the Hammersley-Clifford theorem, the Gibbs distribution (described in section 4.1.1) characterizes a random field, $\mathbf{D}_t$. Due to computational complexity, this method will be implemented using only the first and second order neighborhood systems. In general, any neighborhood system could be used but the improvement in performance diminishes as larger neighborhood systems are used.

Within the neighborhood systems, there are a number of different cliques. In both the first and second order neighborhood systems used in this thesis, only the two element cliques will be used. These cliques are shown in Figure 4.2(c) and (d) for the first order system and in Figure 4.3(c), (d), (e) and (f) for the second order system. While there are many different choices available for the potential function, the one used in this method is

$$V_{\mathbf{d}}(\mathbf{d}_t, c_{\mathbf{d}}) = V(\mathbf{d}(\mathbf{x}_i, t), \mathbf{d}(\mathbf{x}_j, t)) = \| \mathbf{d}(\mathbf{x}_i, t) - \mathbf{d}(\mathbf{x}_j, t) \|^2, \qquad c_{\mathbf{d}} = \{\mathbf{x}_i, \mathbf{x}_j\} \in \mathcal{C}_{\mathbf{d}} \tag{4.11}$$

where $\| \cdot \|$ is a norm in $\mathcal{R}^2$ defined as

$$\| \mathbf{d}(\mathbf{x}_i, t) - \mathbf{d}(\mathbf{x}_j, t) \| = \sqrt{[d_x(\mathbf{x}_i, t) - d_x(\mathbf{x}_j, t)]^2 + [d_y(\mathbf{x}_i, t) - d_y(\mathbf{x}_j, t)]^2} \tag{4.12}$$

The symbols d_x and d_y represent the horizontal and vertical components of the displacement vector respectively. The potential function in Equation 4.11 was chosen

since it favours smooth displacement fields by increasing when the difference between neighboring displacement vectors increases. This intuitively matches what would be expected for "natural"motion fields.

In order to simplify this initial model, it is assumed that the knowledge of a single frame does not affect the computation of the motion field, $\mathbf{d}(\mathbf{x}, t)$; i.e., discontinuities in the pixel amplitudes (object edges) in a frame are not used directly to compute the motion field. This means that the motion field random process, $\mathbf{D}_t$, and the frame random process, $G_{t\_}$, are independent and that the probability, $P(\mathbf{D}_t = \mathbf{d}_t \mid G_{t\_} = g_{t\_})$, can be written as [4]

$$P(\mathbf{D}_t = \mathbf{d}_t \mid G_{t\_} = g_{t\_}) = \pi(\mathbf{d}_t) = \frac{1}{Z_d} e^{-U_\mathbf{d}} = \frac{1}{Z_d} e^{-\sum_{c_\mathbf{d} \in C_\mathbf{d}} V_\mathbf{d}(\mathbf{d}_t, c_\mathbf{d})/\beta_\mathbf{d}} \tag{4.13}$$

where the set of cliques, $\mathcal{C}_\mathbf{d}$, varies depending on the neighborhood chosen (either $\mathcal{N}_\mathbf{d}^1$ or $\mathcal{N}_\mathbf{d}^2$).

4.4 A Posteriori Probability

To form the a posteriori probability, Equations 4.9 and 4.13 can be inserted into 4.4, which results in

$$\pi(\hat{\mathbf{d}}_t \mid G_{t\_} = g_{t\_}, G_{t+} = g_{t+}) = P(\mathbf{D}_t = \hat{\mathbf{d}}_t \mid G_{t\_} = g_{t\_}, G_{t+} = g_{t+}) = \frac{1}{Z} e^{-U(\hat{\mathbf{d}}_t, g_{t\_}, g_{t+})} \tag{4.14}$$

where Z is a normalizing constant composed of the three terms: $P(G_{t+} \mid G_{t\_})$ from Equation 4.4, $(2\pi\sigma^2)^{-M_\mathbf{d}/2}$ from Equation 4.9 and Z_d from Equation 4.13. The overall energy function is

$$U(\hat{\mathbf{d}}_t, g_{t\_}, g_{t+}) = \lambda_g \cdot \sum_{i=1}^{M_\mathbf{d}} [r(\hat{\mathbf{d}}(\mathbf{x}_i, t), \mathbf{x}_i, t)]^2 + \lambda_\mathbf{d} \cdot U_\mathbf{d}(\hat{\mathbf{d}}(\mathbf{x}_i, t)) \tag{4.15}$$

where $\lambda_g = 1/2\sigma^2$ and $\lambda_\mathbf{d} = 1/\beta_\mathbf{d}$. These two parameters control the relationship between the observation model and the displacement field model. For example, maximizing Equation 4.5, the MAP estimate, is equivalent to minimizing the "energy function" defined by Equation 4.15. This energy is a tradeoff between the displaced pel difference, r, and the smoothness of the motion field (represented by $U_\mathbf{d}$). The

[4]The parameter, $\beta_\mathbf{d}$, is the constant from the Gibbs distribution (Equation 4.1)

parameters λ_g and λ_d are very difficult to compute theoretically, and therefore are normally chosen by experience (Konrad 1989).

4.5 Stochastic Solution to Motion Estimation

In order to find a good motion estimate, some method of finding the motion field that maximizes the a posteriori probability must be used. This estimate is known as the maximum a posteriori (MAP) estimate. The method chosen by Konrad (1989) and used in this thesis is stochastic in nature and is known as *simulated annealing.* Deterministic methods are also possible but they tend to get trapped in local minima that decrease the quality of the estimate.

4.5.1 Gibbs Sampler

In order for simulated annealing to be computationally realistic in terms of a frame, it is necessary to use a method that generates samples from the a posteriori conditional distribution in Equation 4.14. The method used in this thesis is called the *Gibbs sampler* and was developed by Geman and Geman (1984).

Let $\mathbf{\Gamma}(\tau)$ be a time-indexed random field whose states are distributed according to the probability measure π after a sufficiently long period of time. The Gibbs sampler is a method of generating these states, where the step $\mathbf{\Gamma}(\tau-1) \rightarrow \mathbf{\Gamma}(\tau)$ is described by Equation 4.16. In order to generate these states, the Gibbs sampler makes a change at only one site at time τ. The site that is changed or at which a replacement is made is denoted by the index, n_τ, $\Gamma_i(\tau)$ refers to location i in the random field, $\mathbf{\Gamma}(\tau)$, and

$$P(\Gamma_{n_\tau}(\tau)=\gamma_{n_\tau}) = \pi(\Gamma_{n_\tau}(\tau)=\gamma_{n_\tau} \mid \Gamma_j(\tau-1)=\gamma_j, \forall j \neq n_\tau) \cdot P(\Gamma_j(\tau-1)=\gamma_j, \forall j \neq n_\tau), \\ (n_\tau, j : (\mathbf{x}_{n_\tau}, t), (\mathbf{x}_j, t) \in \Lambda) \tag{4.16}$$

From Equation 4.16, it is clear that only one site changes at each time step. This means that $\mathbf{\Gamma}(\tau-1)$ and $\mathbf{\Gamma}(\tau)$ can only differ at site n_τ. At each step a new state is chosen by drawing a sample from the local conditional characteristics of distribution π, but because π is a Gibbs distribution, this probability is dependent only on the states of the sites in the neighborhood of site.[5] This means that a state, $\gamma_{n_\tau} \in \mathcal{S}_d$, is

[5]In this application (motion estimation), the current state of a site refers to the current value of

chosen from the conditional distribution of $\Gamma_{n_\tau}(\tau)$ given the states of the neighboring sites, $\Gamma_i(\tau - 1) \quad \forall i : \mathbf{x}_i \in \eta_{\mathbf{d}}(\mathbf{x}_{n_\tau})$ (see below for details of this selection process). This conditional distribution is (Konrad 1989)

$$P(\mathbf{D}(\mathbf{x}_{n_\tau}, t) = \hat{\mathbf{d}}(\mathbf{x}_{n_\tau}, t) \mid \mathbf{D}(\mathbf{x}_j, t) = \hat{\mathbf{d}}(\mathbf{x}_j, t), j \neq n_\tau, G_{t_-} = g_{t_-}, G_{t_+} = g_{t_+}) = \frac{\exp(-U_{\mathbf{d}}^{n_\tau}(\hat{\mathbf{d}}(\mathbf{x}_{n_\tau}, t) | \hat{\mathbf{d}}_t, g_{t_-}, g_{t_+}))}{\sum_{z \in \mathcal{S}_{\mathbf{d}}} \exp(-U_{\mathbf{d}}^{n_\tau}(z | \hat{\mathbf{d}}_t, g_{t_-}, g_{t_+}))} \tag{4.17}$$

where the local energy function, $U_{\mathbf{d}}^{n_\tau}$, is defined as

$$U_{\mathbf{d}}^{n_\tau}(z \mid \hat{\mathbf{d}}_t, g_{t_-}, g_{t_+}) = \lambda_g \cdot [r(z, \mathbf{x}_{n_\tau}, t)]^2 + \lambda_{\mathbf{d}} \cdot \sum_{j:\mathbf{x}_j \in \eta_{\mathbf{d}}(\mathbf{x}_{n_\tau})} V(z, \hat{\mathbf{d}}(\mathbf{x}_j, t)) \tag{4.18}$$

Using the selection process described below, the Gibbs sampler process can be summarized in Figure 4.5.

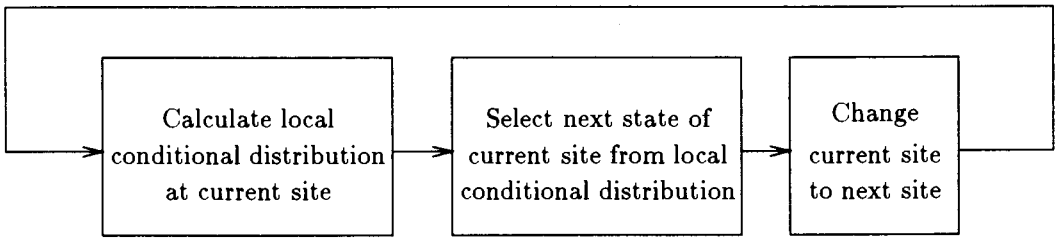


Figure 4.5: Block diagram of Gibbs Sampler algorithm

Selection from the Conditional Distribution

The selection from the conditional distribution at each site is implemented as follows:

1. For each element of the displacement vector state-space, $\mathcal{S}_{\mathbf{d}}$, the value of the energy (Equation 4.18) is calculated.

2. The minimum energy over $\mathcal{S}_{\mathbf{d}}$ is found and all energies are normalized such that when the exp(·) function is applied, the minimum energies will fall within the floating point range of the simulation computer. Any energies that are too large

the displacement vector at that site. Therefore, when a new state is chosten from the conditional distribution, a (possibly) new value of the displacement vector is given to the current site (at n_τ).

after this normalization are insignificant and the corresponding probability is set to zero. This normalization process is based on the relation

$$\frac{\exp(a_k)}{\sum_{i=1}^{N}\exp(a_i)} = \frac{\exp(a_k+b)}{\sum_{i=1}^{N}\exp(a_i+b)} = \frac{b}{b}\frac{\exp(a_k)}{\sum_{i=1}^{N}\exp(a_i)}$$

3. The probability values for each site are then calculated (Equation 4.17).

4. By dividing the interval, $[0,1)$, according to the probability values, a uniform pseudo-random number generator is used to select the next state.

4.5.2 Simulated Annealing

To find the best motion field, it is necessary to compute a MAP estimate. This is difficult since, for a typical $M \times N$ frame, there are MN sites and for each site there is some limited state space of size P. This means that there are MNP possible states for the displacement field, $\mathbf{D}_t$. For a typical frame (360×240) with integer displacement field components (maximum length 8), this results in approximately 2.5×10^7 possible states. Therefore, some method other than direct computation is necessary.

The algorithm used in this thesis to find the MAP estimate is called *simulated annealing* and was first proposed in (Kirkpatrick, Gelatt, and Vecchi 1983) and then later used by both Geman and Geman (1984) and Konrad (1989). The algorithm is based on the annealing of solids, which, when raised to a sufficient temperature at which all particles arrange themselves randomly, and then cooled sufficiently slowly, will result in the configuration of lowest energy. If a system is in thermal equilibrium, then the probability distribution in phase-space is proportional to

$$e^{-U(\gamma)/kT} \tag{4.19}$$

where $U(\gamma)$ is the energy of state γ, k is the Boltzmann constant and T is the surrounding temperature. This concept can also be applied to motion estimation to find the MAP estimate. Figure 4.6 shows a block diagram for the simulated annealing algorithm used in this thesis. The parameters, M_d, T_n, T_0, and T_f represent the number of sites, the temperature at the n^{th} iteration, and the initial and final temperatures respectively. While there are various different annealing schedules, the one used in this thesis is given by

$$T_n = \varphi(T_0, n) = T_0 \cdot a^{n-1} \tag{4.20}$$

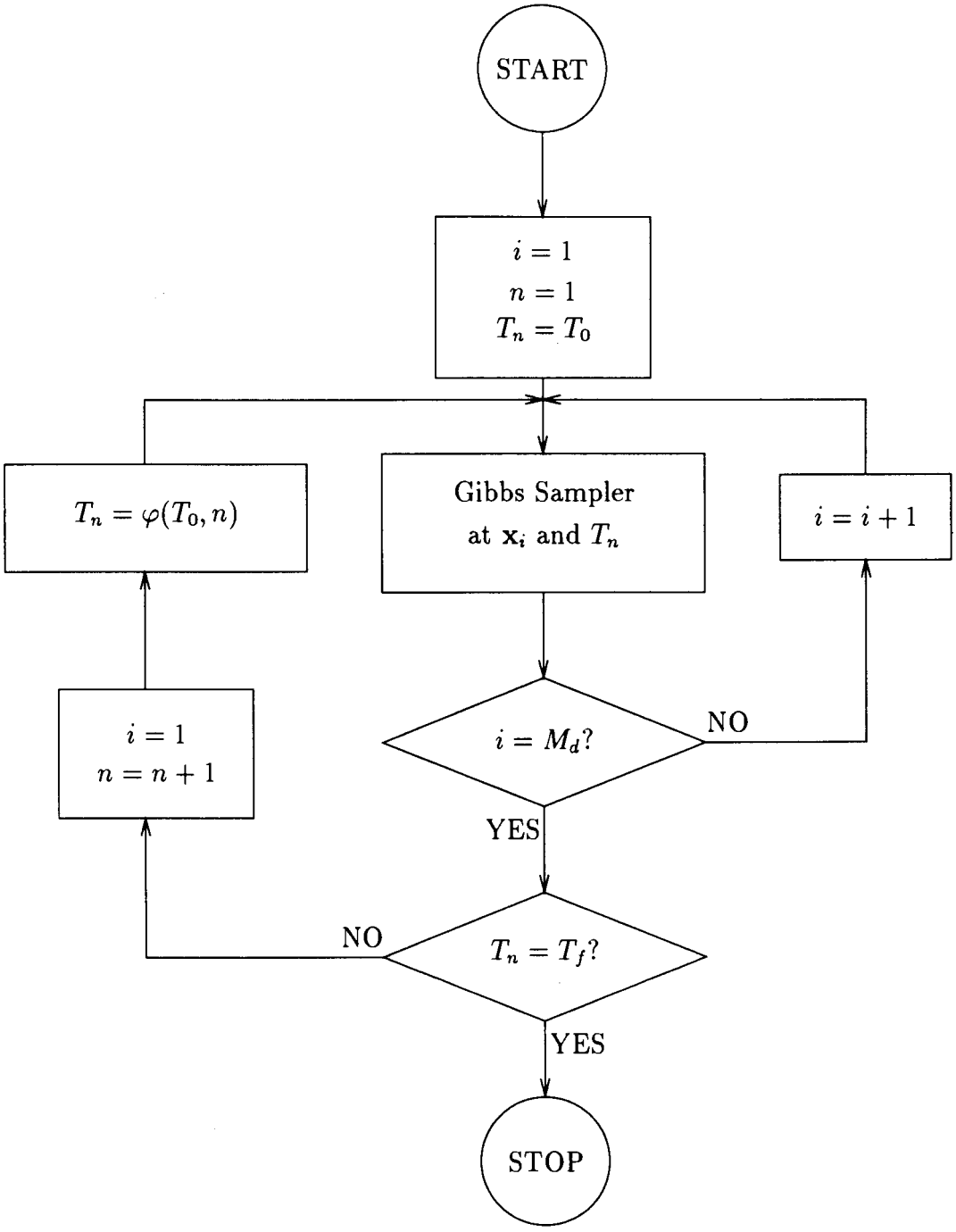


Figure 4.6: Block diagram of simulated annealing algorithm

where $0.0 < a < 1.0$ and T_0 is the initial temperature. The parameter a is chosen as a trade-off between the chance of becoming trapped in a local minimum and the computational time (number of iterations to reach convergence). Typical values for these parameters are $a = 0.9$ and $T_0 = 5.0$.

4.6 Motion Discontinuities

One problem with the previous model derivation is that it assumes a smooth motion field for the entire frame. Obviously this model introduces problems at the edges of moving objects. One suggested method of improving this problem is to introduce a line field (Konrad 1989). This field allows discontinuities in the motion field to exist at the edge of objects and is designated by $l(\mathbf{x}, t)$. Since the line field is not known, it must be included in the estimation, which results in the most likely displacement field, $\hat{\mathbf{d}}_t^*$, and the most likely line field, $\hat{l}_t^*$, being given by the solution to

$$P(\mathbf{D}_t = \hat{\mathbf{d}}_t^*, L_t = \hat{l}_t^* \mid G_{t_-} = g_{t_-}, G_{t_+} = g_{t_+}) \geq$$
$$P(\mathbf{D}_t = \hat{\mathbf{d}}_t, L_t = \hat{l}_t \mid G_{t_-} = g_{t_-}, G_{t_+} = g_{t_+}) \; \forall \hat{\mathbf{d}}_t \in \mathcal{S}_{\mathbf{d}}, \hat{l}_t \in \mathcal{S}_l \qquad (4.21)$$

Using Bayes rule and a similar approach as used earlier in this chapter, the MAP estimate of the pair, $(\hat{\mathbf{d}}_t^*, \hat{l}_t^*)$, is the solution to

$$\max_{(\hat{\mathbf{d}}_t, \hat{l}_t)} [P(G_{t_+} = g_{t_+} \mid \mathbf{D}_t = \hat{\mathbf{d}}_t, L_t = \hat{l}_t, G_{t_-} = g_{t_-}) \cdot$$
$$P(\mathbf{D}_t = \hat{\mathbf{d}}_t \mid L_t = \hat{l}_t, G_{t_-} = g_{t_-}) \cdot P(L_t = \hat{l}_t \mid G_{t_-} = g_{t_-})] \qquad (4.22)$$

4.6.1 Displacement Field Model with Motion Discontinuities

The introduction of a line field does not affect either the structural model (discussed in section 4.3.1) or the observation model (discussed in section 4.3.2). For this reason, only the new displacement field model will be discussed here. The changes to the displacement field model involve revising the potentials so that if the line element between two sites is on (i.e., there is a discontinuity between the two sites), then no penalty is introduced as a result of differing displacement vectors between the two

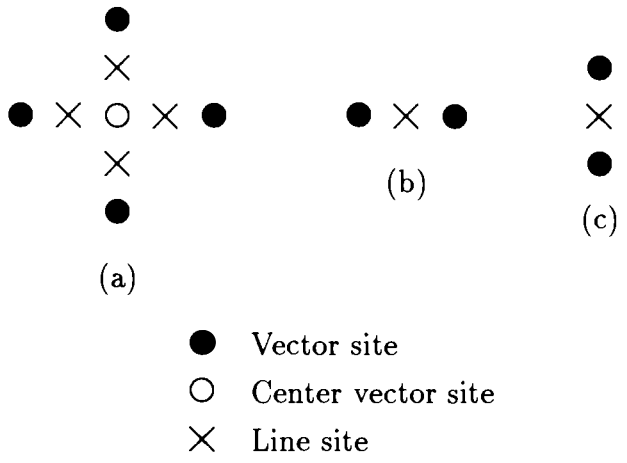


Figure 4.7: (a) First order Neighborhood system $\mathcal{N}^1_{(\mathbf{d},l)}$ for vector field $\mathbf{d}_t$ with line field l_t and the associated (b) horizontal and (c) vertical cliques.

sites. Therefore the probability, $P(\mathbf{D}_t \mid L_t, G_{t-})$, is defined as

$$\pi(\mathbf{d}_t \mid l_t) = \frac{1}{Z_\mathbf{d}} e^{U_\mathbf{d}(\mathbf{d}_t|l_t)/\beta_\mathbf{d}} \tag{4.23}$$

where the conditional energy term is defined as

$$U_\mathbf{d}(\mathbf{d}_t \mid l_t) = \sum_{c_\mathbf{d}=\{\mathbf{x}_i,\mathbf{x}_j\}\in\mathcal{C}_\mathbf{d}} V_\mathbf{d}(\mathbf{d}_t, c_\mathbf{d}) \cdot [1 - l(<\mathbf{x}_i, \mathbf{x}_j>, t)] \tag{4.24}$$

where $V_\mathbf{d}$ is the same potential function as before, and $l(<\mathbf{x}_i, \mathbf{x}_j>, t)$ is the site of the line element that is located between vector sites $\mathbf{x}_i$ and $\mathbf{x}_j$. Each line field site, $l(<\mathbf{x}_i, \mathbf{x}_j>, t)$, may take on either a value of 1 indicating that there is a discontinuity, or a value of 0 indicating that there is no discontinuity. The cliques, $c_\mathbf{d}$, are based on the first order neighborhood system and associated cliques shown in Figure 4.7.

4.6.2 Line Field Model

The line field is assumed to be samples from a binary MRF, L_t, and therefore is described by the Gibbs distribution

$$P(L_t \mid G_{t-}) = \pi(l_t \mid g_{t-}) = \frac{1}{Z_l} e^{U_l(l_t|g_{t-})/\beta_l} \tag{4.25}$$

The line energy function, U_l, is defined as

$$U_l(l_t \mid g_{t-}) = \sum_{c_l\in\mathcal{C}_l} V_l(l_t, g_{t-}, c_l) \tag{4.26}$$

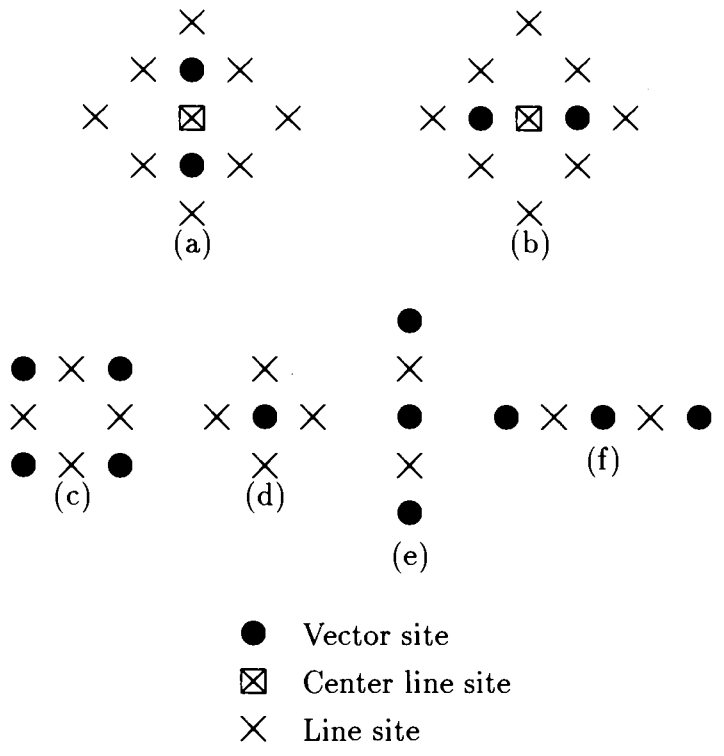


Figure 4.8: Neighborhood system $\mathcal{N}_l$ for line field l_t of (a) horizontal and (b) vertical line element, and the associated (c),(d) four-element and (e),(f) two-element cliques.

where c_l is a line clique and $\mathcal{C}_l$ is the set of all line cliques. The purpose of the line potential function, V_l, is to assign a penalty to the introduction of a line element.

The line field is of a structure such that there are four elements associated with a vector site (above, below, right and left). Each of these elements is shared with another vector site except for those sites on the perimeter of the frame. Therefore, there are two types of line element: vertical and horizontal. The neighborhoods and associated cliques are shown in Figure 4.8. The four element clique in Figure 4.8(c) is used to model the shape of motion boundaries, while the four element clique in Figure 4.8(d) is used to disallow isolated vector sites. The two element cliques in Figure 4.8(e) and (f) discourage the introduction of line elements.

The costs associated with the possible configurations (within a rotation) of the first four element clique (V_{l_4}) and the two element cliques (V_{l_2}) are shown in Figure 4.9(a) and (b) respectively. The second four element cliques is only used when all four line elements are "on" for which the potential is set as $V_{l_4} = \infty$. These costs

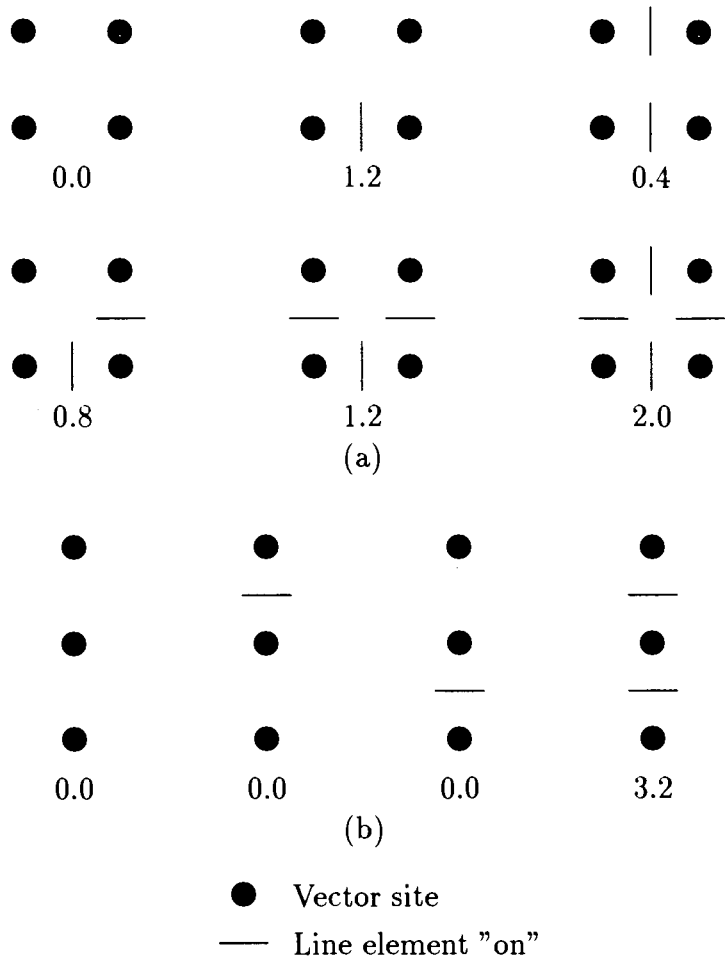


Figure 4.9: Potentials associated with various line element configurations (up to a rotation) for the (a) four-element and (b) two-element cliques.

are chosen in such a way as to encourage realistic (i.e. continuous) line fields while discouraging isolated sites and line field elements (Konrad 1989).

Motion discontinuities generally occur at intensity edges and therefore an additional potential function is introduced for the single element clique:

$$V_{l_1} = \begin{cases} \frac{\alpha}{(\nabla_v g_{t\_})^2} \cdot l_h(<\mathbf{x}_i, \mathbf{x}_j>, t) & \text{for horizontal } c_l = <\mathbf{x}_i, \mathbf{x}_j> \\ \frac{\alpha}{(\nabla_h g_{t\_})^2} \cdot l_v(<\mathbf{x}_i, \mathbf{x}_j>, t) & \text{for vertical } c_l = <\mathbf{x}_i, \mathbf{x}_j> \end{cases} \tag{4.27}$$

where the spatial gradient can be estimated using numerous different techniques. The technique used in this thesis is same as used by Konrad (1989).

The total local potential for the line field is then given by

$$V_l(l_t, g_{t-}, c_l) = V_{l_4}(l_t, c_l) + V_{l_2}(l_t, c_l) + V_{l_1}(l_t, c_l) \tag{4.28}$$

where V_{l_4} and V_{l_2} are given in Figure 4.9 and V_{l_1} is given in Equation 4.27.

4.6.3 Gibbs Sampler for Model with Motion Discontinuities

In order to reduce the computational complexity, updates of the displacement vector field and the line field are done separately, each using the Gibbs sampler method. In (Konrad 1989), the conditional probability for the displacement vector at location $\mathbf{x}_{n_\tau}$ is given by

$$P(\mathbf{D}(\mathbf{x}_{n_\tau}, t) = \hat{\mathbf{d}}(\mathbf{x}_{n_\tau}, t) \mid \mathbf{D}(\mathbf{x}_j, t) = \hat{\mathbf{d}}(\mathbf{x}_j, t), j \neq n_\tau, \hat{l}_t, G_{t-} = g_{t-}, G_{t+} = g_{t+}) = \frac{e^{-U_{\mathbf{d}}^{n_\tau}(\hat{\mathbf{d}}(\mathbf{x}_{n_\tau}, t) \mid \hat{\mathbf{d}}_t, \hat{l}_t, g_{t-}, g_{t+})}}{\sum_{z \in S'_{\mathbf{d}}} e^{-U_{\mathbf{d}}^{n_\tau}(z \mid \hat{\mathbf{d}}_t, \hat{l}_t, g_{t-}, g_{t+})}} \tag{4.29}$$

where the local energy function is defined as

$$U_{\mathbf{d}}^{n_\tau}(z \mid \hat{\mathbf{d}}_t, g_{t-}, g_{t+}) = \lambda_g \cdot [r(, \mathbf{x}_{n_\tau}, t, \delta t)]^2 + \lambda_{\mathbf{d}} \cdot \sum_{j: \mathbf{x}_j \in \eta_{\mathbf{d}}(\mathbf{x}_{n_\tau})} \| \mathbf{z} - \hat{\mathbf{d}}(\mathbf{x}_j, t) \|^2 \cdot [1 - l(<\mathbf{x}_{n_\tau}, \mathbf{x}_j>, t)] \tag{4.30}$$

The conditional probability for a line element at location (y_{n_τ}, t) is

$$P(L(\mathbf{y}_{n_\tau}, t) = \hat{l}(\mathbf{y}_{n_\tau}, t) \mid L(\mathbf{y}_j, t) = \hat{l}(\mathbf{y}_j, t), j \neq n_\tau, \hat{\mathbf{d}}_t, g_{t-}) = \frac{e^{-U_l^{n_\tau}(\hat{l}(\mathbf{y}_{n_\tau}, t) \mid \hat{l}_t, \hat{\mathbf{d}}_t, g_{t-})}}{\sum_{z \in S'_l} e^{-U_l^{n_\tau}(z \mid \hat{l}_t, \hat{\mathbf{d}}_t, g_{t-})}} \tag{4.31}$$

where the local line energy is defined as

$$U_l^{n_\tau}(z \mid \hat{l}_t, \hat{\mathbf{d}}_t, g_{t-}) = \lambda_{\mathbf{d}} \cdot \sum_{l<\mathbf{x}_j, \mathbf{x}_k>=\mathbf{y}_{n_\tau}} \| \hat{\mathbf{d}}(\mathbf{x}_j, t) - \hat{\mathbf{d}}(\mathbf{x}_k, t) \|^2 \cdot [1 - z] + \lambda_l \cdot \sum_{c_l: \mathbf{y}_{n_\tau} \in \mathcal{C}_l} V_l(z, g_{t-}, c_l) \tag{4.32}$$

The solution to this problem is obtained by using simulated annealing and the Gibbs sampler to alternately update the motion and line fields via the two conditional probabilities in Equations 4.29 and 4.31 respectively. However, since the local line energy is dependent on the motion field, it is necessary to obtain a rough estimate of the motion field prior to introducing the line field. This is accomplished by running simulated annealing without the line field for a number of iterations until an estimate is obtained.

4.7 Results

This section presents results of the motion estimation algorithms described in this chapter. In all results presented here, the discrete displacement vector state space, $\mathcal{S}_{\mathbf{d}}$, is composed of integer vectors in the two dimensional range

$$\mathcal{S}_{\mathbf{d}} = \{\mathbf{d} = (d_x, d_y) : -8 \leq d_x \leq 8, -8 \leq d_y \leq 8\} \tag{4.33}$$

Since the energy values are normalized to ensure that the exponential values fall in the floating point range of our workstations, only the ratio, $\lambda_{\mathbf{d}}/\lambda_g$, describing the tradeoff between matching and smoothing, is required to compute the energies. The individual values of $\lambda_{\mathbf{d}}$ and λ_g do, however, affect the choice of the initial temperature, T_0.

The annealing schedule used was given by Equation 4.20 with $T_0 = 5.0$ and $a = 0.9$. Optimization of this schedule could result in slightly improved results and/or a more computationally efficient algorithm.

The motion estimates in this section have been computed using progressive sequences for the case where displacement vectors are estimated only for all known pixel amplitudes on the sampling grid, i.e., $\Lambda_{\mathbf{d}} = \Lambda_g$.

4.7.1 Results for the *comp* sequence

Recall that the *comp* sequence is composed of the first frame of the *pongi* sequence as its background with a 50×50 block from the *missa* sequence moving at a constant displacement of $(+4, +4)$. The displacement vector field for the moving block area of the second frame of this sequence is shown in Figure 4.10.

Figure 4.11 shows the estimated motion field for the moving block area of the second frame of the *comp* sequence. No line field was used in this experiment. It is clear from Figure 4.11 and 4.10 that the algorithm has problems estimating motion near the edges of moving objects. This problem occurs due to a number of reasons including:

1. New areas are exposed when an object moves, which creates matching errors.

2. Regions are hidden behind the moving object, which creates matching problems in a smooth motion field.

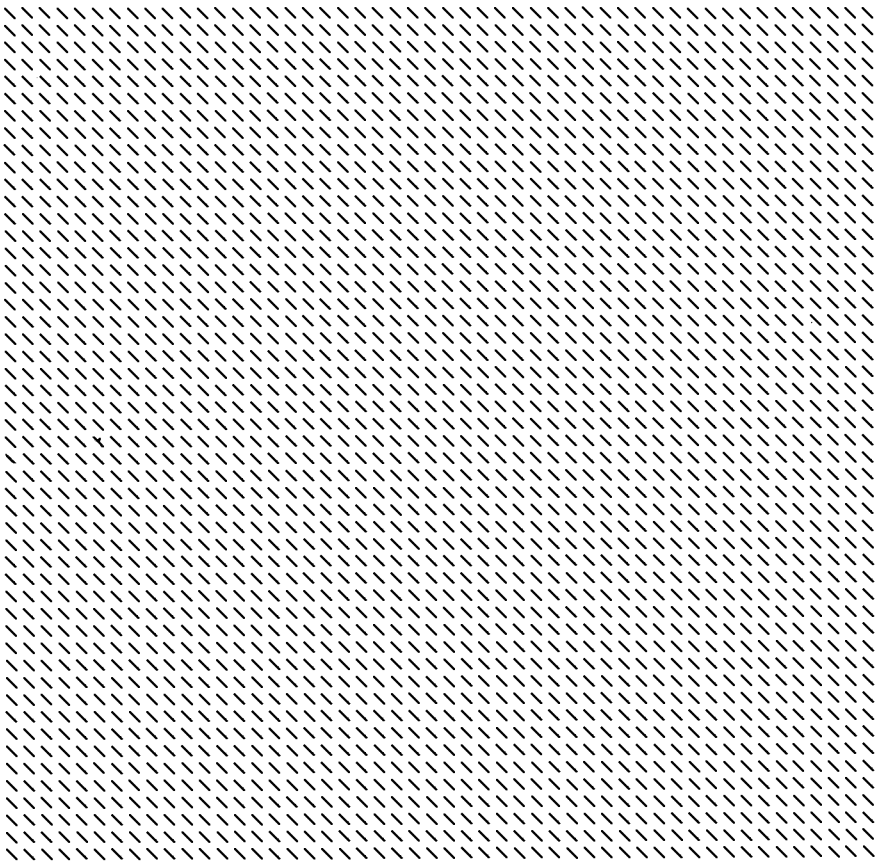


Figure 4.10: Actual motion field for moving block area of the second frame of the *comp* sequence.

3. Motion discontinuities are disallowed, which results in averaging over the motion discontinuity.

The first two problems are beyond the realm of this thesis. They are, however, considered by Dubois and Konrad (1993). The third problem can be reduced through the use of a line field as described in this chapter.

Figure 4.12 shows the resulting motion field for the *comp* sequence when a line field (represented by the dotted line) is used. It is clear that the line field does improve the estimate of the motion field. Note, however, that there is a region where the boundary is not clearly defined on the left of the field. This is largely due to a small discontinuity in the pixel amplitude on the left side of the block, i.e., the background is the same colour as the block on this edge.

Note that while the results using a line field for the progressive test sequences were superior to not using a line field, difficulties were encountered using a line field on an interlaced sequence. This is likely because of the spatial interpolation error in the estimation of the missing lines. This error introduces errors into pixel amplitude discontinuities. For example, horizontal edges are smoothed by the spatial interpolation, which increases the likelihood for an incorrect line field. Incorrect line fields can lead to horribly erroneous motion vectors.

4.7.2 Results for *pongi* sequence

Figures 4.13 and 4.14 show the motion fields for the shoulder and upper arm area of the second frame of the *pongi* sequence without a line field and with a line field respectively. The upwards motion of the arm and shoulder is clearly shown in both figures. In Figure 4.14, the line field corresponds reasonably well to the upper edge of the arm and the collar of the shirt. Note that there is more difficulty estimating the line field at the bottom of the arm. This is due to the newly exposed areas below the arm, which do not match well with areas in the previous frame. This type of problem could be improved by using bidirectional motion estimation (both Υ_{n-1} and Υ_{n+1} are used). In this thesis, however, the main application of this motion estimation is for motion compensation and, in general, the decoder would not have access to future frames.

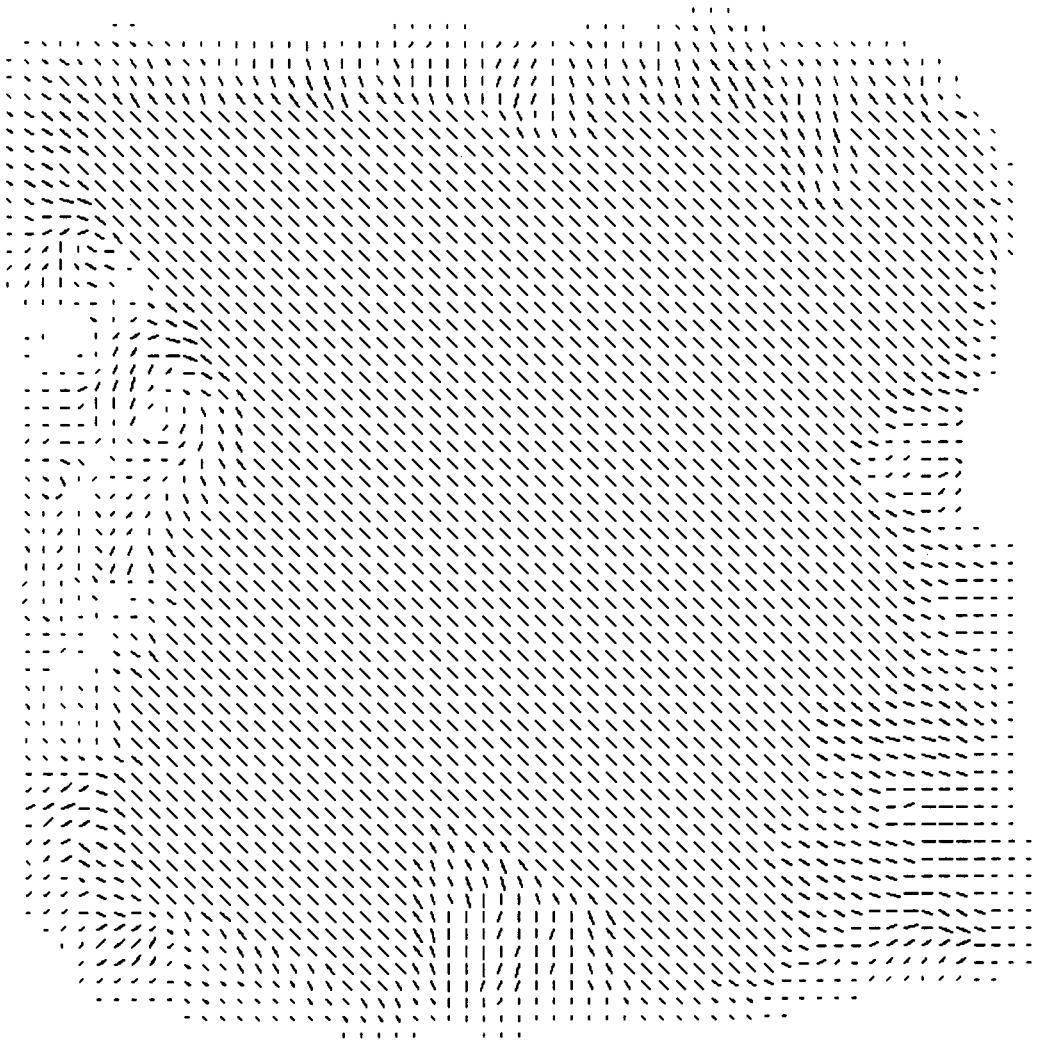


Figure 4.11: Motion field for moving block area of the second frame of the *comp* sequence. No line field is used. $\lambda_{\mathbf{d}} = 0.3, \lambda_g = 0.1$. Exponential annealing schedule. $T_0 = 5.0, a = 0.9, 100$ iterations.

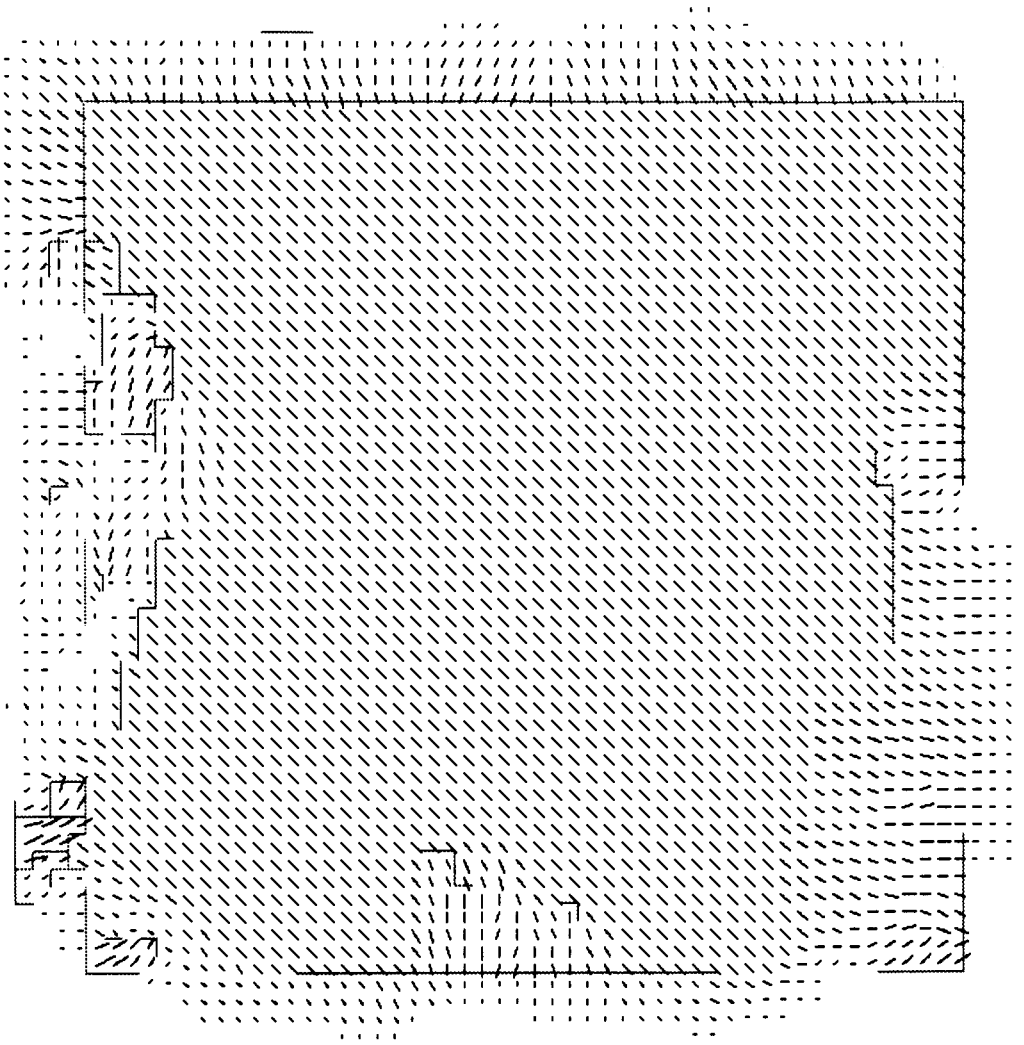


Figure 4.12: Motion field for moving block area of the second frame of the *comp* sequence. Line field after 39 iterations. $\lambda_{\mathbf{d}} = 0.3, \lambda_g = 0.1, \lambda_l = 0.5$. Exponential annealing schedule. $T_0 = 5.0, a = 0.9, 100$ iterations. For line field $T_l = 50T$.

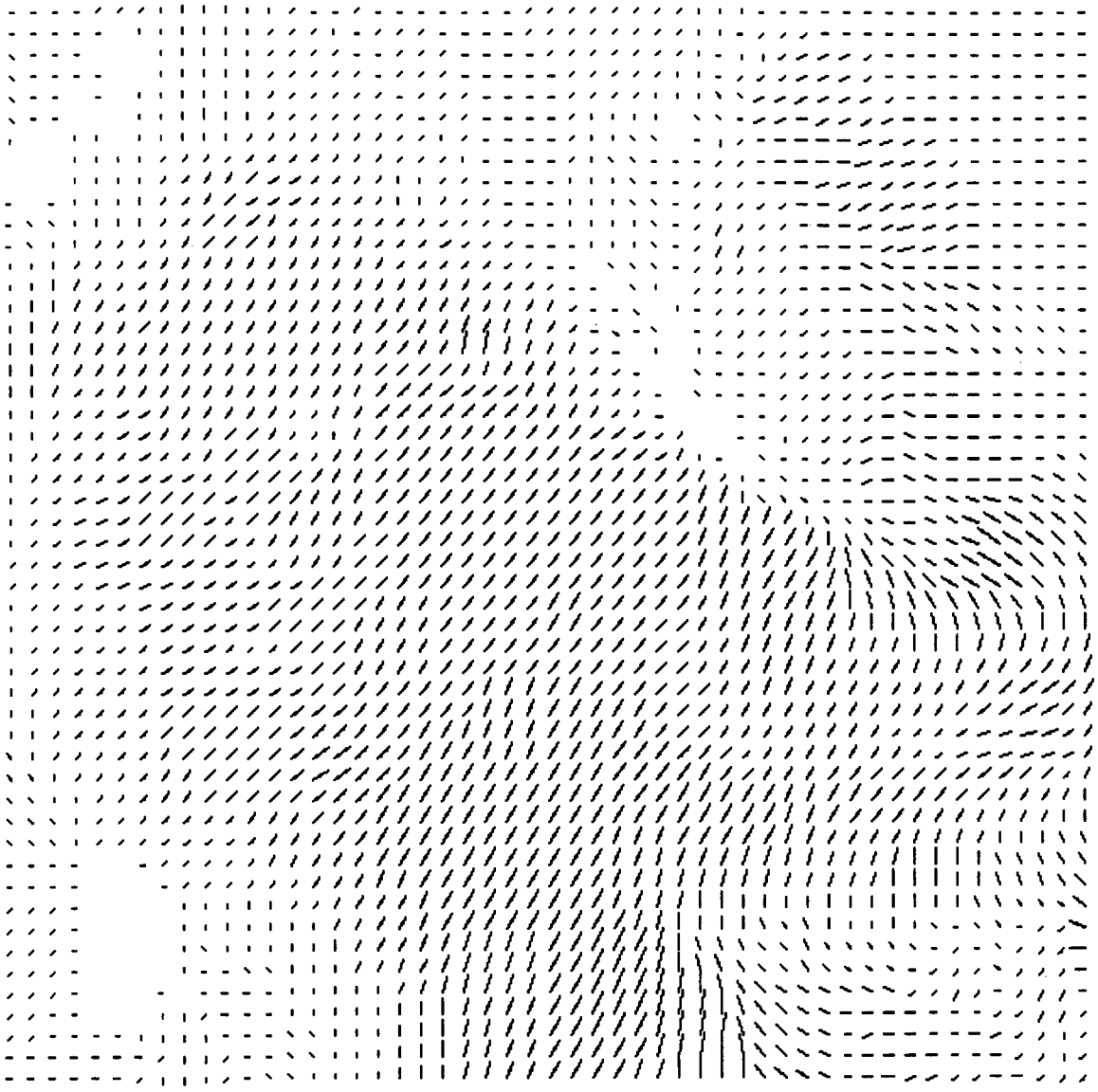


Figure 4.13: Motion field for shoulder and upper arm area of the second frame of the *pongi* sequence. No line field is used. $\lambda_{\mathbf{d}} = 0.07, \lambda_g = 0.0025$. Exponential annealing schedule. $T_0 = 5.0, a = 0.9, 100$ iterations.

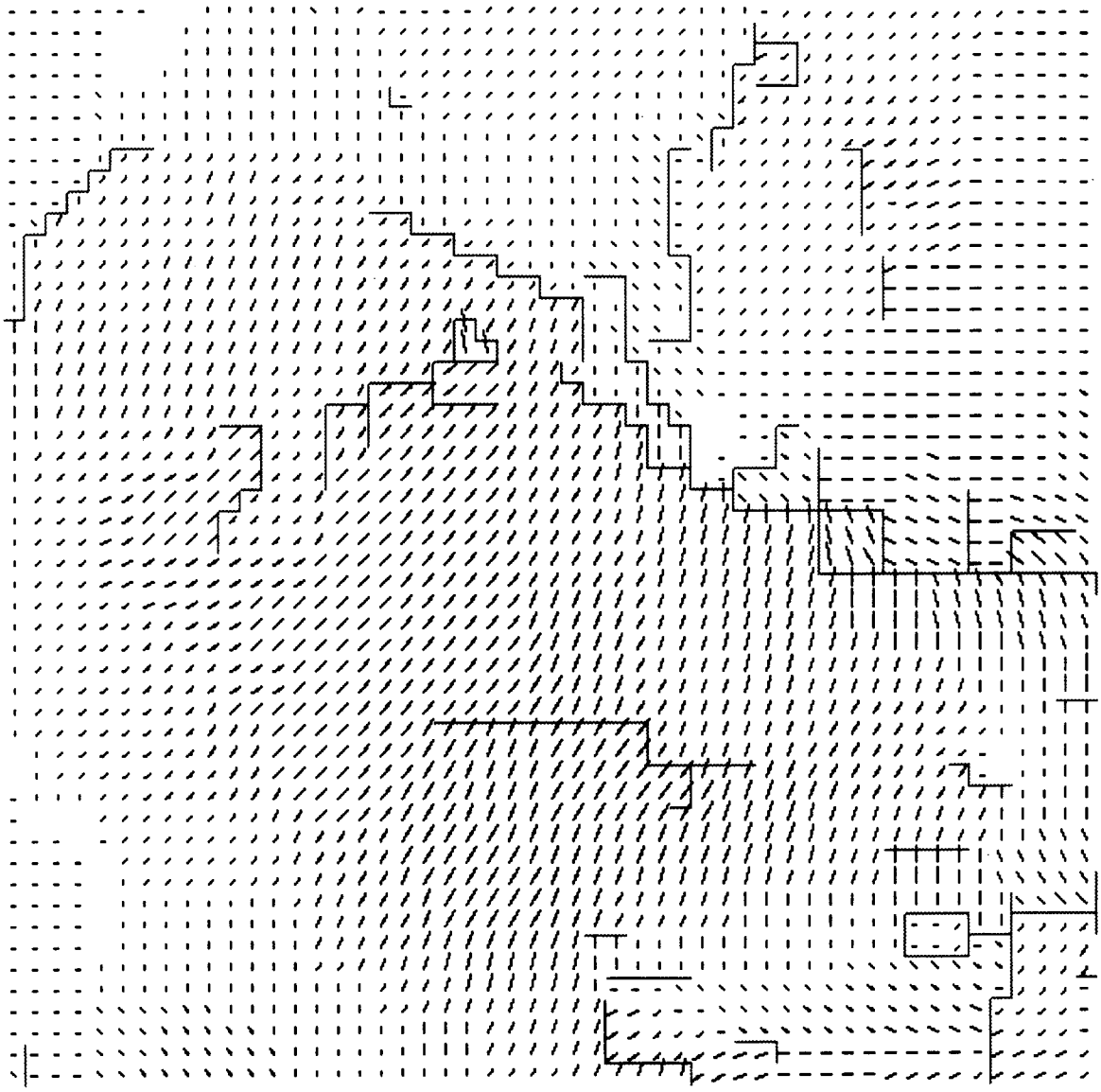


Figure 4.14: Motion field for shoulder and upper arm area of the second frame of the *pongi* sequence. Line field after 39 interations. $\lambda_{\mathbf{d}} = 0.3, \lambda_g = 0.0025, \lambda_l = 0.5$. Exponential annealing schedule. $T_0 = 5.0, a = 0.9, 100$ iterations. For line field $T_l = 50T$.

4.7.3 Computational Cost

The methods presented in this chapter provide an excellent estimate of the motion in a video sequence; however, the computational cost is very high. Let the following quantities be defined as follows:

$\mathcal{F}_{state}$ floating point operations (flops) necessary to calculate Equation 4.17 for a single state in the sample space for one site.

$\mathcal{D}_{\mathbf{d}}$ number of states in the sample space, $\mathcal{S}_{\mathbf{d}}$

$M_{\mathbf{d}}$ number of vectors in a frame's displacement field

$\mathcal{I}_{frame}$ number of iterations per frame

Then, the computational cost is given by

$$CC \approx (\mathcal{I}_{frame})(M_{\mathbf{d}})(\mathcal{D}_{\mathbf{d}})(\mathcal{F}_{state}) \tag{4.34}$$

Depending on the implementation and the choice of parameters, $\mathcal{F}_{state}$ varies between 25 and 35 flops. For a first order neighborhood system, $\mathcal{F}_{state} \approx 25$ and for this implementation of the algorithm without the line field, $\mathcal{D}_{\mathbf{d}} = 289$ and $\mathcal{I}_{frame} = 100$.[6] Then for the *pongi* sequence ($M_{\mathbf{d}} = 86400$) the computational cost is

$$CC \approx 62 \text{ Gflops/frame}$$

Using similar calculations, the approximate computational costs for several other motion estimation methods described in this thesis are shown in Table 4.1. Details of the Horn & Schunk and Gauss-Newton method of MAP estimation are given in Chapter 6. The most expensive method is the MAP estimation with the simulated annealing technique. Note that both of the simulated annealing methods are significantly more expensive than the other three by at least two orders of magnitude. This additional expense must be traded-off against the quality of the motion field estimation depending on the application.

Recently, Miller and Rose (1994) have proposed a deterministic annealing algorithm which seems to work well, and is more computationally efficient than simulated annealing. Future application of this algorithm to Bayesian motion estimation would reduce the computational cost, with possibly limited performance degradation.

[6]The number of iterations could probably be lowered by optimizing the annealing schedule.

Table 4.1: Computational Cost of Various Motion Estimation Methods

Motion Estimation Method	Approximate Computational Cost	Relative Cost
MAP estimation (simulated annealing)	62 Gflops/frame	1.0000
MAP estimation with line field (simulated annealing)	87 Gflops/frame	1.4000
MAP estimation (Gauss-Newton)	170 Mflops/frame	0.0027
Horn & Schunk	540 Mflops/frame	0.0087
Block-based matching estimation	75 Mflops/frame	0.0012

Parallel Processing

The computational costs shown in Table 4.1 are an indication of the number of floating point operations (flops) per frame. After examining the MAP estimation (simulated annealing) algorithm described in this chapter, it is clear that it could easily be implemented in a parallel arrangement. The extreme of this approach would be to have a processor for each site in the motion field. The computational cost would then be virtually independent of the size of the motion field, and the computational cost would be reduced by a factor of $M_{\mathbf{d}}$ which in the case of the *pongi* sequence would reduce the computational cost to

$$CC \approx (\mathcal{I}_{frame})(\mathcal{D}_{\mathbf{d}})(\mathcal{F}_{state}) \approx 0.72 \text{ Mflops}$$

While the monetary cost of such an implementation would be high, it would make the computation possible in real time.

Chapter 5

Interlaced Bayesian Motion Estimation

There are a number of reasons for obtaining good motion estimations for both the known and missing fields in an interlaced sequence. These include interpolation of the sequence (de-interlacing), hybrid motion compensation prediction and motion compensation. The method of Bayesian motion estimation can be applied to these interlaced sequences; however, difficulties arise due to the missing information. This chapter discusses various approaches of applying Bayesian motion estimation to interlaced sequences, including a new method that performs significantly better than previous methods.

5.1 Estimating Motion in Interlaced Sequences

In order to perform Bayesian motion estimation in interlaced sequences, the missing fields must be reconstructed using spatial filtering to form progressive frames on which the method discussed in the previous chapter can be performed.

5.2 Structural Model Fault

In Section 4.3.1, a structural model was assumed such that the pixel intensity is constant along the path of a motion vector (Equation 4.6). In the case of Bayesian

motion estimation applied to spatially interpolated interlaced sequences, this model does not hold well. Since vertical 1D filters do not work well in areas of high detail (especially areas of high vertical spatial frequency), pixel intensities can vary greatly along the motion vectors in this area. This results in very poor motion estimates in these areas and consequently inferior interpolations.

For example, Figure 5.1 shows a small section of a progressive frame containing a narrow object cutting across the frame. When the interlaced field corresponding to this frame is interpolated using one-dimensional vertical filtering the frame section shown in Figure 5.2 is obtained. Note that some parts of the object are lost completely, while other parts are averaged to an incorrect value. The missing and incorrect values will cause a variation of pixel intensities along motion vectors and will result in erroneous motion estimates.

Three algorithms that provide improvements to this model will be discussed in this chapter. Note that all three algorithms are discussed in the context of integer displacement vectors. More precise vectors can be used; however, this results in increased computational cost since the state space discussed in the previous chapter increases in size.

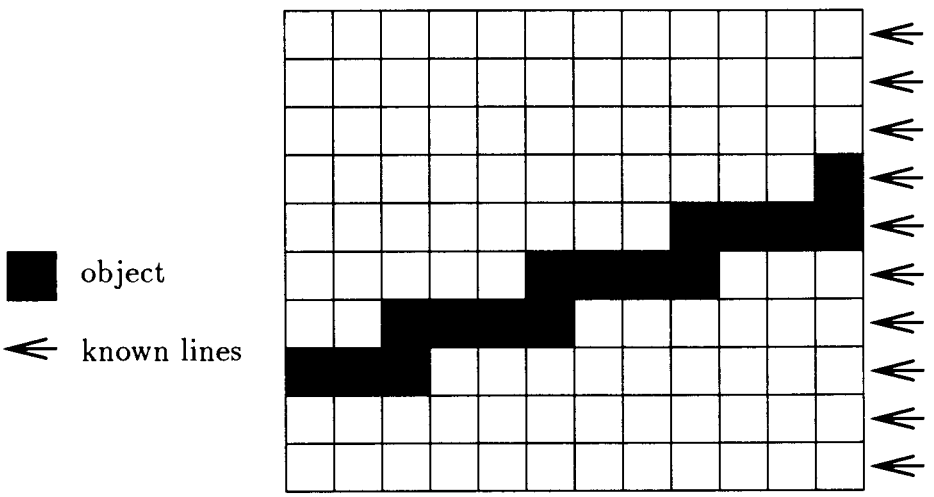


Figure 5.1: Small section of progressive frame

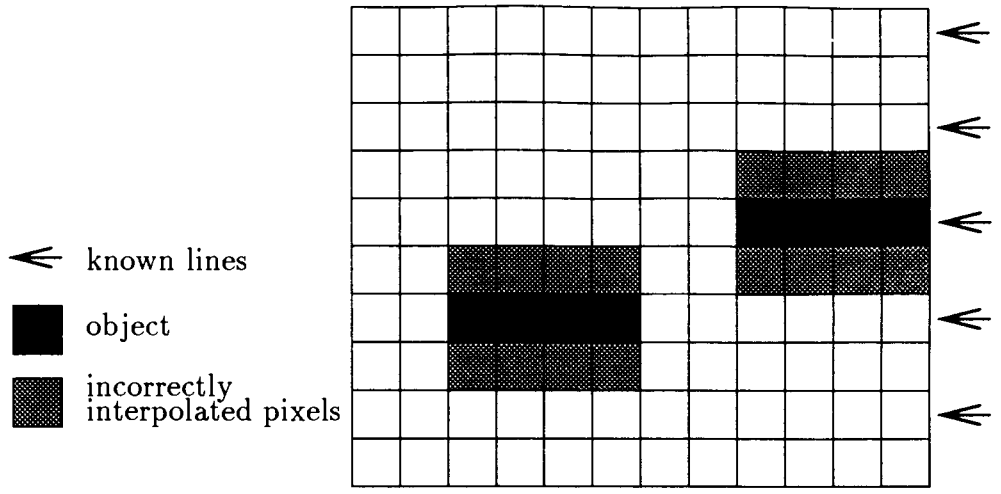


Figure 5.2: Small section of vertically, spatially interpolated interlaced frame corresponding to the progressive frame in the previous figure

5.3 Improved Interpolation

The first method of improving the validity of the structural model is to improve the interpolation of the current frame so that it more closely matches the previous frame. This, however, involves more complicated interpolation methods, which leads us into a circular problem, since better interpolation methods generally involve some sort of motion estimation.

5.4 Pixel Prediction Along Linear Trajectories

An improvement to the structural model for interlaced sequences is proposed by Depommier and Dubois (1992). In Equation 4.18, the energy is composed of two terms: a matching error term (displaced pel difference) and a smoothness term. The matching error term is given by

$$E_u(\mathbf{x}, \mathbf{d}(\mathbf{x}) = \tilde{g}(\mathbf{x}, t_+) - \tilde{g}(\mathbf{x} - \mathbf{d}(\mathbf{x}, t), t_-) \tag{5.1}$$

The notation, tilde, above, denotes an approximate value since, in the case of interlaced sequences, not all required values of $g(\mathbf{x}, t)$ can be observed. The errors in the estimates of $g(\mathbf{x}, t)$ cause the structural model to become less valid.

Since the structural model does not hold well in interlaced sequences that have been spatially interpolated to form progressive sequences, the matching term can increase the energy for the *correct* displacement vector. This incorrect contribution from the matching term is especially prevalent in stationary areas containing high detail. In these areas, the spatial interpolation gives rise to high errors and therefore the matching term is highly inaccurate.

Matching can be improved by including the de-interlaced version of the second previous field, Ω_{n-2}. In stationary areas, spatial interpolation results in approximately the same interpolation error in both Ω_n and Ω_{n-2}, and good matches are found between these two spatially interpolated fields. However, as the y-component of the motion increases, it is better to use the spatially interpolated version of Ω_{n-1}. This results in an error matching term in the form of

$$E_u(\mathbf{x}, \mathbf{d}(\mathbf{x})) = \tilde{g}(\mathbf{x}, t_+) - \alpha(\mid \mathbf{d}_y \mid)\tilde{g}(\mathbf{x} - \mathbf{d}(\mathbf{x}, t), t_-) - [1 - \alpha(\mid \mathbf{d}_y \mid)]\tilde{g}(\mathbf{x} - 2\mathbf{d}(\mathbf{x}, t), t_{-2}) \tag{5.2}$$

where t_{-2} is the position in time of Ω_{n-2}, and $\alpha(\mid \mathbf{d}_y \mid)$ is some function that changes the weighting of the two previous fields in the matching term depending on $\mathbf{d}_y$, the y-component of the displacement vector, $\mathbf{d}(\mathbf{x})$.

The function, $\alpha(\mid \mathbf{d}_y \mid)$, that is used by Depommier and Dubois (1992) is given by

$$\alpha(\mid \mathbf{d}_y \mid) = \begin{cases} \frac{1}{2}(1 - \cos(\frac{\pi \mathbf{d}_y}{Y}) & \text{when } 0 < \mid \mathbf{d}_y \mid \leq Y \\ 1 & \text{when } Y < \mid \mathbf{d}_y \mid \end{cases} \tag{5.3}$$

and is shown in Figure 5.3. This function causes only the second previous field, Ω_{n-2}, to be used when $\mathbf{d}_y = 0$ and only the previous field, Ω_{n-1}, to be used if $\mid \mathbf{d}_y \mid \geq Y$. In the region, $0 < \mid \mathbf{d}_y \mid < Y$, a combination of the two previous fields is used.

5.5 Variable Previous Frame Motion Estimation

A novel approach to improving the structural model for interlaced sequences is called *variable previous frame motion estimation* and involves varying the previous frame depending on the candidate motion vector, and whether the current pixel is known or interpolated. This variation attempts to introduce the same simple interpolation error into the previous frame in order to increase the validity of the structural model

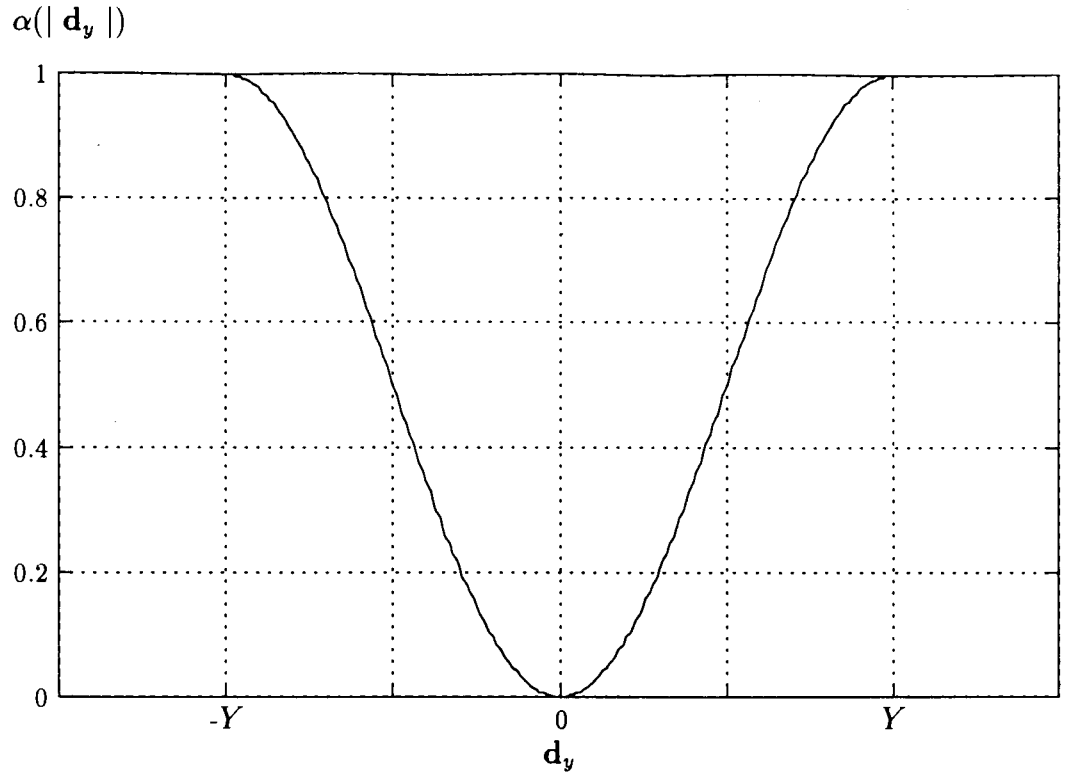


Figure 5.3: Weighting Function

in Section 4.3.1. If the matching error term is given by

$$E_u(\mathbf{x}, \mathbf{d}(\mathbf{x})) = [g(\mathbf{x}, t_+) + \mathcal{E}(\mathbf{x}, t_+)] - [g(\mathbf{x} - \mathbf{d}(\mathbf{x}, t), t_-) + \mathcal{E}(\mathbf{x} - \mathbf{d}(\mathbf{x}, t), t_-)] \quad (5.4)$$

where $\mathcal{E}$ is the interpolation error between g and $\tilde{g}$, then by using the same interpolation to compute both $\tilde{g}(\mathbf{x}, t_+)$ and $\tilde{g}(\mathbf{x} - \mathbf{d}(\mathbf{x}, t), t_-)$, this method attempts to have non-zero but equal errors such that

$$\mathcal{E}(\mathbf{x}, t_+) \approx \mathcal{E}(\mathbf{x} - \mathbf{d}(\mathbf{x}, t), t_-) \neq 0 \quad (5.5)$$

The general concept is to sacrifice accuracy in the estimate of $\tilde{g}$ to get a better motion estimate.

This method assumes that the previous frame has been reconstructed well. If this is the case then the implementation of the Bayesian motion estimation method leads to three separate cases when calculating the displaced pixel difference (DPD):

1. **The current pixel is on a known line in the current frame** – In this

case, the best match will be found in the de-interlaced previous frame, which is designated Ψ_{n-1}.

2. **The current pixel is on a missing line in the current frame and the y-component of the candidate motion vector is even** – In this case, the current pixel is interpolated and, since the y-component is even, it is desirable to spatially interpolate the same parity field in the previous frame in the same way as the current frame. For example, if the even lines are known in the current field then the even lines in the previous frame are used in the spatial interpolation. This results in a replacement of the known pixels by interpolating using unknown (previously interpolated) pixels. The frame generated by this process is designated Ψ_e.

3. **The current pixel is on a missing line in the current frame and the y-component of the candidate motion vector is odd** – In this case, the current pixel is interpolated and the y-component is odd. This means that the match will lie on a lie of the opposite parity in the previous field and therefore the opposite parity lines are used in the previous field interpolation. This frame is referred to as Ψ_o.

Figure 5.4 and 5.5 show an example of this method on a small 7×7 section of a frame starting at location $(9, 5)$. In Figure 5.4, the three versions of the previous spatially, linear interpolated frame, Ψ_{n-1}, Ψ_e and Ψ_o, are shown. Note that since this object is a single-pixel-width line, vertical spatial interpolation is unable to recreate the missing pixels accurately. In some cases the pixels are completely eliminated (eg. location $(10, 10)$ in Figure 5.4(b)) and in other cases the pixels are interpolated to an incorrect value (eg. location $(11, 10)$ in Figure 5.4(b)).

In Figure 5.5, two possible translations (between the previous field and current field) of the object in Figure 5.4(a) are shown. The first of these translations is the case where the y-component of the motion vector is even (in this case it is 2 upwards). We can see that the best match for the pixels on the unknown lines can be found in Ψ_e. This will result in an accurate motion estimation. Similarly, in figure 5.4(b) an odd y-translation is shown (1 upwards) and the best match is found in Ψ_o.

Note that this method will work well if all the pixels used in the vertical interpolation for any given pixel have approximately equal motion vectors. This is a common

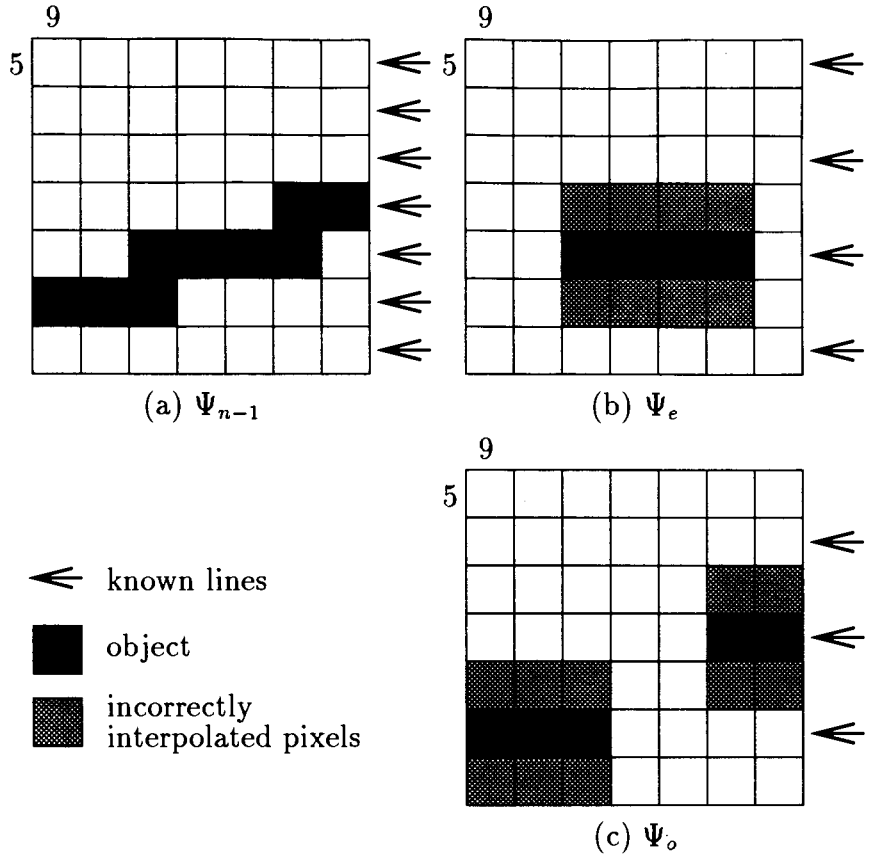


Figure 5.4: Three previous frames used in variable motion estimation

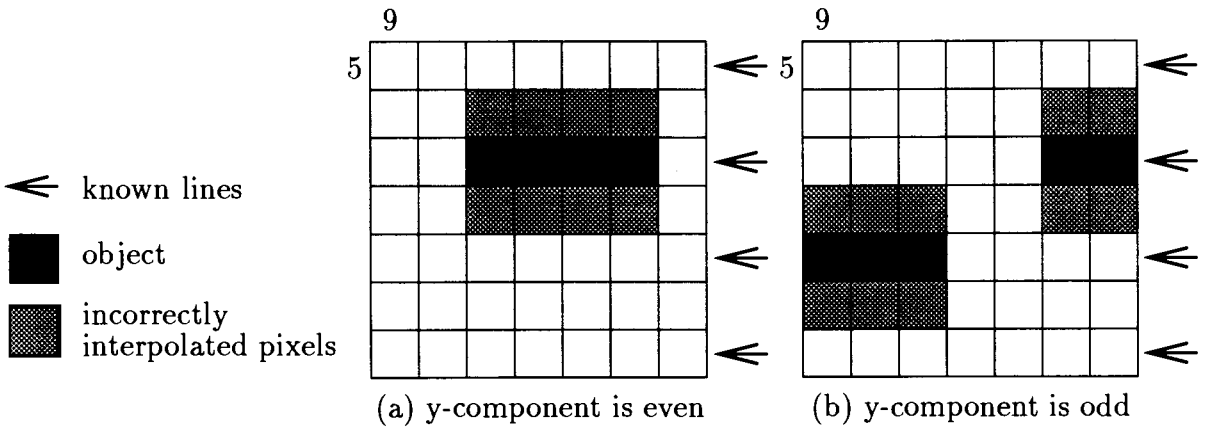


Figure 5.5: Example of two different current frames

feature of many objects and backgrounds and therefore significant improvement can be made over other methods.

5.5.1 Modified Variable Previous Frame Motion Estimation

One of the most difficult components of motion estimation in interlaced sequences is to correctly estimate motion (if any) in the background. In many sequences the background motion is zero corresponding to a stationary camera. However, when the camera is moving, the background will have some non-zero motion. A common type of camera movement is horizontal panning, which results in a non-zero x-component of the motion with a zero y-component of the motion. Therefore, in two common cases (stationary camera and horizontal panning camera), the y-component of the motion will be zero and Ω_{n-2} can be used since it is the same parity as Ω_n. The assumption that is made here is that the camera panning is linear over the three fields, Ω_{n-2}, Ω_{n-1}, and Ω_n. This knowledge can then be used to improve the variable previous frame motion estimation algorithm described in the previous section by using a spatially interpolated version of Ω_{n-2} when the y-component of the motion vector is zero.

5.6 Dual Frame Motion Estimation

Figure 5.6 shows the y-t plane of an interlaced sequence of three consecutive fields. If linear displacement vectors are assumed over three consecutive fields, then it is clear (Figure 5.6) that for *odd* y-components the best match for a known pixel is found in Ω_{n-1} and for *even* y-components the best match for a given pixel is found in Ω_{n-2}. In addition, since the same spatial interpolation is used to form estimates of the missing pixels, the same rules will apply for the missing pixel matches. Therefore, dual frame motion estimation uses two previous spatially interpolated frames:

1. Ψ_{n-1} (from Ω_{n-1}) — y-component of the candidate displacement vector is odd.

2. Ψ_{n-2}, (from Ω_{n-2}) — y-component of the candidate displacement vector is even.

Note that this method is an extension of that proposed by (Depommier and Dubois 1992) since it uses Ψ_{n-2} for more than just zero y-components.

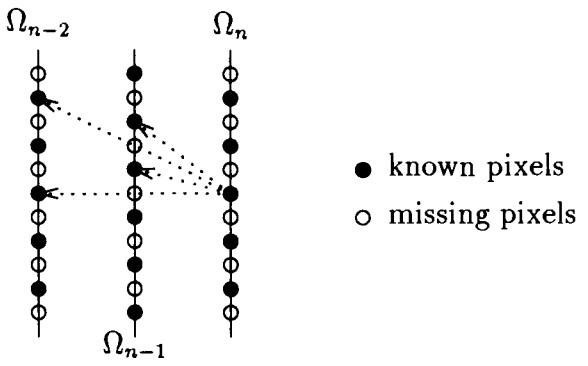


Figure 5.6: Diagram of the y-t plane of an interlaced sequence showing different displacement vectors.

5.7 Results

This section will present two sets of results using the methods for interlaced motion estimation described in this chapter. The first set will be the quality of interpolation attained using each method. The second set will be results from performing motion compensation using the interpolated fields. In order to simplify the descriptions in this section, the labels shown in Table 5.1 will be used to refer to the different interpolation methods.

Table 5.2 shows the parameter values for the results presented in this section. Note that while some work was done to find good values for these parameters, it is likely that better performance could be attained by studying the effects of each parameter in detail. The improvement resulting from the "fine-tuning" of these parameters is not likely to be significant. The work in this thesis, however, is intended to compare the different algorithms of motion estimation and therefore, consistency in the parameter values is more important than absolute values.

5.7.1 Interpolation Results

As discussed in Chapter 1, interpolation of the missing lines in an interlaced sequence is important for many different applications. In this section, interpolation using the motion estimation methods in this chapter will be compared with each other and with the spatial linear interpolation method described in chapter 2. Detailed results will be

Table 5.1: Terminology for various motion estimation methods used in the results presented in this chapter.

Method	**Label**	**Section**
Linear Spatial Interpolation	Method 1	2.1
Pixel Prediction Along Linear Trajectories	Method 2	5.4
Variable Previous Frame Motion Estimation	Method 3	5.5
Modified Variable Previous Frame Motion Estimation	Method 4	5.5.1
Dual Frame Motion Estimation	Method 5	5.6

Table 5.2: Parameters for methods 2 through 5 for the results presented in this chapter.

Parameter	Method 2	Method 3	Method 4	Method 5
T_0	5.0	5.0	5.0	5.0
a	0.9	0.9	0.9	0.9
$\lambda_{\mathbf{d}}$	0.3	0.3	0.3	0.3
λ_g	0.0025	0.0025	0.0025	0.0025
Iterations	100	100	100	100

presented for the *pongi* sequence along with limited results for the *caltrain* and *missa* sequences. This is mainly because the results for the *pongi* sequence clearly show the potential improvement from Bayesian motion estimation.

Motion Compensated Interpolation

Motion compensated interpolation is the interpolation of missing pixels using an estimated of the motion. In this chapter, several methods of estimating motion for the missing pixels in an interlaced video sequence have been presented. Once these motion estimates have been obtained, there are several methods of implementing motion compensated interpolation such as motion adaptive filtering (Delogne, Cuvelier, Maison, Caillie, and Vandendorpe 1993), motion compensated filtering (Dubois 1992), adaptive motion compensated interpolation (Weston 1988), and motion-compensating field interpolation (Girod and Thoma 1985).

In this thesis, we are limited to causal motion-compensated interpolation since the main application is video coding and the decoder will not, in general, have access to future frames/fields of the video sequence. Therefore, for the purposes of comparison of the motion estimation methods, the following simple interpolation method will be

used, where the interpolated pixel value, $\hat{g}(\mathbf{x}_i, t_+)$, is given by:

$$\hat{g}(\mathbf{x}_i, t_+) = \tilde{g}(\mathbf{x}_i - \hat{\mathbf{d}}(\mathbf{x}_i, t), t_-) \tag{5.6}$$

This means that the estimate of the missing pixel is just taken to be the value of the pixel in the previous frame along the estimate of the missing pixel's motion vector.

Interpolation of the *pongi* sequence

The *pongi* sequence is a difficult sequence to interpolate accurately for a number of reasons:

1. The camera is panning horizontally, which makes motion estimation more difficult.
2. The poster in the background has high frequency detail including several places where single-pixel-wide dark horizontal lines exist on a light background.
3. The wall in the background has a "random" texture with pixel amplitudes varying over an approximate range between 120 and 190.
4. Some of the motion (player's arm, spectators' heads and hands) is not translation.
5. Motion of the ball is large between successive frames (approximately 50 horizontal pixels).

For these reasons, the interpolation error in the pongi sequence is higher than in other sequences and there is more potential for improvement by using more complex methods of motion estimation.

Figure 5.7 shows the performance of the five methods on the pongi sequence. On average, all four Bayesian algorithms (methods 2–5) are better than the best spatial interpolation algorithm, linear interpolation (method 1). The best algorithm is method 4, which is 54–70% better than method 1 (best spatial method), and is 15-30% better than method 5 (next best Bayesian method).

Figures 5.8, 5.9, 5.10, 5.11, 5.12, and 5.13 show the actual second frame of the *pongi* sequence along with the reconstructed second frame using methods 1, 2, 3, 4,

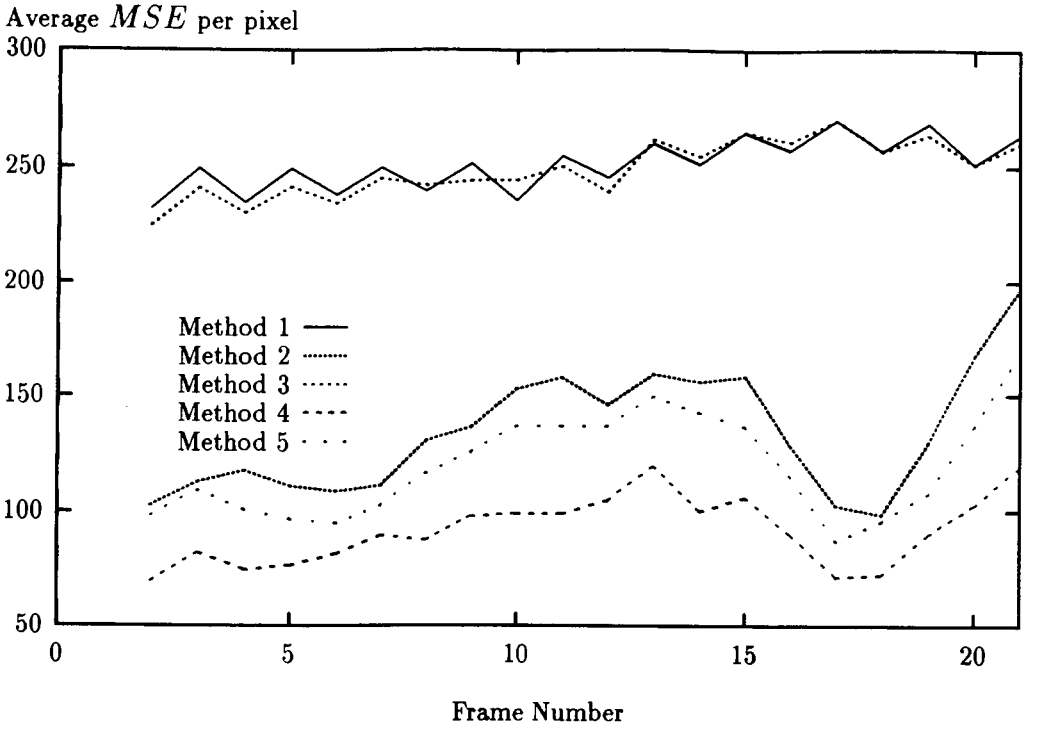


Figure 5.7: Performance of motion-estimated interpolation methods on the *pongi* sequence.

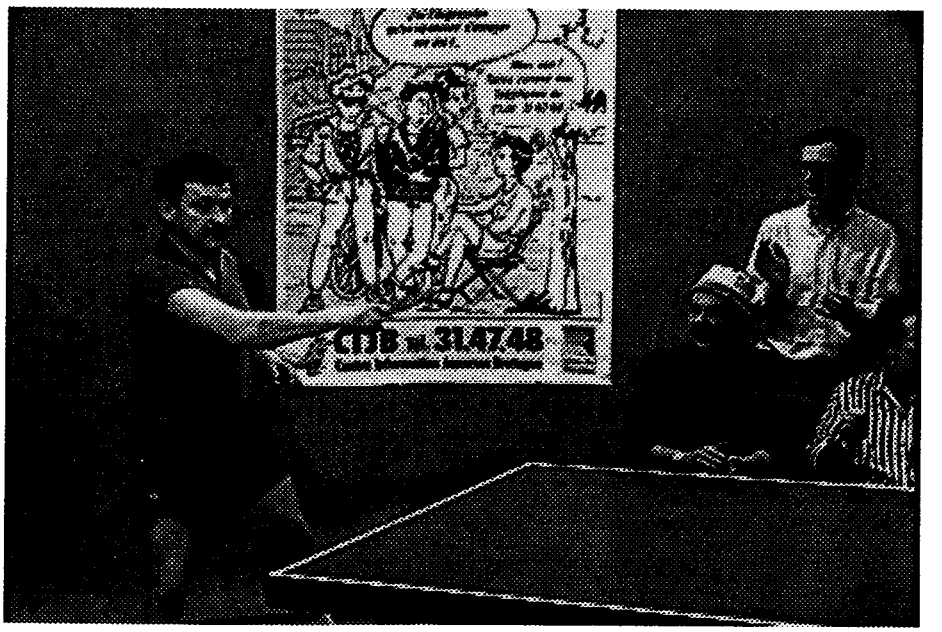


Figure 5.8: Actual second frame of the *pongi* sequence.

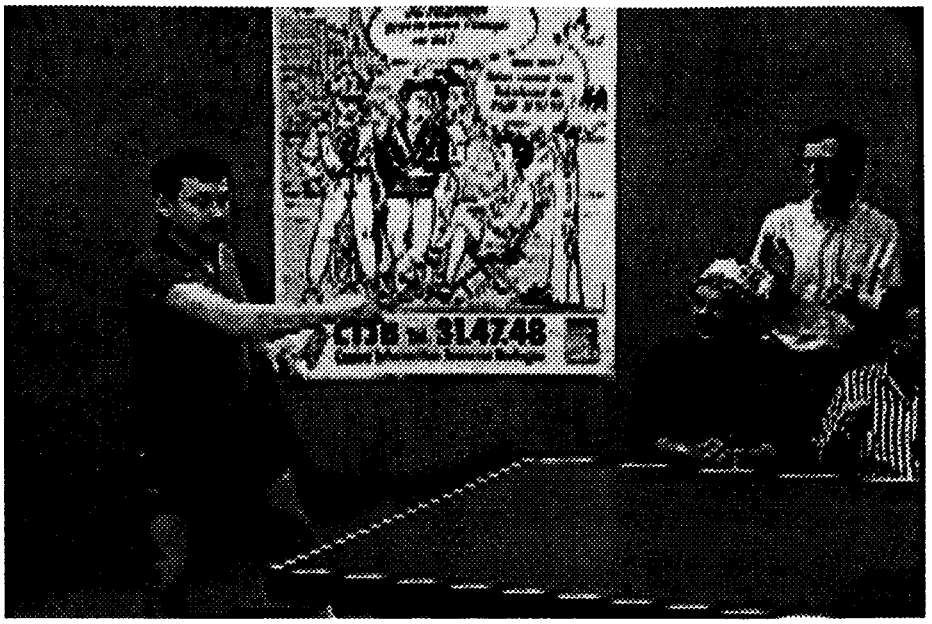


Figure 5.9: Reconstructed second frame of the *pongi* sequence using method 1.

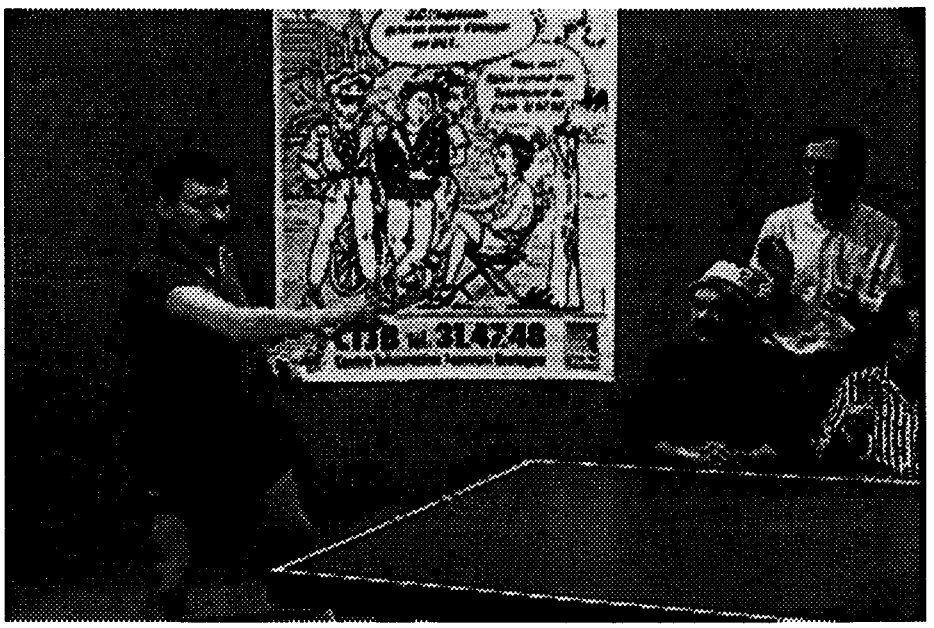


Figure 5.10: Reconstructed second frame of the *pongi* sequence using method 2.

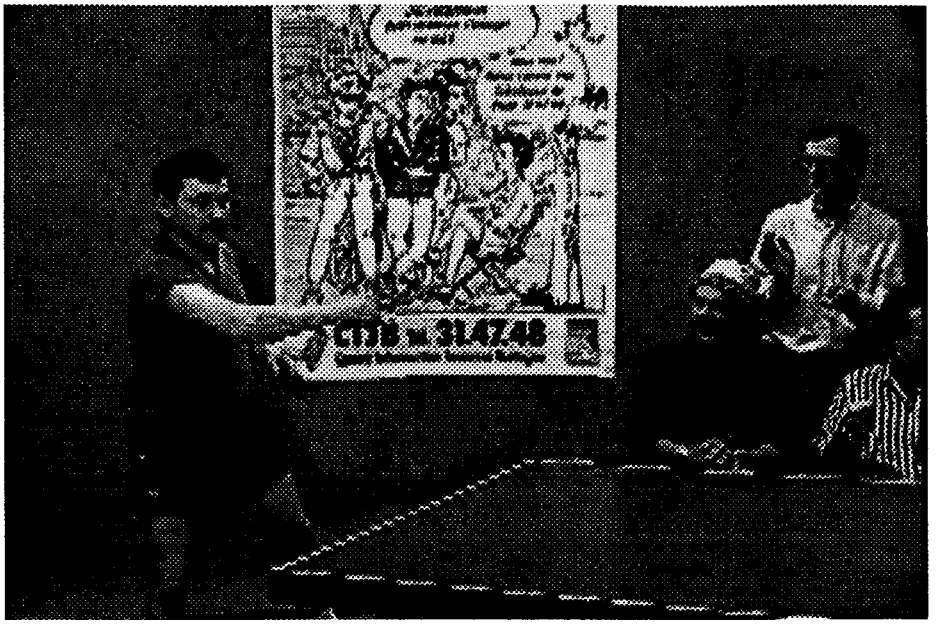


Figure 5.11: Reconstructed second frame of the *pongi* sequence using method 3.

Figure 5.12: Reconstructed second frame of the *pongi* sequence using method 4.

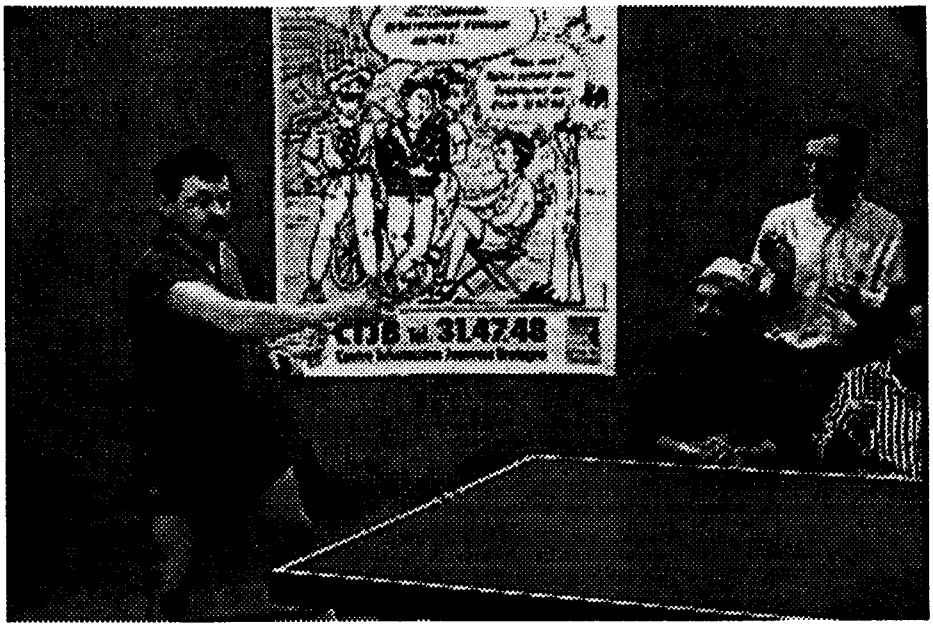


Figure 5.13: Reconstructed second frame of the *pongi* sequence using method 5.

and 5 respectively. The difference in quality is quite clear when comparing the three best methods (2, 4, and 5) with the two worst (1 and 3).

A number of observations can be made regarding the reconstructed frames presented here:

1. Object edges are significantly improved by using the motion estimation methods (2, 3, 4, and 5) over the spatial method (1).

2. The modification made between method 3 and method 4 (section 5.5.1) significantly improves the interpolation of the high detail background areas such as the poster and the table edge in the *pongi* sequence. This results from improved estimation of the panning motion.

3. The background wall appears low-pass-filtered in method 1, whereas in methods 2, 3, 4, and 5, the texture is sharper.

4. In all cases, the areas of non-translatory motion contain noticeable interpolation error. While this degrades the characteristics of individual frames, it is not as noticeable when the reconstructed sequence is played.

5. In the region of the ball, method 1 performs better than methods 2, 3, 4, and 5. This is because the ball moves a greater distance than is contained in the sample space, $\mathcal{S}_{\mathbf{d}}$, of the Bayesian methods.

Interpolation of the *caltrain* sequence

The *caltrain* sequence has a number of characteristics that make it an easier sequence to interpolate than the *pongi* sequence:

1. The background of the sequence (wall-paper) is composed of homogeneous regions (objects and animals) that contrast with a light background. The homogeneous regions are easily interpolated; however, the object edges introduce difficulties.

2. Non-translatory motion exists in a number of regions in the sequence including the rolling ball and the mobile on the left of the sequence.

3. High frequency detail exists in the calendar; however, the detail in the original sequence is blurred reducing the noticeable error in this area.

In order to compare the Bayesian methods with the spatial methods, only the results from the two best methods (1 and 4) are presented here. Figure 5.14 shows the performance of methods 1 (best spatial method) and method 4 (best Bayesian method) on the *caltrain* sequence. While method 4 ranges from 29% better to 10% worse than method 1, the difference in MSE is not really significant. What are significant are the subjective results as shown in Figures 5.15 (actual frame 2), 5.16 (method 1), and 5.17. While these three figures are not as clear as when shown on the computer screen, they do demonstrate the differences especially at high contrast object edges such as the white mobile in the bottom left section of the frame. The error frames for method 1 (Figure 5.18) and method 4 (Figure 5.19) are also shown.

Once again, a number of observations about the *caltrain* sequence results can be made:

1. Clearly in the homogeneous regions of the sequence, both interpolation methods work reasonably well. There are, however, a number of small regions where method 4 does not estimate the motion well and the result are thinly stripped

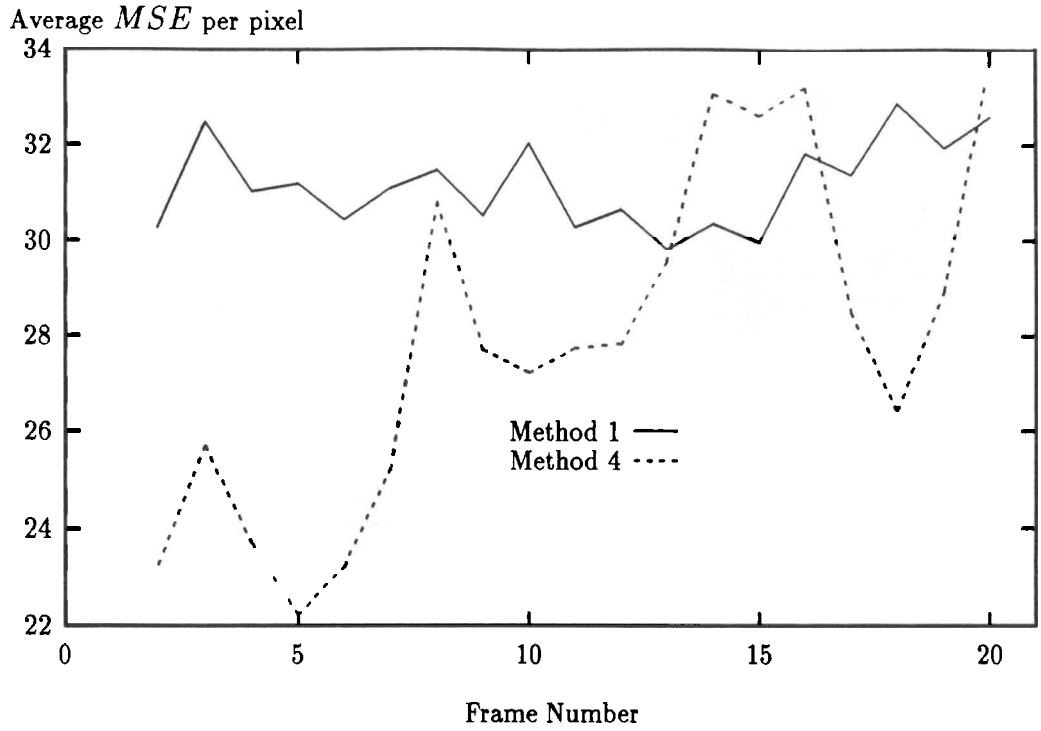


Figure 5.14: Performance of motion-estimated interpolation methods on the *caltrain* sequence.

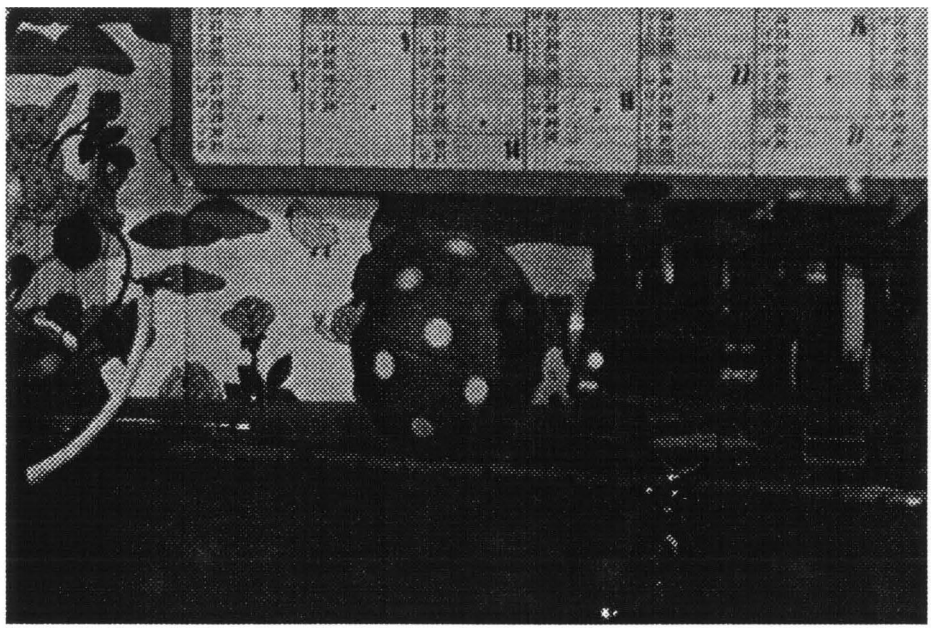


Figure 5.15: Actual second frame of the *caltrain* sequence.

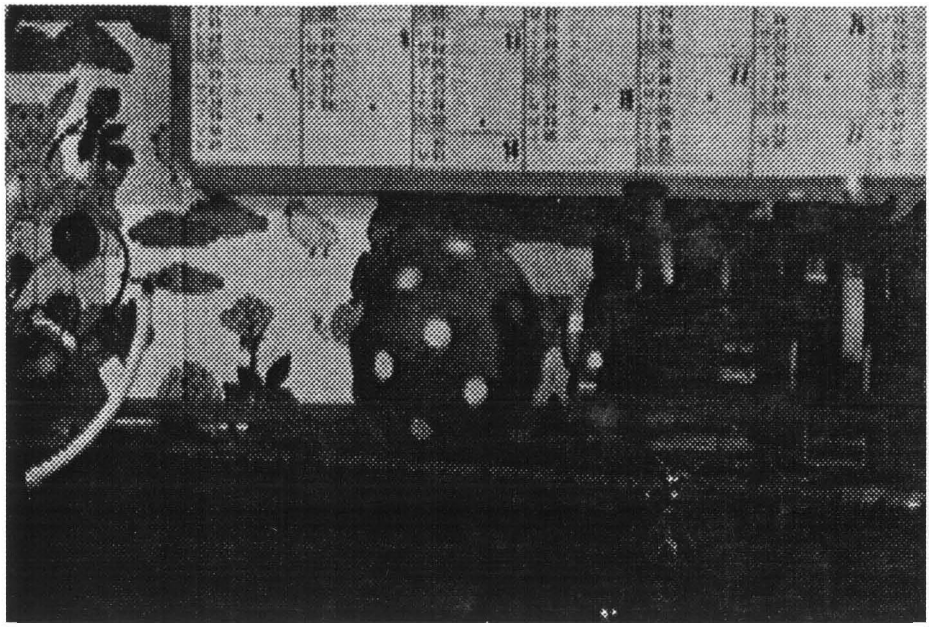


Figure 5.16: Reconstructed second frame of the *caltrain* sequence using method 1.

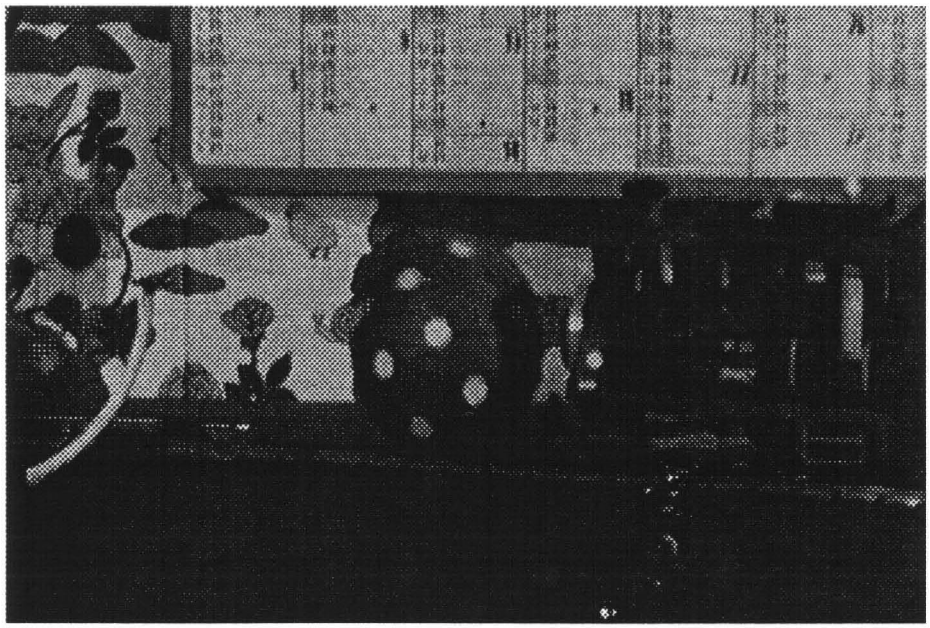


Figure 5.17: Reconstructed second frame of the *caltrain* sequence using method 4.

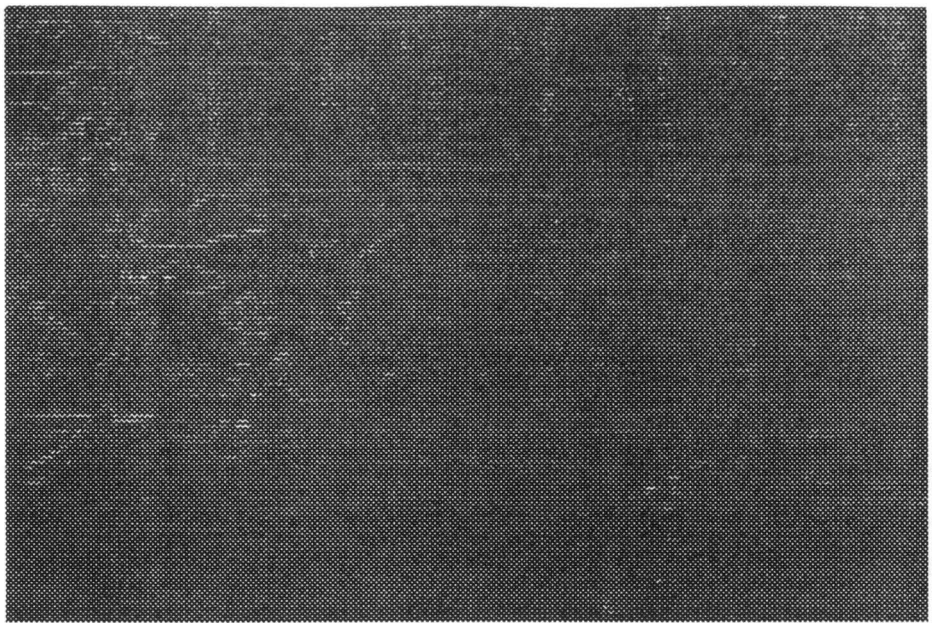


Figure 5.18: Error frame for the reconstruction of the second frame of the *caltrain* sequence using method 1.

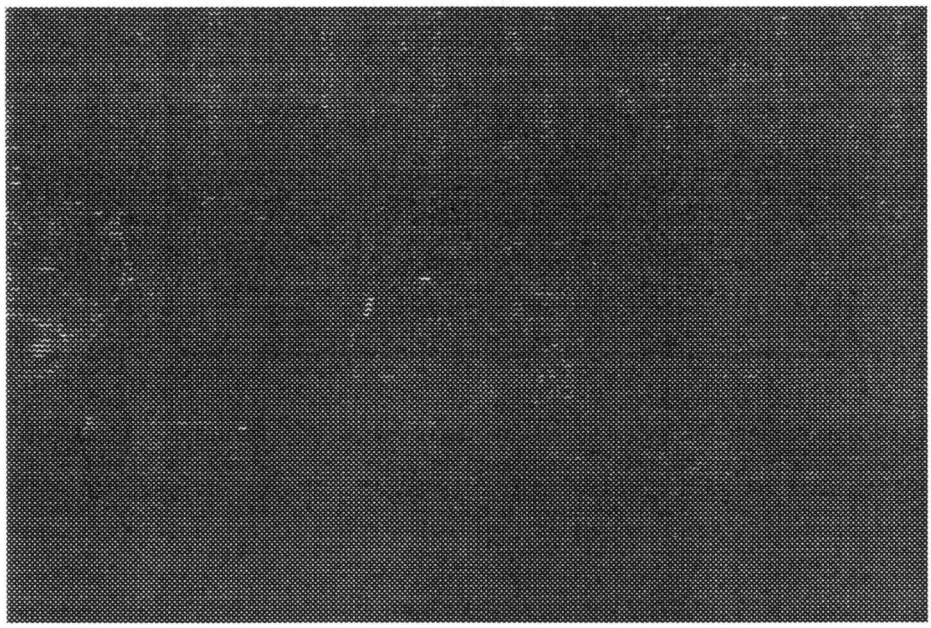


Figure 5.19: Error frame for the reconstruction of the second frame of the *caltrain* sequence using method 4.

patches where the interpolated lines are not well correlated with the known lines. These patches are not present in the method 1 results.

2. Object edges are significantly better in the method 4 results. This can easily be seen on the left side of the error frames (Figures 5.18 and 5.19).

3. Despite the non-translatory motion, the interpolation is acceptable. This is because non-translatory motion can be often be approximated well as translatory motion between successive frames. There are, however, some "peaky" errors in method 4 in the spots on the ball.

4. Errors in the calendar region appear to be approximately equal in both methods. This is because the high frequency detail does not allow good matches between successive frames in method 4. In method 1, the low-pass filtering effect of the linear interpolation introduces inherent errors in high spatial frequency areas.

Interpolation of the *missa* sequence

The *missa* sequence is a typical video-conferencing type of sequence. It contains little high spatial frequency detail and small amounts of motion (only in the mouth and face area). The average MSE per pixel data is shown in Figure 5.20 for methods 1 and 4. Clearly the results are almost identical for both methods. The differences are insignificant and are not noticeable in the resulting reconstructed frames. Note that the "zig-zag" effect is caused by information which is lost in even frames and is unrecoverable by either of these methods. Reconstruction using a more complicated interpolation scheme (see section 5.7.1) could reduce this effect.

5.7.2 Motion Compensation Results

In this section, results will be presented from the application of the motion estimation and interpolation techniques discussed in this chapter to three different motion compensation methods.

Two Field Motion Compensation

In this motion compensation algorithm, two fields are searched:

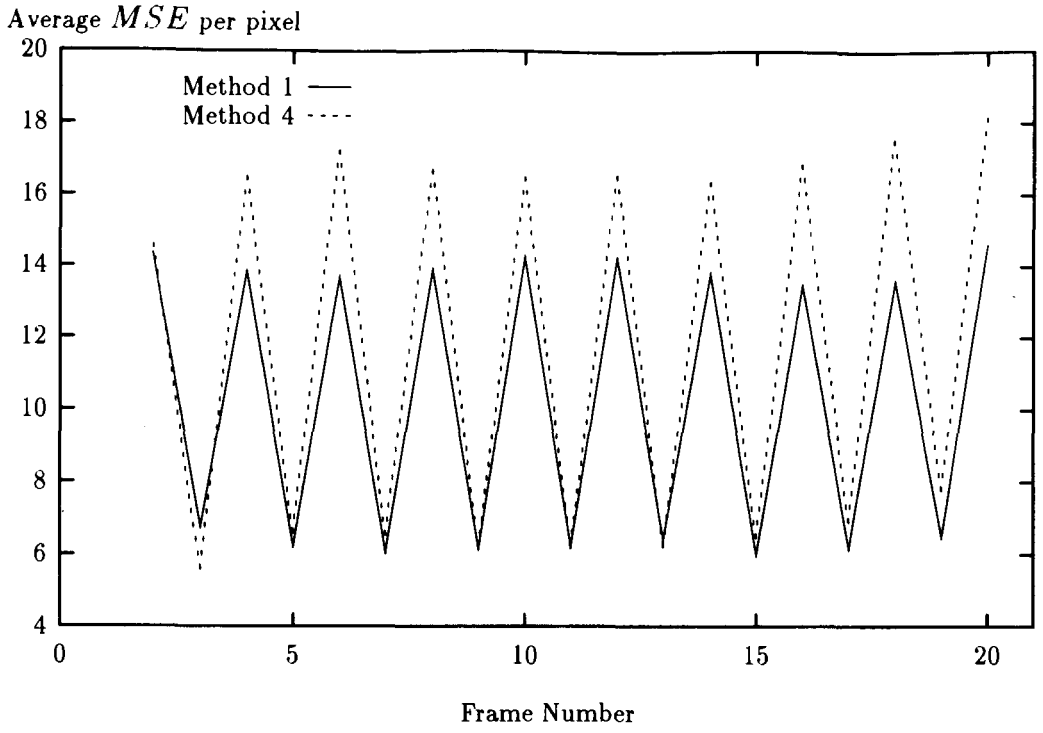


Figure 5.20: Performance of motion-estimated interpolation methods

1. Ω_{n-1} – the previous field (different parity from the current field (Ω_n).

2. $\Omega_{n-1}^{(r)}$ – the interpolated previous field (same parity as Ω_n).

Ideally if the interpolation is sufficiently good, this method should provide excellent results. Figure 5.21 shows the performance of method 1 (spatial interpolation) and method 4 (best Bayesian interpolation) on the *pongi* sequence. As expected, the improved interpolation resulting from method 4 results in significantly lower energy (MSE) than method 1. The order of results for the *caltrain* sequence (Figure 5.22) is the same, despite the fact that method 1 and method 4 had very similar quantitative interpolation errors.

Figure 5.23 shows the performance of the Bayesian motion compensated interpolation methods when applied to two field motion compensation. Method 4 is slightly superior to methods 2, and 5 reflecting the order of interpolation quality. Method 3 is not shown because it is significantly worse than methods 2, 4 and 5.

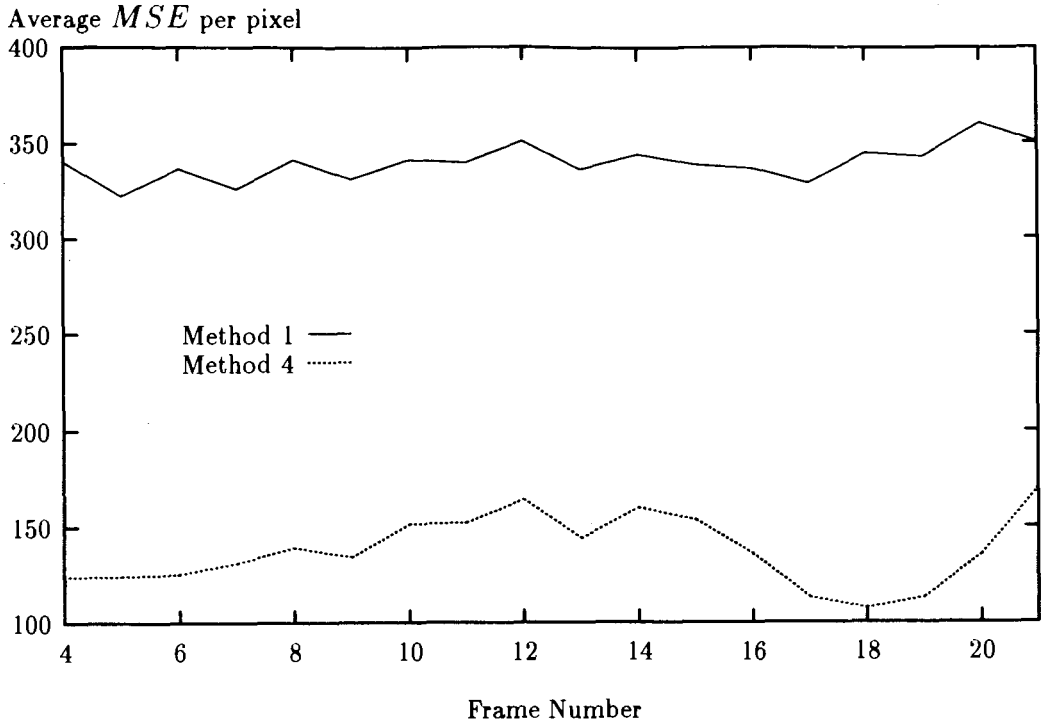


Figure 5.21: Performance of spatial method 1 and Bayesian method 4 using two field motion compensation for the *pongi* sequence

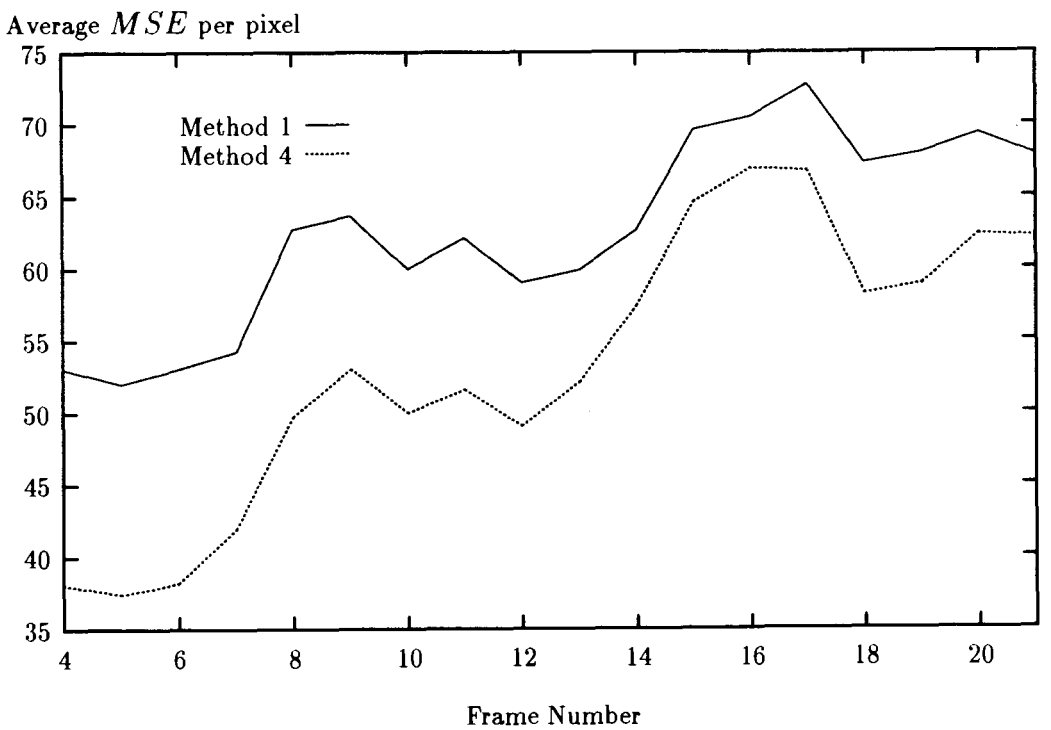


Figure 5.22: Performance of spatial method 1 and Bayesian method 4 using two field motion compensation for the *caltrain* sequence

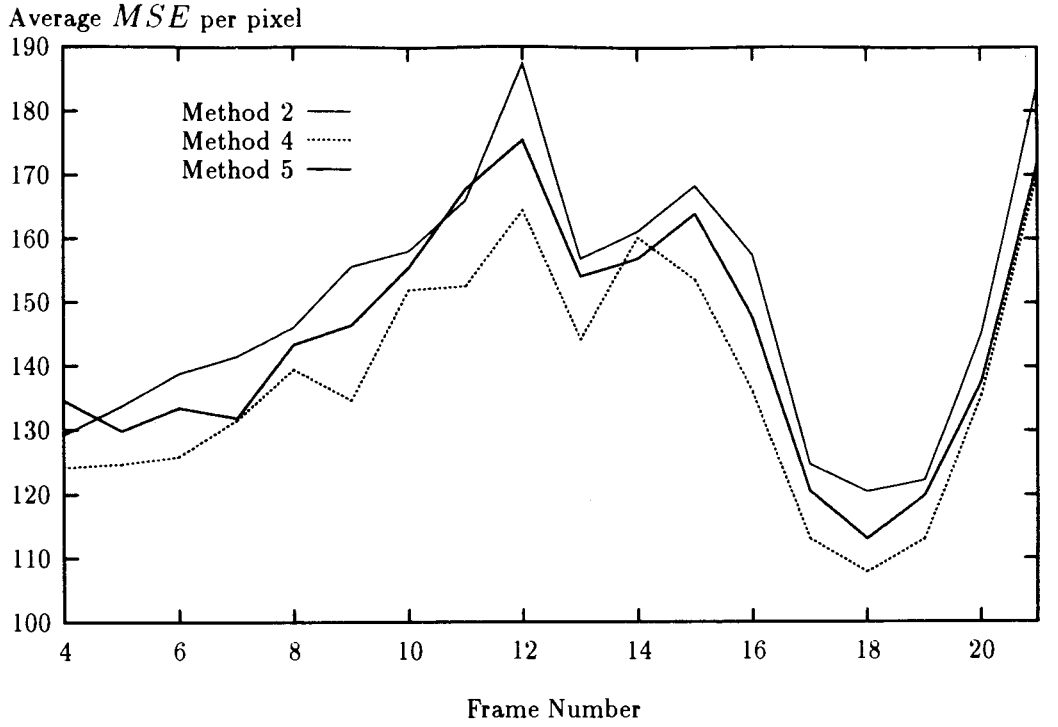


Figure 5.23: Performance of various Bayesian interpolation methods using two field motion compensation for the *pongi* sequence

Three Field Motion Compensation

The second algorithm applied to the interpolation results in this chapter is similar to the two field algorithm described above; however, when the y-component of the candidate block motion vector is zero, the block that is searched is located in the second frame back, Ω_{n-2}. The rational for this modification is the same as in section 5.5.1. Note that this modification does not require any extra information to be sent to the decoder, since, whenever the y-component of the transmitted motion vector is zero, the decoder will simply use Ω_{n-2} in the reconstruction process.

The results of this MC algorithm for the *pongi* sequence are shown in Figure 5.24 (note the change in scale from Figure 5.21). As can be seen from the figure, there is very little difference between the five methods with the addition of the second field. This result is repeated in Figure 5.25 for the *caltrain* sequence. These results show the importance of using the second previous field (Ω_{n-2}) in the motion compensation of interlaced sequences. In fact, the simpler spatial interpolation (method 1) performs slightly better than the more complex Bayesian methods. This is because matches for

the stationary and background regions are found primarily in Ω_{n-2} so the performance difference occurs in the motion regions. While these regions are in general estimated better by the Bayesian algorithms, there are a few pixels that have a large estimation error. The spatial algorithm (method 1), however, performs moderately well in all areas leading to no large errors. Overall, these characteristics of the two types of algorithm result in a slightly better MSE performance for the spatial algorithm and a much lower cost. From these results and conclusions, better results could be obtained on certain very complex, high motion sequences from the Bayesian methods, but for the majority of sequences, the spatial interpolation will perform well.

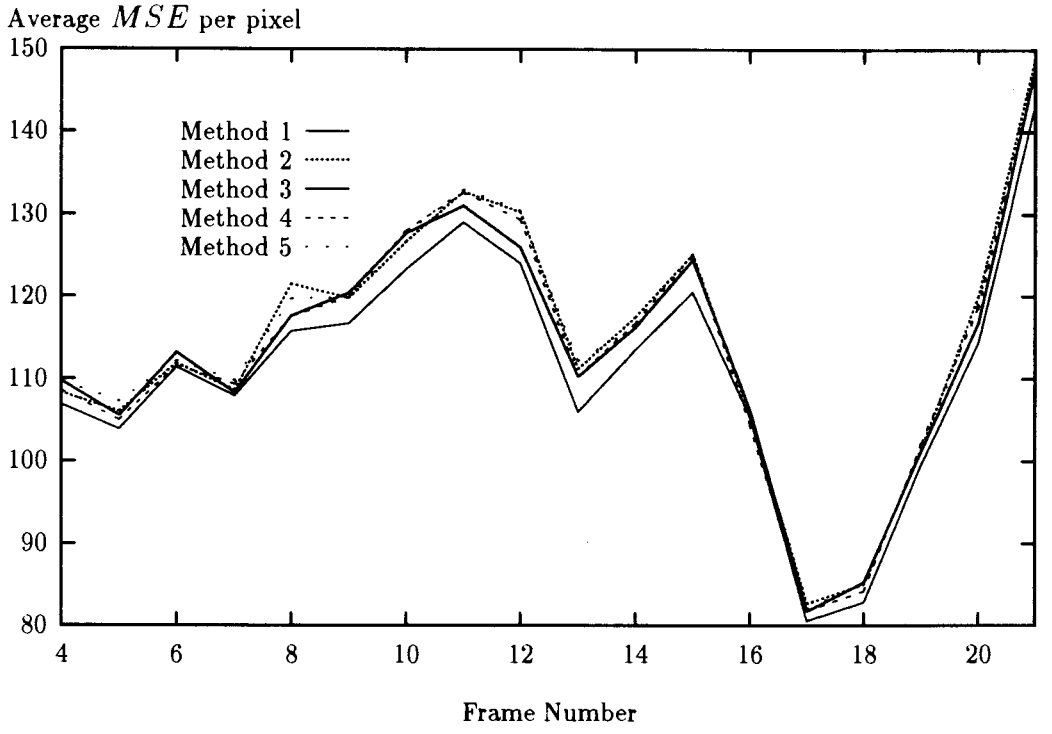


Figure 5.24: Performance of various interpolation methods using three field motion compensation for the *pongi* sequence

Another very recent approach that uses similar ideas to those of this chapter to perform adaptive de-interlacing for the purpose motion compensation of interlaced sequences, was developed by Zaccarin and Liu (1993) (published in May, 1994). This approach combines the motion compensated interpolation with the spatial interpolation using a line correlation technique. Application of such a technique using the Bayesian motion estimation methods given here, could improve interpolation by eliminating the "peaky" errors found in the Bayesian methods (see earlier discussion in

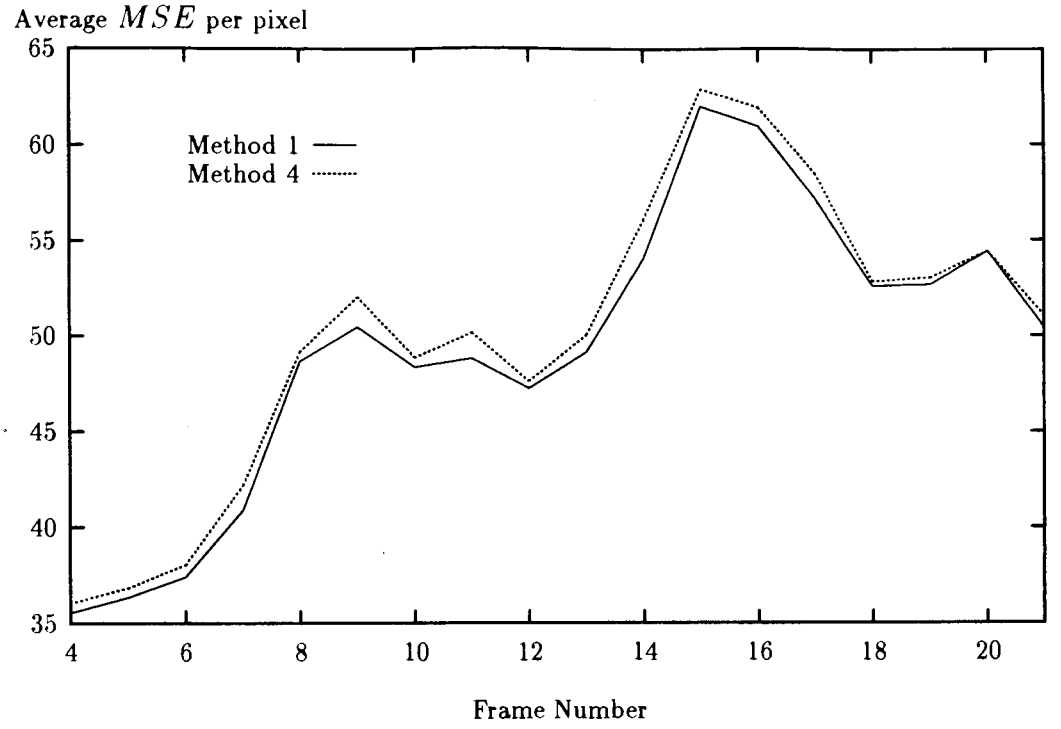


Figure 5.25: Performance of methods 1 and 4 using three field motion compensation for the *caltrain* sequence

this section). This improved interpolation has the potential to improve the motion compensation of interlaced sequences.

In addition, Zaccarin and Liu (1993) do not directly search Ω_{n-2} which has proven to provide significant improvement (three field motion compensation). Improvements in their results would be expected if they included Ω_{n-2}.

Hybrid Motion Compensation

Hybrid motion compensation was described in 3.4 and proved to be the best method of motion compensation from Chapter 3. The results when different interpolation methods are applied to hybrid motion compensation, are shown in Figure 5.26 for the *pongi* sequence (five methods) and Figure 5.27 for the *caltrain* sequence (two methods). In both cases, method 1 performs better than the Bayesian methods. Again, this is due to the more uniform error in the spatial interpolation (method 1).

It should also be noted that the hybrid MC algorithm performs better than other MC algorithms by approximately 10% for each of the interpolation methods tested.

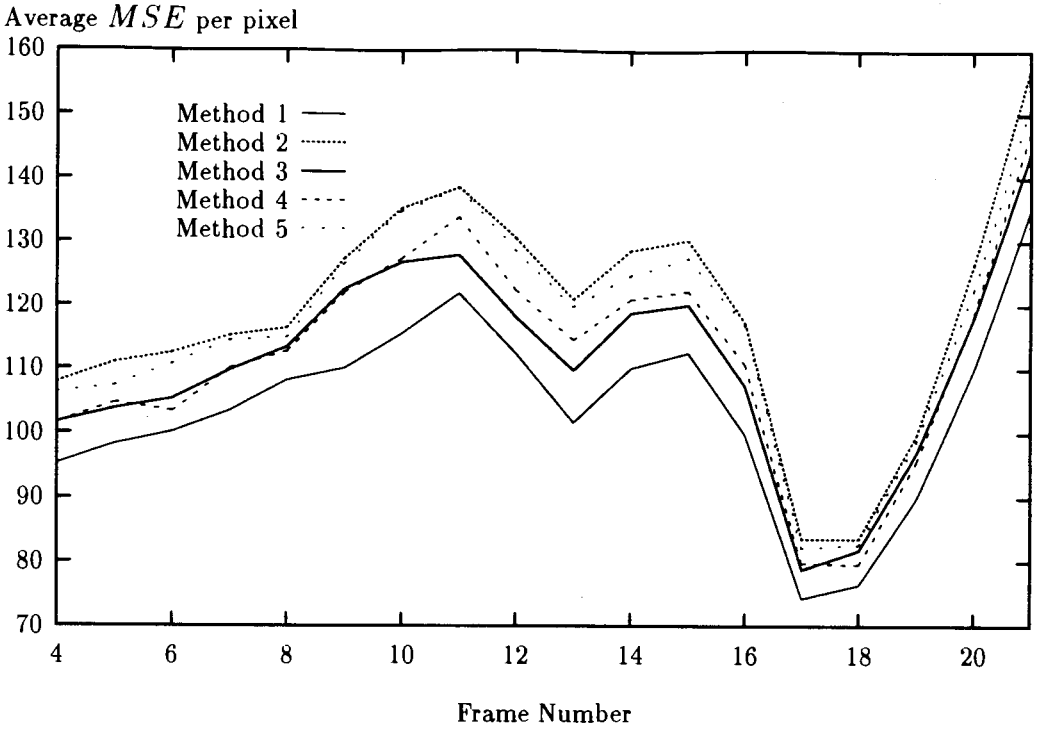


Figure 5.26: Performance of various interpolation methods using hybrid motion compensation for the *pongi* sequence

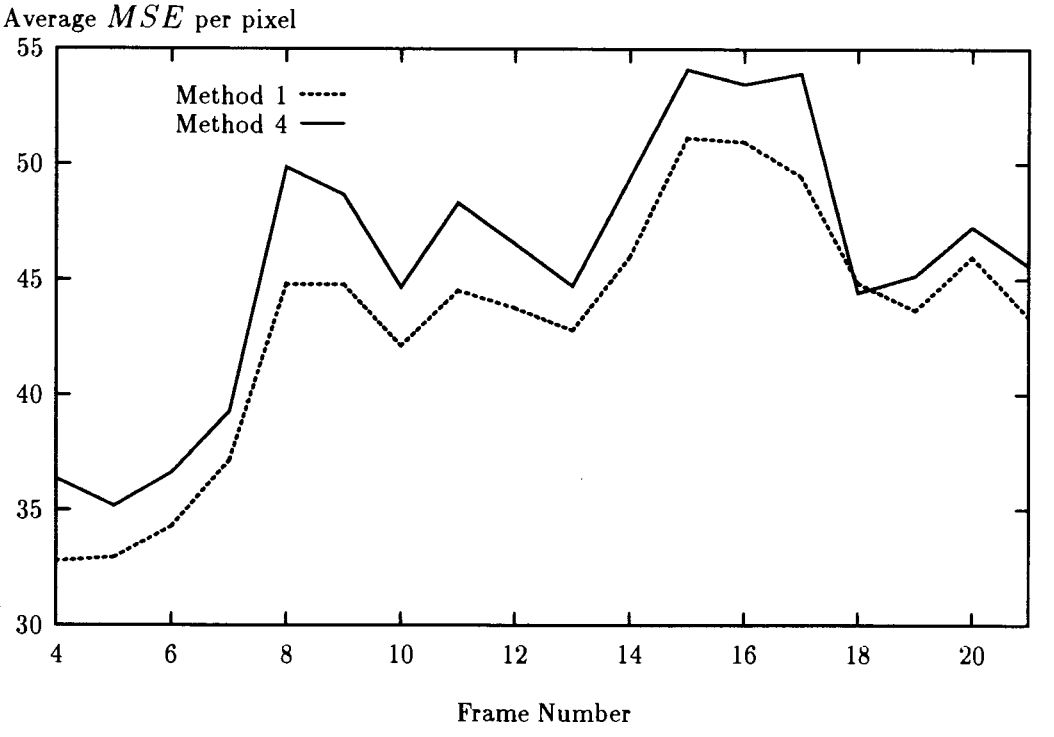
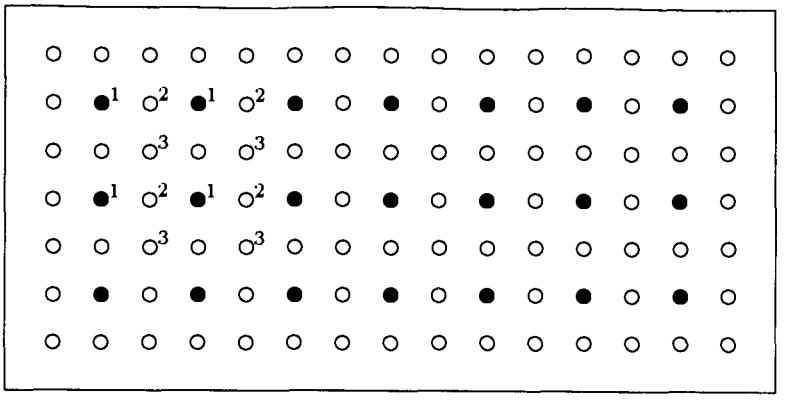


Figure 5.27: Performance of methods 1 and 4 using hybrid motion compensation for the *caltrain* sequence



● –integer locations
○ –fractional locations
1 –original block location ($\mathbf{d} = (0,0)$)
2 –horizontal fractional motion vector ($\mathbf{d} = (\frac{1}{2},0)$)
3 –diagonal fractional motion vector ($\mathbf{d} = (\frac{1}{2},\frac{1}{2})$)

Figure 5.28: Fractional motion compensation

Motion Compensation with Fractional Pixel Accuracy

Girod (1993) has described a technique for vector refinement using fractional pixel accuracy. A similar technique is used in the MPEG-2 standard. Initially, an integer vector is found for each block using the block matching method described in chapter 1. Starting with this vector, all displacements of $\pm\frac{1}{2}$ in both the horizontal and vertical directions are searched using pixel values on a fractional sampling grid. Figure 5.28 shows this process for a 2×2 block. The pixels labelled "1" compose the block located at the best integer vector. These pixels are on the integer sampling grid in the frame. The pixels labelled "2" are located at a displacement of $(\frac{1}{2},0)$ from the integer vector while those labelled "3" are located at a displacement of $(\frac{1}{2},\frac{1}{2})$. Displacements of size $\frac{1}{4}$ pixel are then searched around the best $\frac{1}{2}$ pixel displacement. The process can be continued with displacements of size 2^{-n} until the desired accuracy is obtained.

For the purposes of this thesis, the fractional sampling grid values are generated using linear spatial interpolation. Therefore, for a pixel at location $(x+\frac{1}{2},y)$, its value is given by

$$\Psi(x+\frac{1}{2},y) = \frac{1}{2}\Psi(x,y) + \frac{1}{2}\Psi(x+1,y) \tag{5.7}$$

and a pixel value at location $(x+\frac{1}{2}, y+\frac{1}{2})$ is given by

$$\Psi(x+\frac{1}{2}, y+\frac{1}{2}) = \frac{1}{4}\Psi(x,y) + \frac{1}{4}\Psi(x+1,y) + \frac{1}{4}\Psi(x,y+1) + \frac{1}{4}\Psi(x+1,y+1) \quad (5.8)$$

Other methods such as Wiener filtering can also be used (Girod 1993).

This fractional pixel MC algorithm was applied to both two-field MC and three-field MC (as described above) using the interpolation methods listed in table 5.1. The results are shown for methods 1 and 4 in Figures 5.29 (*pongi* sequence) and 5.30 (*caltrain* sequence). Note that the results for two-field MC are not shown for method 1 since they were significantly worse than those shown. Note that in the case of the *pongi* sequence, the three-field algorithm is superior, while in the *caltrain* sequence, the three curves shown cross several times with insignificant differences between them.

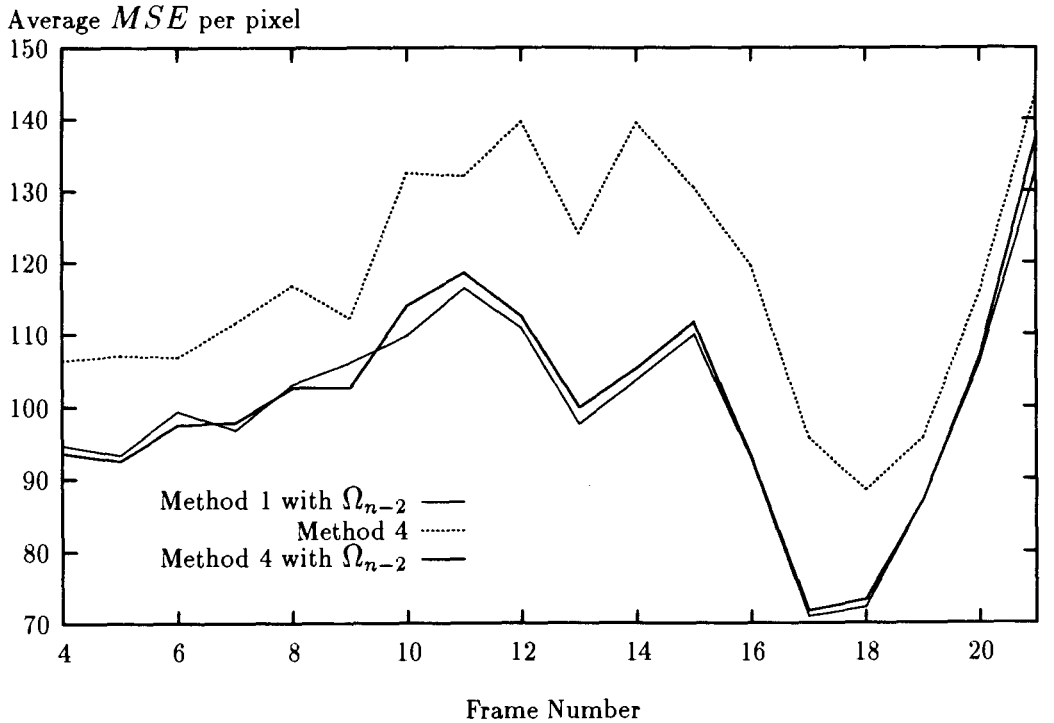


Figure 5.29: Performance of various interpolation methods using fractional pixel motion compensation for the *pongi* sequence

An important observation from Figures 5.29 (*pongi* sequence) and 5.30 (*caltrain* sequence) is that with fractional pixel MC the three-field curves for methods 1 and 4 are closer than without fractional pixel MC. This is as expected, because method 4 is provides a better interpolation of each frame and therefore the fractional offsets are also more accurate.

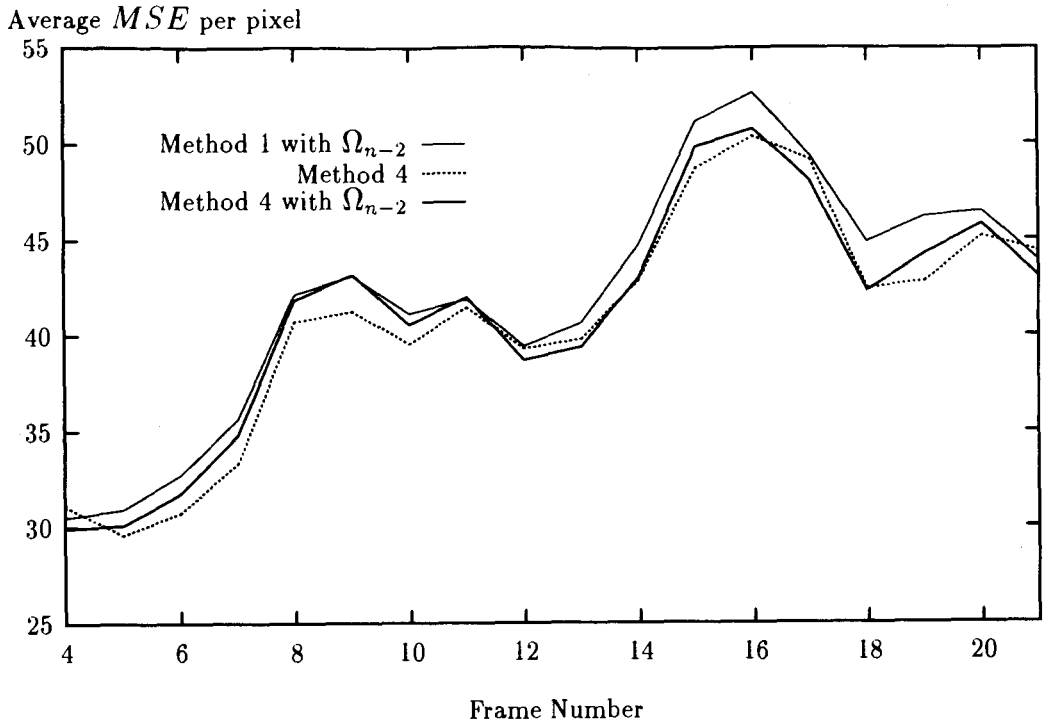


Figure 5.30: Performance of various interpolation methods using fractional pixel motion compensation for the *caltrain* sequence

While fractional motion compensation does improve the performance (approximately 10% in our results), consideration has to be given to the additional computational cost associated with it. However, since fractional motion compensation has been included in the MPEG-2 standard, the consensus is that the additional computation is acceptable.

Chapter 6

Reduced Complexity Motion Estimation

There are numerous methods of estimating motion in a video sequence. A good overview of motion estimation techniques is given by Aggarwal and Nandhakumar (1988). In this chapter, two constraint-based iterative techniques will be investigated. These two techniques were chosen since they are intended to estimate the true motion in the sequence, which is important for the purposes of motion compensated interpolation of interlaced sequence. Both methods have been applied to interlaced sequences and the resulting interpolated fields have been used in the motion compensation algorithms discussed in previous chapters.

In addition, a survey of several other types of motion estimation methods will be presented.

6.1 Determining Optical Flow

One of the more well known methods of estimating optical flow (motion) was developed by Horn and Schunck (1981). The method is based on the assumption that the image intensity, $g(\mathbf{x}, t)$, as a function of spatial position, $\mathbf{x}$ and time, t, is independent of time so that

$$\frac{\mathrm{d}g(\mathbf{x} - \mathbf{d}(\mathbf{x}, t), t)}{\mathrm{d}t} = 0 \tag{6.1}$$

This assumption is the same as that found in the structural model described in section 4.3.1. By using the chain rule for differentiation, a single linear equation with two unknowns can be derived as

$$g_x u + g_y v + g_t = 0 \tag{6.2}$$

where g_x, g_y, and g_t are the partial derivatives of $g(\mathbf{x}, t)$ with respect to x, y, and t respectively and u and v are the velocity components (x-component and y-component of the displacement vector, $\mathbf{d}(\mathbf{x}, t), t)$, respectively).

The second assumption that is made is that the motion field is smooth. This can be obtained by attempting to minimize the square of the magnitude of the gradient of the optical flow velocity, which is given by

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 \text{ and } \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{6.3}$$

By forming two error terms: the rate of change of image intensity,

$$\varepsilon_b = g_x u + g_y v + g_t \tag{6.4}$$

and the smoothness,

$$\varepsilon_c^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{6.5}$$

and combining them together as the total error

$$\varepsilon^2 = \int\int (\alpha^2 \varepsilon_c^2 + \varepsilon_b^2)\, \mathrm{d}x\mathrm{d}y \tag{6.6}$$

the following iterative solution is obtained

$$u^{n+1} = \overline{u}^n - g_x \frac{g_x \overline{u}^n + g_y \overline{v}^n + g_t}{\alpha^2 + g_x^2 + g_y^2} \tag{6.7}$$

$$v^{n+1} = \overline{v}^n - g_y \frac{g_x \overline{u}^n + g_y \overline{v}^n + g_t}{\alpha^2 + g_x^2 + g_y^2} \tag{6.8}$$

Estimates of g_x, g_y, and g_t are obtained at the center of the cube of intensity samples shown in Figure 6.1. The estimates are then given by

$$\begin{aligned} g_x \approx \frac{1}{4}[&g_{i,j+1,k} - g_{i,j,k} + g_{i+1,j+1,k} - g_{i+1,j,k} + \\ &g_{i,j+1,k+1} - g_{i,j,k+1} + g_{i+1,j+1,k+1} - g_{i+1,j,k+1}] \end{aligned} \tag{6.9}$$

$$g_y \approx \frac{1}{4}[g_{i+1,j,k} - g_{i,j,k} + g_{i+1,j+1,k} - g_{i,j+1,k} + g_{i+1,j,k+1} - g_{i,j,k+1} + g_{i+1,j+1,k+1} - g_{i,j+1,k+1}] \quad (6.10)$$

$$g_t \approx \frac{1}{4}[g_{i,j,k+1} - g_{i,j,k} + g_{i+1,j,k+1} - g_{i+1,j,k} + g_{i,j+1,k+1} - g_{i,j+1,k} + g_{i+1,j+1,k+1} - g_{i+1,j+1,k}] \quad (6.11)$$

In addition, the local averages, $\overline{u}$ and $\overline{v}$, are defined as

$$\overline{u}_{i,j,k} = \frac{1}{6}[u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k}] + \frac{1}{12}[u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k}] \quad (6.12)$$

$$\overline{v}_{i,j,k} = \frac{1}{6}[v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k}] + \frac{1}{12}[v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k}] \quad (6.13)$$

The parameter, α^2, is a weighting factor between the two errors (Equations 6.4 and 6.5) and only really influences the algorithm in regions where the brightness gradient is small. This parameter should be approximately equal to the expected noise in the estimate of $g_x^2 + g_y^2$.

6.1.1 Results

This section presents the results obtained by using the algorithm described in section 6.1. Initially, the algorithm was applied to a modified version of the *comp* sequence in which the block only moves by $(+1, +1)$ between successive frames. The resulting motion field is shown in Figure 6.2. On this low-motion modified version of the *comp* sequence, the motion estimation is very good with errors primarily at the edges due to the newly exposed and hidden areas.

Figure 6.3 shows the motion field resulting from the normal *comp* sequence (displacement of block is $(+4, +4)$ between successive frames). As is clear, the algorithm does not perform well on higher motion sequences. This type of problem could be improved by using some type of hierarchical algorithm as discussed in section 6.3.3.

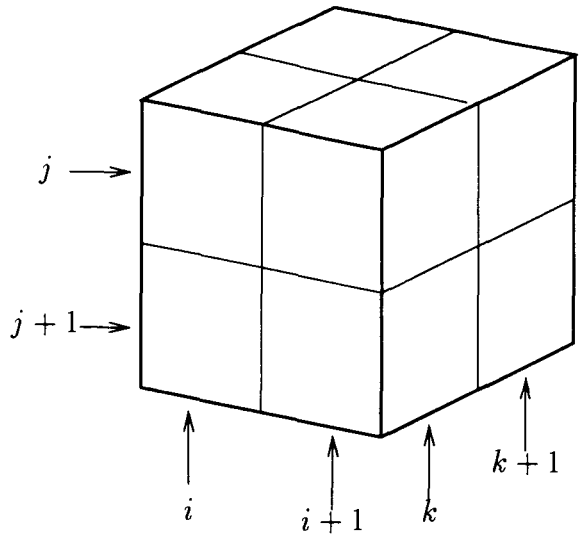


Figure 6.1: Sample location for estimation of the partial derivatives of the brightness

6.2 Gauss-Newton Minimization of the MAP Criterion

Konrad and Dubois (1991) have developed a deterministic solution for minimization of the MAP criterion by expanding the displaced pel difference using the first-order terms of the Taylor series. The estimation process is described by the following iterative update expression:

$$\hat{\mathbf{d}}^{n+1}(\mathbf{x}_i,t) = \bar{\mathbf{d}}^n(\mathbf{x}_i,t) - \frac{\varepsilon_i}{\mu_i}\nabla_{\mathbf{d}}^T\mathbf{r}(\bar{\mathbf{d}}^n(\mathbf{x}_i,t),\mathbf{x}_i,t,\Delta t) \tag{6.14}$$

where ε_i and μ_i are defined as

$$\varepsilon_i = \mathbf{r}(\bar{\mathbf{d}}^n(\mathbf{x}_i,t),\mathbf{x}_i,t,\Delta t) \tag{6.15}$$

and

$$\mu_i = 4\frac{\lambda_{\mathbf{d}}}{\lambda_g} + \parallel \nabla_{\mathbf{d}}\tilde{\mathbf{r}}(\bar{\mathbf{d}}^n(\mathbf{x}_i,t),\mathbf{x}_i,t,\Delta t) \parallel^2 \tag{6.16}$$

The average vector, $\bar{\mathbf{d}}^n(\mathbf{x}_i,t)$, computed over neighborhood, $\eta_{\mathbf{d}}(\mathbf{x}_i)$, is defined as

$$\bar{\mathbf{d}}^n(\mathbf{x}_i,t) = \frac{1}{\xi_i}\sum_{j:\mathbf{x}_i\in\eta_{\mathbf{d}}(\mathbf{x}_i)}\hat{\mathbf{d}}(\mathbf{x}_j,t) \tag{6.17}$$

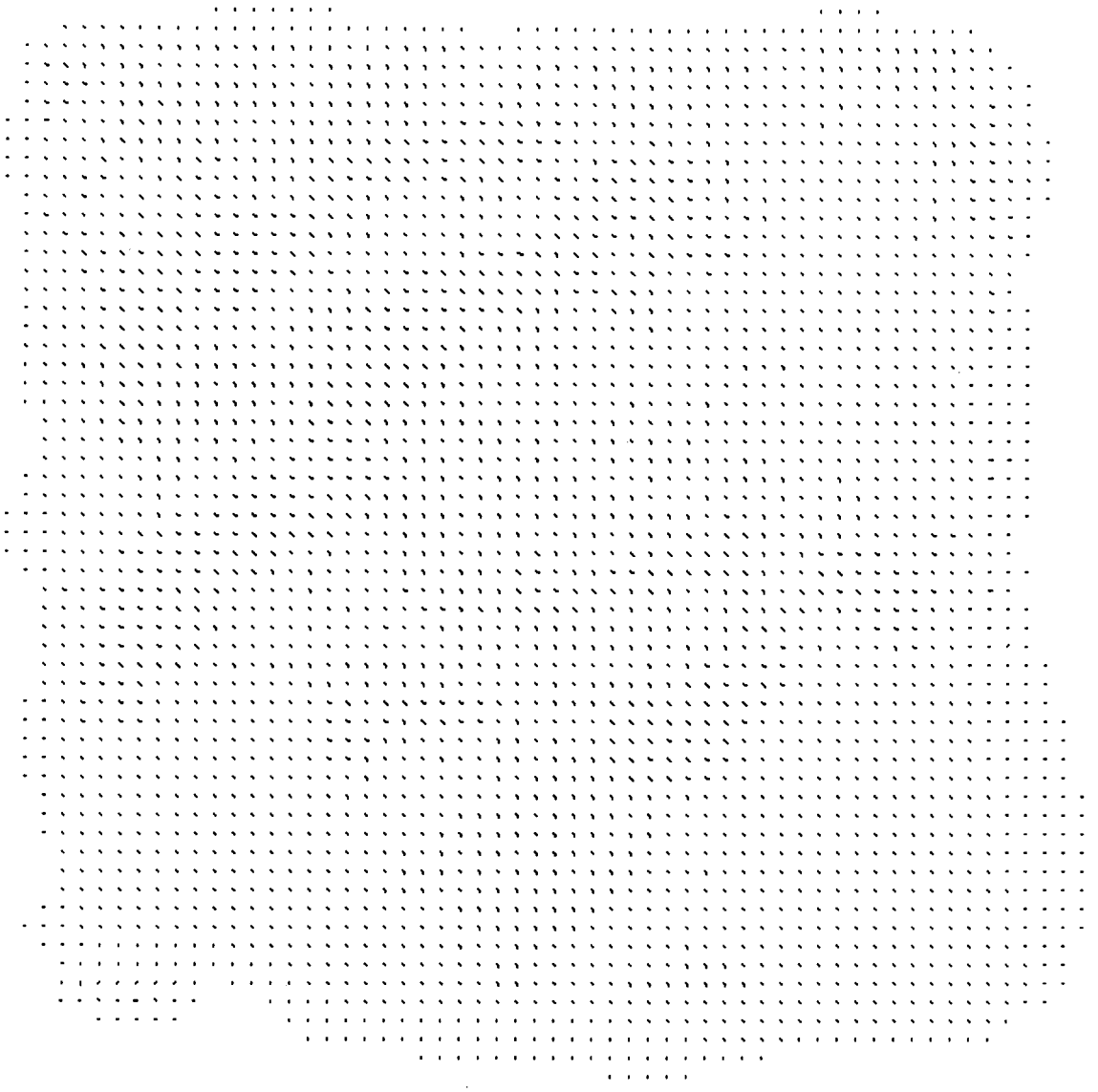


Figure 6.2: Motion Field for the moving block area of the second frame of the modified *comp* sequence generated using the Horn & Schunck algorithm. 100 iterations.

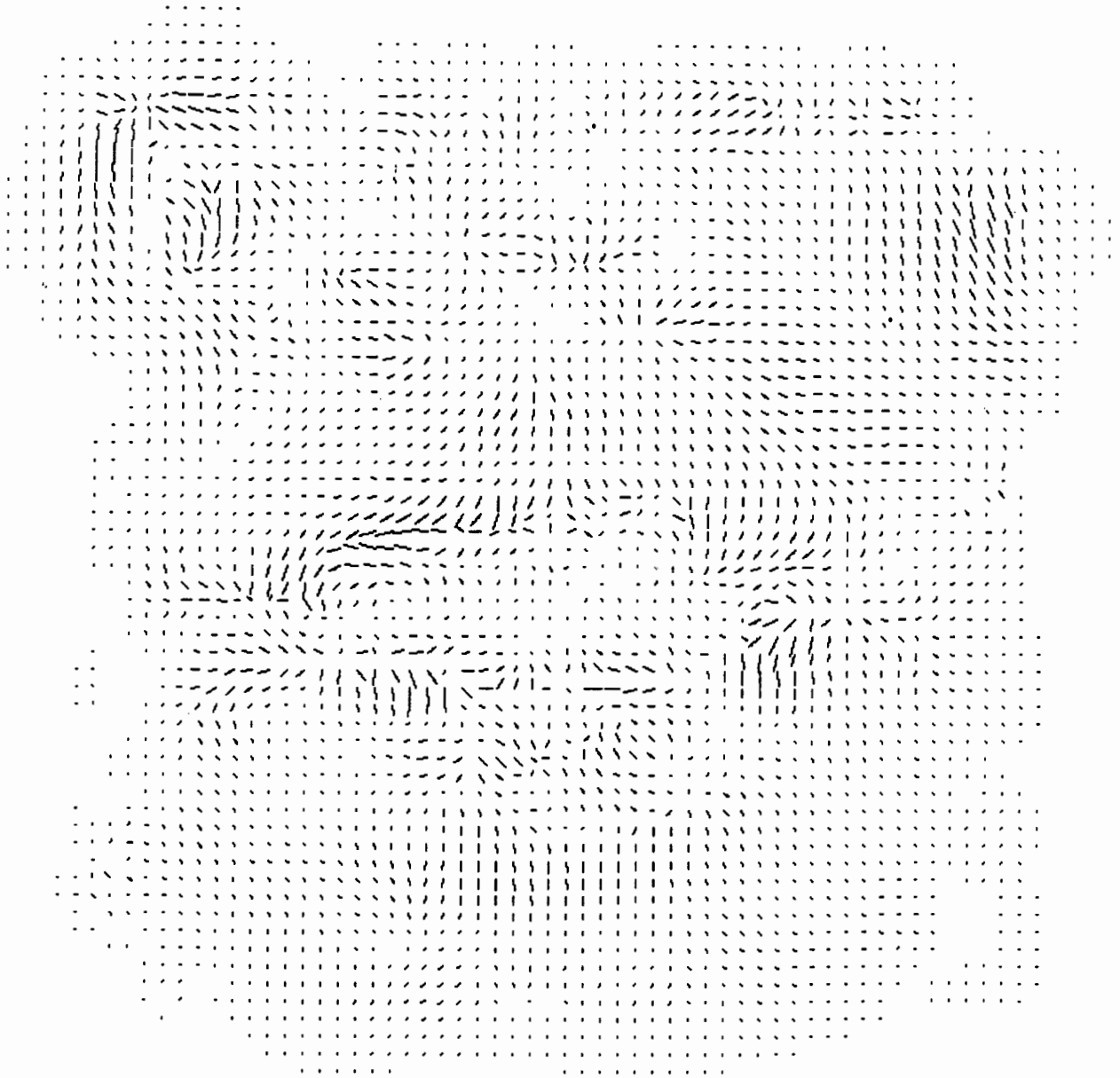


Figure 6.3: Motion Field for the moving block area of the second frame of the *comp* sequence generated using the Horn & Schunck algorithm. 100 iterations.

where ξ_i is the size of the neighborhood. For example, in the first order neighborhood, $\xi_i = 4$. The implementation of this method used in this thesis uses the following estimation of the gradient of the displaced pel difference:

$$\nabla_{\mathbf{d}}\tilde{\mathbf{r}}(\mathbf{d}(\mathbf{x}_i,t),\mathbf{x}_i,t,\Delta t) \approx \begin{bmatrix} \tilde{\mathbf{r}}(\mathbf{d}(\mathbf{x}_i,t)+(1,0),\mathbf{x}_i,t,\Delta t) - \tilde{\mathbf{r}}(\mathbf{d}(\mathbf{x}_i,t)+(-1,0),\mathbf{x}_i,t,\Delta t) \\ \tilde{\mathbf{r}}(\mathbf{d}(\mathbf{x}_i,t)+(0,1),\mathbf{x}_i,t,\Delta t) - \tilde{\mathbf{r}}(\mathbf{d}(\mathbf{x}_i,t)+(0,-1),\mathbf{x}_i,t,\Delta t) \end{bmatrix} \tag{6.18}$$

This algorithm attempts to maintain a smooth motion field by using the average motion vector over the neighborhood as the initial estimate for the next iteration. The second term in Equation 6.14 causes the estimate to move towards a lower displaced pel difference.

There are also a number of other computationally less complex algorithms for minimizing the MAP criterion presented and evaluated by Konrad and Dubois (1991), Bergeron and Dubois (1991), and Konrad (1989).

6.2.1 Results

Figure 6.4 shows the motion field generated using Gauss-Newton minimization of the MAP criterion on the normal progressive *comp* sequence (block displacement of $(+4,+4)$). While the motion field does reflect the general trend of the motion, there are serious defects caused by local minima into which the algorithm falls.

The algorithm was also applied to interlaced sequences for the purpose of interpolating the missing fields. The resulting reconstructed fields were both subjectively and quantitatively worse than the linear spatial interpolator discussed in Chapter 2. This is likely because of the interpolation error in the missing fields when this algorithm is used on interpolated sequences.

6.3 Survey of Other Motion Estimation Methods

This section presents an overview of several popular types of motion estimation algorithms. They are presented here since some of the ideas contained in the techniques could be applied to the methods presented in this thesis to improve the performance.

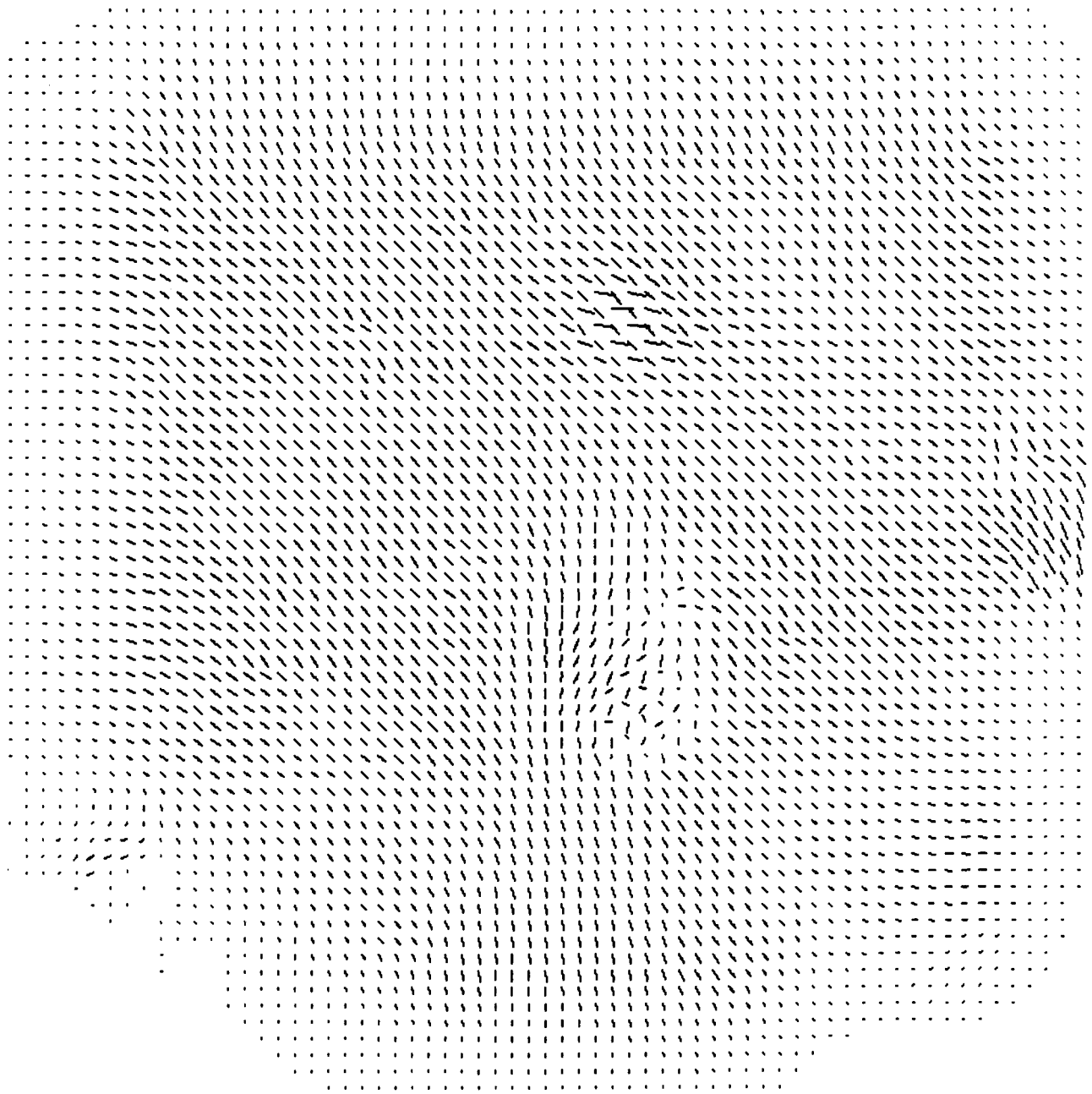


Figure 6.4: Motion Field for the moving block area of the second frame of the *comp* sequence generated using the Gauss-Newton minimization of the MAP criterion. $\lambda_{\mathbf{d}} = 0.3, \lambda_g = 0.0025$, 100 iterations.

6.3.1 Block-based Motion Estimation

Block matching techniques of motion estimation are generally similar to the the block-based motion compensation algorithm described in section 1.2.1. The current frame is divided into blocks which are then matched with the previous frame(s) and the best match is taken as the motion estimate. Variations include the different search and minimization techniques (Orchard 1989) (Zaccarin and Liu 1992), recursive block-matching algorithms (de Haan, Biezen, Huijgen, and Ojo 1993) (Xie, Eycken, and Oosterlinck 1992), and generalized block-matching algorithms that allow more than just translation (Seferidis and Chanbari 1992).

The main advantage of block-matching algorithms is that they are computationally efficient (see section 4.7.3) and therefore are the easiest algorithms to implement in real time. Difficulties arise however, when a block overlaps the edge of a moving object; i.e., part of the block has different motion from the rest of the block. This can lead to a "blocking" effect when used in the reconstruction of interlaced sequences.

To reduce the computational complexity of the MAP estimation presented in this thesis it is possible to use a block-oriented version of MAP estimation as described by (Bergeron and Dubois 1991). Together with the interlaced techniques in this thesis, block-oriented MAP estimation could perform reasonably well with acceptable computational complexity.

6.3.2 Pixel-recursive Motion Estimation

Pixel recursive algorithms attempt to recursively minimize the displaced pixel difference (Hsing 1987). The most common technique uses steepest descent and results in the following iteration (Baron 1990):

$$\hat{\mathbf{d}}^{n+1}(\mathbf{x}_i, t) = \hat{\mathbf{d}}^{n}(\mathbf{x}_i, t) - \epsilon \cdot \mathbf{r}(\hat{\mathbf{d}}^{n}(\mathbf{x}_i, t), \mathbf{x}_i, t, \Delta t) \cdot \nabla \tilde{g}(\mathbf{x} - \mathbf{d}(\mathbf{x}, t_-)) \qquad (6.19)$$

Improvements to Equation 6.19 have been suggested by Walker and Rao (1984) that improve the performance by varying the parameter, ϵ. In addition, simplifications have been made which reduce the computational cost of Equation 6.19 (Netravali and Robbins 1979).

Pel-recursive motion estimation allows for a more accurate estimation of the true

motion. However, it is more computationally complex than the block-matching algorithms. It is comparable to the Gauss-Newton minimization of the MAP criterion and the Hork & Schunck algorithm presented earlier in this chapter. We would expect the Gauss-Newton algorithm to outperform the pel-recursive technique while the Horn & Schunck algorithm would be the worst of the three.

6.3.3 Hierarchical Motion Estimation

Hierarchical motion estimation has been examined by several authors including Konrad (1989), Woods and Han (1991) and Paek, Kim, and Lee (1992). The general idea behind the algorithms is to coarsely estimate the motion and then refine the vectors to obtain an accurate approximation.

The advantage of this technique is that it tries to eliminate many of the local minima by using a coarse (often low pass) version of the video sequence to obtain initial approximations. Then, as more detail is added to the video sequence, the motion estimate is refined. This technique has been applied to interlaced sequences by Woods and Han (1991) and has proven to give better subjective performance than spatial filtering. By applying hierarchical techniques to the Bayesian techniques in this thesis could improve the interpolation and possibly the performance of the motion compensation algorithms.

Chapter 7

Conclusions

This thesis has presented three main topics as they relate to interlaced video sequences:

1. Estimation of motion in an interlaced video sequence using Bayesian motion estimation
2. Conversion of interlaced to progressive video using both spatial and motion compensated methods.
3. Motion compensation of interlaced video for the purposes of data compression.

7.1 Motion Estimation

The motion estimation approach taken in this thesis was to apply a statistical model to the motion field which allowed the Maximum A Posteriori probability (MAP) to be formed. Using the MAP criterion, an "optimal" estimate of the motion field for the given data, was obtained using a simulated annealing technique. While this technique is computationally expensive, it results in an excellent estimate of the motion which was the goal of the research.

Several modifications have been proposed in this thesis for the application of the Bayesian techniques to interlaced video sequence. These techniques have proven to give excellent motion estimates on difficult sequences containing high motion, panning and difficult textures.

In addition, the application of a piecewise-smooth model (line field) to the motion field was also considered. While this model did improve the quality of the motion field

estimation for progressive sequences, it did not perform as well in the case of interlaced sequences. For this reason, it was not included in the application to interpolation or motion compensation. It is possible that with the addition of another concept, perhaps a hierarchical procedure, the motion discontinuity model could be implemented with greater success.

Several computationally efficient motion estimation methods were also considered; however, all were significantly inferior to the simulated annealing procedure. Other algorithms were also mentioned as possible avenues for future work in this area.

7.2 Interpolation of Interlaced Video

Reconstruction of the missing lines of an interlaced sequence was studied using both spatial techniques and the Bayesian motion estimation techniques. As expected, the Bayesian motion estimation techniques were significantly better than the spatial techniques both subjectively and quantitatively. While the computational cost of the Bayesian algorithms is high, it was suggested that the algorithms could easily be implemented in a parallel fashion reducing the computing time significantly. Applications for this implementation would include off-line interlaced-to-progressive converters, video motion analysis and object detection/analysis.

7.3 Motion Compensation of Interlaced Video

Several different methods of motion compensation of interlaced video were presented. The main conclusion from this work was the importance of using the second previous field (same parity) in the motion compensation process. In addition, improved results were obtained when linear combinations of fields were used in the process.

The Bayesian motion estimation techniques were compared to the spatial techniques using the various motion compensation algorithms and the resulting difference was found to be negligible. It was suggested that in the case of very high motion, complex sequences, the Bayesian methods may prove superior, but in the majority of cases, the spatial techniques perform satisfactorily. This is important since the spatial techniques are relatively simple. However, results using "perfect" interpolation (the actual missing lines) have shown that there is potential for better interpolation

methods to significantly improve performance. Further research is needed to discover these methods and decide if the computational expense is warranted.

7.4 Future Work

While there are many directions for future work in this area, the most promising include the development of computationally efficient motion estimation algorithms possibly through the use of hierarchical procedures. This may also allow the favorable implementation of the piecewise-smooth motion model (line field). Also incorporation of characteristics such as newly exposed and hidden regions into the motion model could be productive.

In terms of interpolation, different motion compensated filtering techniques could be explored along with the associated concept of using several frames/fields in the interpolation process.

There may also be room for improvement in interlaced motion compensation algorithms by using multiple fields/frames and by using some of the ideas presented in Chapter 5 of this thesis.

References

Aggarwal, J. K. and N. Nandhakumar (1988, August). On the computation of motion from sequences of images - a review. *Proceedings of the IEEE 76*(8), 917–935.

Baron, A. (1990, April). Motion estimation and compensation. INRS Telecommunication.

Bergeron, C. and E. Dubois (1991, March). Gradient-based algorithms for block-oriented map estimation of motion and application to motion-compensated temporal interpolation. *IEEE Transactions on Circuits and Systems for Video Technology 1*(1), 72–85.

Besag, J. E. (1972). Nearest-neighbor systems and the auto-logistic model for binary data. *J. Royal Stat. Soc. B 34*, 75–83.

de Haan, G., P. W. A. C. Biezen, H. Huijgen, and O. A. Ojo (1993, October). True-motion estimation with 3-d recursive search block matching. *IEEE Transactions on Circuits and Systems for Video Technology 3*(5), 368–379.

Delogne, P., L. Cuvelier, B. Maison, B. V. Caillie, and L. Vandendorpe (1993). Improved interpolation, motion estimation and compensation for interlaced pictures. In *Proceedings SPIE, Visual Communications and Image Processing*, Volume 2094, pp. 326–336.

Depommier, R. and E. Dubois (1992, September). Motion-compensated temporal prediction for interlaced image sequences. In *Proceedings SPIE, Visual Communications and Image Processing*, pp. 264–269.

Dubois, E. (1992). Motion-compensated filtering of time-varying images. *Multidimensional Systems and Signal Processing 3*, 211–239.

Dubois, E. and J. Konrad (1993). Estimation of 2-D motion fields from image sequences with applications to motion compensated processing. In M. Sezan and R. Lagendijk (Eds.), *Motion Analysis and Image Sequence Processing*, Chapter 3. Kluwer.

Geman, S. and D. Geman (1984, November). Stochastic relaxation, Gibbs distributions, and the Bayesian estimation of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 6*(6), 721–741.

Gersho, A. and R. M. Gray (1992). *Vector Quantization and Signal Compression.* Series in Communications and Information Theory. Kluwer Academic Publishers.

Girod, B. (1993, April). Motion-compensating prediction with fractional-pel accuracy. *IEEE Transactions on Communications COM-41*(4), 604–611.

Girod, B. and R. Thoma (1985). Motion-compensating field interpolation from interlaced and non-interlaced grids. In *Proceedings SPIE, Image Coding*, pp. 186–193.

Gothe, M. and J. Vaisey (1993, May). Improving motion compensation using multiple temporal frames. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 157–160.

Horn, B. K. P. and B. G. Schunck (1981). Determining optical flow. *Artificial Intelligence 17*, 185–203.

Horvat, D. C., J. S. Bird, and M. M. Goulding (1992). True time-delay bandpass beamforming. *IEEE Journal of Oceanic Engineering 17*, 185–192.

Hsing, T. R. (1987). Motion detection and compensation coding for motion video coders: Technical review and comparison. In *GLOBECOM '87*, pp. 2.6.1–2.6.4.

Keys, R. G. (1981, December). Cubic convolution for digital image processing. *IEEE Transactions on Acoustics, Speech and Signal Processing ASSP-29*, 1153–1160.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983, May). Optimization by simulated annealing. *Science 220*, 671–680.

Konrad, J. (1989, June). *Bayesian Estimation of Motion Fields from Image Sequences.* Ph. D. thesis, McGill University, Montreal, Canada.

Konrad, J. and E. Dubois (1991, August). Comparison of stochastic and deterministic solution methods in bayesian estimation of 2d motion. *Image and Vision Computing 9*(4), 215–228.

LeGall, D. (1991, April). MPEG: A video compression standard for multimedia applications. *Communications of the ACM 34*(4), 47–58.

Martinez, D. M. and J. S. Lim (1989). Spatial interpolation of interlaced television pictures. In *Proceedings ICASSP'89*, pp. 1886–1889.

Miller, D. and K. Rose (1994, Feb./Mar./Apr.). Combined source-channel vector quantization using deterministic annealing. *IEEE Transactions on Communications 42*(2/3/4), 347–359.

Netravali, A. N. and B. G. Haskell (1988). *Digital Pictures.* AT&T Bell Laboratories.

Netravali, A. N. and J. D. Robbins (1979, March). Motion compensated television coding: Part i. *Bell Syst. Tech. J. 58*, 631–670.

Oppenheim, A. V. and R. W. Schafer (1989). *Discrete-Time Signal Processing.* Prentice Hall.

Orchard, M. T. (1989, November). A comparison of techniques for estimating block motion in image sequence coding. In *Proceedings SPIE, Visual Communications and Image Processing IV*, pp. 248–258.

Paek, H., R. C. Kim, and S. U. Lee (1992, October). On the motion compensated transform coding technique employing sub-band decomposition. In *Proceedings SPIE, Visual Communications and Image Processing*, pp. 253–264.

Seferidis, V. and M. Chanbari (1992, October). Generalized block matching motion estimation. In *Proceedings SPIE, Visual Communications and Image Processing*, Volume 1818, pp. 110–119.

Walker, D. and K. Rao (1984, October). Improved pel-recursive motion compensation. *IEEE Transactions on Communications 32*(10), 1128–1134.

Weston, M. (1988). Fixed, adaptive, and motion compensated interpolation of interlaced tv pictures. In L. Chiariglione (Ed.), *Signal Processing of HDTV*, pp. 401–408. Elsevier Science Publishers B.V.

Woods, J. W. and S.-C. Han (1991, October). Hierarchical motion compensated deinterlacing. In *Proceedings SPIE, Visual Communications and Image Processing*, pp. 805–810.

Xie, K., L. V. Eycken, and A. Oosterlinck (1992). A new block-based motion estimation algorithm. *Signal Processing: Image Communication 4*, 507–517.

Zaccarin, A. and B. Liu (1992, March). Fast algorithms foir block motion estimation. In *Proceedings ICASSP'92*, pp. III–449 – III–452.

Zaccarin, A. and B. Liu (1993). Block motion compensated coding of interlaced sequences using adaptively deinterlaced fields. *Signal Processing: Image Communication 5*, 473–485.