INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the original text directly from the copy submitted. Thus, some dissertation copies are in typewriter face, while others may be from a computer printer.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyrighted material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each oversize page is available as one exposure on a standard 35 mm slide or as a 17" × 23" black and white photographic print for an additional charge.

Photographs included in the original manuscript have been reproduced xerographically in this copy. 35 mm slides or 6" × 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Accessing the World's Information since 1938

300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

Order Number 8821116

Efficient motion-compensated coding for low bit-rate video applications

Puri, Atul, Ph.D.

City University of New York, 1988

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark __✓__.

1. Glossy photographs or pages __✓__
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy _____
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages __✓__
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print _____
11. Page(s) _____ lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Dissertation contains pages with print at a slant, filmed as received _____
16. Other Poor print on pages 1, 25, and 32. Best copy available.

U·M·I

EFFICIENT MOTION-COMPENSATED CODING

FOR

LOW BIT-RATE VIDEO APPLICATIONS

by

ATUL PURI

A dissertation submitted to the Graduate Faculty in Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1988

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosphy.

March 24, 1988
Date

Donald L. Schilling
Chair of Examining Committee

3/24/88
Date

Jacques E. Benveniste
Executive Officer

Dr. Barry G. Haskell

Dr. Ralph C. Brainard

Prof. Joseph Barba

Prof. Norman Scheinberg

Supervisory Committee

The City University of New York

Abstract

EFFICIENT MOTION-COMPENSATED CODING
FOR
LOW BIT-RATE VIDEO APPLICATIONS

by

Atul Puri

Adviser: Professor Donald L. Schilling

Techniques for compression of video-signals seek to exploit various statistical and perceptual redundancies present, to obtain a compact representation, efficient for storage, processing, and transmission. It is also necessary that this efficient representation allow reconstruction of images having *visually acceptable* quality. In this thesis, we develop efficient, image-compression techniques towards *low bit-rate, full-motion,* video-conferencing application. This application typically requires bandwidth compression by a factor of 500 or more; and can generally be achieved [3],[44],[60] via a combination of many bandwidth reduction measures, such as, spatial subsampling, frame decimation, conditional replenishment, motion-compensation, and block coding.

Motion-compensation techniques seek to detect, estimate and compensate for movement of the objects in a scene, and can improve the coding efficiency significantly [4],[5],[37]-[61]. We present an *efficient search technique,* that reduces the computations necessary for estimating motion by block-matching. We also propose several approaches to improve performance, reduce complexity of computations, and reduce overhead. *Statistical properties* of motion-compensated frame-differential (MCFD) signal, are then investigated; and provide a basis for

improving existing, and developing new coding algorithms.

A basic *transform-domain* scheme for coding MCFD signal is investigated, and choices for transforms explored. The basic (single) transform scheme is then modified to allow *multiple-transform* coding. In this scheme, depending on the characteristics of the block being coded, selection of the transform can be made from a small set of pre-decided transforms, on a block-by-block basis.

A *variable block-size* (VBS) scheme is proposed for coding MCFD's, and introduces adaptivity to improve performance. Two configurations are tested: the first, uses blocks for motion-estimation and sub-blocks for coding, whereas, the second uses sub-blocks for motion-estimation and coding. The performance of, both the transform-domain and pel-domain algorithms are compared with and without the VBS approach.

The concept of *cluster* and the method of *cluster coding* is then introduced; this method encodes significant pels that group together in a MCFD block in pel-domain. For blocks with sharp pulsive-transitions in MCFD's, transform coding schemes may require excessive bits. A *hybrid scheme*, combining transform and cluster coding techniques, is therefore recommended and its performance evaluated.

A *complete-coder* that uses fixed step-size is simulated, and bit-rate including motion and other overheads computed. A strategy useful for controlling the buffer status is then outlined. A *complete-coder* that includes *buffer-feedback* to allow buffer control is simulated.

Finally, we discuss some recent trends and developments, and identify directions for future research in this area.

ACKNOWLEDGMENT

I gratefully acknowledge the kind support and help of Dr. Barry Haskell, my Department Head at "AT&T BELL LABS". He not only permitted me to persue my thesis here but also provided me with full guidance, encouragement and support. Further, I would like to thank Dr. Ralph Brainard for his patient explanations, guidance, and insightful lunch-time discussions on various research related and other matters. I am also very thankful to Dr. Hseuh-Ming Hang for very fruitful technical interaction on various research topics, and for providing constant encouragement. To Barry, Ralph and Hseuh-Ming; I owe a lot. I would especially like to thank Dr. William Ninke for his kind encouragement and support. I am highly indebted to other colleagues at Bell Labs who were willing to provide help and suggestions when needed. I also deeply appreciate the generous permission to use the facilities, and the financial support of "AT&T BELL LABS".

I am sincerely grateful to my thesis adviser, Prof. Donald Schilling, for his guidance, advise, and being a constant source of encouragement. Prof. Schilling provided constant support and motivation, and was very kind in permitting the flexibility of pursuing the thesis off-campus, while allowing regular consultation. The kind suggestions and encouragement of thesis committee members, Prof. Joseph Barba and Prof. Norman Schienberg are also deeply appreciated; I would especially like to thank them for their willingness to put up with delayed deadlines. I would also like to thank Prof. Eichman, chairman of Deptt. of Electrical Engineering and other faculty members at City College; from whom I learned a lot.

I am ever grateful to my parents and brother for constant encouragement and support for pursuing the Doctoral degree. Last but not the least, whole-hearted support of Diane is fondly appreciated; this thesis would have been difficult to complete without her patience, understanding and sincere effort over past many years.

CONTENTS

CHAPTER 1: TECHNIQUES FOR IMAGE DATA COMPRESSION

1.1 INTRODUCTION 1
1.2 PREDICTIVE CODING 2
1.2.1 The Predictor 3
1.2.1.1 The Theory of Linear Prediction 3
1.2.2 The Quantizer 5
1.2.2.1 MMSE Quantizer 5
1.2.2.2 Subjective Quantizer 7
1.2.3 Channel Code Assigner 7
1.2.4 Adaptive Predictive Coding 8
1.2.4.1 Adaptive Prediction 8
1.2.4.2 Adaptive Quantization 8
1.3 TECHNIQUES FOR TRANSFORM CODING 9
1.3.1 General Discrete Transform Operation 10
1.3.1.1 The Karhunen-Loeve Transform (KLT) 12
1.3.1.2 The Discrete Sine Transform (DST) 15
1.3.1.3 The Discrete Cosine Transform (DCT) 15
1.3.1.4 The Discrete Fourier Transform (DFT) 17
1.3.1.5 The Walsh-Hadammard Transform (WHT) 17
1.3.2 Transform Efficiency 18
1.3.3 Transform Coefficient Distributions 20
1.3.4 Quantization and Bit Assignment of Transform coefficients 21
1.3.5 Adaptive Techniques for Transform Coding 22
1.3.6 Block Size, Performance and Implementation trade-offs 22
1.4 HYBRID CODING TECHNIQUES 23
1.5 COMPONENT versus COMPOSITE PROCESSING 24
1.6 EFFICIENT IMAGE-SEQUENCE COMPRESSION 24
1.7 DISCUSSION 25

CHAPTER 2: MOTION ESTIMATION: A REVIEW

2.1 INTRODUCTION 26
2.2 CLASSIFICATION OF VARIOUS TECHNIQUES 27
2.3 SEARCH BASED ON DERIVATIVES 28
2.3.1 Velocity Estimation Technique 29
2.3.2 Differential Technique 30

2.3.3 Block Motion by Recursive Method 32
2.3.4 The Theory of Steepest Descent 34
2.3.5 Pel Recursive Technique 36
2.4 SEARCH WITHOUT DERIVATIVES: Matching 37
2.4.1 Area Correlation Techniques 37
2.4.2 Distortion Function and Search Area 38
2.4.3 Independent Block Matching Motion Estimation 40
2.4.3.1 Log Search Algorithm 40
2.4.3.2 Three Step Search 41
2.4.3.3 Modified Log Search 42
2.4.3.4 One at a Time Search 43
2.4.3.5 Orthogonal Search 44
2.4.4 Dependent Block Matching Motion Estimation 45
2.4.4.1 Motion With an Initial Estimate 45
2.4.4.2 Reduced Search With an Initial Estimate 47
2.5 MISCELLANEOUS TECHNIQUES 47
2.6 PROBLEMS AND PROSPECTS: A DISCUSSION 48
CHAPTER 3: EFFICIENT MOTION-COMPENSATION AND STATISTICAL ANALYSIS
3.1 INTRODUCTION 53
3.2 THE MOTION DETECTOR AND SEARCH RANGE 54
3.3 CRITERION FOR MATCH 54
3.4 DISTORTION MODEL 56
3.5 GOALS OF BLOCK MATCHING MOTION-ESTIMATION ALGORITHMS 57
3.6 ALGORITHMS FOR INDEPENDENT SEARCH 58
3.6.1 Search by the Orthogonal Technique 58
3.6.2 Techniques for measuring smaller displacements 60
3.6.2.1 Algorithm 2.1 61
3.6.2.2 Algorithm 2.2 61
3.6.2.3 Algorithm 2.3 61
3.7 ALGORITHMS FOR DEPENDENT ESTIMATION 62
3.7.1 Algorithm 3.1: Search With Temporal Initial Estimate 63
3.7.2 Algorithm 3.2: Search With Spatial Initial Estimate 64
3.8 SEARCH COMPLEXITY COMPARISON 65
3.9 BLOCK-SIZE, OVERHEAD, SAMPLES, AND COMPLEXITY 66

3.10 IMAGE SEQUENCES AND PREPROCESSING 67
3.11 SIMULATION RESULTS AND STATISTICAL ANALYSIS 68
3.12 DISCUSSION 74
CHAPTER 4: MOTION-COMPENSATED TRANSFORM CODING
4.1 INTRODUCTION 105
4.2 MOTION-COMPENSATED BLOCK-CODING: A REVIEW 105
4.2.1 Coding Algorithms 105
4.2.1.1 Coding Algorithm 1 107
4.2.1.2 Coding Algorithm 2 108
4.2.1.3 Coding Algorithm 3 108
4.2.1.4 Coding Algorithm 4 108
4.2.2 Block-size Choice and Low bit-rate Image-Sequence Coding 109
4.3 MOTION-COMPENSATED TRANSFORM CODING SCHEME 109
4.3.1 Introduction to Basic Approach 112
4.3.2 Motion Detection and Estimation 112
4.3.3 Orthogonal Transforms on the Motion-Compensated Frame-Differences 114
4.3.3.1 Block Classification and Transform Coding System 117
4.3.3.2 Transform Choices for Coding 117
4.3.4 Simulation Results for single Transform Coding 119
4.4 MULTIPLE-TRANSFORM APPROACH FOR MOTION-COMPENSATED CODING 121 123
4.4.1 Entropy versus VWL Coding Bits 125
4.4.2 Block-Correlations and Transforms 125
4.4.3 Multiple Transform Approach 126
4.4.3.1 One Transform Selection per Block 127
4.4.3.2 Correlation-Direction Based Transform Selection 128
4.4.4 Simulation Results for Multiple Transform Scheme 128
4.5 FILTER IN THE CODING LOOP 130
4.6 DISCUSSION AND CONCLUSIONS 132
CHAPTER 5: VARIABLE BLOCK-SIZE MOTION-COMPENSATED CODING
5.1 INTRODUCTION 171
5.2 CODING SYSTEM, MOTION, AND FRAME-DIFFERENCES 172
5.3 VARIABLE BLOCK-SIZE MOTION ESTIMATION 174
5.4 TRANSFORM DOMAIN CODING WITH AND WITHOUT VARIABLE BLOCK SIZE MOTION-COMPENSATION 177

5.5 PEL DOMAIN CODING WITH AND WITHOUT VARIABLE BLOCK-SIZE MOTION-COMPENSATION 180
5.6 CONCLUSION 182
CHAPTER 6: CLUSTER/TRANSFORM HYBRID CODING
6.1 INTRODUCTION 207
6.2 MOTIVATION 208
6.3 VECTOR, SCALAR, AND TRANSFORM CODING 209
6.4 CLUSTER CODING 212
6.4.1 What is a Cluster? 213
6.4.2 Cluster Sizes and Shapes 214
6.4.3 Extracting Clusters 215
6.4.4 Representing Clusters 216
6.5 A CLUSTER/TRANSFORM HYBRID CODING ALGORITHM 218
6.6 PERFORMANCE 220
6.7 DISCUSSION 222
CHAPTER 7: A COMPLETE MOTION-COMPENSATED CODER
7.1 INTRODUCTION 236
7.2 A FIXED STEP-SIZE CODER 237
7.2.1 Performance of Coder 237
7.3 A COMPLETE CODER WITH BUFFER FEEDBACK CONTROL 238
7.3.1 Desirable Attributes of Buffer Feedback Control 240
7.3.2 Buffer-size and Status 241
7.3.3 Buffer Fullness & Control (BFC) Algorithm 241
7.3.4 Performance of Algorithm 244
7.4 CONCLUSION 245
CHAPTER 8: DISCUSSION AND DIRECTIONS
8.1 DISCUSSION 262
8.2 TRENDS, APPLICATIONS, AND DIRECTIONS 266
8.3 PROSPECTS AND PROBLEMS 267
8.3.1 Hybrid Motion-compensation 268
8.3.2 Hybrid Cluster/Transform Coding 268
8.3.3 Motion Compensated Interpolation 268
8.3.4 Non-orthogonal Transforms & Model-based Approach 269
8.3.5 Post-processing 269
REFERENCES 270

CHAPTER 1: TECHNIQUES FOR IMAGE COMPRESSION

1.1 INTRODUCTION

Compression of image (video signals) is possible because, a large part of the information in an image is redundant. Image compression techniques seek to exploit various statistical redundancies present in the signal and the perceptual limitations of the human vision in order to obtain a compact representation, thereby minimizing the amount of information necessary to represent an image for transmission or storage purposes. It is necessary that the reconstructed image from this compact representation, be of visually acceptable quality. It is also usually required that the complexity of an image compression system should be low. Various requirements of an application, and economic constraints usually determine the transmission capacity available, which in turn places a bound on coding, and the fidelity with which reconstruction can be achieved. Various image coding applications such as broadcast TV, video-conferencing, still frame etc., involve processing monochrome or color signals of still or moving images containing natural scenes, graphics or text and have to fulfill different subjective requirements.

Most picture processing applications, require a discrete (digital) representation of the signal for ease of manipulations, storage and transmission. The basic PCM (Pulse Code Modulation) coding-system provides such a discrete-time and discrete-amplitude representation. This is obtained by first sampling a continuous video signal, followed by quantizing and assignment of code words. Often in PCM representation of a monochrome signal, the image luminance has to be quantized to 8 bits per sample to allow reconstruction of good quality. For video signals sampled at Nyquist-rate, this

generates a raw data-rate of over 64 Mbit/s. If the video-signal is available in R,G,B color format then the raw data-rate generated is triple of that of monochrome signal. Thus, various practical and economic reasons make it necessary to seek a bandwidth compressed (compact) representation of the information contained in video signals. Of course it is necessary that distortion in image reconstructed from bandwidth compressed representation be small, causing only minimum visual artifacts.

The digitized PCM-representation of a still-scene is often referred to as a frame (or field). If the contents of the scene vary with time, several snap-shots are necessary to exhibit all the changes taking place in time. Such a collection of continuous snapshots (frames or fields) is referred to as an image-sequence.

1.2 PREDICTIVE CODING

The adjacent samples (pels) in the PCM representation of the video signal tend to be often quite similar. This similarity (or correlation) in the adjacent pels is exploited in the predictive coding systems, where a prediction of a pel to be encoded is made from previously coded pels that have already been transmitted. A prediction difference signal is generated by subtracting the prediction signal from the original signal. This error signal is now quantized into a set of discrete amplitude levels. A channel encoder then transmits binary words of fixed or variable length representing the quantized difference (prediction error) signal. The predictive coder thus has three basic components (1) Predictor, (2) Quantizer and (3) Channel Code Assigner. If the quantizer uses two levels then the predictive coder is also called the Delta-Modulator [24],[25], when the number of levels is greater than two it is generally referred to as a Differential PCM (DPCM) coder.

1.2.1 The Predictor

The predictor employed in a DPCM system may be linear or non-linear depending on whether the prediction is obtained by a linear or non-linear combination of previously transmitted sample values. A linear predictor is defined by a set of predictor coefficients, which when applied to previously transmitted pels generate the prediction signal. Predictor for an image-frame can either be one dimensional or two dimensional, depending on whether the previously transmitted sample values forming the prediction signal belong to only the current line or to previous lines as well. The prediction scheme that uses the previous samples from the same image-frame is also referred to as *spatial* prediction. When a scene is available as sequence of image-frames, the samples in adjacent frames are also related. The interframe-predictors exploit this property, and use the pels located in previously transmitted frame for prediction of the current frame pels. This is referred to as *temporal* prediction. It is also possible to combine pels from current line, previous lines and previous frame to form a three dimensional prediction, also called as spatio-temporal prediction. We now review a statistical procedure [1]-[2],[9]-[10], [23] for designing a linear predictor.

1.2.1.1 The Theory of Linear Prediction

Let a sequence of sample values (pels or coefficients) be given by s_1, s_2, ..., s_N, and are assumed to have a zero mean and interelement covariance of r_{ij}. The sequence is assumed to be stationary so that $r_{NN} = r_{N-1N-1} = \ldots\ldots = r_{11} = \sigma^2$, and interelement correlation coefficients are $\rho_{ij} = \frac{r_{ij}}{\sigma^2}$. We now predict the value of current element s_N by using a weighted sum of previously observed values $\hat{s}_1$, ..., $\hat{s}_{N-1}$ such that,

$$\hat{s}_N = a_1\hat{s}_{N-1}+a_2\hat{s}_{N-2}+.....+a_{N-1}\hat{s}_1=\sum_{i=1}^{N-1} a_i\hat{s}_{N-i} \tag{1.1}$$

Where $\hat{s}_N$ represents prediction signal and is computed from previously transmitted samples $\hat{s}_{N-1},.....,\hat{s}_1$, and a set of predictor coefficients $a_1,.....,a_{N-1}$, so that the receiver can generate $\hat{s}_N$. The prediction error is then given by,

$$e_N=s_N-\hat{s}_N \tag{1.2}$$

and is to be quantized and coded for transmission purposes. The various weighting coefficients $a_1,.....,a_{N-1}$ can now be optimized to minimize the mean square signal as follows.

$$\sigma_e{}^2=E(e_N{}^2)=E((s_N-\hat{s}_N)^2)=E((s_N-\sum_{i=0}^{N-1} a_i\hat{s}_{N-1})^2) \tag{1.3}$$

For minimizing the error we take partial derivatives of the above and set them to zero.

$$i.e.\quad \frac{\partial\sigma_e{}^2}{\partial a_1}=0,\ .....,\ \frac{\partial\sigma_e{}^2}{\partial a_{N-1}}=0 \qquad \text{or in general} \qquad \frac{\partial\sigma_e{}^2}{\partial a_i}=0 \tag{1.4}$$

Therefore $-2E((s_N-\sum_{i=1}^{N-1} a_i\hat{s}_{N-i})\hat{s}_{N-i})=0$. Now the optimum predictor coefficients minimizing the mean square error can be generated as :

$$\begin{bmatrix} a_1 \\ a_2 \\ . \\ a_{N-1} \end{bmatrix} = \begin{bmatrix} E(\hat{s}_{N-1}\hat{s}_{N-1}) & E(\hat{s}_{N-1}\hat{s}_{N-2}) & . & E(\hat{s}_{N-1}\hat{s}_1) \\ E(\hat{s}_{N-2}\hat{s}_{N-1}) & E(\hat{s}_{N-2}\hat{s}_{N-2}) & . & . \\ . & . & . & . \\ E(\hat{s}_1\hat{s}_{N-1}) & . & . & E(\hat{s}_1\hat{s}_1) \end{bmatrix}^{-1} \begin{bmatrix} E(s_N\hat{s}_{N-1}) \\ . \\ . \\ E(s_N\hat{s}_1) \end{bmatrix} \tag{1.5}$$

In general the variance of error can be computed as :

$$\sigma_e{}^2=E[(s_N-\hat{s}_N)^2]=E[s_N(s_N-\hat{s}_N)]-E[\hat{s}_N(s_N-\hat{s}_N)] \tag{1.6}$$

If all a_i's are optimum then $E[(s_N-\hat{s}_N)\hat{s}_{N-i}]=0$. and this results in dropping of the term $E[\hat{s}_N(s_N-\hat{s}_N)]$ in eqn (1.6). Hence,

$$\sigma^2_{emin} = E[s_N(s_N - \hat{s}_N)] \tag{1.7}$$

The minimum mean square error variance can then be given by

$$\sigma^2_{emin} = E[s_N(s_N - a_1\hat{s}_{N-1} - a_2\hat{s}_{N-2} - \ldots\ldots\ldots a_{N-1}\hat{s}_1)] \tag{1.8}$$

On further simplification eqn. (1.8) yields,

$$\sigma^2_{emin} = \sigma^2 - \sum_{i=1}^{N-1} a_i E(s_N \hat{s}_{N-i}) \tag{1.9}$$

This can also be written as,

$$\sigma^2_{emin} = \sigma^2 - \sum_{i=1}^{N-1} a_i r_{NN-i} \tag{1.10}$$

1.2.2 The Quantizer

In the DPCM systems the prediction-error signal has a much smaller dynamic range as compared to original signal (that has many statistical redundancies), and can therefore be much coarsely quantized than the original signal, allowing some compression. Three types of degradations can occur if the quantizers are not very well designed. They are, (i) the granular noise, (ii) edge busyness, and (iii) the slope overload. Various methods for optimizing the design of a quantizer have been reviewed in [17] and [21]. Here, we outline a statistical procedure for designing a quantizer.

1.2.2.1 MMSE Quantizer

We follow Max's [22] minimum mean square error procedure to design an optimum K level quantizer. We define r_1, r_2, $\ldots\ldots r_K$ to be a set of representative levels and d_1, d_2, $\ldots\ldots d_{K+1}$ be a set of decision levels. Now if the input signal 's' is such that $d_i < s \leq d_{i+1}$, for $i=1,\ldots\ldots,K$, it is represented by r_i. The error due to quantization

over any interval is given by,

$$e_i = \int_{d_i}^{d_{i+1}} f(s-r_i)\, p(s)\, ds \tag{1.11}$$

where *p(s)* is the probability density function for the signal and $f(.)$ is a non negative function.

$$e_q = \sum e_i = \sum_{i=1}^{K} \int_{d_i}^{d_{i+1}} f(s-r_i)\, p(s)\, ds \tag{1.12}$$

If $f(.) = (.)^2$ then total mean square quantization error can be given by the following,

$$\sigma_q^2 = \sum_{i=1}^{K} \int_{d_i}^{d_{i+1}} (s-r_i)^2\, p(s)\, ds \tag{1.13}$$

For minimizing $\sigma_q{}^2$ following conditions need to be satisfied :

$$d_i = \frac{r_{i-1}+r_i}{2} \quad \text{for } i=2,.....,K \tag{1.14}$$

$$\int_{d_i}^{d_{i+1}} (s-r_i)\, p(s)\, ds = 0 \quad \text{for } i=1,.....,K \tag{1.15}$$

The above equation when simplified further yields,

$$r_i = \frac{\int_{d_i}^{d_{i+1}} s\, p(s)\, ds}{\int_{d_i}^{d_{i+1}} p(s)\, ds} \tag{1.16}$$

In most image coding applications, *p(s)* is given by a *laplacian function* as,

$$p(s) = \frac{1}{\sqrt{2}\sigma_s}\, e^{\left(\frac{-\sqrt{2}|s|}{\sigma_s}\right)} \tag{1.17}$$

For unit variance, eqn. (1.17) simplifies to $p(s) = \frac{1}{\sqrt{2}}\, e^{(-\sqrt{2}|s|)}$, which on further substituting in eqn. (1.15) and solving yields,

$$r_i = \frac{\int_{d_i}^{d_{i+1}} s\ p(s)\ ds}{\int_{d_i}^{d_{i+1}} p(s)\ ds} = \frac{\left(d_{i+1}+\frac{1}{\sqrt{2}}\right)e^{-\sqrt{2}d_{i+1}} - \left(d_i+\frac{1}{\sqrt{2}}\right)e^{-\sqrt{2}d_i}}{e^{-\sqrt{2}d_{i+1}} - e^{-\sqrt{2}d_i}} \tag{1.18}$$

The decision and reconstruction levels can now be computed by the recursive use of the equations (1.14) and (1.18). An estimate of the resulting mean square error is obtained form the following:

$$\sigma_q^2 = \frac{1}{12K^2}\left[\int_{d_i}^{d_{i+1}} p(s)^{1/3} ds\right]^3 \tag{1.19}$$

1.2.2.2 Subjective Quantizer

Quantizers designed on the basis of minimum mean-square-error criterion are not able to exploit the human visual-properties very well, and so, several researchers have attempted to design improved quantizers by taking into account the subjective-properties and limitations of the human visual system. Quantizers designed on such psycho-visual basis [17] try to either keep the quantization noise under the threshold of visibility or to minimize a weighted mean square error function for better subjective performance.

1.2.3 Channel Code Assigner

The frequency of occurrence of output levels of a quantizer is usually not uniform and therefore variable-word-length (VWL) codes can be used to represent these levels. The inner levels of a quantizer are generally used more frequently than the outer levels, and therefore the inner levels are usually assigned smaller-size codewords than the outer levels.

1.2.4 Adaptive Predictive Coding

These techniques use adaptive prediction and/or adaptive quantization to better handle the local variations in properties of an image. Adaptive techniques may be necessary for the reason of bit rate savings, or improvement of picture-quality when the transmission bandwidth is fixed. Such techniques usually increase the hardware complexity of the system.

1.2.4.1 Adaptive Prediction

As image signals usually have highly nonstationary statistics [1], it seems beneficial to adapt the prediction to the local properties of the image. Predictor selection procedures are usually based on either *forward* or *backward* criteria. In the forward criterion, information unavailable at the receiver is used to select a predictor at the transmitter, and so explicit predictor selection information is necessary to be sent as overhead to the receiver. In the backward criterion, an explicit switching information is not needed to be sent as it is implicitly available, and is obtained from the previous coded sample values at the receiver. A special type of adaptive prediction employed in interframe-coding environment is called the *motion compensated* prediction. In this scheme, depending on the direction and magnitude of the motion, the previously coded pel-values can be adaptively selected for prediction. This technique has been highly successful [1],[19],[21],[42] in reducing the magnitude of frame differential signal for moving image-sequence compression and will be discussed in detail in the next chapter.

1.2.4.2 Adaptive Quantization

Due to variation of activity in various regions of a picture, the required fidelity of

reproduction is usually different for each region. In adaptive-quantization schemes local properties of the signal are used to select a quantizer from a set, such that properties of a region and limitations of visual-system can be exploited for each such region. As in the case of adaptive prediction, the quantizer selection information may be explicitly sent as overhead, or may be implicitly available at the receiver.

1.3 TECHNIQUES FOR TRANSFORM CODING

With the rapid advancement in digital circuitry, transform coding techniques have attracted a great deal of attention [30]-[45]. Transform coding techniques have a superior ability to decorrelate the spatial data and thus minimize the statistical redundancy; these techniques can therefore achieve a high level of coding efficiency. In transform-domain even though the data is in a more 'fragile' form, and errors in transmission may affect several samples of the group; however the successive groups of coefficients are processed independently, and so, the channel errors within a group do not propagate to others. Transform coding techniques are therefore quite popular and well-suited for applications that require a high degree of compression; i.e., they have a superior ability to decorrelate a set of related pels to a rather uncorrelated set of coefficient values. The transformation often used are *linear* and *unitary*. The operation usually starts with subdividing a picture in to blocks of fixed-size. The sample data values in each block are then converted into set of coefficient values which are then quantized. This basic-procedure completely neglects the redundancies that exist between the blocks. Solely on the statistical basis, it seems beneficial to use larger size blocks; however from implementation simplicity, and to exploit the local properties of the video signal small block-sizes are considered advantageous. Since most of the compression in transform-domain results from dropping coefficients with

small values of energy and retaining only the large-energy coefficients, it is desirable to choose a transform that compacts the image-energy into as few coefficients as possible. Several choices for selecting transforms exist; a few are discussed.

1.3.1 General Discrete Transform Operation

From [1]-[2],[8] we state the general transform operation as follows. Let the matrix of a transform be represented by **T**. This transform operates on a sample data array **S** and a transform array **C** such that,

$$\left[\mathbf{C}\right] = \left[\mathbf{T}\right]\left[\mathbf{S}\right] \quad \text{and} \quad \left[\mathbf{S}\right] = \left[\mathbf{T}\right]^{-1}\left[\mathbf{C}\right] \tag{1.20}$$

In this use, **T** is called a *Linear transform.* Moreover, if

$$\left[\mathbf{T}\right]^{-1} = \left[\mathbf{T}\right]^{T} \tag{1.21}$$

then **T** is also called an ***orthonormal*** or ***unitary*** transform, where superscript ^{T} represents the transpose of a matrix. This means that if we denote q-th column of matrix $\mathbf{T}^{T}$ by a column vector $\mathbf{t}_q$ then eqn. (1.21) implies that

$$\mathbf{t}_q\mathbf{t}_p = 1 \quad \text{for } p = q, \quad \text{and} \quad \mathbf{t}_q\mathbf{t}_p = 0 \quad \text{for } p \neq q \tag{1.22}$$

Now, if a one-dimensional data vector is represented by **s**, the corresponding transform coefficient vector **c** can be obtained by

$$\left[\mathbf{c}\right] = \begin{bmatrix} c_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ c_N \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1N} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ t_{N1} & t_{N2} & \cdots & t_{NN} \end{bmatrix} \begin{bmatrix} s_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ s_N \end{bmatrix} \tag{1.23}$$

Extending to the two-dimensional case, let the data vector be represented by **S** such as,

$$\left[\mathbf{S}\right] = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1N} \\ \vdots & \vdots & \cdots & \vdots \\ s_{N1} & s_{N2} & \cdots & s_{NN} \end{bmatrix} \tag{1.24}$$

We now apply the basis matrix $\mathbf{T}$ to the data vector $\mathbf{S}$, and considering only the case of *separable-transformation* :

$$\left[\mathbf{C}'\right] = \begin{bmatrix} t_{11} & \cdots & t_{1N} \\ \vdots & \cdots & \vdots \\ t_{N1} & \cdots & t_{NN} \end{bmatrix} \begin{bmatrix} s_{11} & \cdots & s_{1N} \\ \vdots & \cdots & \vdots \\ s_{N1} & \cdots & s_{NN} \end{bmatrix} \tag{1.25}$$

We note that in eqn. (1.25), we have exploited the correlation in vertical direction only, and significant correlation in $\mathbf{C}'$ still remains in horizontal direction. We now transpose $\mathbf{C}'$, so that the coefficients now have a correlations in the vertical direction, and re-apply the basis matrix; this however requires another transpose at the end to bring coefficients back in right order.

We avoid having to take two transpose operations on the coefficients by instead transposing the basis array before multiplying it with $\mathbf{C}'$. We now have :

$$\left[\mathbf{C}\right] = \begin{bmatrix} c'_{11} & \cdots & c'_{1N} \\ \vdots & \cdots & \vdots \\ c'_{N1} & \cdots & c'_{NN} \end{bmatrix} \begin{bmatrix} t_{11} & \cdots & t_{N1} \\ \vdots & \cdots & \vdots \\ t_{1N} & \cdots & t_{NN} \end{bmatrix} \tag{1.26}$$

In a general case, the basis matrix of a transform can be real or complex so that the forward operation for the general transform can be written as :

$$\left[\mathbf{C}\right] = \left[\mathbf{C}'\right]\left[\mathbf{T}^*\right]^T = \left[\mathbf{T}^*\right]\left[\mathbf{S}\right]\left[\mathbf{T}^*\right]^T \tag{1.27}$$

Where superscript * means the *complex conjugate*. The inverse-transform operation can be written as,

$$\left[\mathbf{S}\right] = \left[\mathbf{S}'\right]\left[\mathbf{T}^*\right] = \left[\mathbf{T}^*\right]^T\left[\mathbf{C}\right]\left[\mathbf{T}^*\right] \tag{1.28}$$

For most transforms elements of **T** are however real, and eqn (1.28) can be rewritten as :

$$\left[\mathbf{C}\right] = \left[\mathbf{C}'\right]\left[\mathbf{T}\right]^T = \left[\mathbf{T}\right]\left[\mathbf{S}\right]\left[\mathbf{T}\right]^T \tag{1.29}$$

and its inverse as :

$$\left[\mathbf{S}\right] = \left[\mathbf{S}'\right]\left[\mathbf{T}\right] = \left[\mathbf{T}\right]^T\left[\mathbf{C}\right]\left[\mathbf{T}\right] \tag{1.30}$$

The forward transform operation can be alternatively written as :

$$c_{uv} = \sum_{p=0}^{N-1}\sum_{q=0}^{N-1} s_{pq}\, t_{up}\, t_{vq} \quad \text{where} \quad \mathrm{u,v} = 0,\ldots,\mathrm{N}-1 \tag{1.31}$$

1.3.1.1 The Karhunen-Loeve Transform (KLT)

To compute the KLT (in [1]), we first find the eigenvalues of image data covariance matrix biased to have a zero mean. The correlation-matrix is given by **R**,

$$\mathbf{R} = E[(\mathbf{s} - E(\mathbf{s}))(\mathbf{s} - E(\mathbf{s}))] \tag{1.32}$$

Alternately, in terms of elements of correlation matrix we have,

$$s(p,q) = E[(s_p - E(s_p))(s_q - E(s_q))] \tag{1.33}$$

Defining the statistical average μ,

$$\mu = \frac{1}{N}\sum_{p=1}^{N} s_p = \frac{1}{N}\sum_{q=1}^{N} s_q \tag{1.34}$$

Following a slightly suboptimum procedure we now have,

$$r(p,q) = E[(s_p - \mu)(s_q - \mu)] \tag{1.35}$$

The KLT basis vectors are then real orthonormalized eigenvectors of $\mathbf{R}$ and satisfy

$$\mathbf{R}\mathbf{t}_m = \lambda_m \mathbf{t}_m \tag{1.36}$$

Where,

$$(\mathbf{t}_m{}^T)\mathbf{t}_n = \begin{cases} 1 & \text{if } m=n \\ 0 & \text{if } m \neq n \end{cases} \tag{1.37}$$

It is well known [1],[2],[6]-[9], that KLT is theoretically a superior transform for decorrelating data, but we note that it requires:

[1] Estimation of data covariance matrix.

[2] Both row and column processing in a 2-D coding scheme as horizontal and vertical correlations may be different.

[3] Generation of eigen-vectors to compute the basis matrix.

[4] All of the above for a new image or new set of images as the basis vectors of a KLT are specific to data source.

[5] Basis vectors to be transmitted to receiver along with coded data.

KLT is therefore complex, and hence less popular for image coding applications. This is not a big loss, as for data having significant interelement correlation the difference between the KLT and other transforms is minimal.

In any event for the separable first-order Markov model assumption basis vectors can be determined by solving the following as suggested in [2], [63] as,

$$\tan N\omega_p = \frac{-(1-\rho^2)\sin\omega_p}{(1+\rho^2)\cos\omega_p - 2\rho} \tag{1.38}$$

Here N is the order of the transform and ρ the interelement correlation. Equation (1.38) is satisfied for N values of ω_p which can in turn be substituted in the following:

$$\lambda_p = \frac{1-\rho^2}{1+\rho^2-2\,\rho\,\cos\omega_p} \tag{1.39}$$

From ω_p, and eigen values λ_p, the terms of the basis vector can be generated by,

$$t_{pq} = \left(\frac{2}{N+\lambda_p}\right)^{1/2} \sin\left[\omega_p\left(q-\frac{(N-1)}{2}\right)+\frac{(p+1)\pi}{2}\right] \tag{1.40}$$

where ω_p can take values between 0 and π.

Following the first order Markov model given by eqns. (1.38)-(1.40) the basis matrices of a separable KLT for correlation values of 0.36, 0.50 and 0.91 can be written as follows,

$$\mathbf{KLT8(.36)}=\begin{bmatrix}
.2064 & .3185 & .3999 & .4427 & .4427 & .3999 & .3185 & .2064\\
.3648 & .4568 & .3716 & .1422 & -.1422 & -.3716 & -.4568 & -.3648\\
.4520 & .3499 & -.0553 & -.4126 & -.4126 & -.0553 & .3499 & .4520\\
.4707 & .0688 & -.4359 & -.2894 & .2894 & .4359 & -.0688 & -.4707\\
.4345 & -.2437 & -.3856 & .3211 & .3211 & -.3856 & -.2437 & .4345\\
.3581 & -.4490 & .0434 & .4102 & -.4102 & -.0434 & .4490 & -.3581\\
.2535 & -.4655 & .4339 & -.1752 & -.1752 & .4339 & -.4655 & .2535\\
.1311 & -.2915 & .4124 & -.4772 & .4772 & -.4124 & .2915 & -.1311
\end{bmatrix} \tag{1.41}$$

$$\mathbf{KLT8(.50)}=\begin{bmatrix}
.2300 & .3259 & .3946 & .4304 & .4304 & .3946 & .3259 & .2300\\
.3916 & .4512 & .3539 & .1334 & -.1334 & -.3539 & -.4512 & -.3916\\
.4642 & .3217 & -.0806 & -.4178 & -.4178 & -.0806 & .3217 & .4642\\
.4657 & .0315 & -.4478 & -.2857 & .2857 & .4478 & -.0315 & -.4657\\
.4186 & -.2720 & -.3779 & .3286 & .3286 & -.3779 & -.2720 & .4186\\
.3389 & -.4610 & .0576 & .4115 & -.4115 & -.0576 & .4610 & -.3389\\
.2374 & -.4652 & .4417 & -.1796 & -.1796 & .4417 & -.4652 & .2374\\
.1220 & -.2879 & .4134 & -.4809 & .4809 & -.4134 & .2879 & -.1220
\end{bmatrix} \tag{1.42}$$

$$\mathbf{KLT8(.91)}=\begin{bmatrix} .3226 & .3492 & .3645 & .3722 & .3722 & .3645 & .3492 & .3266 \\ .4732 & .4239 & .2925 & .1042 & -.1042 & -.2925 & -.4239 & -.4732 \\ .4694 & .2183 & -.1692 & -.4510 & -.4510 & -.1692 & .2183 & .4694 \\ .4278 & -.0756 & -.4832 & -.2788 & .2788 & .4832 & .0756 & -.4278 \\ .3655 & -.3411 & -.3576 & -.3495 & .3495 & -.3576 & -.3411 & .3655 \\ .2878 & -.4863 & .0914 & .4151 & -.4151 & -.0914 & .4863 & -.2878 \\ .1985 & -.4626 & .4590 & -.1896 & -.1896 & .4590 & -.4626 & .1985 \\ .1012 & -.2793 & .4154 & -.4890 & .4890 & -.4154 & .2793 & -.1012 \end{bmatrix} \quad (1.43)$$

1.3.1.2 The Discrete Sine Transform (DST)

The basis vector terms can be generated from the following :

$$t_{pq} = \sqrt{\frac{2}{N+1}} \sin\left[\frac{(p+1)(q+1)\pi}{N+1}\right] \quad (1.44)$$

Here p, q vary from 0 to $N-1$. For the case of N = 8 the basis matrix can be given by the following :

$$\mathbf{DST8} = \begin{bmatrix} .1612 & .3030 & .4082 & .4642 & .4642 & .4082 & .3030 & .1612 \\ .3030 & .4642 & .4082 & .1612 & -.1612 & -.4082 & -.4642 & -.3030 \\ .4082 & .4082 & .0000 & -.4082 & -.4082 & .0000 & .4082 & .4082 \\ .4642 & .1612 & -.4082 & -.3030 & .3030 & .4082 & -.1612 & -.4642 \\ .4642 & -.1612 & -.4082 & .3030 & -.3030 & -.4082 & -.1612 & .4642 \\ .4082 & -.4082 & .0000 & .4082 & -.4082 & .0000 & .4082 & -.4082 \\ .3030 & -.4642 & .4082 & -.1612 & -.1612 & .4082 & -.4642 & .3030 \\ .1612 & -.3030 & .4082 & -.4642 & .4642 & -.4082 & .3030 & -.1612 \end{bmatrix} \quad (1.45)$$

We note that the zeroth order basis vector for the DST is not constant.

1.3.1.3 The Discrete Cosine Transform (DCT)

The basis vector terms can be generated from the following :

$$t_{pq} = tt\sqrt{\frac{2}{N}} \cos\left[\frac{(p)(q+\frac{1}{2})\pi}{N}\right] \quad (1.46)$$

where $tt = \frac{1}{\sqrt{2}}$ when p = 0, otherwise $tt = 1$ for p, q = 0 to N-1. For the case N =

8 the basis matrix is given by the following:

$$\mathbf{DCT8} = \begin{bmatrix} .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 \\ .4904 & .4157 & .2778 & .0975 & -.0975 & -.2778 & -.4157 & -.4904 \\ .4619 & .1913 & -.1913 & -.4619 & -.4619 & -.1913 & .1913 & .4619 \\ .4157 & -.0975 & -.4904 & -.2778 & .2778 & .4904 & .0975 & -.4157 \\ .3536 & -.3536 & -.3536 & .3536 & .3536 & -.3536 & -.3536 & .3536 \\ .2778 & -.4904 & .0975 & .4157 & -.4157 & -.0975 & .4904 & -.2778 \\ .1913 & -.4619 & .4619 & -.1913 & -.1913 & .4619 & -.4619 & .1913 \\ .0975 & -.2778 & .4157 & -.4904 & .4904 & -.4157 & .2778 & -.0975 \end{bmatrix} \tag{1.47}$$

We see from above that the zeroth order basis vector for DCT is a constant, and each element of this vector has a value of $\frac{1}{\sqrt{N}}$.

The cosine transform is very popular in image coding, and for this reason we specify an alternate representation of the transformation when the input sample sequence is two dimensional. Let the sample sequence be given by $s(p,q)$, where p, q = 0,1,....,$N-1$. The transform coefficients can be generated as,

$$c(u,v) = \frac{4tt(u)tt(v)}{N^2} \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} s(p,q) \cos\left[\frac{(2p+1)u\pi}{2N}\right] \cos\left[\frac{(2q+1)v\pi}{2N}\right] \tag{1.48}$$

Where $u,v = 0, 1, ..., N-1$ and,

$$tt(w) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } w=0 \\ 1 & \text{if } w=1,2,...,N-1 \end{cases} \tag{1.49}$$

The inverse transform is then given by,

$$s(u,v) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} tt(u)tt(v)c(u,v) \cos\left[\frac{(2p+1)u\pi}{2N}\right] \cos\left[\frac{(2q+1)v\pi}{2N}\right] \tag{1.50}$$

for $u,v = 0,1,....,N-1$.

If the input data has significant correlation then the DCT is quite efficient in decorrelation efficiency and its performance is comparable to that of KLT.

1.3.1.4 The Discrete Fourier Transform (DFT)

The basis vector terms for the Discrete Fourier Transform can be generated from the following :

$$t_{pq} = \frac{1}{\sqrt{N}} e^{-j2\pi pq/N} \tag{1.51}$$

Alternately, eqn. (1.51) can also be written as,

$$t_{pq} = \frac{1}{\sqrt{N}} \left[\cos\left(\frac{2\pi pq}{N}\right) - j \sin\left(\frac{2\pi pq}{N}\right) \right] \tag{1.52}$$

For the case N = 8 the basis matrix for real part is given by the following:

$$\mathbf{DFT8(R)}= \begin{bmatrix} .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 \\ .3536 & .2500 & .0000 & -.2500 & -.3536 & -.2500 & .0000 & .2500 \\ .3536 & .0000 & -.3536 & .0000 & .3536 & .0000 & -.3536 & .0000 \\ .3536 & -.2500 & .0000 & .2500 & -.3536 & .2500 & .0000 & -.2500 \\ .3536 & -.3536 & .3536 & -.3536 & .3536 & -.3536 & .3536 & -.3536 \\ .3536 & -.2500 & .0000 & .2500 & -.3536 & .2500 & .0000 & -.2500 \\ .3536 & .0000 & -.3536 & .0000 & .3536 & .0000 & -.3536 & .0000 \\ .3536 & .2500 & .0000 & -.2500 & -.3536 & -.2500 & .0000 & .2500 \end{bmatrix} \tag{1.53}$$

The basis matrix for imaginary part can be stated as:

$$\mathbf{DFT8(I)} = \begin{bmatrix} .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & .2500 & .3536 & .2500 & .0000 & -.2500 & -.3536 & -.2500 \\ .0000 & .3536 & .0000 & -.3536 & .0000 & .3536 & .0000 & -.3536 \\ .0000 & .2500 & -.3536 & .2500 & .0000 & -.2500 & .3536 & -.2500 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & -.2500 & .3536 & -.2500 & .0000 & .2500 & -.3536 & .2500 \\ .0000 & -.3536 & .0000 & .3536 & .0000 & -.3536 & .0000 & .3536 \\ .0000 & -.2500 & -.3536 & -.2500 & .0000 & .2500 & .3536 & .2500 \end{bmatrix} \tag{1.54}$$

1.3.1.5 The Walsh-Hadammard Transform (WHT)

The lowest order basis matrix can be written as:

$$\mathbf{H2} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1.55}$$

$$\mathbf{H(N)} = \frac{1}{\sqrt{2}}\begin{bmatrix} H(N/2) & H(N/2) \\ H(N/2) & -H(N/2) \end{bmatrix} \tag{1.56}$$

Here $N = 2^n$ where n is an integer. For example,

$$\mathbf{H4} = \frac{1}{\sqrt{4}}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{1.57}$$

For the case N = 8 the basis matrix is given by the following:

$$\mathbf{H8} = \frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \tag{1.58}$$

The basis vectors in above can be ordered in terms of increasing number of zero crossings (sequency) to get,

$$\mathbf{H8} = \frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \tag{1.59}$$

1.3.2 Transform Efficiency

Using a first-order Markov model for a specified value of interelement correlation ρ. We have a $N \times N$ covariance matrix given by,

$$COV(\mathbf{S}) = E[(\mathbf{S}-\overline{\mathbf{S}})(\mathbf{S}-\overline{\mathbf{S}})^T] \tag{1.60}$$

where $\mathbf{S}$ is the data vector and $\overline{\mathbf{S}}$ is the mean vector. This can be rewritten as,

$$COV(\mathbf{S}) = E(\mathbf{S}\mathbf{S}^T) - \overline{\mathbf{S}}\overline{\mathbf{S}}^T \tag{1.61}$$

We can similarly define a transform covariance matrix,

$$COV(\mathbf{C}) = E[(\mathbf{C}-\overline{\mathbf{C}})(\mathbf{C}-\overline{\mathbf{C}})^T] \tag{1.62}$$

which can be simplified as before,

$$COV(\mathbf{C}) = E(\mathbf{C}\mathbf{C}^T) - \overline{\mathbf{C}}\overline{\mathbf{C}}^T \tag{1.63}$$

Using the transform coding equations the relation between the data domain covariance matrix and the transform domain covariance matrix for a two-dimensional separable-transform can be written as,

$$COV(\mathbf{C}) = [\mathbf{T}][COV(\mathbf{S})][\mathbf{T}]^T \tag{1.64}$$

The decorrelation efficiency η_C from [2] is given by,

$$\eta_C = 1 - \frac{\sum_{p,q=1,\, p\neq q}^{N} |C_{pq}|}{\sum_{p,q=1,\, p\neq q}^{N} |S_{pq}|} \tag{1.65}$$

where $\sum\mathbf{S}$ is the sum of off-diagonal terms of data covariance matrix and $\sum\mathbf{C}$ is the sum of off-diagonal terms of transform covariance.

We are now interested in the energy packing ability of the transforms. For this we turn to the diagonal elements in the data and transform covariance matrices to determine the relative energy in the first M of total of N diagonal components. From [2] we define energy packing efficiency η_E as,

$$\eta_E = \frac{\sum_{p=q=1}^{M} |C_{pq}|}{\sum_{p=q=1}^{N} |C_{pq}|} \tag{1.66}$$

In most intraframe (single frame) image coding applications a high degree of correlation (>0.8) exists between the adjacent samples both along horizontal and vertical directions. The KLT is considered to be theoretically optimum for mean square error performance criterion. Image data is generally highly nonstationarity and KLT is complex to implement. This is not a major problem as DCT can be employed for this application as long as there is high degree of correlation in the data it performs very close to the optimal KL Transform. In fact at present the DCT is the most popular efficient transform employed for image coding. WHT is often used because of its simplicity. Its basis vectors contain only +1 or -1 's and have an advantage from hardware implementation point of view, however this transform is less efficient then the DCT or the KLT.

1.3.3 Transform Coefficient Distributions

For coding of images by a two-dimensional discrete cosine transform (DCT) different assumptions have been made regarding the distribution of transform coefficients. Pratt [9] states that histogram of the DC coefficient should have a Rayleigh distribution since it is the sum of positive values, and that based on the central limit theorem other coefficients should have a Gaussian distribution. This conjecture is verified by Netravali and Limb [7], who also state that histograms of non-DC coefficients are roughly bell shaped. Tescher [10] however states that non-DC coefficients have a Laplacian rather then Gaussian Distribution. Others have assumed DC coefficients to have a Gaussian and non-DC coefficients to have a

Laplacian distribution. Reininger and Gibson [74] have even performed a goodness of fit test test on many images and conclude that DC coefficient should be considered to have a Gaussian and non-DC coefficients to have a a Laplacian distribution.

1.3.4 Quantization and Bit Assignment of Transform coefficients

From the theory of MMSE quantization presented earlier, we relate the variance of quantization error to the variance of input signal. Now, representing the variance of quantization error ${\sigma_q}^2$ by e and assuming a unit variance input signal we have,

$$e = l\,K^p = l\,2^{-p\,b_i} \tag{1.67}$$

where p usually takes a value of 2, and l is a constant which depends on the probability-distribution assumed. Assuming n coefficients are retained and each is assigned bits b_i, then the total number of bits per block will be,

$$\sum_{i=1}^{n} b_i = M \tag{1.68}$$

A general expression relating the overall average bit-rate b_i to a set of variances ${\sigma_i}^2$ can be stated from [2] as,

$$b_i = \frac{1}{2}\log_2{\sigma_i}^2 - \frac{1}{2n}\left[\log_2\prod_1^n{\sigma_i}^2 - 2M\right] \tag{1.69}$$

and can be alternately specified as,

$$b_i = \frac{1}{2}\log_2\left[{\sigma_i}^2 - D\right] \tag{1.70}$$

where D is related to ${\sigma_i}^2$ by,

$$\log_2 D = \frac{1}{n}\left[\log_2\prod_1^n{\sigma_i}^2 - 2M\right] \tag{1.71}$$

Eqn. (1.70) is also known as Shannon s famous rate-distortion result for gaussian

sources, and implies that bits should be assigned to various variables in proportion to *log* of their variances.

1.3.5 Adaptive Techniques for Transform Coding

In the adaptive techniques for transform coding blocks can be classified on the basis of spatial activity into low detail and high detail blocks. The low detail blocks generally require only a few bits for representing its contents where as high detail blocks need more bits for its representation. One of the popular schemes [71] uses DCT and Max's quantizer (discussed earlier), with blocks of size of 16×16 to classify the blocks of original image into one of the 4 classes on the basis of block AC energy. The variance of each coefficient in each class is calculated and used with eqn. (1.70) to determine a 16×16 bit assignment for each class.

1.3.6 Block Size, Performance and Implementation trade-offs

From coding point of view, smaller blocks require less hardware for performing transform operation and handling the blocked data. Besides, another advantage is that statistics with in a small block are stationary over the area of the block. Adaptive coding schemes can be used to compress the data within a block dependent on the type of data present for a particular block. A disadvantage of coding smaller-size blocks is that the inter-block correlation is neglected in compressing of data, resulting in inferior compression. In compression with large block-sizes correlations in data can be better exploited thus resulting in a higher compression. However, as block size increases stationarity of data from one block to another becomes greater. This allows a fixed coder to be employed instead of an adaptive one. Several choices in selection of a transform for coding exist. They are usually based on various

criterion such as ease of hardware implementation, decorrelation and energy compaction efficiency, minimum mean square reconstruction error etc. 1-D transforms provide decorrelation of data in one direction, and allow coding of single-frame images with reasonable visual quality at about 3 bits/pel. 2-D transforms decorrelate data in both directions and give good performance at much lower rates. For coding interframe signals 3-D transforms can be used to allow even higher compression, however, the implementation of 3-D transforms is considered to be quite complex.

1.4 HYBRID CODING TECHNIQUES

Hybrid coding techniques offer a compromise between the complexity of implementation of transform coding schemes and relatively low compression factors obtainable by predictive coding techniques [2]. As an improvement to 1-D transform, a hybrid coding procedure is investigated [15] that provides decorrelation in vertical direction by employing a previous element DPCM type coder. This approach results in an improvement over 1-D transform coding as decorrelation is further provided in the vertical direction by predicting the present coefficient by the coefficient at same spatial location in the previous line. The hardware complexity of such a scheme is also lower then that of a 2-D transform coding scheme.

In interframe image-sequence transform-coding the correlation in spatial and temporal direction can be fully exploited by employing a 3-D transform based scheme. Implementation of such a scheme is quite complicated as, for transform size of $8\times8\times8$ a storage of up to 8 frames is required, in addition to, speed, timing, size, and various other constraints. Hybrid interframe-coding techniques can help in exploiting both the spatial and temporal redundancy in a image sequence, thus resulting in good compression performance, without excessively complicating the hardware

implementation. We can for example exploit the spatial redundancy first, by a 2-D transform and then, the temporal redundancy, by means of predictive operation between the consecutive coefficients of the temporally adjacent blocks. The order of operations can also be interchanged such that the predictive operation between frames is done first, followed by a 2-D transform on the frame difference (FD) signal.

1.5 COMPONENT versus COMPOSITE PROCESSING

One can choose to process composite signal or independent component color signals [26]. In component processing R,G,B color signals are mapped to a more desirable Y,U,V [16] color space. Component processing offers some flexibility in choice of individual resolutions for luminance (Y) and the chrominance (U,V) signals. Moreover, luminance and chrominance signals can be processed based on their individual subjective-requirements. On the negative side, with component coding the implementation may be more complex. In composite processing even though a single color signal is used, due to the color information on the sub-carrier choices for processing are somewhat limited. The composite format therefore offers some simplicity in hardware design at the expense of flexibility. In brief, the preference of composite to component processing or vice-versa is dependent on the economic considerations, performance, ease of modifications and the application for which the systems are designed.

1.6 EFFICIENT IMAGE-SEQUENCE COMPRESSION

As stated earlier, in hybrid interframe-coding, a one dimensional predictor is employed in the temporal direction, followed by a 2-D transform. Now consider the scheme which uses a one dimensional previous element prediction in temporal

direction. If the scene is relatively still then the prediction (interframe) difference so obtained is likely to be small and, hence a 2-D transform of this signal will allow efficient compression. If on the other hand there is movement in the scene, the interframe prediction difference is small only in non-moving areas and is larger in the moving areas. This is so because, for moving areas, the previous element prediction along temporal direction can not very well predict the sample in current frame. As a solution to this problem, adaptation of the prediction signal, such that it lies in the direction of motion is possible, thereby giving small interframe differences. A special procedure for adapting prediction such that it is consistent with the motion is called *motion compensation procedure*. It is important to have a good *motion estimator* for the interframe differences to be small, and be efficiently coded by a transform-domain or a pixel-domain coding approach.

1.7 DISCUSSION

We have reviewed various basic image compression techniques such as predictive coding and transform coding and have discussed the theory behind these techniques. We have also discussed hybrid coding schemes that combine predictive and transform coding techniques; such schemes have found wide-application in both intraframe and interframe image coding. We introduced *motion-compensation* as a special form of adaptive-prediction, and briefly pointed out its importance in interframe-coding of moving image-sequences. In order to generate the motion-compensated prediction error signal that can be efficiently encoded, a good estimate of motion of objects in the image-sequence is necessary. We now devote the entire next chapter to review various techniques used for motion-estimation.

CHAPTER 2: MOTION-ESTIMATION : A REVIEW

2.1 INTRODUCTION

The estimation of motion in a sequence of images can have various applications such as interframe coding, scene and pattern analysis, robotics, telemetry, remote guidance, environmental etc. Choice of a specific approach for motion estimation is largely governed by the application it is aimed for. Most computer vision, robotics, scene and pattern analysis applications seek to track regular objects often having sharp edges. These edges usually retain a rigid shape during movement, and facilitate translational as well as rotational motion measurements by use of methods e.g. templates matching, and also in addition, several views of the object may also be available to aid in motion measurement. Above approach is not very suitable for measuring motion from television images of natural scenes as these images often do not possess sharp moving edges that can be tracked, and besides, objects can undergo a change in shape as well as size. In addition one of the aims of motion estimation here is to aid in image data compression (interframe coding) and therefore, the overhead data which may be generated for describing motion parameters to receiver has to be kept minimal. Most video-conference scenes consist of a head and shoulders view of a person/group of persons engaged in an active conversation. The motion in such scenes may be from camera panning or from movement of participants. This type of motion can be primarily modeled as being translatory in nature and with possibly, a secondary rotation component. This type of motion is possibly the most we can hope to compensate for, as complexity of motion estimation scheme has to be kept low for a real-time video-conferencing system. Hence the application and the

allowed complexity of implementation justify the right choice.

At first, various techniques for motion estimation appear to be quite different from each other but on closer inspection they seem to have a common theoretical basis. An overall understanding of various principles involved in motion estimation procedures is helpful for developing future techniques as well as for general image and vision understanding.

We review various approaches for motion estimation with low bit-rate video-conferencing application in mind. An important goal of motion estimation procedures that we review is to allow efficient compression of the motion compensated frame differential signal with little overhead.

2.2 CLASSIFICATION OF VARIOUS TECHNIQUES

Let us try to classify motion estimation techniques in to different categories. First a loose classification can be done on the basis of the size of data vector the techniques operate on. This means that the motion estimation techniques which estimate the motion of individual pels (1x1 size data vector) can be called *pel based* and the techniques estimating motion of a larger size data vector can be called *block based* techniques. Another approach some authors [4] have chosen is to classify these techniques to be forward acting or backward acting with regard to interframe coding depending on whether an explicit displacement vector needs to be transmitted to the receiver for motion compensation or not. An example of the first type is the *block matching* and that of second type is *pel recursive* technique. In addition classification can also be done on the basis of whether each technique specifies a single displacement estimate (like velocity estimation scheme) or more then one displacement estimates.

In any event it is significant to note at this point that all motion estimation techniques seek a so called *best estimate* of displacement given some current and past information regarding the nature of variation of input data (for e.g. luminance signal). Such techniques that seek a *best estimate* of data have their roots in optimization theory, and a classification already exists there. Under this classification, first, we have the search techniques which use derivatives (gradients) and second, we have techniques that do not use the derivatives. With reference to known schemes for motion compensation pel recursive techniques would fall under the former category where as the block matching schemes would fall under the later category. A third *miscellaneous* category is defined here for the sake of completeness and includes schemes that combine the two basic approaches. An example of this category can be the feature based motion estimation approach.

2.3 SEARCH BASED ON DERIVATIVES

Small variations of the apparent velocity of objects in an image sequence can be measured by comparing the frame differences with the spatial variations of the luminance. For a more precise displacement measurement, it is necessary that the spatial spatial and temporal luminance derivatives should be somewhat large. More complex, but continuous motion fields can be estimated by extensions of the differential method that uses recursion. The practical results are usually reasonable for image coding, but may not be good enough to obtain a smooth velocity field. We now discuss various techniques for motion estimation which use gradients for displacement measurement.

2.3.1 Velocity Estimation Technique

The use of velocity estimate for predictive coding was introduced by Haskell and Limb [28], and Rocca [31]. Limb and Murphy [37] developed a method to obtain an estimate of velocity of a single moving object undergoing pure horizontal translation in a sequence of frames. Let $S_k(x,y)$ be the luminance at location (x,y) in frame k and $S_{k-1}(x,y)$ be the luminance value at the previous frame. Let FD represent the temporal gradient between $S_k(x,y)$ and $S_{k-1}(x,y)$, and ED & LD represent spatial luminance gradients around location (x,y) in frame $k-1$. The components of the displacement vector D are $\hat{dx}$ and $\hat{dy}$ and can be given as follows,

$$\hat{dx} = \frac{\sum_{M} |FD|}{\sum_{M} |ED|} \tag{2.1}$$

Here the $|FD|$ is the magnitude of the frame difference signal, $|ED|$ is the magnitude of the element difference signal and the summation is done over the entire moving area M. A pixel located at (x,y) in frame k belongs to the moving area M if the frame difference $|FD|$ for it is greater then a preset threshold T.

$$\hat{dy} = \frac{\sum_{M} |FD|}{\sum_{M} |LD|} \tag{2.2}$$

This algorithm can be easily understood if we consider a single edge moving in x direction. Then in eqn. (2.1) we can interpret $\sum_{M} |FD|$ to be the area of a parallelogram which is generated by the horizontal movement of this edge between the frames, $\sum_{M} |ED|$ to be the height and the required displacement estimate to be the base of parallelogram.

2.3.2 Differential Technique

Cafferio and Rocca [39],[40] start with a different approach and show that Limb and Murphy's procedure is a special case of their general approach. Let $S_k(x,y)$ be the luminance at location (x,y) in frame k and $S_{k-1}(x,y)$ be the luminance value at the previous frame. For an object that moves in translation with a displacement vector D, and neglecting the uncovered background the frame difference signal can be written as,

$$FD(x,y) = S_k(x,y) - S_{k-1}(x,y) \tag{2.3}$$

However,

$$S_{k-1}(x,y) = S_k(x+dx,y+dy) \tag{2.4}$$

substituting in eqn. (2.3) we obtain,

$$FD(x,y) = S_k(x,y) - S_k(x+dx,y+dy) \tag{2.5}$$

Using Taylor's series expansion we have,

$$S_k(x+dx,y+dy) = S_k(x,y) + \frac{\partial S_k(x,y)}{\partial x}dx + \frac{\partial S_k(x,y)}{\partial y}dy + \eta(x,y) \tag{2.6}$$

where $\eta(x,y)$ represents higher order terms. Solving from above and neglecting the higher order terms we have,

$$FD(x,y) = -\frac{\partial S_k(x,y)}{\partial x}dx - \frac{\partial S_k(x,y)}{\partial y}dy \tag{2.7}$$

In the matrix notation we have,

$$FD(x,y) = -\begin{bmatrix} dx & dy \end{bmatrix}\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} S_k(x,y) \tag{2.8}$$

or,

$$FD(x,y) = -[D]^T \nabla S_k(x,y) \tag{2.9}$$

where ∇S_k is the gradient of S_k with components element difference (ED), $\frac{\partial S_k(x,y)}{\partial x}$ and line difference (LD), $\frac{\partial S_k(x,y)}{\partial y}$.

If the displacement is only along horizontal direction we have,

$$\hat{D} = \hat{dx} = \frac{\frac{-FD(x,y)}{\partial S_k(x,y)}}{\partial x} \tag{2.10}$$

For two dimensional displacement we obtain by linear regression and by neglecting the cross terms,

$$\hat{dx} = -\frac{E\left[FD(x,y)\,\frac{\partial S_k(x,y)}{\partial x}\right]}{E\left[\left(\frac{\partial S_k(x,y)}{\partial x}\right)^2\right]} = -\frac{\sum_M FD(x,y)\,ED(x,y)}{\sum_M (ED(x,y))^2} \tag{2.11}$$

and,

$$\hat{dy} = -\frac{E\left[FD(x,y)\,\frac{\partial S_k(x,y)}{\partial y}\right]}{E\left[\left(\frac{\partial S_k(x,y)}{\partial y}\right)^2\right]} = -\frac{\sum_M FD(x,y)\,LD(x,y)}{\sum_M (LD(x,y))^2} \tag{2.12}$$

In the matrix notation we now have,

$$\hat{D} = -\begin{bmatrix} \frac{\sum_M FD(x,y)\,ED(x,y)}{\sum_M (ED(x,y))^2} \\ \frac{\sum_M FD(x,y)\,LD(x,y)}{\sum_M (LD(x,y))^2} \end{bmatrix} \tag{2.13}$$

To simplify further we define,

$$sign(x) = \begin{cases} 0 & \text{if } z=0 \\ \dfrac{z}{|z|} & \text{otherwise } z \neq 0 \end{cases} \tag{2.14}$$

$$\hat{D} = -\left[\begin{array}{c} \dfrac{\sum_M FD(x,y)\ sign(ED(x,y))}{\sum_M |ED(x,y)|} \\ \dfrac{\sum_M FD(x,y)\ sign(LD(x,y))}{\sum_M |LD(x,y)|} \end{array} \right] \tag{2.15}$$

2.3.3 Block Matching by Recursive Method

The previous algorithm works for small displacements only. As motion (D) increases quality of approximation becomes poor. Netravali and Robbins [42] and Cafferio and Rocca [41] recommended use of recursive technique for better estimation. Linearizing the displacement estimate around an initial value,

$$\hat{D}_i = \hat{D}_{i-1} + U_i \tag{2.16}$$

where $\hat{D}_i$ and, $\hat{D}_{i-1}$ are the displacement estimates at i and $i-1$ iterations and U_i is the update term. We now define the displaced frame difference,

$$DFD(x,y,\hat{D}_{i-1}) = S_k(x,y) - S_{k-1}(x-\hat{d}x_{i-1}, y-\hat{d}y_{i-1}) \tag{2.17}$$

We use $DFD(x,y,\hat{D}_{i-1})$ to compute $\hat{D}_i$; substituting in eqn. (2.17) we obtain,

$$DFD(x,y,\hat{D}_{i-1}) = S_k(x,y) - S_k(x+dx-\hat{d}x_{i-1}, y+dy-\hat{d}y_{i-1}) \tag{2.18}$$

By Taylor's Series expansion after neglecting the higher order terms we have,

$$S_k(x+(dx-\hat{d}x_{i-1}), y+(dy-\hat{d}y_{i-1})) = S_k(x,y) - (dx-\hat{d}x_{i-1})\frac{\partial S_k(x,y)}{\partial x} - (dy-\hat{d}y_{i-1})\frac{\partial S_k(x,y)}{\partial y} \tag{2.19}$$

Substituting from eqn. (2.18) in above and solving we get,

$$DFD(x,y,\hat{D}_{i-1}) = -(dx-\hat{d}x_{i-1})\frac{\partial S_k(x,y)}{\partial x} - (dy-\hat{d}y_{i-1})\frac{\partial S_k(x,y)}{\partial y} \tag{2.20}$$

This can be written in matrix notation as,

$$DFD(x,y,\hat{D}_{i-1}) = -\begin{bmatrix} dx-\hat{d}x_{i-1} & dy-\hat{d}y_{i-1} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} S_k(x,y) \tag{2.21}$$

or,

$$DFD(x,y,\hat{D}_{i-1}) = -[D-\hat{D}_{i-1}]^T \nabla S_k(x,y) = [\hat{D}_{i-1}-D]^T \nabla S_k(x,y) \tag{2.22}$$

where ∇S_k is the gradient of S_k with components element difference (ED), $\frac{\partial S_k(x,y)}{\partial x}$ and line difference (LD), $\frac{\partial S_k(x,y)}{\partial y}$.

By linear regression and neglecting the cross terms, the displacement estimates in x and y direction can be written as,

$$\hat{d}x = \hat{d}x_{i-1} - \frac{E\left[DFD(x,y,\hat{D}_{i-1})\,\frac{\partial S_k(x,y)}{\partial x}\right]}{E\left[\left(\frac{\partial S_k(x,y)}{\partial x}\right)^2\right]} = \hat{d}x_{i-1} - \frac{\sum_M DFD(x,y,\hat{D}_{i-1})\,ED(x,y)}{\sum_M (ED(x,y))^2} \tag{2.23}$$

$$\hat{d}y = \hat{d}y_{i-1} - \frac{E\left[DFD(x,y,\hat{D}_{i-1})\,\frac{\partial S_k(x,y)}{\partial y}\right]}{E\left[\left(\frac{\partial S_k(x,y)}{\partial y}\right)^2\right]} = \hat{d}y_{i-1} - \frac{\sum_M DFD(x,y,\hat{D}_{i-1})\,LD(x,y)}{\sum_M (LD(x,y))^2} \tag{2.24}$$

In the matrix notation we have,

$$\hat{D} = \hat{D}_{i-1} - \left[\begin{array}{c} \dfrac{\sum_{M} DFD(x,y,\hat{D}_{i-1})\, ED(x,y)}{\sum_{M} (ED(x,y))^2} \\ \dfrac{\sum_{M} DFD(x,y,\hat{D}_{i-1})\, LD(x,y)}{\sum_{M} (LD(x,y))^2} \end{array} \right] \tag{2.25}$$

This can be simplified as before,

$$\hat{D} = \hat{D}_{i-1} - \left[\begin{array}{c} \dfrac{\sum DFD(x,y,\hat{D}_{i-1})\, sign(ED(x,y))}{\sum_{M} |ED(x,y)|} \\ \dfrac{\sum_{M} DFD(x,y,\hat{D}_{i-1})\, sign(LD(x,y))}{\sum_{M} |LD(x,y)|} \end{array} \right] \tag{2.26}$$

2.3.4 The Theory of Steepest Descent

From [36] we briefly review the technique of steepest descent, before proceeding with the *pel-recursive algorithm.* For an adaptive system, if output of a system can be iteratively adjusted to obtain the desired response by minimizing an error signal, it implies that the mean square of error is a quadratic function of weights. This function never goes negative, and can be pictured to form a concave hyper-paraboloidal shaped surface. Adjusting the weights involves descending on this surface with the objective of reaching the unique minimum point (or the bottom of the bowl). Gradient methods are commonly used for this purpose. A well known method for adjusting the response of an adaptive system is the steepest-descent method. Adaptation by this method starts with an arbitrary initial value of the weight vector. The gradient of the mean square error function is then measured, and weight vector altered in accordance with negative of its value. This procedure is

repeated, causing error to be successively reduced as the weight-vector approaches the optimal value. In a general form the method of steepest descent is given by:

$$W_j = W_{j-1} + \mu(-\nabla_j) \tag{2.27}$$

where W_j is the present weight vector, W_{j-1} is the previous weight vector, μ controls the stability and the rate of convergence and ∇ is the gradient on the error surface. The term *steepest descent* arises as descent along the negative gradient maximizes the reduction in error function generated by first order Taylor series approximation. In the LMS algorithm for measurement of this gradient an estimate can be obtained by squaring the single value of ξ_i and differentiating it as if it were the mean square error.

$$\hat{\nabla}_i = \begin{bmatrix} \frac{\partial}{\partial w_1} \\ \cdot \\ \cdot \\ \frac{\partial}{\partial w_n} \end{bmatrix} \xi_i^2 = 2\xi_i \nabla_W \xi_i \tag{2.28}$$

Here $\nabla_W \xi_i$ can be obtained from approximation of the derivatives as,

$$\frac{\partial \xi}{\partial w_k} = \frac{\xi(w_k + a) - \xi(w_k - a)}{2a} \tag{2.29}$$

where k can take any value from 1 to i. Now the equation for steepest differential descent can be written as:

$$W_i = W_{i-1} - 2\mu(\xi_i \nabla_W \xi_i) \tag{2.30}$$

As a new estimate is obtained with each data sample an adaptive iteration is effected with arrival of each sample, so index j has been replaced by index i. Thus to approach the optimal point weights are adjusted and correction is made in a direction so as to to achieve the unique minimum of parabolic surface.

2.3.5 Pel Recursive Technique

This technique was developed by Netravali and Robbins [42]. If the displaced frame difference is defined by DFD(.) as before, and the previous recursive estimate in either the horizontal, vertical or temporal direction is given by $\hat{D}_{i-1}$ then the new estimate will be,

$$\hat{D}_i = \hat{D}_{i-1} + U_i \tag{2.31}$$

Where U_i is an update term. This term is got here by minimizing recursively $(DFD(x,y,\hat{D}_{i-1}))^2$. using the steepest descent algorithm explained above. Following method of steepest descent we have,

$$\hat{D}_i = \hat{D}_{i-1} - \frac{\epsilon}{2} \nabla_{\hat{D}_{i-1}} (DFD(x,y,\hat{D}_{i-1}))^2 \tag{2.32}$$

Further,

$$\hat{D}_i = \hat{D}_{i-1} - \epsilon\, DFD(x,y,\hat{D}_{i-1})\, \nabla_{\hat{D}_{i-1}} (DFD(x,y,\hat{D}_{i-1})) \tag{2.33}$$

and,

$$\hat{D}_i = \hat{D}_{i-1} - \epsilon\, DFD(x,y,\hat{D}_{i-1})\, \nabla S_{k-1}(x-\hat{dx}_{i-1}, y-\hat{dy}_{i-1}) \tag{2.34}$$

where,

$$\nabla S_{k-1}(x-\hat{dx}_{i-1}, y-\hat{dy}_{i-1}) = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} S_{k-1}(x-\hat{dx}_{i-1}, y-\hat{dy}_{i-1}) \tag{2.35}$$

To smooth out the effect of noise the update can be calculated from weighted estimates of several pixels as follows,

$$\hat{D}_i = \hat{D}_{i-1} - \frac{\epsilon}{2} \nabla_{\hat{D}_{i-1}} \sum w_j (DFD(x,y,\hat{D}_{i-1}))^2 \tag{2.36}$$

Here $\sum w_j = 1$ and $j \in M$.

We note, that the evaluation of DFD and ∇S_{k-1} requires bilinear interpolation of luminance for non-integral displacement vector $\hat{d}x_i$ and $\hat{d}y_i$. This can be done by,

$$S_{k-1}(x-\hat{d}x_i, y-\hat{d}y_i) = (1-\delta_y)[(1-\delta_x)S_A + \delta_x S_B] + \delta_y[(1-\delta_x)S_C + \delta_x S_D] \quad (2.37)$$

where δ_x and δ_y are the fractional displacements in x and y directions, and S_A, S_B, S_C, S_D are the luminance values at pels A, B, C, D in the previous frame which surround the location defined by the displacement vector $\hat{d}x_i$, $\hat{d}y_i$.

Pel recursive technique has been used for estimating the motion of component color signals in [43], and has also been adopted in [44] as part of a proposed video-codec for 50Kb/s transmission-rate. In [45] a general recursive-technique that also uses gradients is discussed.

2.3.5.1 Modifications

Several suggestions have been made [21],[41],[46],[48] to improve the rate of convergence and accuracy of pel-recursive algorithms. These modifications differ in the assumptions used in computing the update term in the algorithm. It has been recommended that ϵ should be made a variable instead of being a constant, and should reflect the effect of variations because of second order luminance gradients.

2.4 SEARCH WITHOUT DERIVATIVES: Matching

We now consider ways of estimating displacements in a sequence of frames by search techniques that do not use derivatives. Area correlation and block matching are primary examples of this type of technique.

2.4.1 Area Correlation Techniques

Cross correlation measure has been used for motion estimation between two images in

aerial guidance, video-conferencing and other applications. In the cross correlation measurement between two images, the location of the peak of the correlation function gives the estimate of displacement. Generally cross-correlations can be computed via the fast Fourier transform (FFT). The accuracy of area correlation methods is considered to be poor when the block size is small and blocks are not undergoing pure translation [49].

2.4.2 Distortion Function and Search Area

In block matching each frame is divided in to blocks of fixed size. A criterion called the *distortion function*, is then selected to determine when we have the *best match*. If $(u,v) \in$ SR, and pel $s_k(.)$ and pel $s_{k-1}(.)$ belong to frames k and k-1 respectively then some distortion functions $DF(.)$ [21],[1] for frame to frame matching can be stated as,

(i) normalized cross-correlation function (NCCF):

$$NCCF(u,v) = \frac{\sum_{y=1}^{M}\sum_{x=1}^{N} s_k(x,y)s_{k-1}(x+u,y+v)}{\left[\sum_{y=1}^{M}\sum_{x=1}^{N} s_k^2(x,y)\right]^{1/2}\left[\sum_{y=1}^{M}\sum_{x=1}^{N} s_{k-1}^2(x+u,y+v)\right]^{1/2}} \tag{2.38}$$

(ii) mean square error (MSE):

$$MSE(u,v) = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}\left[s_k(x,y) - s_{k-1}(x+u,y+v)\right]^2 \tag{2.39}$$

If $\sum\sum s_{k-1}^2(.,.)$ is spatially invariant in the match area, MSE is then equivalent to NCCF.

(iii) mean absolute frame difference (MAD):

$$MAD(u,v) = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N} | s_k(x,y) - s_{k-1}(x+u,y+v) | \qquad (2.40)$$

(iv) Number of thresholded Differences (NTD):

$$NTD(u,v)= \sum_{y=1}^{M}\sum_{x=1}^{N} \left[N(T_0, f(s_k(x,y) - s_{k-1}(x+u,y+v)) \right] \text{ and } N(T_0,f(.)) = \begin{cases} 1 & T_0 < f \\ 0 & T_0 \geq f \end{cases} \qquad (2.41)$$

Here $f(.)$ in NTD could be either the MSE or MAD function. NTD can be interpreted as a counting function of thresholded differences.

Next, we need to know as to how many times the distortion function is to be evaluated per block. To understand this consider, a *search region SR* in which every point contained is a candidate for being an cptimal displacement vector. Say, that we are seeking a match for a $N \times N$ block from present frame in to a SR in the previous frame and let p be the maximum displacement allowed in any direction for that block. This means that the size of the match area in previous frame will have to be $(N+2p)^2$ to allow detection of optimal displacement vector. Also the number of evaluations required for the distortion function per block will be given by the number of points contained in the *SR* i.e. $(2p+1)^2$. Figure 2.1 shows possible match area in the previous frame where the best match of a block could be located, and the corresponding displacement vector; Figure 2.2 shows an expanded view. We discuss the issue of search complexity in detail in the next chapter. For now, it is important to realize at this point that if the maximum displacement of $p = \pm 6$ is allowed for a block belonging to the current frame, then we need to compute the distortion function at 169 possible displaced locations in the previous frame. This computation has to be performed for every block in the frame, and for every frame in the image-sequence. Therefore the enormous computational burden caused by *exhaustive matching* become apparent. Fortunately, *search schemes* come to the rescue, such that based on certain

assumptions regarding the distortion function a *reduced search* that eases the computational burden by a large extent can be performed effectively.

2.4.3 Independent Block-Matching Motion-Estimation

A distortion function is evaluated at several displaced locations in the previous frame. Depending on the distortion-function selected the maximum or minimum value of this function gives an estimate of motion. The search proceeds in several steps where depending on the outcome of previous step the search is directed towards the correct direction in the following step. The procedure is continued until the allowed match area from previous frame has been covered.

2.4.3.1 Log Search Algorithm

Jain and Jain [49] use a 2D-log search procedure to search for best displacement estimate. They assume a MSE distortion criterion DF(.), and search for a direction of minimum distortion (DMD). They further assume that DF(.) increases monotonically as one moves away from DMD and state that such an assumption is satisfied for most images as their covariance is usually a decreasing function of displacement in each of the quadrants of search.

Using a *log-search* scheme in two dimensions, and having chosen MSE as the distortion measure, the search proceeds seeking to obtain the minimum of the MSE function. This algorithm can best be explained by an example. Let a maximum displacement of ± 6 in any direction (a typical value) be allowed for a block from present frame into a possible search region from previous frame. The search starts in the first step seeking a minimum direction by comparing value of MSE for the block at 5 search positions (points). The minimum value of the MSE function identifies the

correct direction of approach. In the next step around the minimum value obtained from the previous step, values of distortion (MSE) at 3 new search positions are compared to obtain direction of updated minimum. This procedure is continued until the plane of the search reduces to 3×3 size. At this point value of MSE at the remaining 8 neighboring points are compared to locate an overall minimum. The location corresponding to this final minimum MSE is called the *optimal displacement vector*. An example of search by this algorithm is shown in Figure 3.3(a), where the maximum displacement allowed is ±6 in any direction (a typical value). A step number uniquely identifies all the positions searched at that step, and the order in which the search progresses. An X marks the location of minimum-point at each search step, an o marks the other search points at that step. The best-match location obtained by searching the entire area is also shown in the final step.

2.4.3.2 Three Step Search

This scheme was devised by Koga et al [50]. It aims to provide a fixed regular procedure well suited for hardware implementation. Here the search for the optimal displacement is done in a predetermined number of steps. Moreover each step contains a fixed number of search points. The number of steps are kept few to allow employing of only a few decision making stages in hardware implementation. A nonlinear function (called cost function) approximating the error power was selected. A search for an approximate displacement vector is then conducted using this function and a coarsely spaced search region. In the next step a more accurate displacement is obtained by conducting a finer search around the vector obtained from previous step. This next step is repeated until the displacement estimate with required accuracy is obtained.

For the case of $p=\pm 6$ search is always done in three steps and except for the first step a set of 8 search positions is allowed in each step. In the first step in addition to 8 search positions (points) an additional search position corresponding to zero displacement vector is also used. The coarse displacement estimate obtained from the first step is refined by performing a search around the estimate obtained from the first step. In the third step and the final step all the neighboring 8 positions around the minimum from the second step are checked to obtain the overall optimal displacement vector. An example of this procedure for a maximum allowed displacement of $p=\pm 6$ in any direction is shown in Figure 3.3(b).

2.4.3.3 Modified Log Search

Kappagantula and Rao [56] combine the *log-search* and the *three step* scheme and propose a scheme that performs on the average better then either of the two schemes. They use mean of absolute difference (MAD) as the distortion function and perform a search that uses thresholds at various stages to determine if sufficiently precise displacement estimates have been obtained.

The search typically starts out in the first step by comparing MAD values at four search positions in a diamond-shaped formation around, and in addition to, the zero displacement position. an initial direction of minimum is therefore identified. Next, MAD values at two additional locations around the position where MAD function had minimum value in the first step are computed. Based on the comparisons, the direction of minimum from first step is modified if necessary. In the next step, spacing between search positions is reduced and we again compare MAD values at four search positions with that of minimum MAD value from previous step, as before to obtain modified direction of minimum. Next, MAD values at two additional search

positions are compared to the minimum obtained so far and the direction of search updated. The two previous steps are repeated with reduced spacing until the minimum of the MAD with in the search region is found. The displacement vector at this minimum is then taken to be the optimum estimate. An example of this procedure for a maximum allowed displacement of $p = \pm 6$ in any direction is shown in Figure 3.3(c). This scheme would use twice as many search steps as the 3 step search if thresholding at various stages is not allowed, and the savings in the number of search points would not be very significant as compared to the *three-step* scheme or the *log-search* scheme. Use of thresholds to exit search from a stage may cause premature conclusion and may change the result of the search. The penalty with premature exits may be quite significant at times.

2.4.3.4 One at a Time Search

Srinivasan and Rao [55] propose a search from optimization theory called *one at a time search*. This scheme also uses the MAD as the distortion function and aims to find a minimum first in the x-direction and then in the y-direction. It therefore searches one direction first until it reaches the global minimum for the MAD function in that direction and then searches the conjugate direction for the same.

It is proposed that a search for minimum direction start at the middle of the search area and three points in the horizontal direction, centered at and including the zero displacement vector, be checked to obtain an initial estimate of direction. In the second step an additional search position is checked and compared with the past position. If the new search position is yields a MAD value larger then the previous position then search for horizontal direction has just ended and a similar search in the vertical direction is then performed. Otherwise, if MAD function evaluated at the new

search position yields a smaller value then the second step is repeatedly applied to evaluate the value of MAD at adjacent displacement locations. Search continues in this direction until a minimum value of the MAD is bounded by 2 higher values or we reach end of the search region in this direction. A similar search is then conducted in the conjugate direction. An example of this procedure for a maximum allowed displacement of $p=\pm 6$ in any direction is shown in Figure 3.3(d). We note that one major problem in this approach may be that the noise may influence the search at any step and the search may get misdirected as it requires that the MAD function display a well defined monotonic behavior at each point in the search region. We know that this is a difficult condition to meet as neighboring displacement vectors often generate nearly similar MAD values.

2.4.3.5 Orthogonal Search

All possible requirements desired from a good search algorithm are difficult to satisfy due to inherent conflicts in them. A search scheme for instance requires that it should be possible to obtain the optimal displacement vector in minimum number of search steps, while minimizing the number of search points. It is also required that an algorithm be insensitive to noise and produce robust estimates. The goal of this scheme [101] is to satisfy many of the above desired requirements with some compromise on others that cannot be fully satisfied. This scheme can be considered to be an adaptation of *three-step* scheme as it performs coarser to finer search in fixed number of steps, applied for searching the space in perpendicular directions such as in *conjugate-search* scheme. It however outperforms many search-schemes in its ability to locate the optimum point, while it maintains a complexity comparable or less than the schemes discussed. Further details of *orthogonal-search* scheme are discussed in

the next chapter. Comparison of performance of this scheme with others, on the basis of number of search points and number of search steps is shown in Table 3.1 (at the end of next chapter).

2.4.4 Dependent Block Matching Motion Estimation

So far we have discussed various algorithms for independent displacement vector estimation for each of the blocks belonging to a frame. However if the block sizes are kept small then the displacement estimates of spatially or temporally adjacent blocks are usually quite correlated. This assumption is used in following procedures for motion estimation.

2.4.4.1 Motion With an Initial Estimate

Ninomiya and Ohtsuka's proposal [51] starts with choice of a logarithmic measure as their matching criterion. Next they propose an iterative scheme consisting several stages, where each stage contains some allowed displacement vectors with higher and higher resolution to estimate motion accurately. They quickly point out that for real-time hardware implementation only one such stage could be allowed, with menu consisting of 25 displacement vectors many selected corresponding to displacements of ± 2 and a few corresponding to displacements of as much as ± 5. It was recommended that displacement estimates of the previous field block, identically located, as the current field block could be used as an initial estimate around which the best vector minimizing the matching function could be selected from the menu to give an estimate of the relative motion between the fields. This approach is reasonable when no temporal subsampling is employed, however many low bit-rate video-conferencing applications require 4:1 (or even higher) temporal subsampling rate due to

compression requirements, and in these circumstances a much more elaborate procedure will be needed for motion estimation.

Matsuda et. al. [54], suggest a solution to the problem of temporal subsampling and motion estimation. First they propose to evaluate the MAD function at various displaced positions in the reference block set (search area) which consists of area belonging to the previous field for moving area blocks, and one previous frame block from the same location as the current frame block for the stationary area. The idea behind using previous field reference space is to keep the hardware complexity low as only small displacements will be allowed. Next, they suggest a way of extending their approach to estimate displacement for a temporal subsampled sequence, without increasing the hardware complexity. To prevent an increase in size of search area and hence the hardware when temporal subsampling is allowed, they introduce a vector tracking method. In this method skipped fields are used to compute the incremental displacements by using the motion vector of a block at the same spatial location in the previous field as the current field block and can be used as an initial estimate around which a new search region of small size is defined. A minimum value of MAD function in the small region of allowed displacement vectors gives the optimal displacement vector. These incremental displacement vectors can be combined to obtain a single estimate for the transmitted fields. Even though an attempt to track motion on a field by field basis is made here, this approach also has some drawbacks. First, the motion estimation algorithms have to operate at normal rate even when temporal subsampling is used and second, the total displacement vector between two available fields may not accurately be representable by sum of several *quantized* displacements.

2.4.4.2 Reduced Search With an Initial Estimate

Recently techniques [102],[103] have been proposed which consider various problems such as effects of temporal subsampling, restriction of search to a small area, keeping the complexity of implementation and transmission overhead low and offer possible solutions to them. These techniques first attempt to make previous estimates more reliable before using them as initial estimates for the current block. This is done by employing an averaged estimate of displacements from the previous moving area such that motion estimation procedure becomes somewhat insensitive to noisy initial estimates. For the case of temporal subsampling, next, a search scheme is used to reduce the number of computations necessary. The previous estimates could belong to the previous field or the present field. These algorithms are discussed in more detail in the next chapter.

2.5 MISCELLANEOUS TECHNIQUES

Several other techniques have also been proposed for motion estimation. Such techniques are still in a stage of infancy and are only briefly mentioned here.

First technique under this category is based on contour [2] or feature extraction [21]. The main principle behind such a scheme is that separation of significant derivative information yields information regarding the contours (or edges) and then matching techniques can then be employed to obtain displacement estimates at the contours. These estimates can be used to interpolate a full displacement field for the entire image.

Next we consider a hierarchical motion estimation technique [100] where displacement estimates are first obtained using large block sizes and can be refined subsequently by

going to smaller block sizes if needed. This technique uses a gradient measurement method [59] to compute displacements.

2.6 PROBLEMS AND PROSPECTS: A DISCUSSION

Almost all of the above motion estimation techniques neglect the the effect of gain variation in a sequence of images. Changes in the gain would be incorrectly classified as motion in most cases and would probably cause motion estimator to give inconsistent estimates. Further, the effect of *uncovered area* has so far been neglected which also gives rise to inconsistent displacement estimates. It is also not clear as to how to best compensate for rotations which are often encountered even in typical video-conferencing scenes. Segmenting of various kinds of areas where motion estimates are inconsistent is expensive as it requires extra overhead.

We also notice that gradient schemes seem to offer motion estimates to fractional pel accuracy. These schemes when used in interframe coding applications do not need any overhead information for transmission of displacement estimates. For these schemes to work well reasonable luminance gradients are necessary otherwise displacement estimates from these schemes tend to be noisy. The weighted error from neighboring area may need to be minimized to eliminate the effect of noise. Interpolation to get pel values at non integer locations is often required. The derivative methods are thus *complex* and maybe less suitable for real-time video applications. The simplifications to gradient schemes, often suggested for hardware implementation, seem to seriously effect the performance.

The *basic matching schemes* require excessive computations to estimate motion of a block. Reduced search schemes can be quite effective in reducing the number of

computations necessary. *Matching schemes* usually require displacement vectors to be transmitted explicitly and this overhead can become quite significant. Various ways of reducing this overhead are currently under study by several researchers and the results seem promising.

We have reviewed various techniques for motion estimation. By careful consideration of these techniques we have tried to analyze the inherent assumptions common in these techniques, and emphasize various relationships that exist between them.

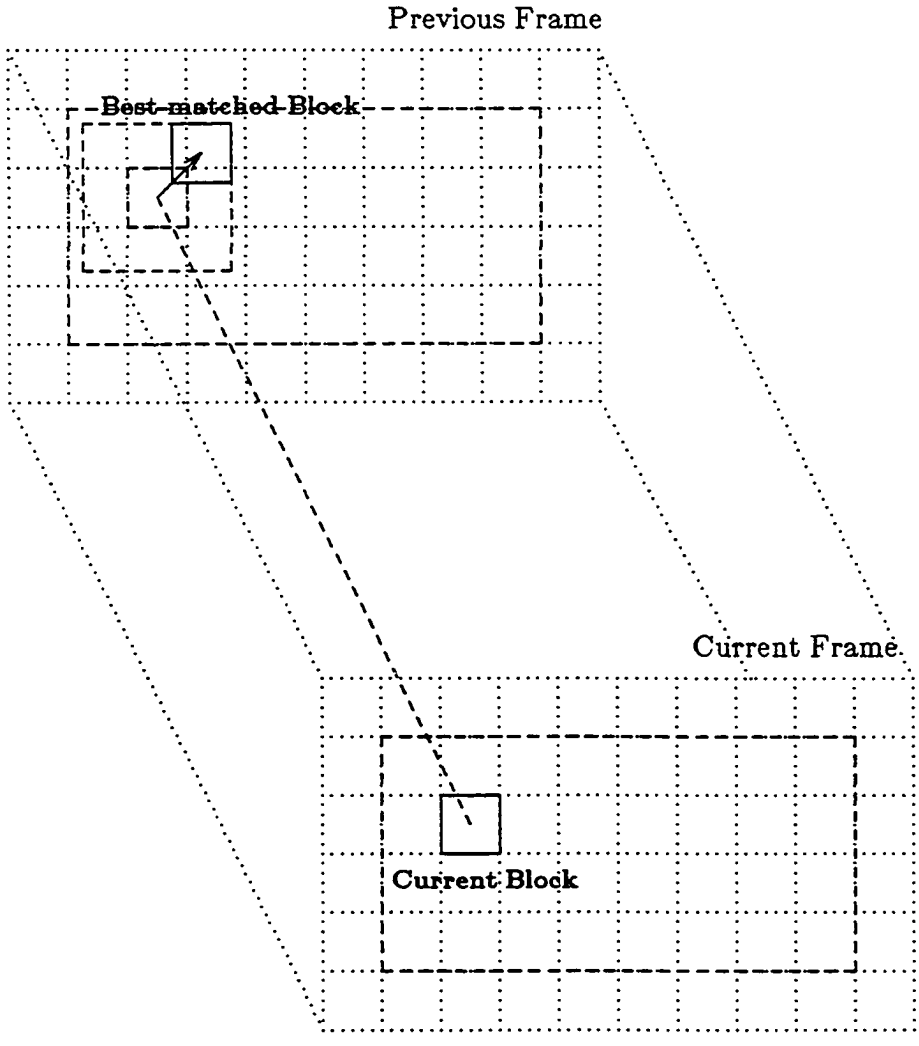
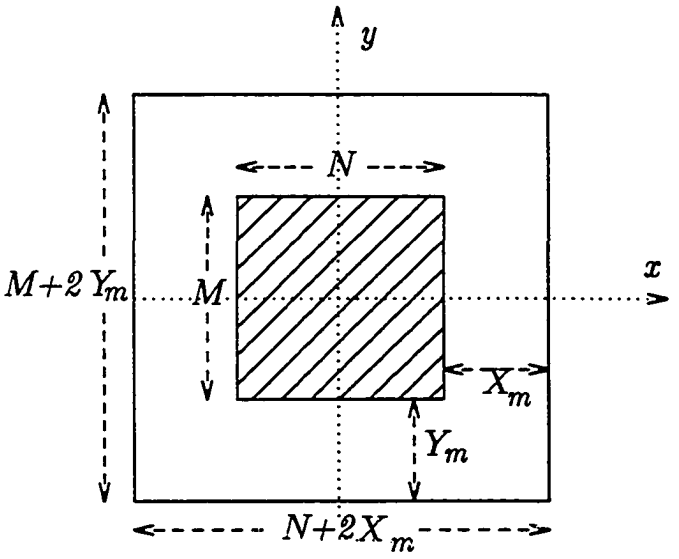
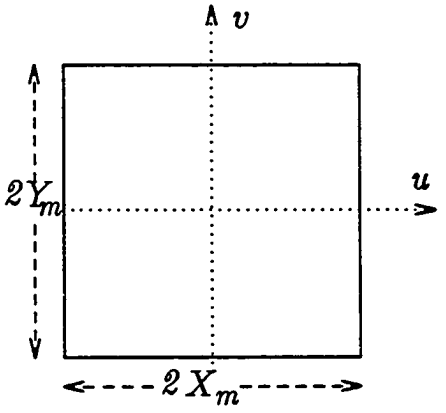


Figure 2.1 Current and Previous frames for Block Motion Estimation



(a)



(b)

Figure 2.2 Match Area and Search region in Block Motion Estimation

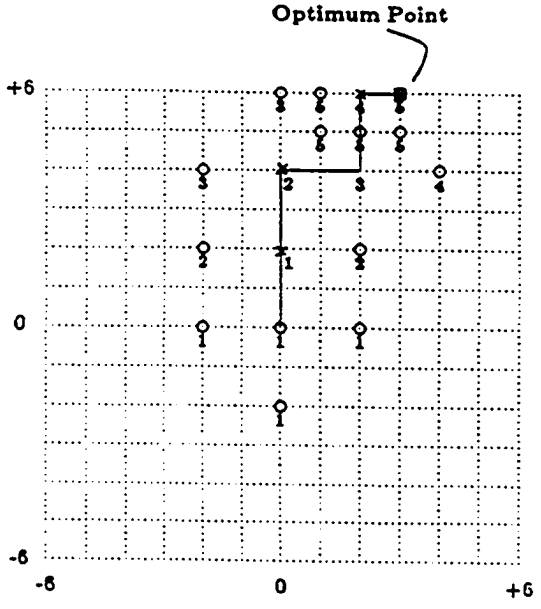


Figure 2.3(a) 2D-Log Search

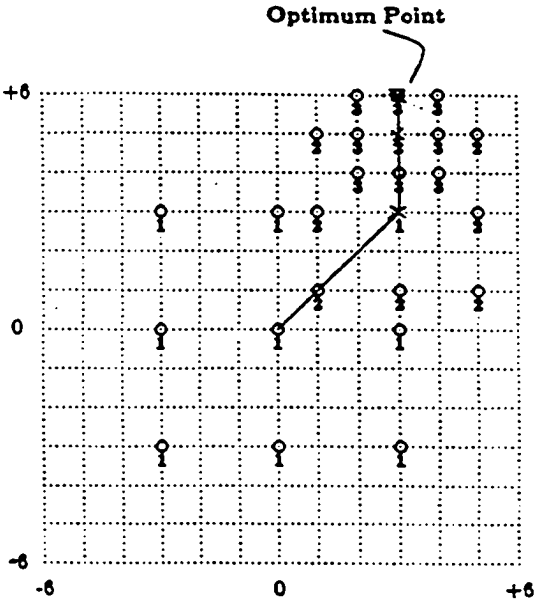


Figure 2.3(b) Three Step Search

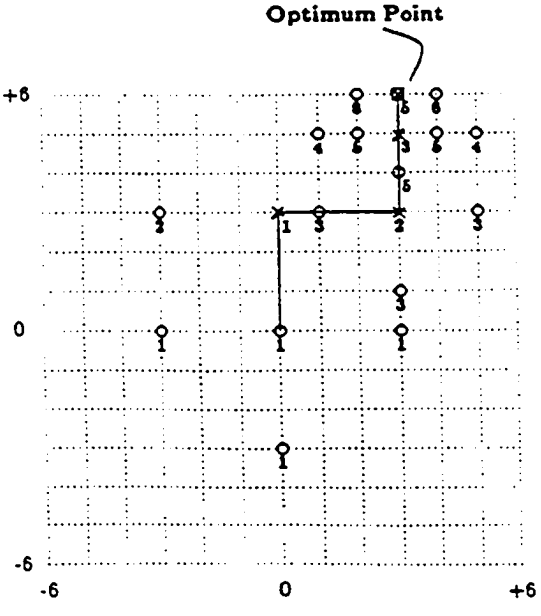


Figure 2.3(c) Modified Log Search

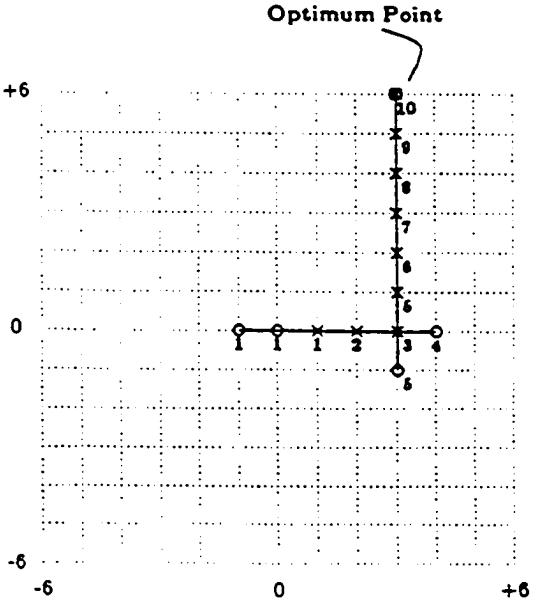


Figure 2.3(d) One-at-a time Search

CHAPTER 3: EFFICIENT MOTION-COMPENSATION AND STATISTICAL ANALYSIS

3.1 INTRODUCTION

Motion in many video-conferencing scenes is relatively small and can be modeled to be translatory in nature. For such an application primary goal of a motion estimation procedure may be to allow efficient compensation for image-sequence compression. The differences between the current frame of an image-sequence and the motion-compensated previous frame are usually small and can be much efficiently encoded. Thus, a movement compensation procedure can be regarded as an effective adaptive prediction technique in image-coding applications . When motion is complex in nature because of rotations, scale changes of objects, appearance of uncovered area etc, good estimation is difficult and so the prediction for coding purposes is not likely to be very good. If estimation and compensation of motion is done for a group of pels then the size of the group (or block) has to be kept small to allow compensation for multiple moving objects, small rotations etc. Because of various practical reasons many block coding techniques for low bit rate coding employ blocks of size 8×8. This has been found to be a reasonable tradeoff between the reliability of a motion estimate, ability to compensate for various types of motion likely to be encountered and overhead costs.

A complete block motion-compensated coding system consists of three stages: (1) a *motion detector* which detects the moving blocks, (2) a *displacement estimator* which estimates the displacement vectors of moving blocks, and (3) a *data compression*

algorithm which encodes the pel differences after motion compensation.

3.2 THE MOTION DETECTOR AND SEARCH RANGE

The motion detector is used to first classify a block to be *moving* or *non-moving* before any motion estimation can be done for that block. This is essential as a few non-moving blocks might otherwise end up being classified as moving just because they happen to contain areas of low detail and we may find a slightly better match for them in the previous frame due to noise. This would unduly increase the number of blocks that need motion estimation and compensation and coding. The motion detector is thus necessary to ensure a zero displacement vector for the unchanged region. The design of our motion detector is motivated by [1],[57]. Two parameters, T_0 and N_0, are used. Pel (x,y) is called *changed* if $|s_k(x,y) - s_{k-1}(x,y)| > T_0$, where pels $s_k(.,.)$ and $s_{k-1}(.,.)$ are pels in frames k and k-1 respectively. A block is called *moving* if the number of changed pels in that block is greater than or equal to N_0. For the image-sequence "MsUSA", the motion-detection parameters, $T_0=3$ and $N_0=10$ work adequately for 8 bit pels in the range 0-255 when the block size is 8×8. In the "Salesman" sequence, somewhat higher (camera) noise is present, and so motion-detection thresholds are modified to $T_0=6$ and $N_0=7$.

The goal of a displacement estimator is to find the best match of a *reference block* from frame k in a suitable *match area* in the previous frame k-1. The detectable displacement is assumed to be less than X_m along horizontal (x) axis and Y_m along vertical (y) axis. If the size of a reference block is $M\times N$, the match area is $(M+2X_m)\times(N+2Y_m)$ where X_m and Y_m assume discrete value for digitized images.

The above problem consists of seeking for a best match location by evaluating a

match function at every point in the *displacement search region* (SR) of size $(2X_m)\times(2Y_m)$ (which has $(2X_m+1)\times(2Y_m+1)$ points). This is illustrated in Figure 3.1. In most practical cases, M is often chosen to be the same as N and $X_m = Y_m = p$ and hence SR can have $(2p+1)^2$ points. Each point in SR is a candidate for being the correct displacement vector and is called a *search point. Fast search algorithms* are invented to obtain the optimum (best match) point in SR with fewer computations than that required by the exhaustive search. They achieve this goal by performing the search in a few stages, such that the result of each stage guides the search towards the optimal point. Each stage is referred to as a *search step.*

3.3 CRITERION FOR MATCH

A criterion called, *distortion function,* has to be selected to decide as to when we have the *best match.* Some distortion functions $DF(.)$ for frame to frame matching were stated in [Chapter 3], [21],[57]. Here we state two simple distortion functions that we find quite useful. If $(u,v) \in$ SR, and pels $s_k(.,.)$ and pel $s_{k-1}(.,.)$ belong to frame k and k-1 respectively then,

(i) mean absolute frame difference (MAD):

$$MAD(u,v) = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N} |\, s_k(x,y) - s_{k-1}(x+u,y+v)\,| \tag{3.1}$$

(ii) Number of thresholded Differences (NTD):

$$NTD(u,v) = \sum_{y=1}^{M}\sum_{x=1}^{N}\Big[N(T_0, f(s_k(x,y) - s_{k-1}(x+u,y+v))\Big] \text{ and } N(T_0,f(.)) = \begin{cases} 1 & T_0 < f(.) \\ 0 & T_0 \geq f(.) \end{cases} \tag{3.2}$$

Here $f(.)$ in NTD could be the MAD function. NTD can then be interpreted as a counting function of absolute value of thresholded frame differences.

3.4 DISTORTION MODEL

We also assume that the frame differential signal satisfies the *quadrant-monotonic* model proposed by Jain and Jain [49].

Definition: quadrant-monotonic

Suppose $O=(x_O,y_O)$ is the optimum search point, and $A=(x_A,y_A)$ is any other point in SR as shown in Figure 3.2. DF(.) is called *quadrant monotonic,* if $DF(X)<DF(A)$ for any $X(=(x_X,y_X) \in$ SR) that satisfies the following conditions: (a) X and A belong to the same quadrant with respect to O, that is, x_X-x_O (and y_X-y_O) has the same sign as x_A-x_O (and y_A-y_O), and (b) either

$|x_X - x_O| < |x_A - x_O|$ and $|y_X - y_O| \leq |y_A - y_O|$, or

$|x_X - x_O| \leq |x_A - x_O|$ and $|y_X - y_O| < |y_A - y_O|$

The following rules are derived based on the quadrant-monotonic model assuming that (1) $O=(x_O,y_O)$ is the optimum (minimum) point in SR, and (2) two distinct search points $A=(x_A,y_A)$ and $B=(x_B,y_B)$ have been placed in SR, and $DF(A) > DF(B)$.

Property 1 If $y_A=y_B$ and $x_A>x_B$, then O can*NOT* exist in the half plane defined by $\{(x,y)\in SR \mid x\geq x_B\}$.

Property 2 If $x_A>x_B$ and $y_A>y_B$, then O can*NOT* exist in the quadrant defined by $\{(x,y)\in SR \mid x>x_B \text{ and } y>y_B\}$.

Remark The other cases where the locations of A and B are different from the above but can easily be inferred from the above examples are omitted here.

The above rules are still valid for three search points or more. By examining every pair of search points, we can eliminate sequentially the regions that prohibit the

optimum point. This technique called *(interval) elimination* in optimization theory [9], is legitimate here because *DF*(.) is quadrant-monotonic. The remaining area, which contains the optimum point, is called *region of uncertainty* (RUC).

3.5 GOALS OF BLOCK MATCHING MOTION-ESTIMATION ALGORITHMS

One of the goals of block matching search algorithm is to minimize the total number of search points without sacrificing the performance. The meaning of the *best* search algorithm is somewhat vague. We can list a set of requirements desired from a *good* algorithm, but any realistic algorithm may not be able to achieve all the goals simultaneously because of the inherent conflicts among them. From this set of divergent requirements some have to be inadequately fulfilled so as to better satisfy the others. In addition, we do not consider the algorithms requiring derivatives here because *DF*(.) may not be differentiable and, besides, taking derivatives is often computationally expensive.

We now list many requirements that we would like a *good* search algorithm to satisfy, again keeping in mind that any realistic algorithm will perhaps fully satisfy only some of these fully and will only partially satisfy the remaining requirements.

(1) Convergence -- It converges to the optimum point in finite steps for a finite discrete SR.

(2) Fewer search points -- The total number of search points necessary for finding the optimum point should be as small as possible. This measure could be evaluated either in the worst case which is used most often in the existing literature, or in the average case which is often more difficult to analyze.

(3) Fewer search steps -- The total number of search steps necessary for finding the optimum point should be as small as possible.

(4) Noise immunity -- Its convergence is insensitive to the noise in the data.

In addition to the criteria listed above, other issues may also be equally important, for example, the hardware complexity. An algorithm having a smaller number of search points may need a complicated decision structure that requires a lot more hardware than a regular simple structure.

3.6 ALGORITHMS FOR INDEPENDENT SEARCH

Each block can assume an independent displacement estimate that can be very different from estimates of previous temporally or spatially adjacent blocks.

3.6.1 Search by the Orthogonal Technique

A new search algorithm, called *orthogonal search*, is proposed here. This algorithm involves searching for best match in orthogonal directions at each step and decreasing the step size as we get closer to the optimum point. This effective search technique requires the least number of search points among the known algorithms while it keeps the number of search steps to be reasonably small.

<Orthogonal Search>

Search region, SR = $(2p)\times(2p)$

Step Number i = 1

Initial Step Size $l = \left\lceil \frac{p}{2} \right\rceil$*

a. Step 1: Three search points are placed horizontally in the center of SR. The distance between every two neighboring points equals to the Step Size as shown in Figure 3.4. The minimum point is selected as the center for the next step. Step Number i← (i+1).

b. Vertical Step: Two more search points are placed vertically around the minimum from the previous step. The distance between every two neighboring points equals to the Step Size as shown by the example in Figure 3.4. The minimum point is the center at the next step.

c. Stopping Rule: The remaining RUC now has an area of $4(l-1)\times(l-1)$. If $l=1$, stop. Otherwise, $l \leftarrow \lceil l/2 \rceil$, i← (i+1), and continue.

d. Horizontal Step: Two more search points are placed horizontally around the minimum from the Vertical Step. The minimum point is the center for the next step. i← (i+1), and go back to **b.**

The convergence of the above algorithm is clear. At each step, three search points could eliminate half of the RUC according to the properties in Section 2. For example, if point 1.3 in Figure 3.4 is the minimum among points 1.1, 1.2 and 1.3, then only the area to the left of point 1.2 is the RUC in which O could exist. Hence, for $2k+1$ search points, the remaining RUC is roughly $(\frac{1}{2})^k$-th of the initial SR.

The total number of search points in this algorithm is $2(2\lceil \log_2(p) \rceil)+1$. For example, for $p=6$ (Figure 3.4) we need 6 steps and the Step Sizes are (3,3,2,2,1,1); that is, Step

Size=3 for the first two steps, Step Size=2 for the following two steps, and Step Size=1 for the last two steps. The same number of steps and search points are needed for $p=4$, $p=5$. The total search points are 13 in all the above cases.

We have developed one algorithm for block motion estimation in detail and have tried to emphasize that one main goal of any such algorithm is to reduce the complexity of search while being able to obtain a reliable estimate of the motion. Another equally important consideration that comes to mind is that after motion vectors have been obtained they have to be transmitted to the receiver. For low bit rate coding schemes this may turn out to be a rather large overhead. One way suggested by Koga et al [53] was to transmit the motion vectors as differentials with respect to some previous displacement vectors thus reducing the amount of overhead which would have been otherwise needed.

3.6.2 Techniques for measuring smaller displacements

At times, the motion may always be small, say never exceeding 3 pels from reference in any direction. This may happen if small amount (4:1 or less) of temporal subsampling is employed. In most low bit rate video-conferencing applications for above temporal subsampling rates displacements rarely exceed 3 pels from reference. This means that by exhaustive search 49 comparisons are still required per block to obtain the best motion vector. Search schemes can also be employed to reduce the number of comparisons needed. We can hope to still be able to obtain a reduction in the number of search points (say by a factor of 4) by adopting a search procedure. Such search procedures that seek small displacement vectors by following a set of assumptions mentioned earlier, may be somewhat more sensitive to noise. Therefore some savings in search points may have to be traded off to obtain reliable

displacement estimates.

We can now outline several search procedures. These can mostly be derived as a subset of large displacement measurement algorithms when the first larger step is skipped.

3.6.2.1 Algorithm 2.1

This can be derived from Koga et al's [50] three step algorithm by skipping the large step with step size 3 and performing the search with steps of size 2 and size 1 only. The number of search steps required are 2, where as number of search points required are 17.

3.6.2.2 Algorithm 2.2

This can be derived from Kappagantula and Rao's [56] algorithm by skipping the large steps in x and y with step size 3 and performing the search in x and y with steps of size 2 followed by search with step size 1. The number of search steps required are 4, where as number of search points required are 13.

3.6.2.3 Algorithm 2.3

This can be derived from our Orthogonal search algorithm [101] by skipping the large steps in x and y with step size 3 and performing the search in x and y with steps of size 2 followed by search with step size 1. The number of search steps required are 4, where as number of search points required are 9. To obtain a more robust estimate in the last step we can increase the number of search points by 2, giving total number of search points of 11.

3.7 ALGORITHMS FOR DEPENDENT ESTIMATION

Even though several displacement measuring algorithms are available which differ in a tradeoff of number of search points with number of steps performance wise they only slightly different. Independent displacement measurement algorithms (including the orthogonal search)[49],[50],[55],[56],[101] were designed to keep the search computationally simpler and they pretty much achieve their objective. Separate procedure are now required to minimize the amount of motion vector information that needs to be transmitted to the receiver. Dependent estimation algorithms try to approach the problem of simplifying the search and reducing the overhead in transmitting motion vectors to the receiver simultaneously. We can do that because it is highly improbable that displacement estimates in adjacent blocks are independent. In fact they are correlated and this correlation is higher if when block sizes are small. In fact this was the idea in support of large block sizes. Since the displacement estimates in adjacent block are related we can use this information to effectively reduce the search area, number of search points, and the number of search steps for obtaining the optimal displacement vectors.

In such schemes, an initial estimate of movement of a block is made from the adjacent blocks whose estimates have already been found. A matching criterion e.g. mean of absolute difference (MAD) is selected and an attempt to match the block of present frame into the displaced position blocks belonging to allowed search region (SR) in the previous frame is made. The allowed search region for dependent estimation is usually smaller then the full search as the displacement vectors are measured with respect to previous spatially or temporally adjacent blocks displacement vectors and a high degree of correlation exists between them. A best match can thus be obtained by

a small search centered at previous displacement estimate and this relative displacement vector is called the *update* term. We can formulate Let $\mathbf{z}$ be a vector representing the spatial location of the block. Now, let $\hat{D}_{k-1}(\mathbf{z})$ and $\hat{D}_k(\mathbf{z})$ represent the displacement vector obtained from a region around a identically located block in the previous and present frames.

3.7.1 Algorithm 3.1: Search With Temporal Initial Estimate

Correlation in movement between blocks identically located in the adjacent frames of an image sequence can be exploited to reduce the computations by using the previous frame displacement estimate as the initial search point. Displacement estimates for a current block can thus be measured with respect to previous frame estimates provided that objects move in smooth translational movement from frame to frame. For such movements there exists a high degree of correlation in displacement estimates in corresponding frames and so estimates of the blocks from previous frame which surround the location of the current frame block can be used to form an initial starting estimate for each block around which a smaller search can be performed [51], [54]. Proceeding formally, we seek an update term $U_k(\mathbf{z})$ such that,

$$\hat{D}_k(\mathbf{z}) = \hat{D}_{k-1}(\mathbf{z}) + U_k(\mathbf{z}) \tag{3.3}$$

Where $U_k \in SR$, and is obtained by a search procedure that minimizes the distortion function. For the reasons of robustness and noise reduction a linear weighted estimate of the region in the previous frame may be chosen to form the initial estimate such that,

$$\hat{D}_k(\mathbf{z}) = \sum_{m=-M/2}^{M/2} w_m \hat{D}_{k-1}(\mathbf{z}-m) + U_k(\mathbf{z}) \tag{3.4}$$

Where, for blocks belonging to moving area M in the previous frame the displacement

estimates have already been found and w_m are positive weights associated with displacement vectors such that $\sum_{m=-M/2}^{M/2} w_m = 1$.

A non-linear technique can also be used to select an initial displacement estimate. For example a counting function can detect which displacement vector appears most often in the region of interest in the previous frame and this can be selected as the initial estimate for the current block.

Thus center of current SR is first shifted by the displacement vector estimate obtained from the previous-frame blocks. A search technique like, orthogonal search can then be applied to this shifted SR. This enables us to reduce the computations necessary for finding the best match as well as reduce the motion vector overhead that is required to be transmitted.

3.7.2 Algorithm 3.2: Search With Spatial Initial Estimate

If objects move with smooth translational movement from frame to frame then there exists a high degree of correlation in displacement estimates of blocks spatially adjacent. This fact was inherently used in [53] to reduce the amount of overhead information that needs to be transmitted to receiver. We propose to exploit the spatial correlation in displacement estimates by starting a search to obtain the best match of a block in an initial region as obtained from estimates of the neighboring previously computed motion vectors. This initial estimate is then corrected minimally by employing a reduced search scheme. The advantage of this scheme as compared to ones before is its insensitivity to temporal subsampling, scene change change in motion direction etc. In addition here we have managed to keep the complexity of the search algorithm low as well as minimized the motion vector information that needs

to be transmitted.

Proceeding formally, if the initial estimate is chosen to be the last estimate along the current line then,

$$\hat{D}_k(\mathbf{z}) = \hat{D}_k(\mathbf{z}-1) + U_k(\mathbf{z}) \tag{3.5}$$

More generally we can derive an initial estimate from several blocks of the current frame, to improve the robustness and smoothen out the effect of noise.

$$\hat{D}_k(\mathbf{z}) = \sum_{m=1}^{M/2} w_m \hat{D}_k(\mathbf{z}-m) + U_k(\mathbf{z}) \tag{3.6}$$

Where, for blocks belonging to moving area M in current frame the displacement estimates have previously been found, and w_m are positive weights associated with displacement vectors such that $\sum_{m=1}^{M/2} w_m = 1$.

A non-linear technique can also be used, as before, to select an initial displacement estimate. A counting-function detects the displacement vector that appear most often among the blocks already processed in the present frame, and can be selected as the initial estimate for the current block.

3.8 SEARCH COMPLEXITY COMPARISON

Several block-matching algorithms for image-sequence coding have been proposed in the literature ([21],[49]-[56]). The search complexity of many of these algorithms are compared with that of orthogonal search in Table 3.1. It is clear from this table that for a typical search where the maximum displacement is limited to 5 or 6 pels between adjacent frames, the orthogonal search requires the least number of search points in the worst case. As far as the number of search steps is concerned, the orthogonal search has about the same complexity as many others. An additional useful feature of

the orthogonal algorithm is its regularity i.e. it needs the same number of search points and steps no matter where the optimum point is located. This regular structure should be important from hardware consideration Some authors show better performance of their algorithm by reducing the number of search points and search steps, by thresholding the distortion function at end of various steps, to quit the search. This approach may not be very reasonable in practice as it causes algorithms to terminate prematurely because of improper threshold values. The thresholds are dependent on various parameters such as picture contents and activity, amount of motion, uncovered area etc.

3.9 BLOCK-SIZE, OVERHEAD, SAMPLES, AND COMPLEXITY

The choice of block-size for motion-compensation may depend on many factors. It could depend on the nature of source image sequences likely to be encountered, be aimed towards minimizing either the overhead or complexity, or simply may depend on the application. Larger block-sizes allow good global estimates of motion, where as smaller block-sizes are necessary for accurate local estimations. As pointed out earlier, in image-compression application, the motion-vector overhead is necessary to be transmitted to the receiver. The motion estimation techniques such as the "dependent search" may help somewhat in reducing the motion-vector overhead. However, if very small-size blocks are used, the law of diminishing return comes into play causing little additional saving in coding performance with further reduction of block-size (increase of overhead). Thus the choice of block-size for motion estimation maybe somewhat crucial to the overall performance of the coding system.

We have discussed several search-algorithms aimed at reducing the complexity of computations for block motion-estimation in image sequences. As an additional

possibility for simplification of complexity, the number of samples per block used for computing the chosen distortion measure can also be reduced. In absence of additional knowledge regarding which pels in each block are more important, and should be included, a fixed sub-sampling structure is adopted and the pels located on the structure for each block are employed in computing distortion (matching) criteria. The number of computations required per block-match is thus reduced for all the block-matching displaced positions, reducing considerably the overall complexity of motion estimation procedure. There is, of course, some trade-off involved, as reducing the number of samples for matching may affect the quality of motion-estimation.

3.10 IMAGE SEQUENCES AND PREPROCESSING

The original "MsUSA" and "Salesman" sequences sampled at twice the color subcarrier frequency, are available in NTSC format digitized to 8 bits per pel. They are converted to component format consisting of luminance signal Y and chrominance signals U and V. A line of luminance contains 368 pels per line, and 240 lines per field. Since our target is low rate video-conferencing we further reduce the spatial resolution by subsampling in horizontal direction by a factor of 2. Each image now consisting $184 pels \times 240 lines$ is referred to as a "frame". Performance of motion compensation algorithms is investigated at temporal subsampling rates of 4:1 and 8:1, and thus we have 15 or 7.5 frames per second respectively. The motion activity in typical video-conferencing scenes is expected to be quite limited; therefore, even after temporal subsampling, movement in such scenes may rarely exceed 5 to 6 pels in any direction.

The "MsUSA" sequence consists of a closeup of a lady engaged in conversation. The head and shoulders view covers an area of about 60 percent of each frame. There are

only a few sharp edges in this sequence. The frame-to-frame changes in the scene are primarily due to local movements of eyes and lips, and global movements of head and shoulders. The amount of motion in different parts of the sequence changes somewhat gradually. In the active part of the sequence, the head and shoulders in the scene undergo relatively smooth, translatory motion. During movement, a part of the background is uncovered; however, the background is uniform, smooth and of low intensity.

The "Salesman" sequence consists of a view of a man sitting behind a desk in a office, holding a small box shaped item in his hand, and actively engaged in sales talk. The head, shoulders and arms view of the salesman covers about 40 percent of the area in the scene. There are also comparatively, a larger number of sharp edges. This sequence is much more active than the "MsUSA" sequence, and frame-to-frame changes in the scene are caused by active movements, change in reflections, shadows, and uncovering of the background. The movements in the scene are caused primarily by rather, fast motion of arms and hands, somewhat slower motion of shoulders and head, and non-translatory movement of the box. The background in the scene consists of part of a chair, plants, bookcases containing books and other objects. Relatively fast movements cause background to be uncovered quite often. The uncovered background is however highly non-uniform and complex.

3.11 SIMULATION RESULTS AND STATISTICAL ANALYSIS

The performance of orthogonal algorithm is compared to exhaustive search on the basis of entropy (Figure 3.6) to judge the effectiveness of this algorithm. We observe that both algorithms perform close to the exhaustive search algorithms for most cases. The estimate of the block at same location in previous frame can be used to

form an initial estimate for the current block. To justify that the previous frame displacement provides a reasonable initial estimate for current frame block, we show in Figure 3.7, a 3-D histogram for an exhaustive search of ± 6 with respect to corresponding previous frame displacement estimates. We can see that most of the estimates are centered around the previous frame values and an update of ± 3 is usually what we need to achieve the correct displacement vector. We can also show similarly that previous blocks estimates from current frame form a reasonable initial estimate for the current block.

We now study various properties of the motion-compensated frame differential signal for several reasons. First, it provides us with additional statistical criterion which helps us to evaluate the performance of various motion estimation algorithms. Second, and most important of all it gives us some insight into the choice of an effective coding strategy for motion compensated differential signals.

We measure some statistical properties of FD (frame difference) for the cases of uncompensated and compensated signals. These results are displayed in Figure 3.5 through Figure 3.25. We now discuss these results, with some observations and explanations, whenever necessary.

(1) Results of *motion activity* in "MsUSA" sequence, shown in Figure 3.5, indicate that the most active part of the sequence, lies between fields 120 and 240. Also, the motion activity, with temporal subsampling rate of 4:1 is roughly mid-way between, that with rates of 2:1 and 8:1. Some results on selection of block-size are shown in Figure 3.10. Selection of motion detection thresholds depends on the noise and motion activity; results of altering motion detection thresholds for the "Salesman" sequence are displayed in Figure 3.20.

(2) The *entropy* of the motion-compensated frame-difference (MCFD) signal (Figure 3.6) is smaller then the uncompensated signal, by as much as 35% (depending on the motion activity) for the "MsUSA" sequence. This shows that our motion estimation procedure works well, and therefore motion compensated difference signal can be efficiently encoded. Also, in comparing the performance of search schemes, on the basis of entropy computations, the *Orthogonal search* performs very close to the *Exhaustive search.* In Figure 3.21, the performance of exhaustive search for the "Salesman" sequence is shown; we observe that motion-compensation with block-size of 8×8 results in less than 20 percent improvement, on the basis of entropy of MCFD signal.

(3) The displacement estimates for the blocks identically located in adjacent frames are highly correlated (Figure 3.7). This allows us to linearize the displacement estimates around, previous temporally adjacent estimates. A small update can be formed by help of a search scheme to get the optimal displacement vector.

(4) The displacement vectors for the blocks spatially adjacent are also highly correlated. Thus the displacement estimates of spatially adjacent previous blocks also forms, a good initial prediction of the displacement for the current block. A small update on the initial displacement, is performed to get the optimal displacement vector. The results compared with that of exhaustive search in Figure 3.8, show little differences.

(5) The dependent motion estimation algorithms [13] (Figures 3.8 and 3.9) which update on a single previous estimate, work as long as, the temporal-motion, does not change directions suddenly, or is not very different in going from one end of the moving area in image frame to the other, spatially. The single estimates can also be

sometimes noisy (because of the limitations in the distortion criterion or the near constant nature of luminance). As a solution to this problem a weighted measure of surrounding block displacements may provide a more robust estimate.

(6) With *entropy* as the basis for comparison (Figure 3.11), the Variable Block-size (VBS) scheme, for the "MsUSA" sequence nearly attains the performance of scheme that uses fixed blocks of size 4×4, although the difference in entropy performance for blocks of 8×8 and 4×4 is small. The results for the "Salesman" sequence (Figure 3.21), show that with VBS the entropy performance is much improved, as compared to that when blocks of 8×8 size are used.

(7) In simplifying motion-estimation computations further, various tradeoffs involved, such as, reduction in performance with reduction in samples used in matching and the choice of subsampling structure are investigated; the results for "MsUSA" sequence are shown in Figure 3.12. For reliable motion-estimates, at least 16 samples out of 64, were found necessary to included in computations. Among various subsampling structures compared, when using 16 samples, the structure with 2:1 subsampling in both horizontal and vertical directions and with staggered samples, was found to perform much better than others.

(8) The *variance* of frame difference (FD) in the moving area of the signal after motion-compensation is substantially lower (Figure 3.13), as compared to that without motion-compensation, indicating that motion compensation works well in many areas of "MsUSA" sequence. We also observe that variance, after motion-compensation, is nearly constant, over the entire sequence. The results for the "Salesman" sequence are displayed in Figure 3.22; the variance after motion-compensation is roughly one third of that without motion-compensation. This shows that, even though variance is

much lower then that without compensation, for "Salesman" sequence far fewer blocks get well-compensated as compared to the "MsUSA" sequence, when block-size of 8×8 is used.

(9) The one-step spatial correlation coefficients of FD (Figure 3.14) in the moving area of "MsUSA" sequence is substantially lower, after motion compensation than that without compensation. The correlations typically have a value in the range of 0.25-0.6, which is smaller than that of correlation values computed in either the uncompensated difference signal, or the original image. The one-step correlation coefficients computed over the entire moving area of the frame, may also differ by as much as 0.25, in horizontal and vertical directions. The correlation-coefficient calculation results for "Salesman" sequence are displayed in Figure 3.23.

(10) The *spatial correlations* after motion compensation (Table 3.2 and Figure 3.15), indicate that up to 3 steps along horizontal or vertical directions correlations fall roughly by a factor of 2, in one direction . The rate of fall usually is somewhat steep up to 4 steps but is not as steep beyond. Also, the one step diagonal correlation can be well approximated, by the product of one step horizontal and one-step diagonal correlation coefficients.

(11) The *temporal correlations* (Figure 3.16), between frames of the "MsUSA" sequence are substantially decreased in the moving area, during periods of high motion activity in "MsUSA" sequence. With motion-compensation, the loss in temporal-correlations is corrected, resulting in near constant constant temporal correlations for the entire sequence. For this sequence, the values without motion compensation range from 0.86-0.99, where as the values after compensation are found to lie above typically above 0.98. This also indicates that of our motion compensation algorithms are

effective.

(12) Spatial correlations are also computed on a block basis, and are used to categorize blocks into 3 categories; blocks with correlation-coefficients, less than -0.2, between 0.2 and 0.4, and greater than 0.4. The results of classification for the "MsUSA" sequence are shown in Figures 3.17(a) and 3.17(b), and illustrate that most of the blocks have low correlations. The block-correlation histogram of Fig 3.18 shows that the correlation categories of 0.0 to 0.20, 0.20 to 0.40 and 0.40 to 0.60 contain over 75 percent of all moving blocks of the entire sequence. The results of block classification for the "Salesman" sequence are shown in Figures 3.23(a) and 3.23(b); and the results for block-correlation histogram are shown in Figure 3.24.

(13) The overall characteristics of the motion-compensated frame-differential signal for the two sequences are substantially different and can be summarized here. The "MsUSA" sequence has mostly translational motion that changes gradually from frame-to-frame, whereas the "Salesman" sequence contains relatively faster and non-translatory motion. The MCFD signal therefore also has different characteristics. Figure 3.19 shows the original frames of "MsUSA" sequence as well as FD and MCFD signal, when temporal subsampling rate of 8:1 is employed. Figure 3.25 shows original frames of "Salesman" sequence, FD and MCFD signal, when temporal subsampling rate of 4:1 is employed. On the basis of statistical results and in comparing the visual appearance of MCFD signal, we conclude that "MSUSA" sequence even at 8:1 temporal subsampling rate can be better compensated by block matching than the "Salesman" sequence at temporal subsampling rates of 4:1.

3.12 DISCUSSION

Fast search schemes, designed on the basis of a few regularity assumptions about image sequence data, may fail occasionally to produce the optimum result because of the model mismatch. If the assumed model fits the real data reasonably then the search performs well. For such a case the entropy of the motion compensated frame difference signal may be smaller then the uncompensated signal by as much as 40% (depending on the motion) indicating that motion compensated difference signal can be efficiently encoded. The nature of motion may however be such that the distortion is very different from the assumed model and therefore search algorithms may not perform very well. The model mismatch may result in suboptimum displacement vectors. Compensation by these vectors would result in an increase in the entropy of motion-compensated FD signal. Model mismatch may also effect the visual perception and possibly the coding efficiency. Improvements possible to minimize the penalty paid during mismatch include checking of displacement vectors obtained for the current block with other previously obtained displacement vectors and correcting them if necessary before transmission to the receiver, choosing a better distortion measure etc. Another possible approach for estimating reliable displacement vectors by use of block matching search algorithms may be to employ somewhat larger block sizes to first obtain a globally consistent displacement field and then perform extra searches locally to improve the accuracy of the estimates. This however requires extra transmission overhead to be able to specify local motion variations.

Various statistical measures employed provide valuable information regarding the nature of MCFD signal as well as basis for comparison of the quality of motion estimation with various block motion-estimation algorithms. An investigation into

the statistics of MCFD signals for the two sequences, shows some similarities and important differences. The two sequences represent different types of motion i.e., in "MsUSA" sequence motion is gradual and mostly translational in nature, where as in "Salesman" sequence motion is faster and only partly translational in nature. While movement in "MsUSA" sequence can be compensated by block-matching techniques, to better compensate the movement in the "Salesman" sequence much more sophisticated algorithms are required.

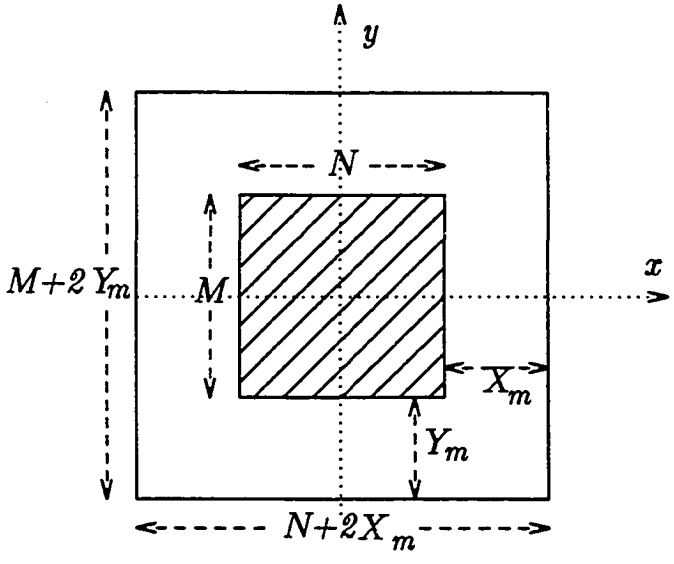
Table 3.1 Worst Case Performance of Fast Search Algorithms

Algorithm	Max. Disp. = 3		Max. Disp. = 6	
	Search Points	Search Steps	Search Points	Search Steps
Exhaustive Search	49	1	169	1
2-D-logarithmic	18	4	21	7
3 Step	17	2	25	3
One-at-a-time	9	6	15	12
Mod. 2-D-logarithmic	13	4	19	6
Orthogonal Search	9	4	13	6

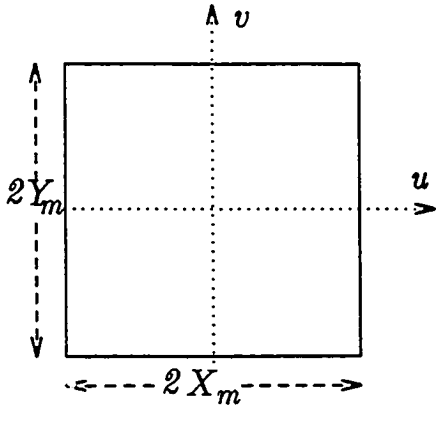
Table 3.2 Measured Correlation Coefficents of Compensated Frame Differences

Measured Correlation coeffs in moving blks of seq.: msusa ,8:1 tss, Fld=160								
Separation	0	1	2	3	4	5	6	7
0	1.0000	.4308	.2322	.1181	.0815	.0679	.0393	.0336
1	.3490	.1399	.0847	.0684	.0661	.0467	.0087	-.0099
2	.1906	.0657	.0127	-.0098	-.0013	.0026	-.0160	-.0214
3	.1281	.0662	.0185	.0103	.0090	-.0013	-.0076	-.0001
4	.1005	.0547	.0174	.0134	.0261	.0344	.0264	.0365
5	.1010	.0745	.0366	.0127	.0163	.0132	.0186	.0317
6	.1046	.0748	.0440	.0344	.0266	.0171	-.0062	.0053
7	.0826	.0563	.0277	.0172	.0137	.0169	.0033	.0048

Measured Correlation coeffs in moving blks of seq.: msusa, 8:1 tss, Fld=80								
Separation	0	1	2	3	4	5	6	7
0	1.0000	.4971	.2348	.1543	.1294	.1018	.0642	.0270
1	.4443	.2390	.0970	.0546	.0491	.0368	-.0069	-.0291
2	.1729	.1027	.0445	.0179	.0115	.0114	-.0125	-.0204
3	.0856	.0741	.0318	.0181	.0140	.0077	.0007	-.0003
4	.0551	.0396	.0202	.0183	.0165	.0125	.0072	.0039
5	.0179	.0039	-.0105	-.0200	-.0213	-.0115	.0010	.0007
6	.0077	-.0147	-.0253	-.0357	-.0383	-.0208	-.0017	-.0085
7	.0156	-.0117	-.0087	-.0273	-.0251	-.0097	-.0046	-.0018



(a)



(b)

Figure 3.1 Match Area and Search region in Block Motion Estimation

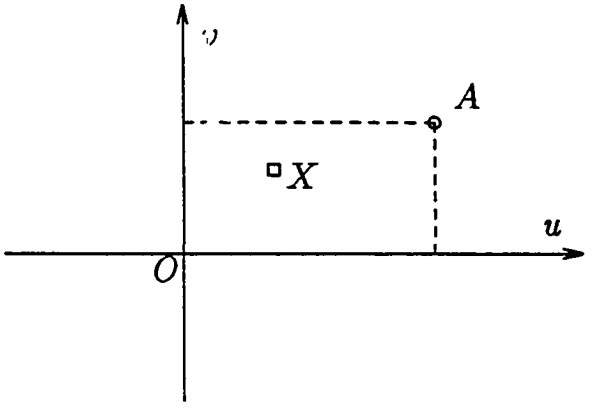


Figure 3.2 Quadrant Monotonic Model

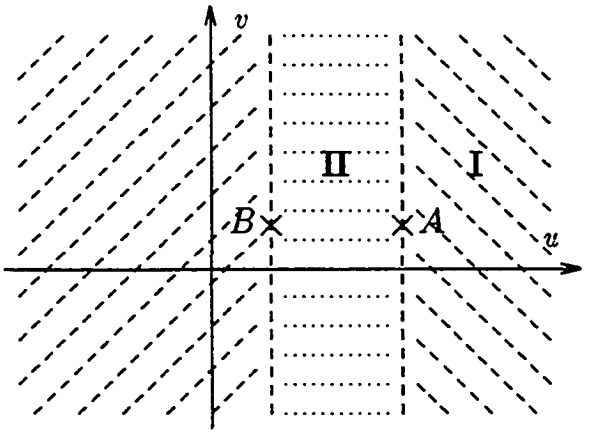


Figure 3.3(a) Property 1 and Remark 1

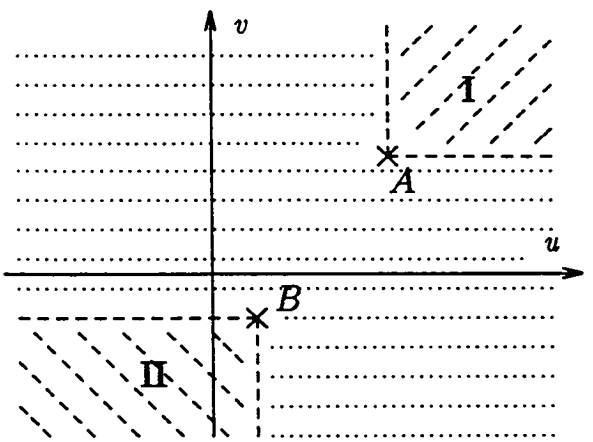


Figure 3.3(b) Property 2 and Remark 2

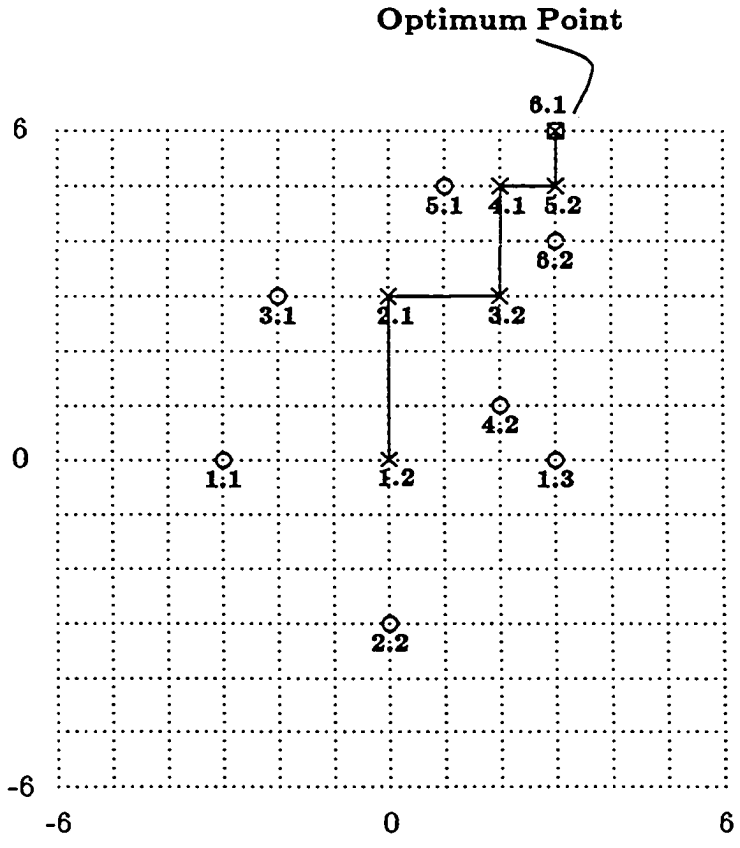


Figure 3.4 Orthogonal Search (p=6)

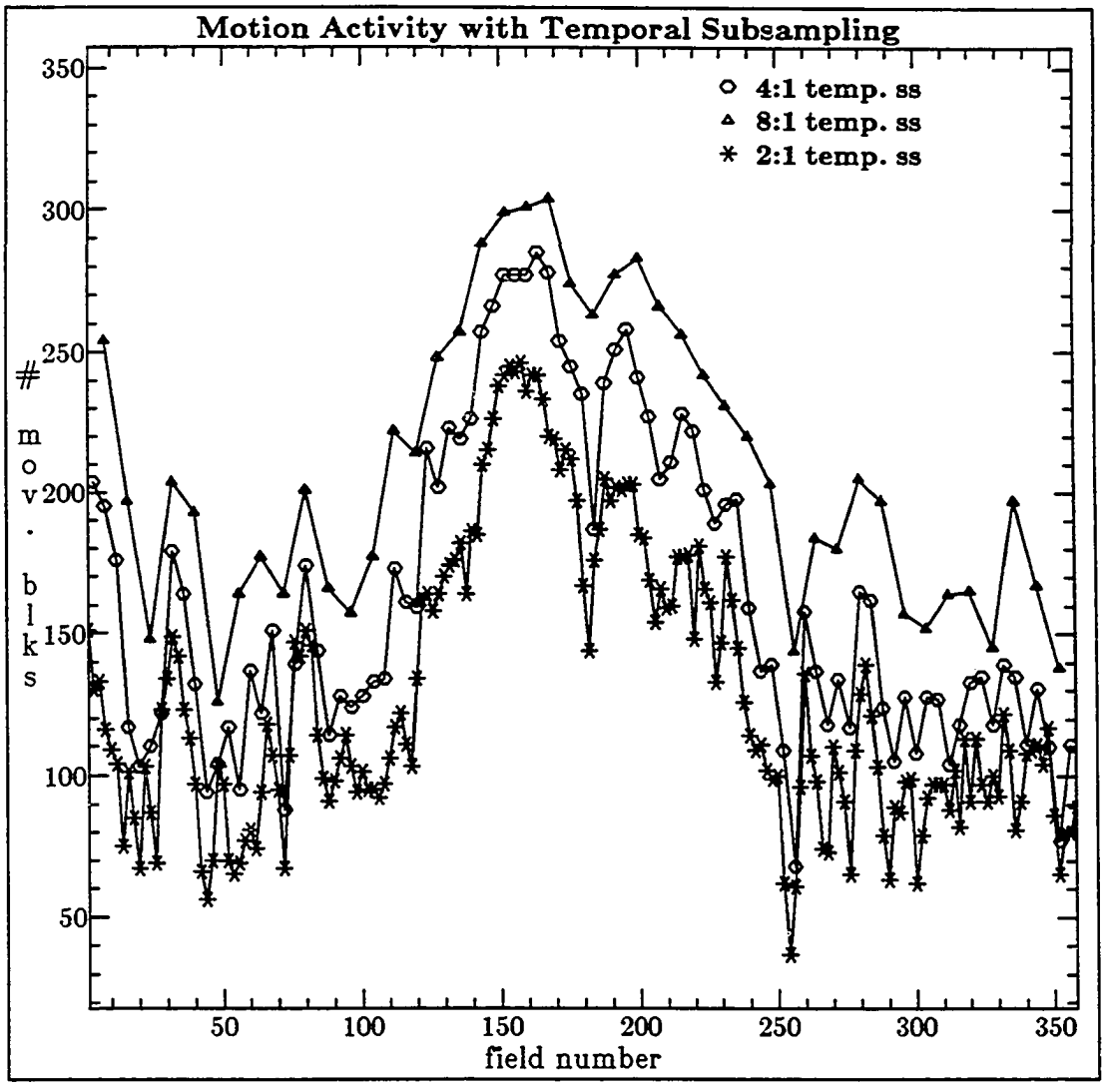


Figure 3.5 Motion Activity in MsUSA Sequence

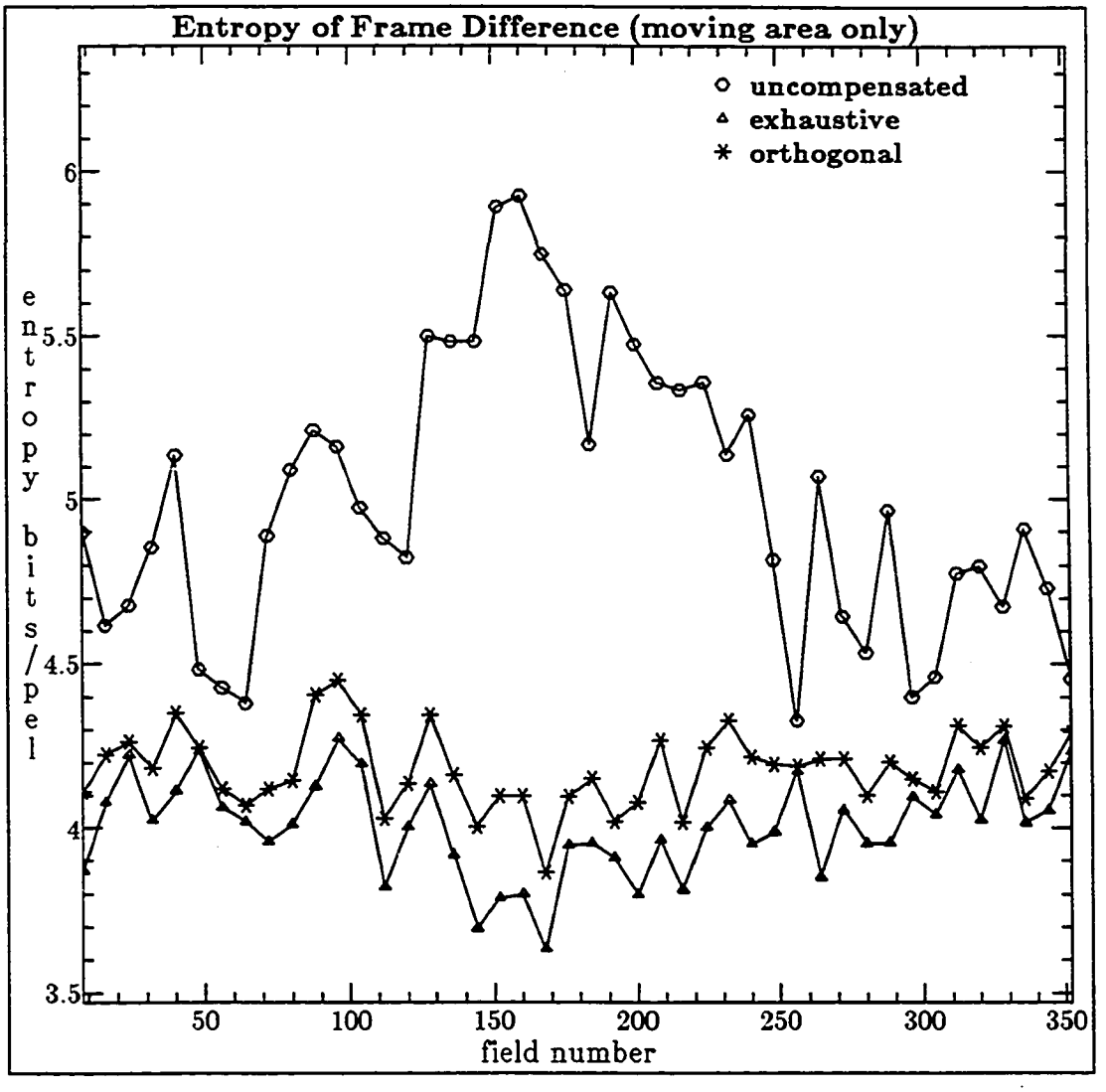


Figure 3.6 Performance of Orthogonal Search: MsUSA Sequence

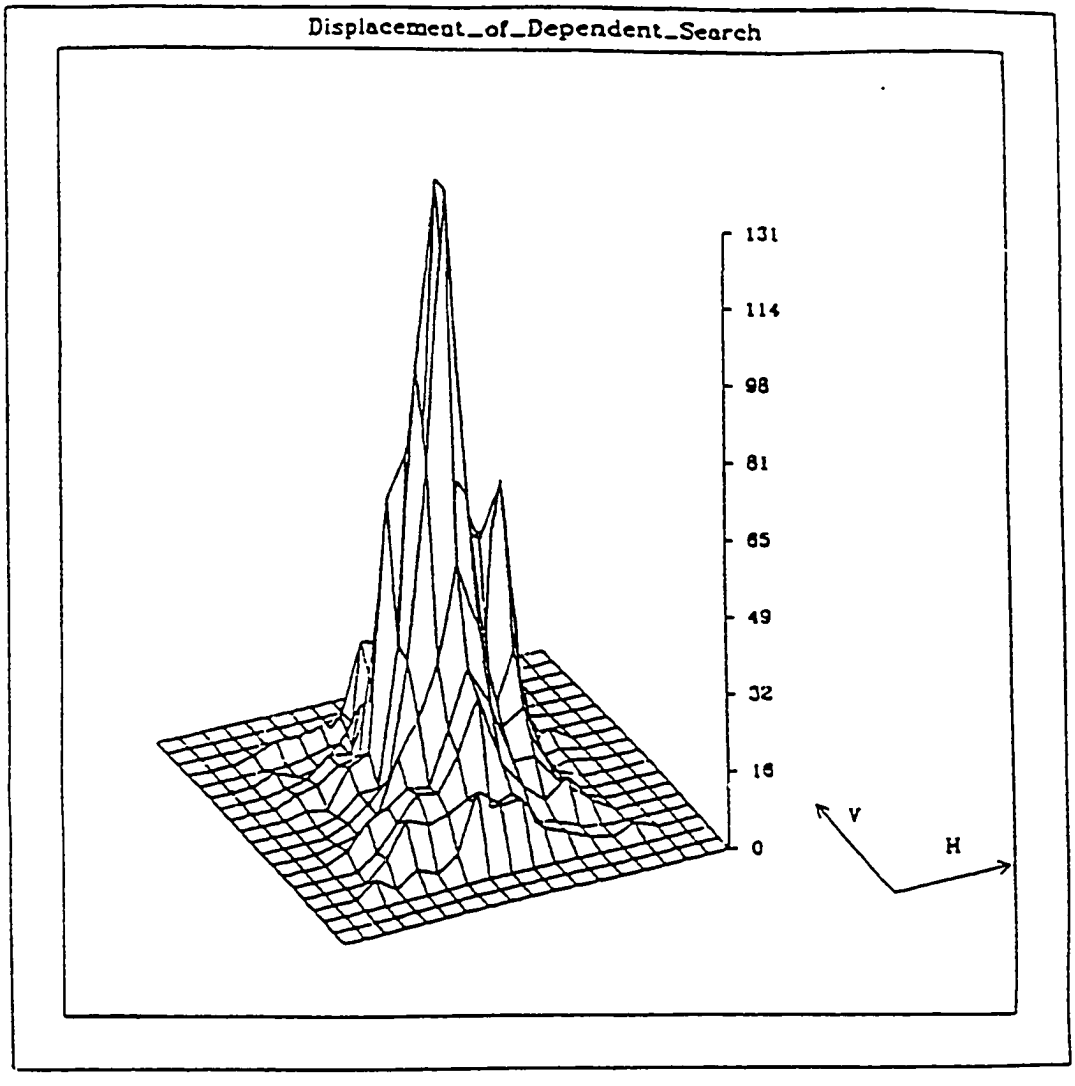


Figure 3.7 Histogram of Dependent-Search Displacement Vectors at 8:1 temporal sub-sampling in active portion of "MsUSA" sequence.

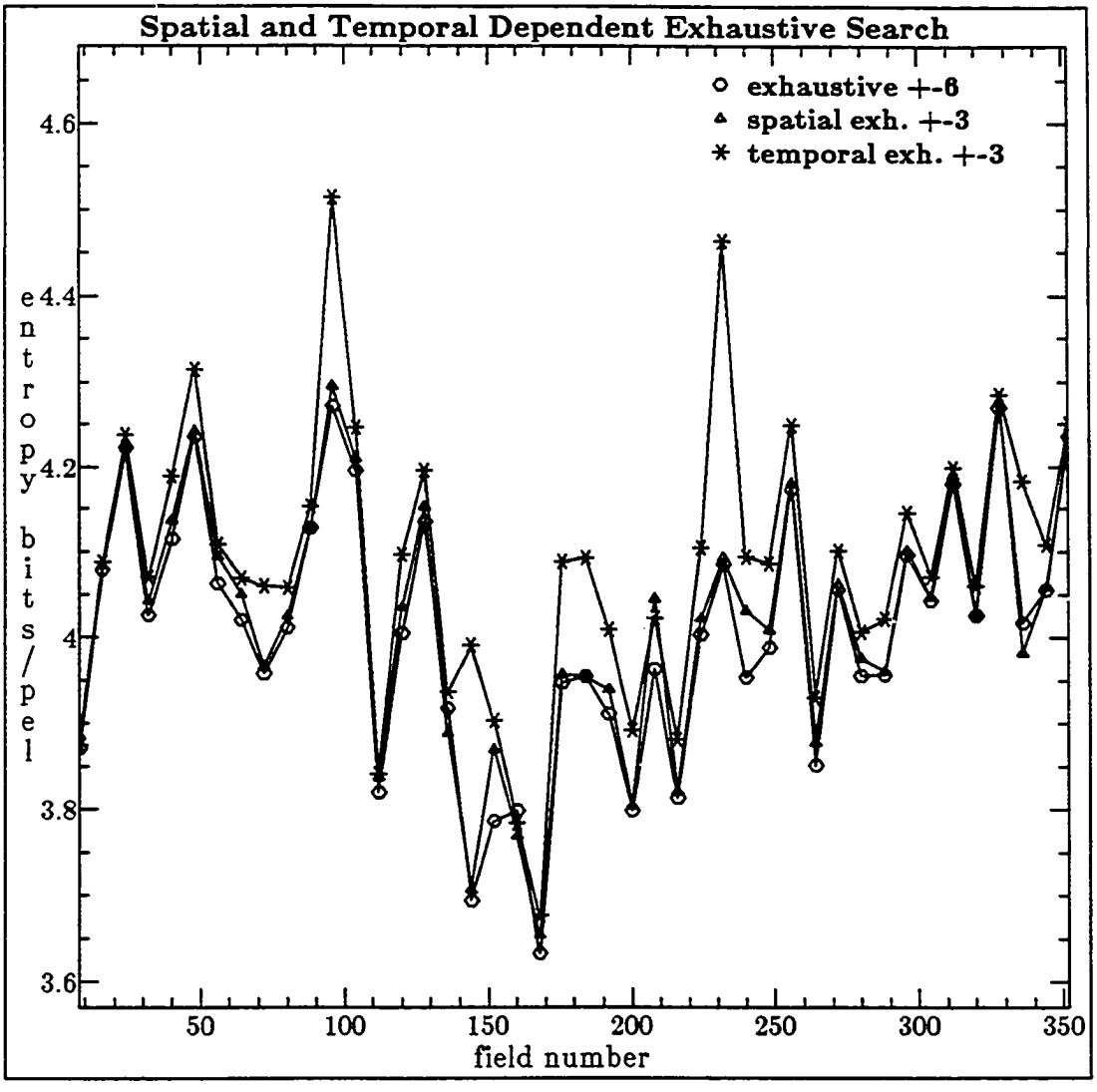


Figure 3.8 Comparison of spatial and temporal Dependent Search : "MsUSA" sequence

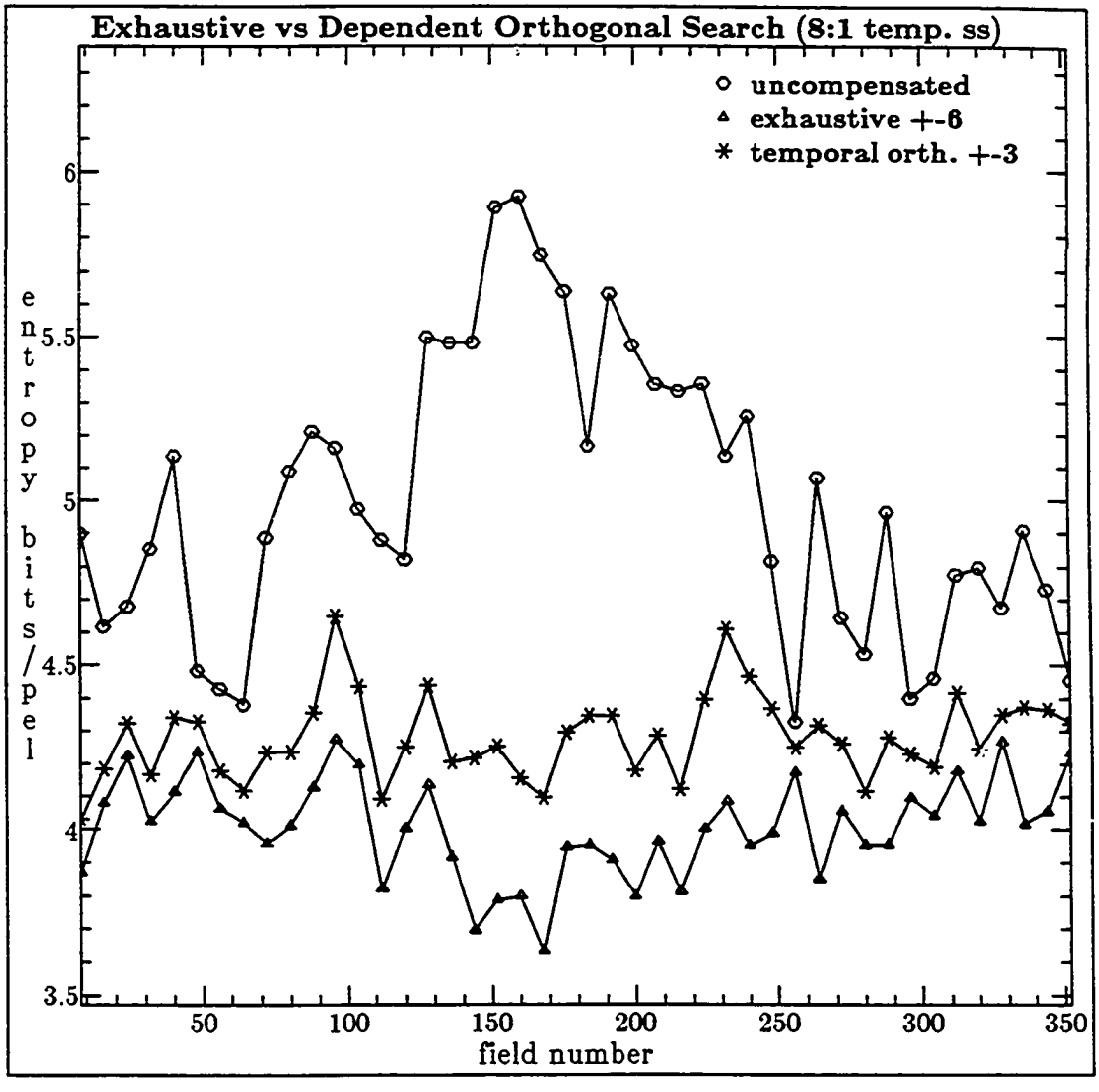


Figure 3.9 Exhaustive and Orthogonal Temporal Dependent Search : "MsUSA" sequence

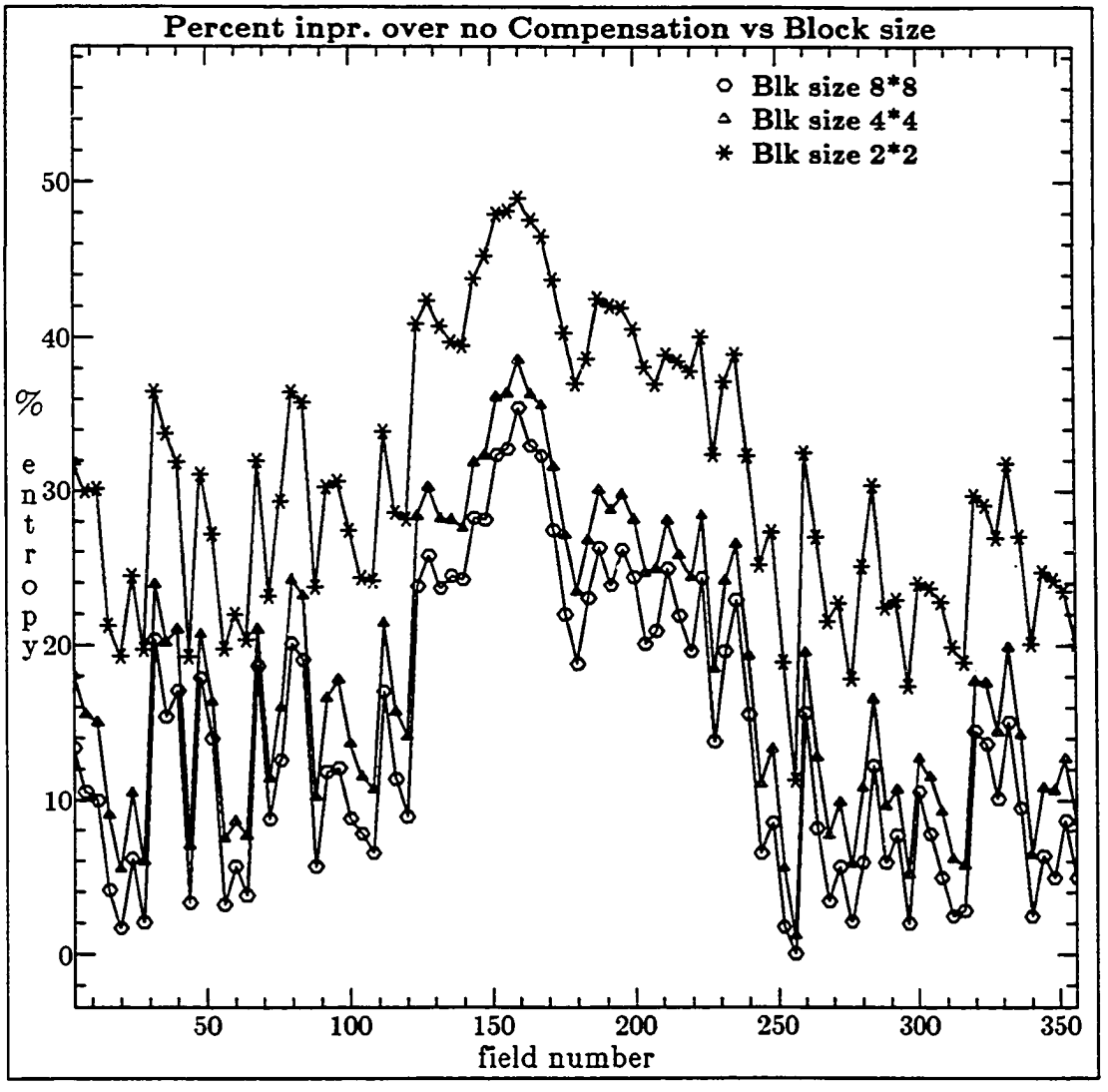


Figure 3.10 Percentage improvement vs Block size at 4:1 temporal subsampling over no compensation for "MsUSA" sequence (Motion vector overhead excluded)

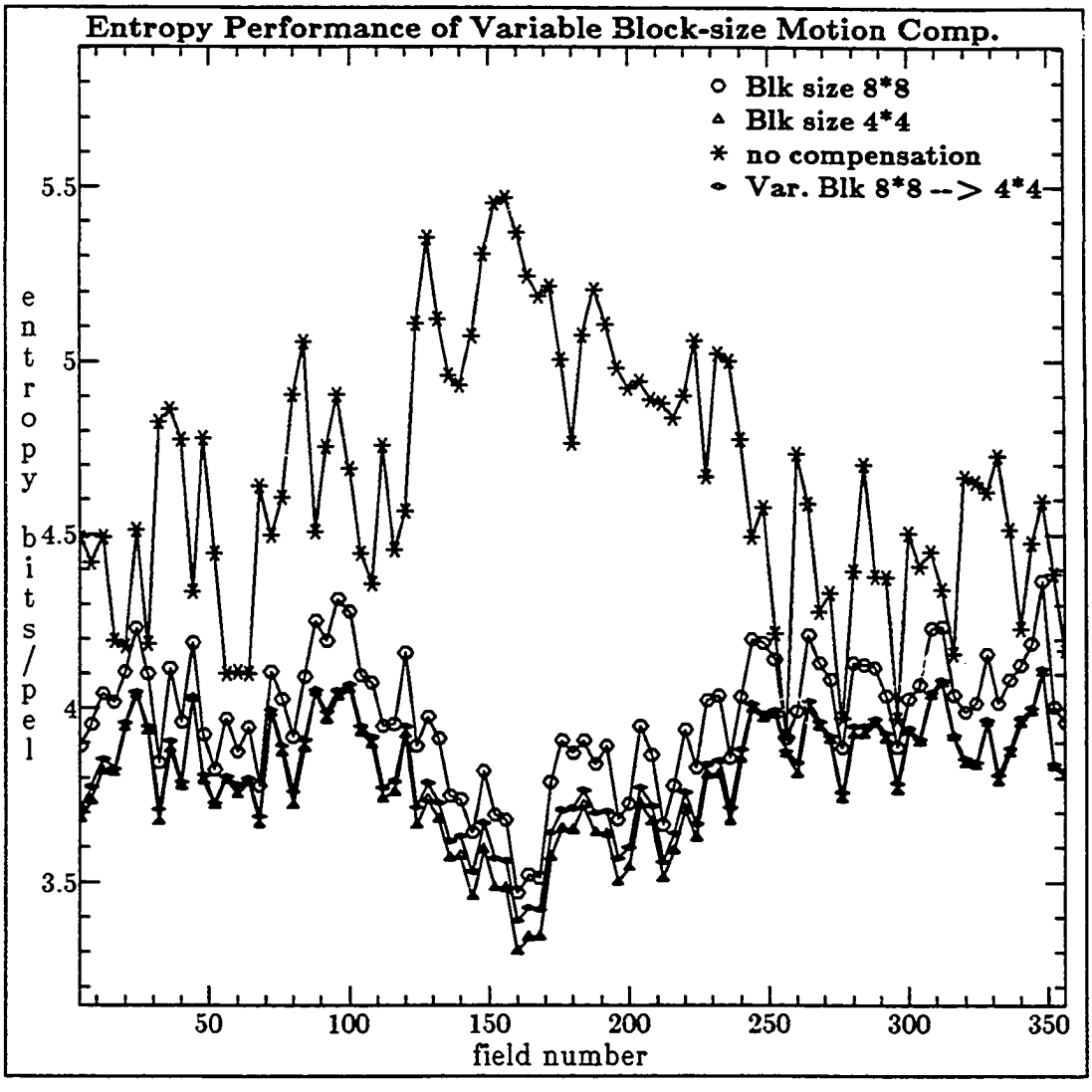


Figure 3.11 Entropy Comparison of 8*8 and 4*4 fixed Block Size scheme with Variable Block-size at 4:1 temporal subsamp. : "MsUSA" sequence

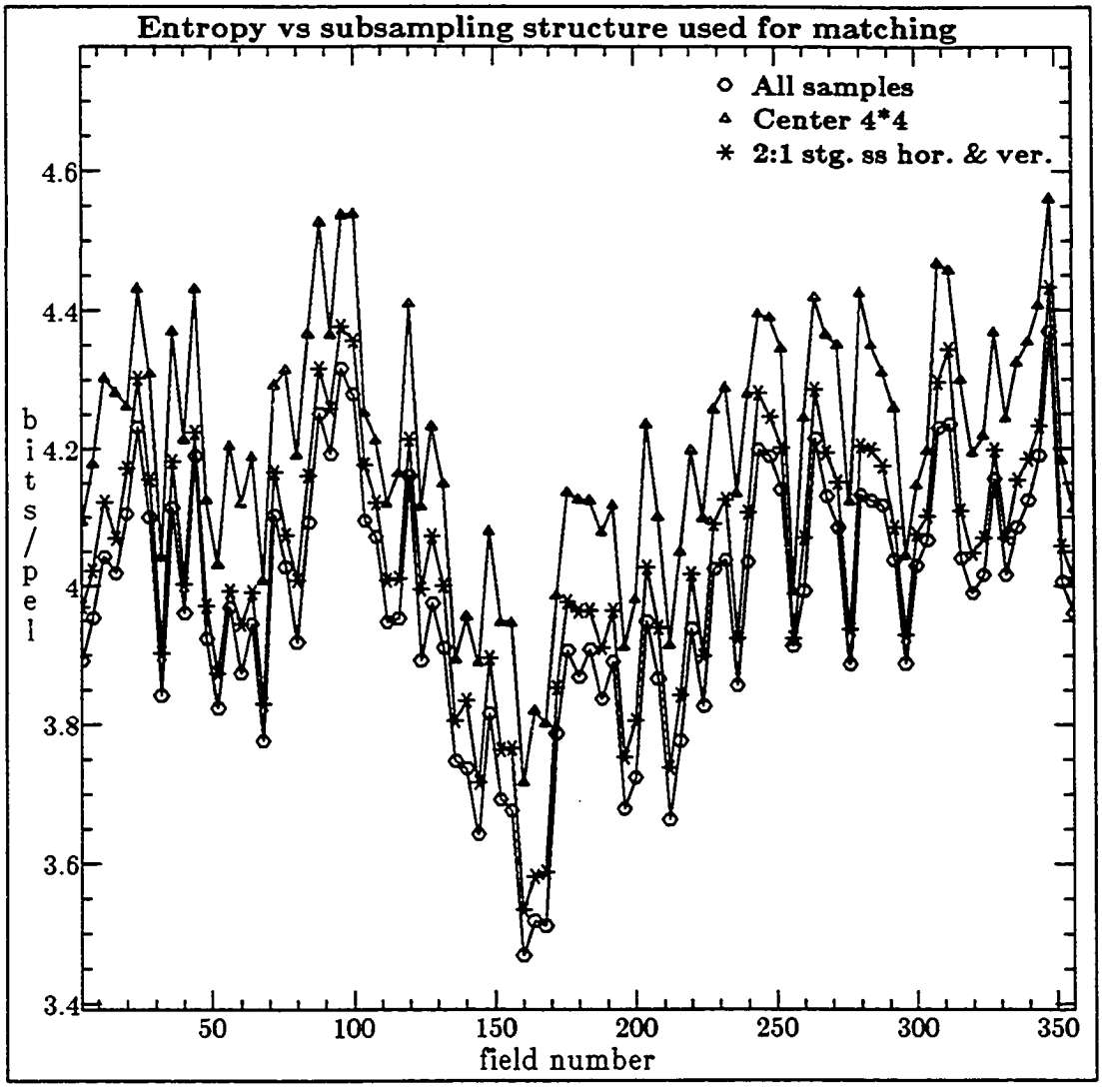


Figure 3.12 Entropy comparison for Block Matching when reduced number of samples are used in matching, 4:1 temporal subsamp. : "MsUSA" Sequence

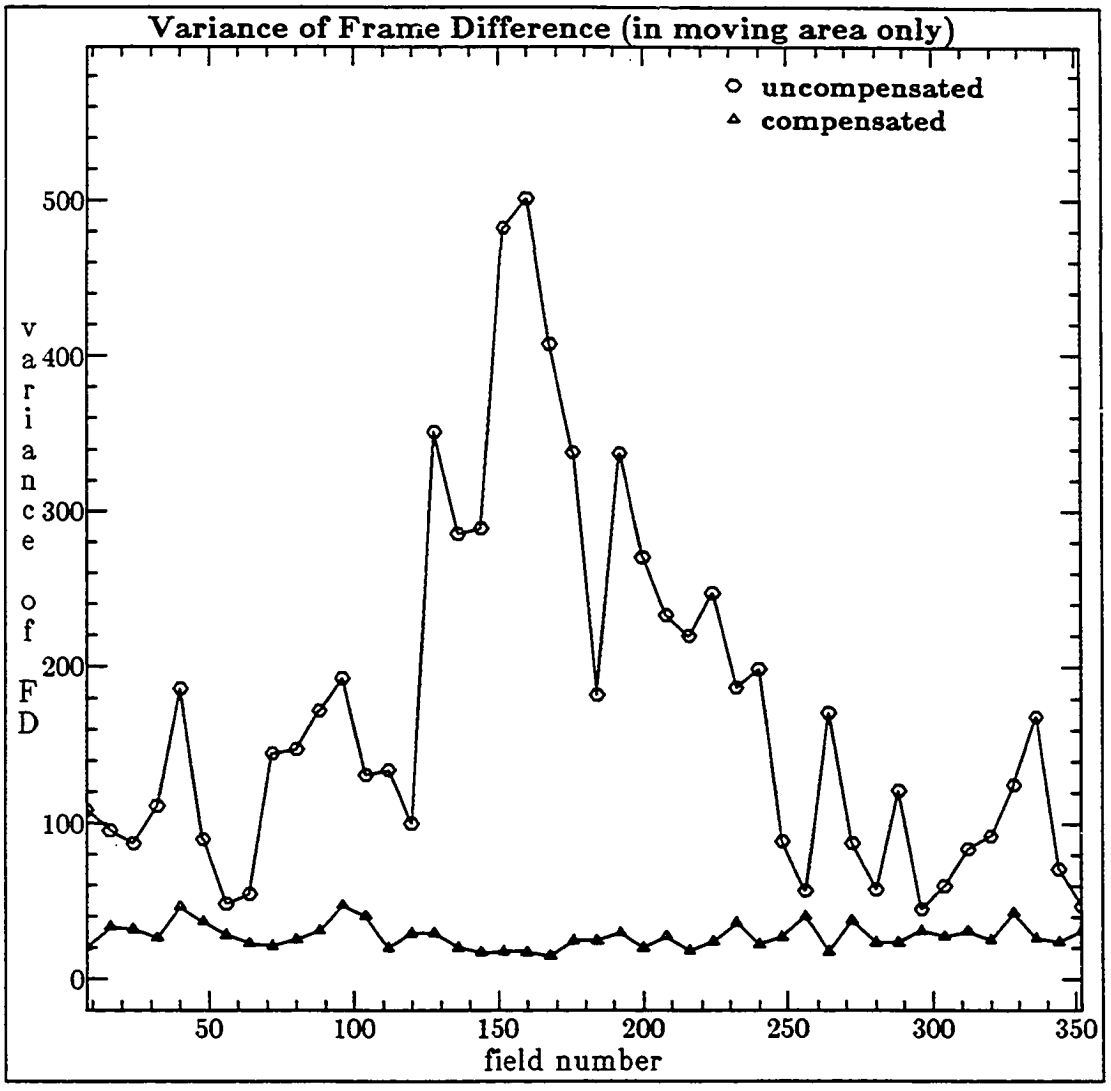


Figure 3.13 Variance of FD in moving area, 8:1 temp. ss : "MsUSA" sequence

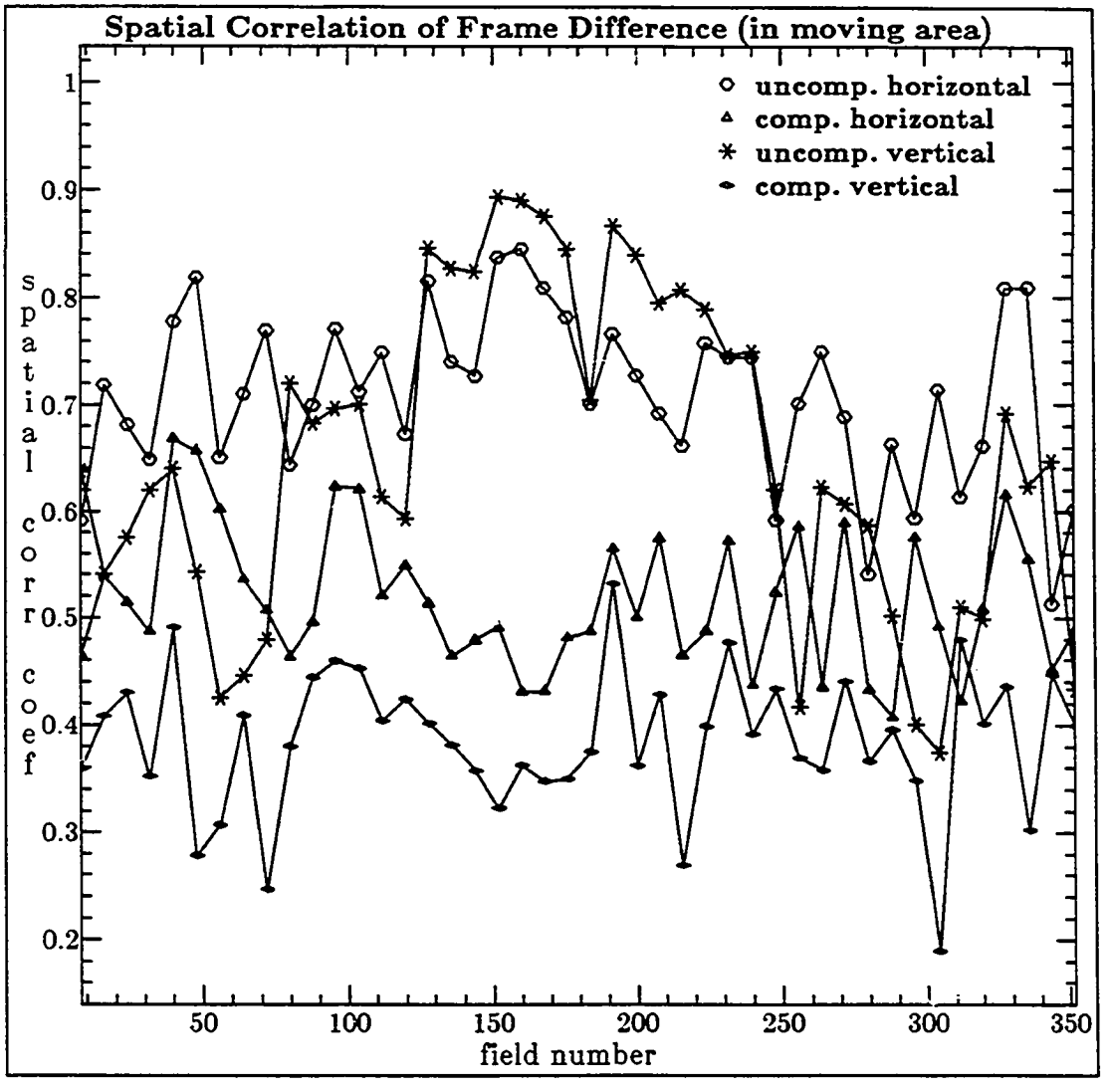


Figure 3.14 Spatial Correlation of FD in the moving Area : "MsUSA" sequence

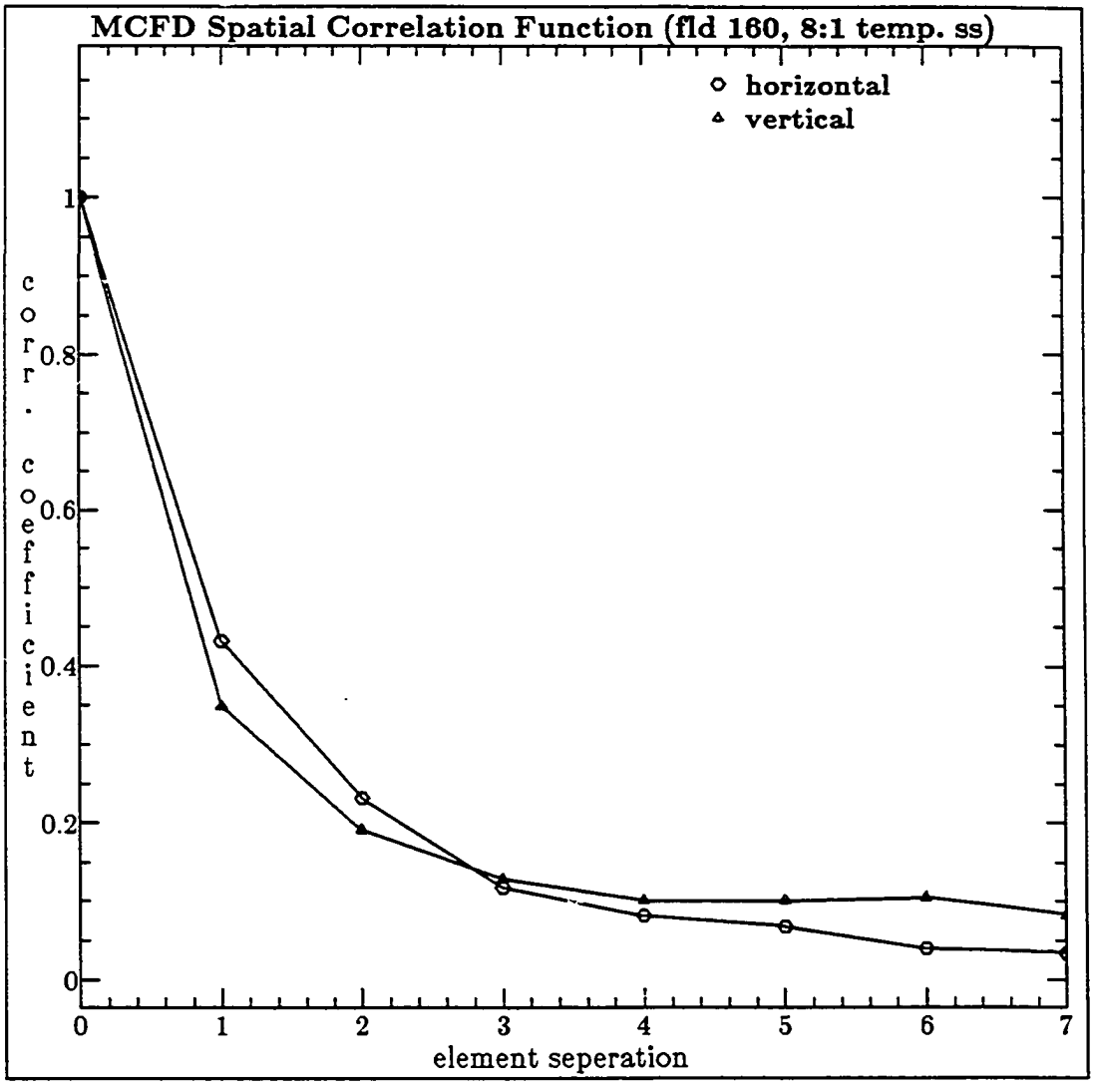


Figure 3.15 MCFD Spatial Correlation Functions : "MsUSA Sequence" (fld. 160)

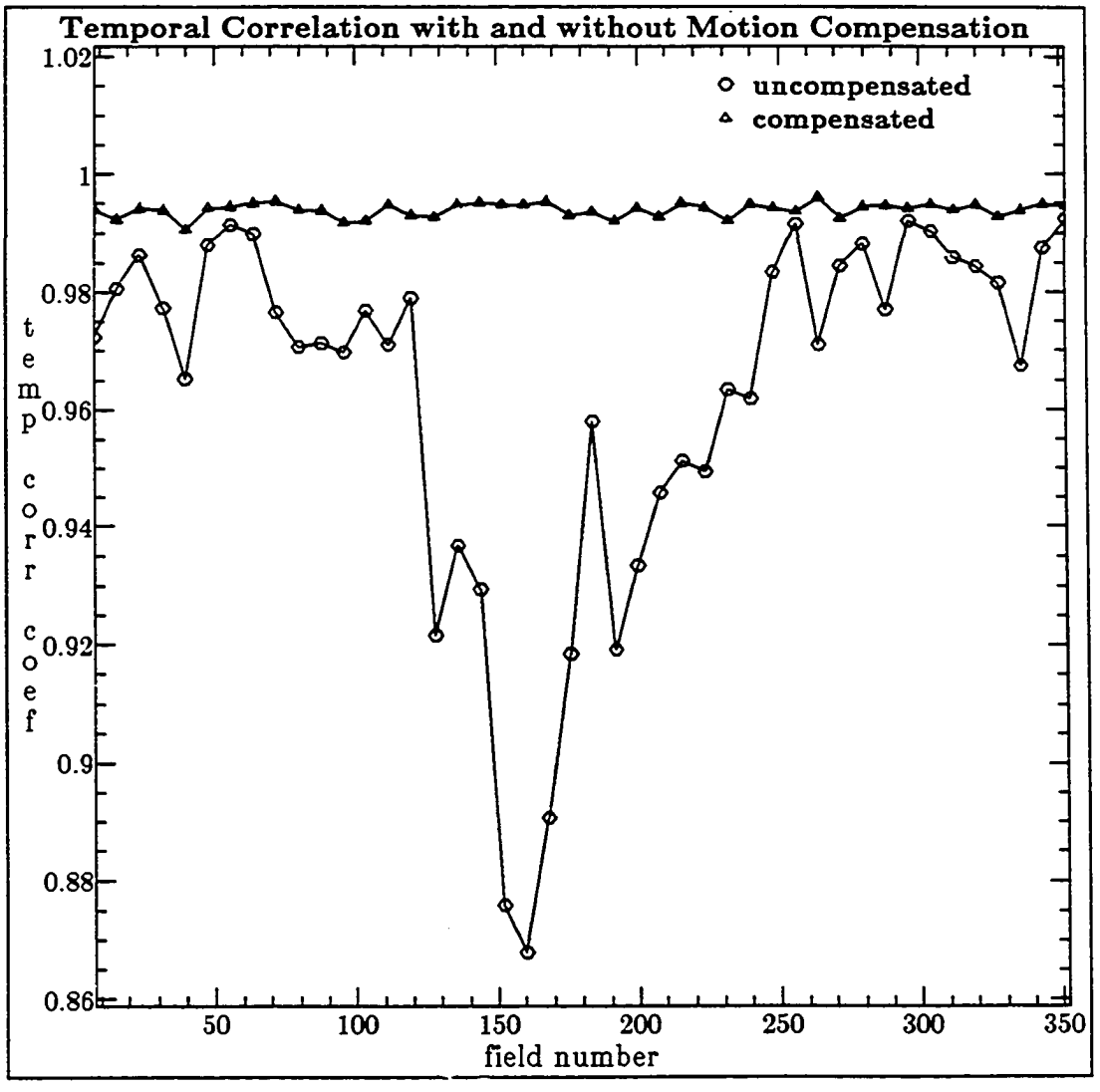


Figure 3.16 Temporal Correlation between Adjacent Frames : "MsUSA" sequence

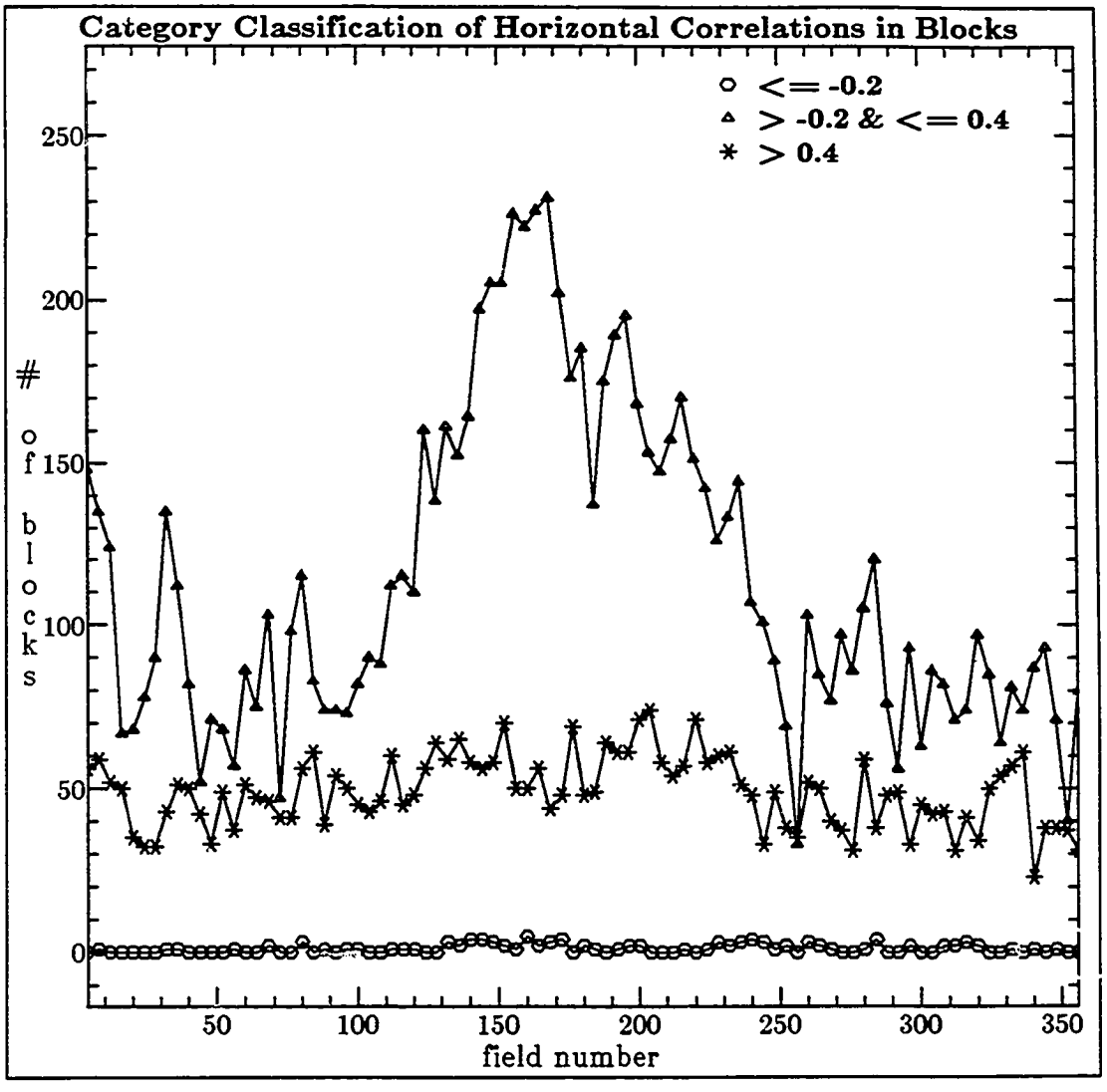


Figure 3.17(a) Category Classification of Correlation-coefficient in horizontal direction for MCFD Blocks : "MsUSA" sequence

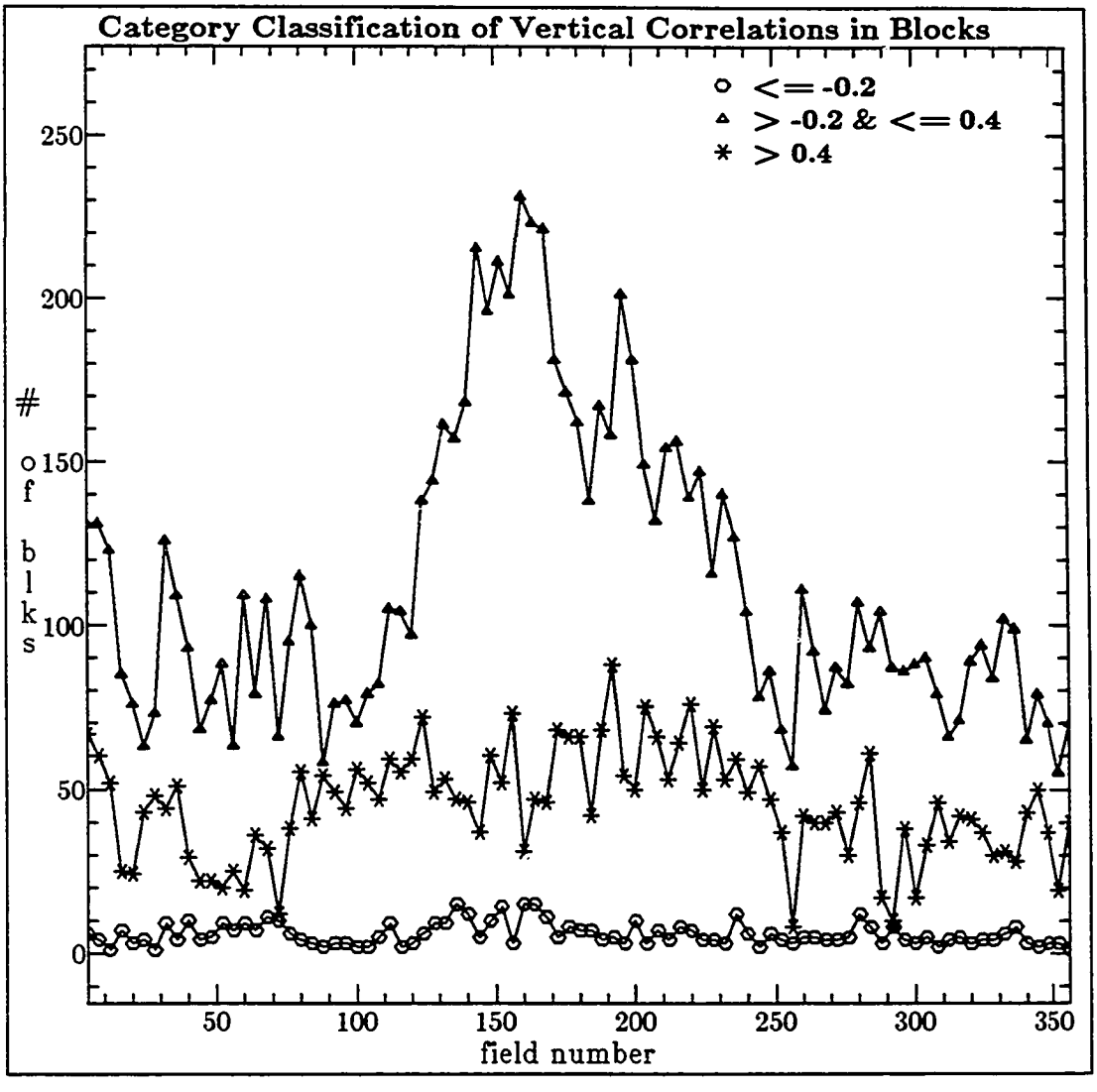


Figure 3.17(b) Category Classification of Correlation-coefficient in vertical direction for MCFD Blocks : "MsUSA" Sequence

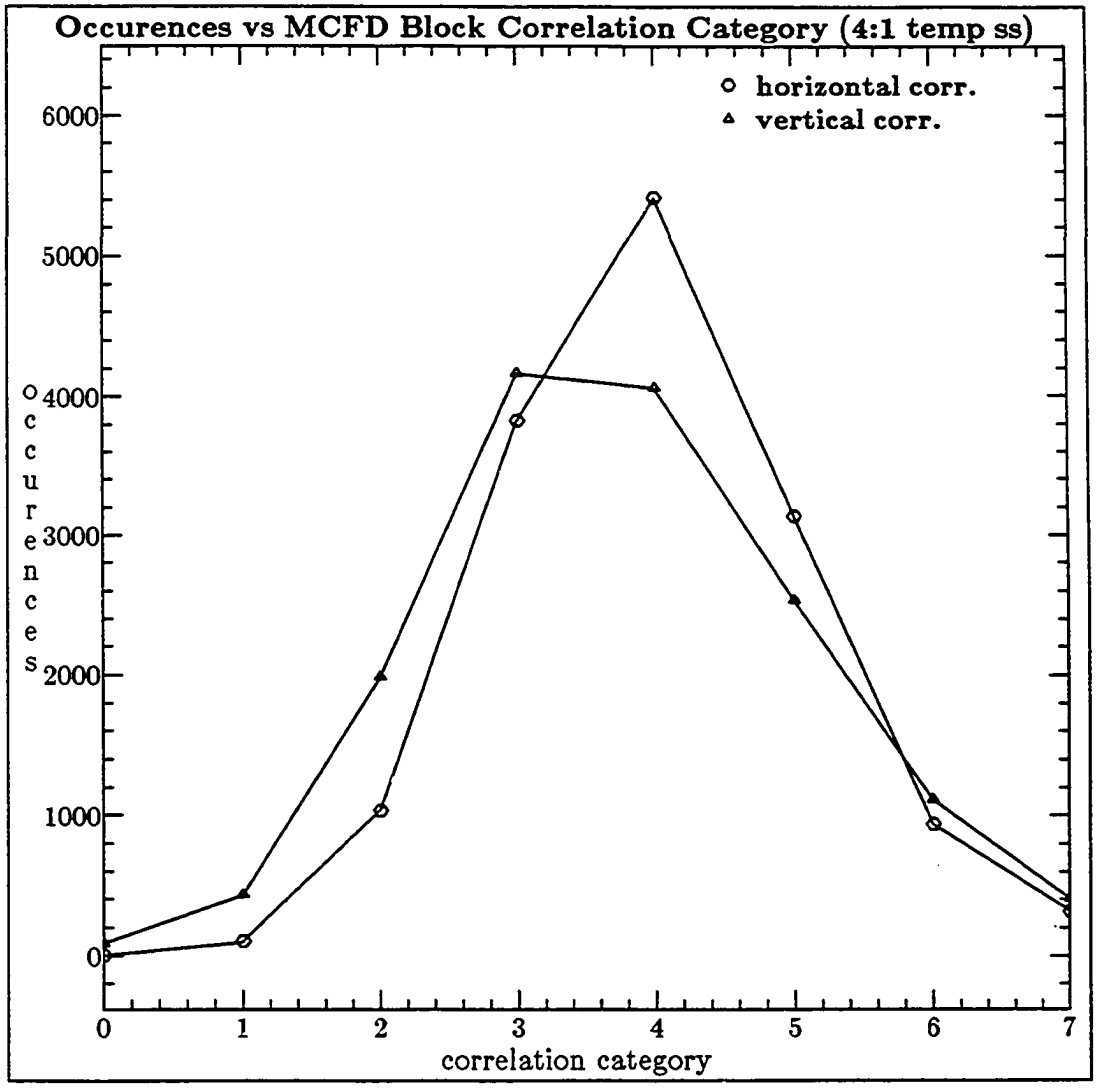


Figure 3.18 Number of blocks in MCFD Block Correl. Category : "MsUSA" Sequence

correlation category	correlation range	correlation category	correlation range
0	-1.00 to -0.41	4	0.21 to 0.40
1	-0.40 to -0.21	5	0.41 to 0.60
2	-0.20 to 0.00	6	0.61 to 0.80
3	0.01 to 0.20	7	0.81 to 1.00

Figure 3.19(a) Original Images from "MsUSA" sequence (at 8:1 temporal ss)

Figure 3.19(b) Frame Difference and Motion-compensated Frame Difference images

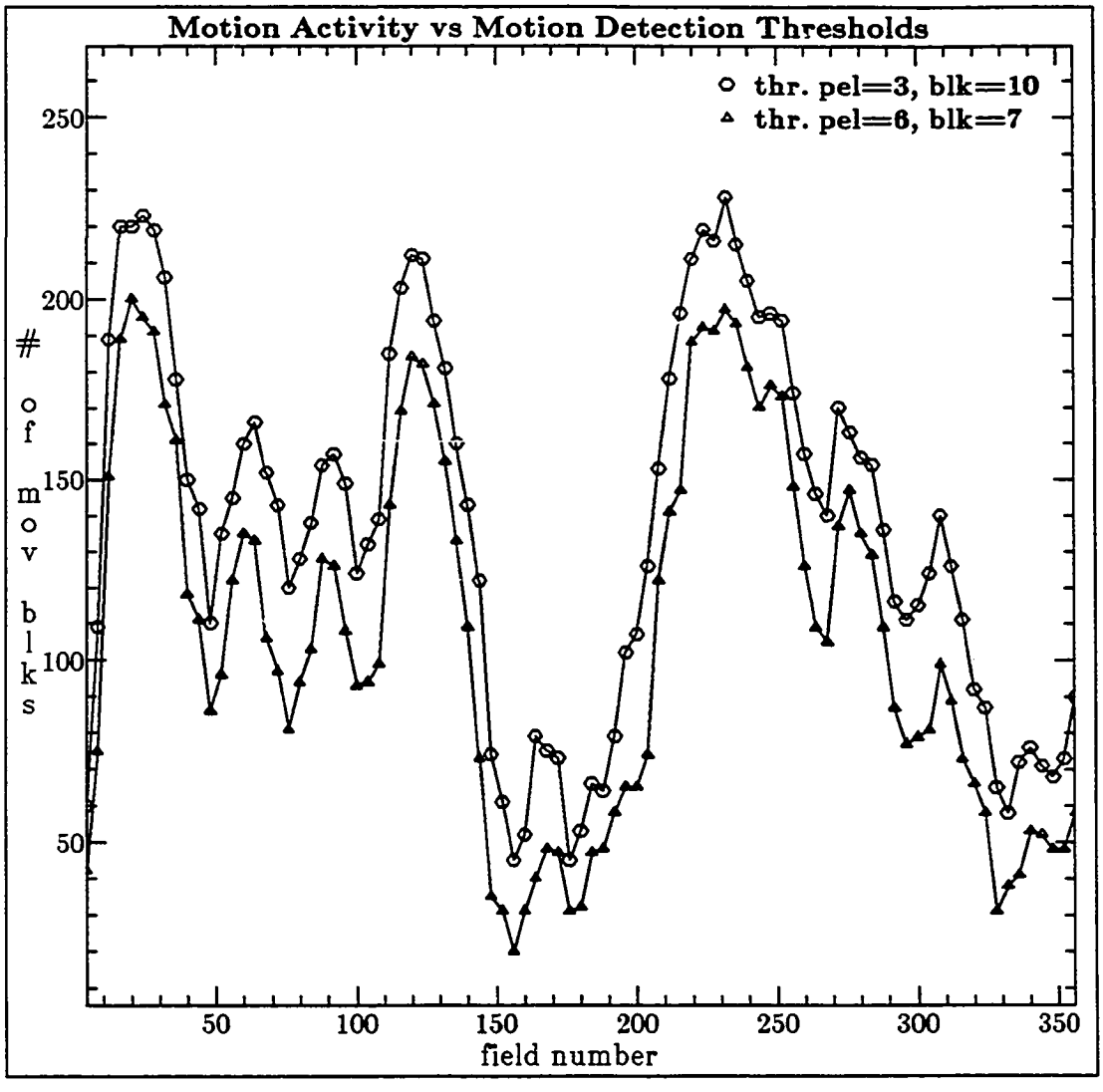


Figure 3.20(a) Variation in Motion Activity with Motion Detection Thresholds for "Salesman" sequence

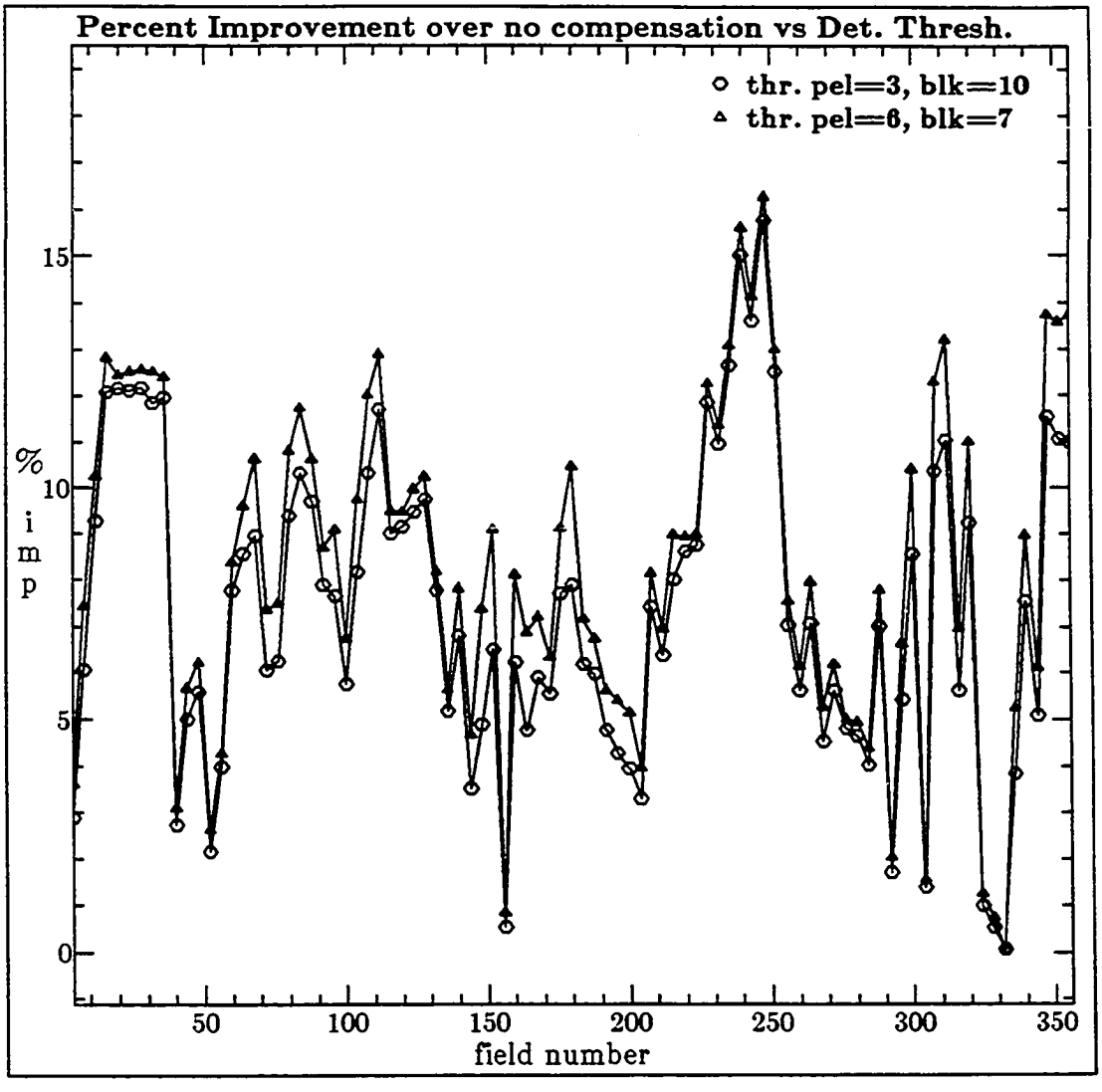


Figure 3.20(b) Percent improvement over no motion compensation with Variation in Motion Detection Thresholds for "Salesman" sequence.

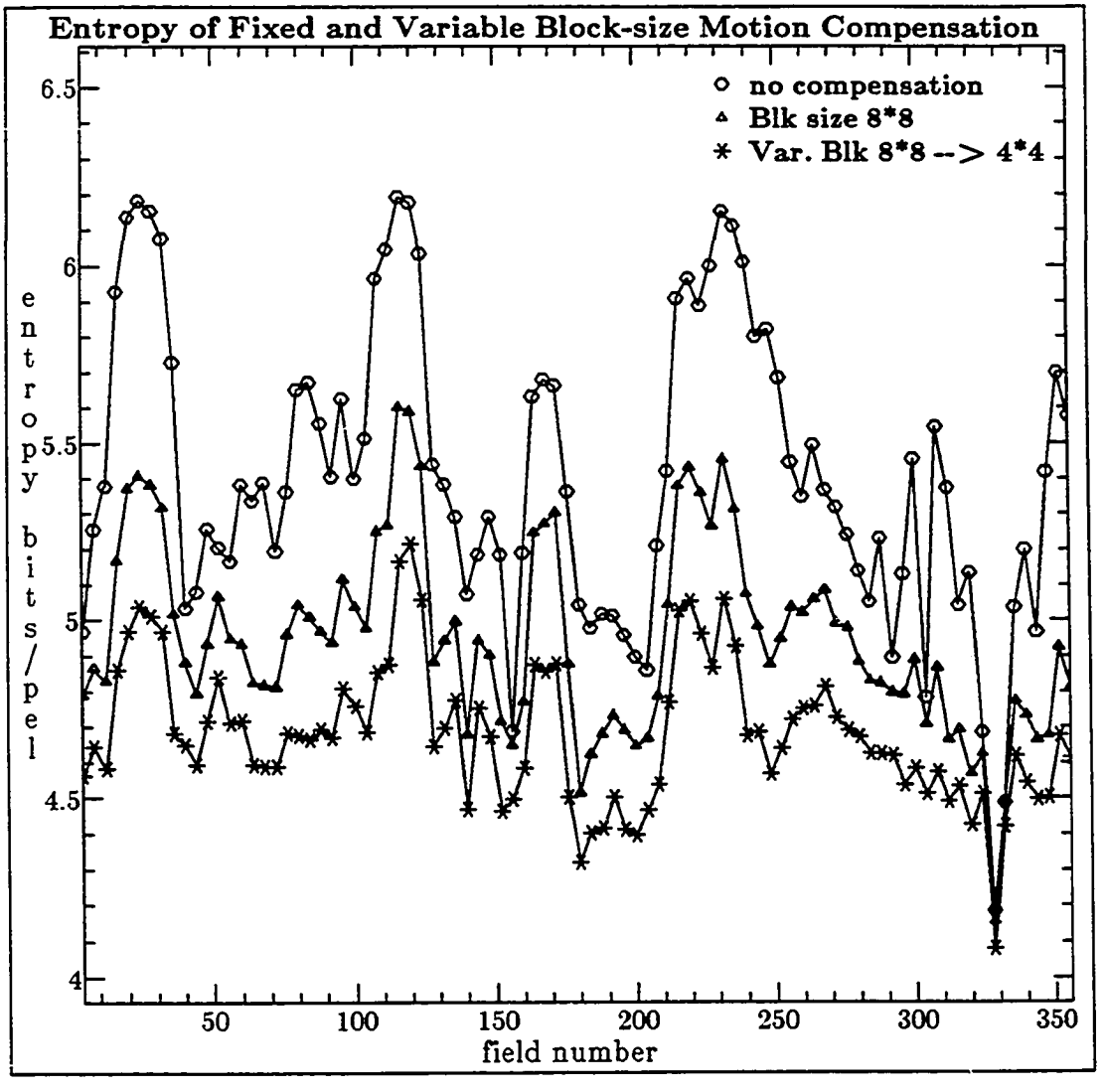


Figure 3.21 Entropy Comparison of 8*8 and VBS Block Size (8*8 --> 4*4) scheme thr. pel=6, blk=7, 4:1 temporal subsamp. : "Salesman" sequence

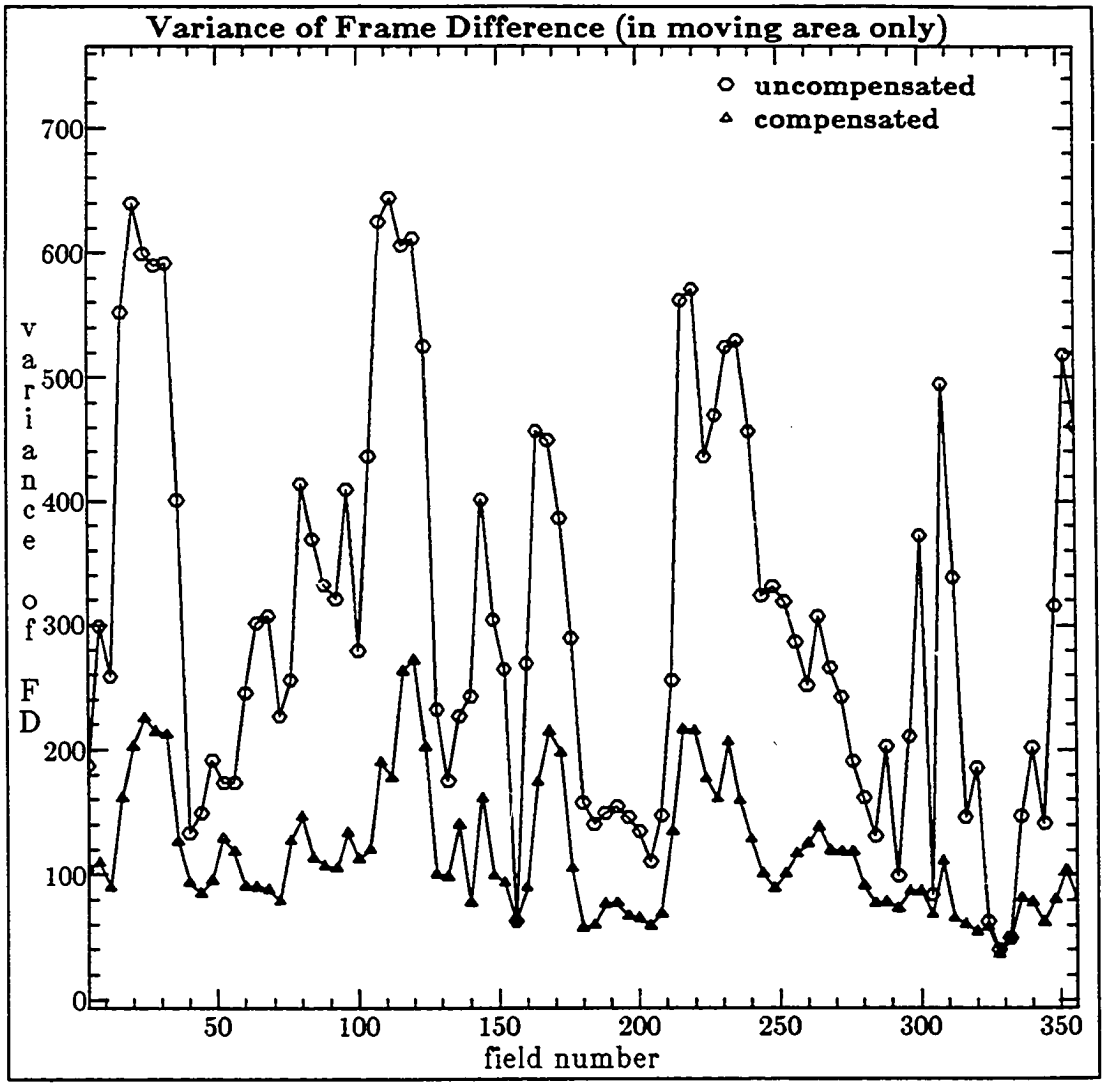


Figure 3.22 Variance of FD in moving area, 4:1 temp. ss : "Salesman" sequence

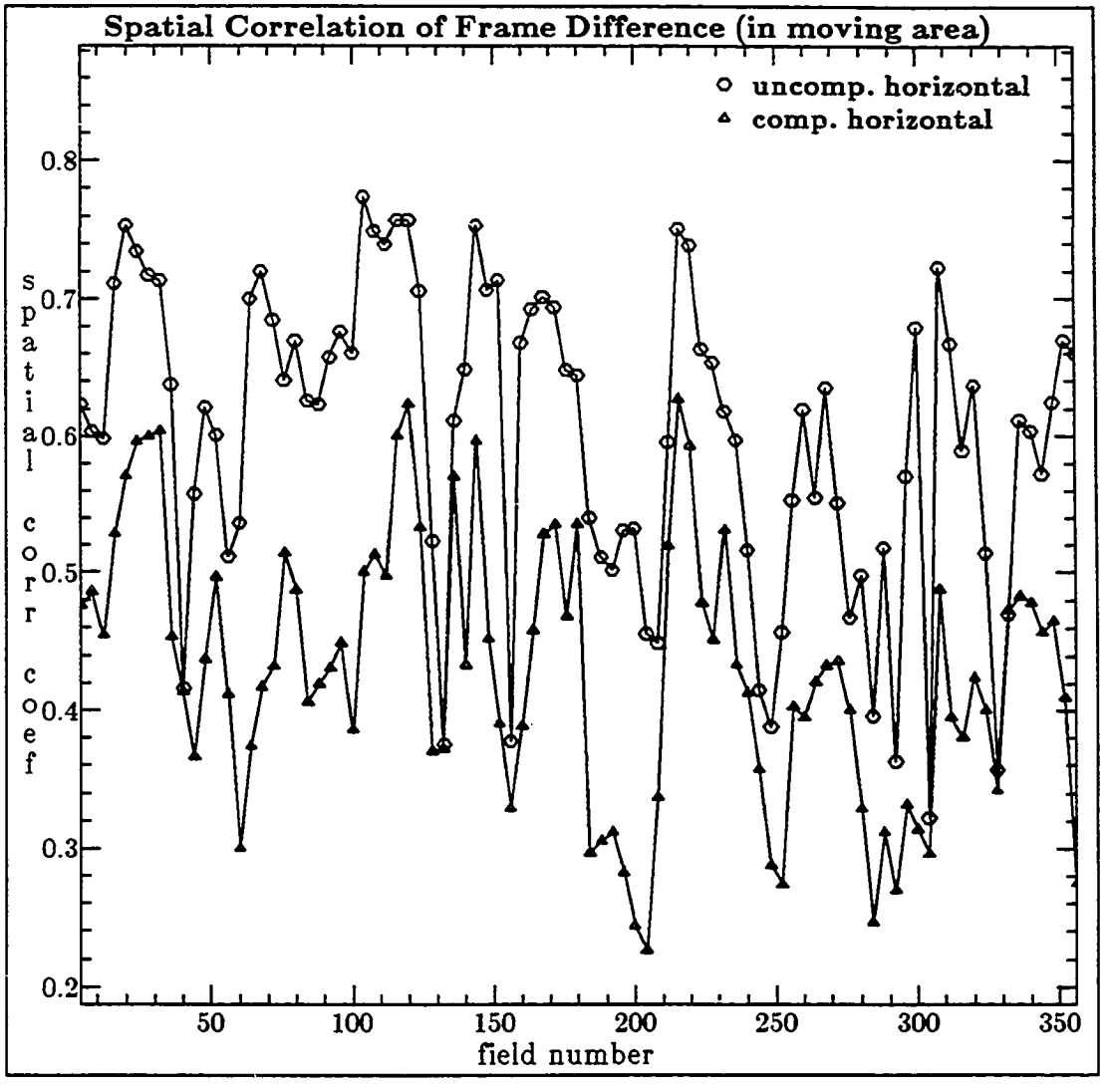


Figure 3.23(a) Spatial Correlation of FD in the moving Area : "Salesman" sequence

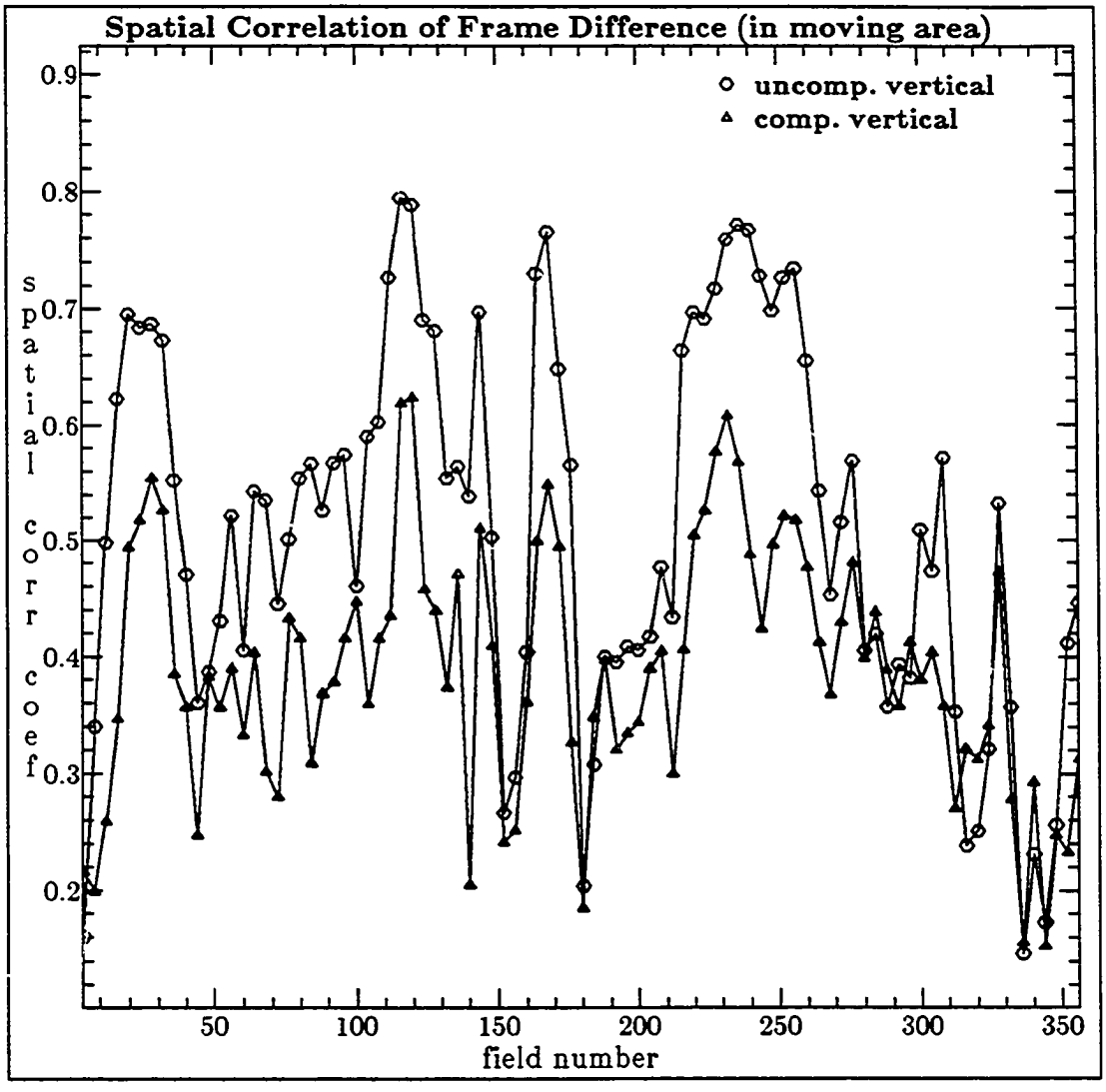


Figure 3.23(b) Spatial Correlation of FD in the moving Area : "Salesman" sequence

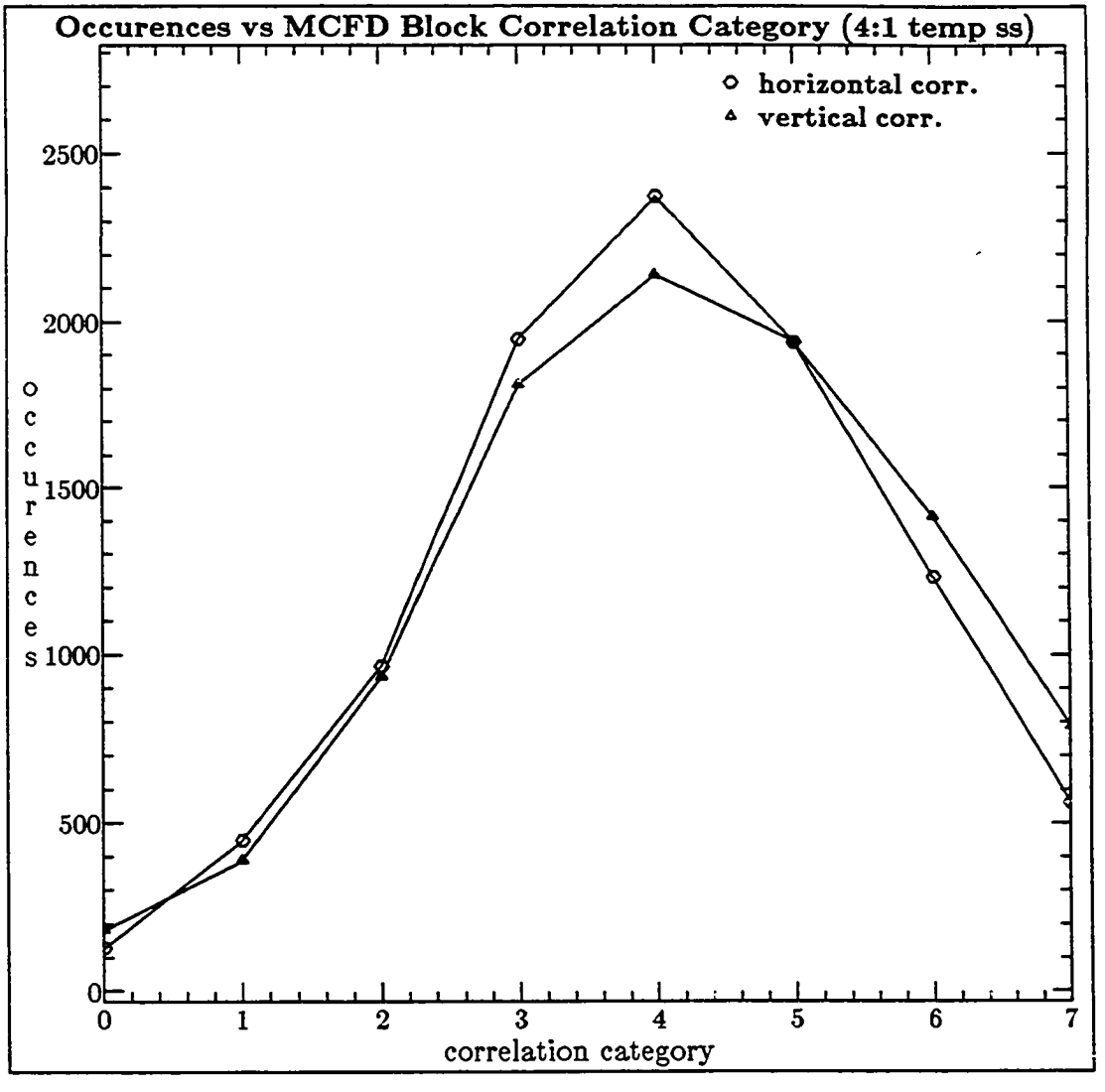


Figure 3.24 Number of blocks in MCFD Block Correl. Category: "Salesman" sequence

correlation category	correlation range	correlation category	correlation range
0	-1.00 to -0.41	4	0.21 to 0.40
1	-0.40 to -0.21	5	0.41 to 0.60
2	-0.20 to 0.00	6	0.61 to 0.80
3	0.01 to 0.20	7	0.81 to 1.00

Figure 3.25(a) Original Images from "Salesman" sequence (at 4:1 temporal ss)

Figure 3.25(b) Frame Difference and Motion-compensated Frame Difference images

CHAPTER 4: MOTION-COMPENSATED TRANSFORM CODING

4.1 INTRODUCTION

In the previous Chapter we discussed several block-matching search algorithms that efficiently estimate and compensate for motion in a scene, and we measured several statistical properties of frame differential (FD) and motion compensated frame differential (MCFD) signal. A thorough understanding of the properties of MCFD signal is important, as it describes the statistical nature of the signal, which may aid us, in designing suitable coding algorithms for MCFD signal. In addition, the knowledge of statistical nature of the signal, may also be beneficial in evaluating and comparing the expected performance, if various existing coding schemes are applied. Our aim in this chapter, is to investigate and compare the performance of various low bit-rate coding algorithms that use transform coding schemes for motion-compensated coding. In this context, we first briefly review some recent techniques [49],[88],[93]-[95], used in low bit-rate motion-compensated coding. This will serve two purposes, first, it will allow us to compare relative performance of various algorithms in low bit-rate motion-video coding, and to seek out promising ideas for new algorithms (to be discussed in this and the following chapters), second, combined knowledge of specific limitations and difficulties of various algorithms when used for motion-video coding, may help us in suggesting possible improvements.

4.2 MOTION-COMPENSATED BLOCK-CODING: A REVIEW

Many techniques for low bit-rate coding of images have been proposed in the past [1]-[10]. Many recently proposed schemes for motion-video compression, use block approach for simplicity of processing; here, we are interested in schemes, that

specifically use blocks for motion-estimation and compensation, and subsequently perform block coding [49],[88],[89],[94],[95]. At the outset, we note that many of above listed schemes incorporate block matching approach for motion-estimation, and 2-D DCT for transform-domain coding. Typically, a hybrid technique that combines predictive and transform-domain approach is used. Two configurations are possible, depending on whether the predictive operation is applied first, and then followed by the 2-D transform or vice-versa. We describe the complete operation of the coder in two configurations as follows.

In the *first* configuration, the motion-estimate for each current frame block is obtained by following a block-matching techniques like the ones discussed in Chapters 2 and 3. We then form a displaced previous frame by compensating the previous frame with this displacement estimate. The predictive operation is applied first, i.e., the corresponding block from previous frame is differenced on a pel by pel basis to form a block of frame differences (MCFD). The transform operation is applied next on the MCFD block of pel differences, i.e. they are 2-D DCT transformed. The block of transform coefficients is then scanned and significant ones retained, quantized, and transmitted.

In the *second* configuration, a motion estimate of current frame block is obtained as before. The blocks of previous frame are then compensated by this estimate to form a displaced previous frame. The transform operation is applied now, i.e., 2-D DCT transform is performed on the original blocks of the current frame to obtain a block of coefficients. A 2-D DCT transform is also applied on the blocks of displaced previous frame to obtain blocks of coefficients. The predictive operation is applied next, i.e., the corresponding blocks of coefficients from these two frames are differenced to

obtain coefficient difference blocks, that can be efficiently encoded. The differential coefficients for every block are then scanned and significant ones retained, quantized, and transmitted.

Studies [49],[89] chose to investigate the two hybrid configurations discussed, and evaluate their performance in coding real image-sequences; the first configuration was preferred. Other studies [94],[95],[105] also use block-matching motion compensation and DCT coding in the first configuration. Several ways of improving the performance of basic configuration are being investigated. They include, techniques for efficient representation of motion-vectors, selection of methods for scanning coefficients, selecting zones for retaining coefficients, combining zone selection and zig-zag scanning, saving bits by exploiting large zero runs, assigning bits to block of coefficients proportional to the energy contained in the block, selecting between a set of variable word length codes to better match statistics of coefficients, combining transform and pel domain schemes, various vector and scalar representations for blocks of coefficients or pels, ways of economizing on overheads to specify locations of significant pels in the block, etc. We review a few coding algorithms that use some of the approaches mentioned above.

4.2.1 Coding Algorithms

Even though the basic scheme combining block-matching and block-coding is employed in the following algorithms, several different ways of improving the performance are investigated in [94],[95],[104],[105]. We briefly outline some improvements, and details coding algorithms employed to encode the motion-compensated frame difference signal.

4.2.1.1 Coding Algorithm 1

The approach for motion-compensated coding followed in [95],[108] uses the second configuration, discussed earlier. Special variable word-length codes are used to inform the receiver of zero coefficient runs, and thus allow their efficient representation. Several ways of retaining the significant transform coefficients and sending the addressing information to the receiver are considered. A technique found to be effective for this involves defining rectangular zones within the transformed block. Extra overhead address bits are then used to identify these zones and are transmitted to the receiver. The tradeoff's in transmission of motion-vectors by 1-D versus 2-D codes is also investigated.

4.2.1.2 Coding Algorithm 2

The approach followed in [94], [107] also uses the second configuration, discussed earlier. The first time prescence of pulsive components in the motion-compensated frame difference signal has been acknowledged. These pulsive components are found to cause an undue spread of significant coefficients in the transform domain. To solve this problem, a separation filter is designed, to separate pulsive and non-pulsive regions. The non-pulsive region in the block is DCT coded. The DCT coefficients inside a zone are linearly quantized and DCT coefficients outside the zone are linearly quantized. The quantization results are variable word length coded. The pulsive components are dealt with spatial domain quantization techniques. Their addresses as well as their magnitudes are sent to the receiver.

4.2.1.3 Coding Algorithm 3

The basic approach in [105], [106], uses the same configuration. The DCT

transformed blocks are classified depending on the coefficient power distribution. Several schemes for classification are compared, one is based on position of significant coefficients in the block, another is based on power of significant coefficients in the block. To identify the position of significant coefficients, a mask containing 1's and 0's is created, and encoded. A vector method predefining, arrangements and magnitudes of coefficients is also investigated. Each retained, DCT coefficient is uniformly quantized and then variable word-length coded (VWLC). The VWLC's are changed to suit probability density of each quantization index.

4.2.1.4 Coding Algorithm 4

The approach for motion-compensated coding in [104], investigates a Vector Quantization (VQ) based coding approach. A new motion-vector transmission method is investigated. A local decoder-filter in the coding loop, is proposed and is controlled adaptively by the motion detection information. An adaptive intra/interframe mode can be selected, based on energy comparisons in motion-compensated prediction error signal and the original block of image. Vector of pels are quantized (VQ) together by searching for the best match in a pre-designed codebook containing standard vectors. A hybrid quantizer combines the vector quantization (VQ) and scalar quantization (SQ) approach. Two stages of vector-codebooks are used for quantization, if the differences are still large, the rest of pels are coded by scalar quantization.

4.2.2 Block-size Choice and Low bit-rate Image-Sequence Coding

From motion estimation point of view small size blocks are more adaptive to local image motion and can be effective in compensation where motion is complex. They however require a larger overhead for transmission of now increased number of

displacement estimates. Thus there is a tradeoff of block size versus overhead. As pointed out earlier, from coding point of view smaller blocks require less hardware for transform operation and handling of blocked data. Larger block sizes allow exploitation of similarities present within the block and allow efficient compression. However if the block size is too large then the benefit may not be much as correlation values are usually high only for neighboring pels. Choice of block size of 8×8 has been found to hold a reasonable compromise for keeping the complexity of implementation low for motion estimation and compensation as well as for transform coding.

In coding of single images, studies have shown that for typical scenes, a high degree of correlation (>0.8) exists between the adjacent samples both along horizontal and vertical directions. For such applications the KLT is theoretically considered to be optimum. Image data is highly nonstationarity and KLT is complex to implement. This is not a major problem as DCT can be employed for this application and for a high degree of correlation in the data performs very close to the optimal KL Transform. In fact at present the DCT is the most popular efficient transform employed for image coding. This approach can be extended to interframe coding of signals, where a one dimensional predictor in temporal direction is employed in conjunction with a 2-D transform to exploit the redundancy. Correlation coefficients measured in the horizontal and vertical directions on the prediction error signal [Chapter 3] have shown that correlation values for interframe difference signal usually lie above 0.6. For such signals a transform like DCT might still be able to decorrelate the data with high efficiency, and therefore be a adequate choice. Our correlation coefficient measurements on block motion compensated frame difference signal [Chap 3] show that as long as motion compensation works well the correlation coefficients are much lower as compared to the previous case and usually range between 0.25 and

0.55. The results were identical if either the entire frame was used for these measurements or only the moving blocks were used for these measurements. Based on [72],[80] it seems that for these low values of correlation DCT may not be the best choice. We can perhaps also employ either separable KLT (for correlation of .36 or 0.5) , or DST. However, because of complex motion such as rotations, scale changing, hidden area appearance etc., if the block matching (with blocks of size 8x8) does not work very well then the correlations might again be significantly high requiring the need of transform such as KLT or DCT.

The performance of various transforms on real low correlation image data has not been investigated, the decorrelation and energy packing results for low-correlation data have been based on the first order Markov model assumptions.

From [17],[21] we note that low bit-rate coding system, employed for compression of image-sequences reveals that several bit rate reducing measures like spatial and temporal subsampling, motion compensation, conditional replenishment, and block coding are necessary to achieve a very high compression. The need of motion estimation and compensation procedures for reduction of information rate as well as for improved visual perception of reconstructed picture sequence, has been thoroughly acknowledged, and the importance of block coding methods like *transform techniques* has been emphasized for efficient representation and compression of video signal for transmission purposes. Further, the image-sequence coding schemes are still in a stage of infancy as compared to intraframe coding schemes. The techniques commonly applied for coding single images have basically been applied to the motion compensated signal, despite the differences in statistical nature. Various researchers are presently studying the peculiarities of this signal, in hope of suggesting

modifications that can significantly improve the existing techniques, and allow better encoding at lower bit-rates. So far, we have briefly reviewed a few block-based approaches for encoding this signal for low bit-rate applications. We note that even though some of the techniques listed are promising, additional investigations are necessary to improve performance thereby allowing good quality reconstruction at low bit-rates.

4.3 MOTION-COMPENSATED TRANSFORM CODING SCHEME

4.3.1 Introduction to Basic Approach

We present studies on the motion-compensated transform coding schemes using the block-matching technique. First, we present an efficient block motion-tracking algorithm that provides good motion estimation while minimizing the computations. Next, we investigate the effectiveness of various orthogonal transforms on the motion-compensated frame differential signal. Finally, the results of coding a real image sequence using various motion-compensated transform coding algorithms are discussed.

Many recent efforts towards efficient image sequence compression employ techniques to detect, estimate and compensate for motion of objects in a scene. The so-called *motion-compensated coding schemes* estimate the displacements of moving objects and only encode the pel differences between the current frame and the translated previous frame in the moving areas of the picture. Researchers have shown that this technique can increase the coding efficiency significantly [Chapter 3],[21],[49]-[51],[88].

One popular technique to estimate the motion displacement is the block matching approach [21]. In such a technique each picture frame is first partitioned into fixed-

size blocks and the estimation of displacement vectors is then performed on each block by a matching technique.

Our proposed block-matching motion-compensated coding scheme has three elements: (1) a *motion detector* which detects the moving blocks, (2) a *displacement estimator* which estimates the displacement vectors of moving blocks, and (3) a *data compression algorithm* which encodes the inter-frame differences after motion-compensation. In this Chapter, we focus on the performance comparison of various orthogonal transforms for encoding motion-compensated frame differences. In order to reduce the computational complexity for motion estimation, we adopt a motion tracking algorithm in which an initial guess of the displacement is obtained from the previous frame displacements, and a small update is then calculated to correct this guess.

We first examine the effects of the motion tracking algorithm on a test picture sequence assuming perfect picture reconstruction. This ideal-case simulation is helpful in measuring the performance of the basic motion estimation algorithm. Three orthogonal transforms (DCT, DST and 1-D KLT) are then chosen as possible candidates for coding the motion-compensated frame-differential signal. We then compare the transform domain behavior of motion-compensated frame differential signals for the chosen transforms.

Next, a (theoretically) asymptotically-optimum quantizer is added into the coder loop to judge the coding performance on the test pictures. Among the three transforms (DCT, DST and 1-D KLT) in our study, DST seems to have a small performance margin in the coded picture quality and the compressed data rate.

4.3.2 Motion Detection and Estimation

A motion detector is first used to classify a block to be *moving* or *non-moving* before any motion estimation can be done for that block. It screens the original image blocks for motion so that the displacement estimator, which usually requires much more computations than the motion detector, works only on a smaller number of blocks which are actually moving. In addition, it also helps in reducing the number of blocks that may be erroneously classified as moving by the displacement estimator, when blocks are in low detail region and/or have random camera noise. The motion detector is thus necessary to ensure a zero displacement for the unchanged region. The design of our motion detector is motivated by earlier works [27]-[29]. Two parameters, T_0 and N_0, are used. A pel at location (x,y) in frame k is called *moving* if $|s_k(x,y) - s_{k-1}(x,y)| > T_0$, where $s_k(x,y)$ and $s_{k-1}(x,y)$ represent pels in frames k and k-1 respectively. A block is called *moving* if the number of moving pels in that block is greater than or equal to N_0. For "MsUSA" sequence, the parameters, $T_0=3$ and $N_0=10$, work adequately for blocks of size 8×8 and 8-bit pels in the range 0-255. For "Salesman" sequence, the parameters, $T_0=6$ and $N_0=7$, are used as discussed in Chapter 3.

The goal of a block-displacement estimator is to find the best match of a block from frame k in a suitable *match area* in the previous frame k-1. The (absolute) detectable displacement is assumed to be less than X_m along horizontal (x) axis and Y_m along vertical (y) axis. As shown in Figure 3.1(a), if the size of a reference block is $M\times N$, the match area is $(M+2X_m)\times(N+2Y_m)$ where X_m and Y_m assume discrete values for digitized images.

The above problem consists of seeking for a best match location by evaluating a matching criterion at every point in the displacement *search region* (SR) having $(2X_m+1)\times(2Y_m+1)$ points. In most practical cases, M is often chosen to be the same as N and $X_m=Y_m=p$ and hence SR has $(2p+1)^2$ points. Each point in SR is a candidate for being the correct displacement vector and will be called a *search point.*

A matching criterion called *number of thresholded absolute differences (NTAD)* is employed in our motion estimation scheme and is defined as [6]:

$$NTAD(u,v) = \sum_{y=1}^{M}\sum_{x=1}^{N}\left[f(T_0,|s_k(x,y) - s_{k-1}(x+u,y+v)|)\right] \text{ and } f(T_0,a)= \begin{cases} 1 & T_0 < a \\ 0 & T_0 \geq a \end{cases} \quad (4.1)$$

This criterion seems to perform slightly better than the often used *mean absolute (frame) difference (MAD).*

If no prior information about the movement of object is known, every search point in SR is an equally likely candidate for the best-match. However, due to high temporal correlation in the movement between frames [59],[88],[109] the previous-frame displacement vectors in the neighborhood of the current block form a good initial estimate of the current displacement vector. In other words, a best-match can be obtained by a small search centered at the initial displacement estimate and this search need only be conducted on a small search region.

The motion tracking technique employed is similar to that of dependent search of Chapter 3. Let $D_k(x,y)$ represent the displacement vector of the current block and $D_{k-1}(x,y)$ be the displacement estimate of the block at the same location in the previous-frame. Then,

$$D_k(x,y) = D_{k-1}(x,y) + U_k(x,y), \quad (4.2)$$

where $U_k(x,y)$ represents a small *update* or a correction term. The center of current

SR is therefore shifted by the displacement vector of the previous-frame block ($D_{k-1}(x,y)$) at the same location before an update can be made. A small update on the initial estimate is usually sufficient to obtain the best estimate of motion for the current block. The update term is found by comparing the matching criteria (eqn. (4.1)) at all the positions in the small search region. As discussed in Chapter 3, to account for the limitations of this approach such as moving objects changing directions rapidly and to increase the robustness of motion estimation, the zero-displacement vector (no motion) is also included in the final selection of displacement vector. In other words, one additional search point (zero displacement) is checked in finding the best-match.

In order to evaluate the effectiveness of the motion estimation algorithm, we conduct a experiments on a real video sequence. The original "MsUSA" sequence and "Salesman" are digitized from NTSC signals sampled at twice the color subcarrier frequency and 8 bits per pel. The sequences are then converted to component form consisting of a luminance signal Y and chrominance signals U and V. A line of luminance contains 368 pels, and there are 240 such lines per field. Since our application is low-bit rate video-conferencing, we further reduce the spatial resolution by subsampling in the horizontal direction by a factor of 2. The image field now consists of 184 *pels* $\times$ 240 *lines*. This is converted into block format with block size of 8$\times$8. Each field now contains 23 Y blocks in the horizontal direction and 30 Y blocks in the vertical direction, and will be referred to as a "frame" in the terms such as *frame difference* used in here. Performance of motion compensation algorithms is investigated at a temporal subsampling rate of 4:1, i.e., 15 fields (also referred to here as frames) per second. The displacements in the high-motion areas in the image-sequences still do not exceed 5 to 6 pels. This is typical of most movements in a

video-conferencing environment.

Figure 4.1 shows that the entropy of the motion compensated frame difference signal is smaller than the uncompensated signal by as much as 35% (depending on the motion activity). Further, comparing the performance of motion-tracking algorithm for $p=3$ to that of the simple search (exhaustive with zero initial estimate) with $p=6$, we also note that the two algorithms perform identically. Since the motion-tracking algorithm is computationally less complex while it maintains the performance of exhaustive search, it is employed in our inter-frame coding system.

4.3.3 Orthogonal Transforms on the Motion-Compensated Frame-Differences

The frame differences after motion compensation (motion-compensated frame difference, MCFD) have to be coded and sent to the receiver if their values are significant. Many coding algorithms may be used to encode such (MCFD) signals. The popular ones are predictive coding and transform coding [21],[49]-[52]. In this Chapter, we concentrate on techniques for transform coding of MCFD signals. We now discuss some details of a block-matching motion-compensated transform coding system.

4.3.3.1 Block Classification and Transform Coding System

We separate the MCFD blocks into two categories: *compensable* and *uncompensable*. The ones that can pass the motion detector -- small-magnitude MCFD blocks -- are *compensable*. Therefore, all the image blocks are classified into one of the following three types as shown in Figure 4.2.

Type 1: *nonmoving blocks* -- Their values are small and hence identified by the motion detector as non-moving. They usually belong to the unchanged area like the background in images. In a real coder a special short code will be necessary to indicate the nonmoving blocks to the receiver.

Type 2: *compensable moving blocks* -- These blocks are originally identified as moving by the motion detector, but after motion compensation their values are found below the motion activity threshold. i.e., they belong to the changed areas where motion compensation works well. For such blocks only the motion vectors need to be transmitted.

Type 3: *uncompensable moving blocks* -- These blocks are labeled as moving by the motion detector but cannot be well-compensated by a simple block translation because of the newly exposed region (uncovered background) or the rotational movement of regions. Both motion vectors and coded MCFD values are required for reconstruction of these blocks.

Based on simulation results, assuming perfect reconstruction, approximately half of the overall MCFD blocks are found to be of Type 1; i.e., they are not moving. Among the moving blocks, around half of them are well-compensable (Type 2) and the rest of them are classified as uncompensable. This means that the data compression algorithms need only be applied to about a quarter of the original blocks in each frame after motion-compensation. In a real coding system the quantizer characteristics might change the number of blocks for each category as some coding errors might not be negligible and may result in inaccurate representation of the original picture. The difference between the original and the reconstructed picture may not be significant if a fine quantizer is employed but it may be substantial if a coarse quantizer is

employed.

Figure 4.3 shows the block diagram of a complete block-matching motion-compensated transform coding system. Figure 4.3(a) shows the transmitting end of the system, which includes a motion detector, an estimator and compensator, and a transform encoder. Figure 4.3(b) shows the receiver which includes a motion compensator and a transform decoder.

4.3.3.2 Transform Choices for Coding

The discrete cosine transform (DCT) has been recognized as an efficient coding technique for single picture (intraframe) coding; however, it is not clear that DCT would also be an efficient coding technique for MCFD because the statistical properties of MCFD are very different from those of the original signals [Chapter 3]. For instance, the pel-pel spatial correlation coefficient of MCFD is typically in the range 0.30-0.55, which is much lower than the pel-pel correlations typical (0.9) of original images. For highly-correlated signals, DCT has a good energy compaction and decorrelation property, if the signal source can be modeled as a first-order Markov model [2]. However, in the cases of low-correlation signal sources, other transforms may have a better energy compaction and decorrelation property [2],[72],[78],[80]. Of course, this does not imply DCT may be a bad candidate in encoding MCFD because the first-order Markov model may not be adequate for the MCFD signals in the first place. Rather, the above observation motivates us to compare different transforms on the MCFD's.

Three orthogonal transforms for coding MCFD signals are studied: DCT, discrete sine transform (DST) and Karhunen-Loeve transform (KLT). The DCT and the DST are well known ([Chap 1],[2],[72]) and will not be stated here. The true 2-D KLT derived

based on the autocorrelation matrix of test data is complex and picture-dependent. In this study, in order to obtain a simple and practical KLT, we assume that the MCFD signal source is a separable first-order Markov model. Hence, the 2-D KLT in this case is an outer product of two 1-D KLT's and the 1-D KLT basis vectors have a closed-form representation [2] which is controlled by only one single parameter -- the pel-pel correlation coefficient. To get the KLT basis vectors, we first solve ω_p (between 0 and π) in the following equation:

$$\tan N\omega_p = \frac{-(1-\rho^2)\sin\omega_p}{(1+\rho^2)\cos\omega_p - 2\rho}, \tag{4.3}$$

where N is the size of the transform and ρ the pel-pel correlation. The KLT basis vectors $\{t_{pq}\}$ are then obtained from

$$t_{pq} = \left(\frac{2}{N+\lambda_p}\right)^{1/2} \sin\left[\omega_p\left(q - \frac{\left(N-1\right)}{2}\right) + \frac{\left(p+1\right)\pi}{2}\right], \quad 0 \le p,q \le N-1, \tag{4.4}$$

in which the eigenvalue λ_p is calculated by

$$\lambda_p = \frac{1-\rho^2}{1+\rho^2-2\,\rho\cos\omega_p}. \tag{4.5}$$

We use ρ=.5 as the parameter to generate our KLT since it roughly approximates the overall measured correlation coefficient of the MCFD in our previous study [Chap 3], [101].

We first calculate the mean and variance of the transform coefficients (Tables 4.1, 4.2 and 4.3) for the uncompensable moving blocks of the three transforms (DCT, KLT(0.5) and DST). The tables show that the energy compaction ability for all of the three transforms appears to be nearly same. Typically about 30% or more transform coefficients are significant, and the DC component variance is of the same magnitude

as the next few higher frequency coefficients.

We further analyze the contents of few sample blocks of MCFD and their transform coefficients (Tables 4.4 and 4.5). The MCFD blocks contain irregular distribution of patterns of positive and negative values, some significant and others insignificant. Looking at the coefficients in transform-domain for each of the transforms we again cannot find a regular structure of the dominant coefficients. The large-magnitude coefficients seem to be widely spread in all cases and retaining only a few low-frequency components may not be the best strategy. One possible solution is transmitting the locations of the dominant coefficients for each block; however, it requires large overhead. So far, it is not very clear which transform is more suitable for encoding the MCFD's.

4.3.4 Simulation Results for single Transform Coding

We have discussed various parts of our block motion-compensated transform coding system, except for the quantizer. Since our statistics of transform coefficients do not seem to provide additional information for quantizer design and bit allocation, we therefore use a simple quantizer: a uniform midtread quantizer with small quantization steps. In theory, a fine uniform quantizer together with an entropy coding on the quantization indices would perform close to the (nonadaptive) optimum entropy-constrained quantizer designed for known probability distributions [1],[2],[6]. A modified threshold sampling scheme is used to select the dominant coefficients: only the coefficients larger than the threshold (= twice of the quantization step) are quantized. In addition, we discard the three highest frequency components that are significant and retain the 4 lowest frequency components even if they are small. Finally, we measure the entropy of the quantized coefficients. Overall, this quantizer

should be universal and close to the theoretical optimum quantizer limit.

We include various performance measures for comparing transform coding schemes using fixed quantization with step sizes of 3, 6 and 9. The performance results on "MsUSA" sequence for DCT, KLT(0.5) and DST are shown in Figures 4.4, 4.5 and 4.6 respectively. Figures. 4.4(a), 4.5(a) and 4.6(a) show the motion activity in "MsUSA" sequence for each of the three transforms when various quantization step sizes are used. We see that in all cases, with a coarser quantizer the number of uncompensable blocks increases whereas the number of compensable blocks decreases slightly as expected. For each of the transforms, increasing the quantization step-size increases the mean square reconstruction error (MSRE) as shown in Figures 4.4(b), 4.5(b) and 4.6(b), and reduces the entropy bit rate generated of quantized pels (Figures 4.4(c), 4.5(c) and 4.6(c)) as one would anticipate. For the "Salesman" sequence some results for DCT coding when quantization step-sizes of 5, 9 and 13 are used, are displayed in Figure 4.8. In Figures 4.8(a) and 4.8(b), the results of entropy performance and MSRE's of uncompensable blocks are displayed. We note that difference in entropy coding bits is smaller in going from a step size 9 to 13, as compared to that in going from step size 5 to 9; the difference between MSRE's is larger in going from step size 9 to 13, as compared to that in going from step sizes 5 to 9. The entropy and MSRE comparison of DCT, KLT(0.5) and DST transform for "Salesman" sequence is shown in Figures 4.9(a) and 4.9(b); the differences are very small.

Statistically, the overall performance of these three transforms is very close both for "MsUSA" and "Salesman" sequence, and there is no clear winner. To allow a simple comparison of various transform coding algorithms, some results from Figures 4.4, 4.5 and 4.8 are selected and redrawn in Figure 4.7. The comparison results for step size of

9 (entropy bits and MSRE's) of DCT, KLT(0.5) and DST coding schemes are displayed in Figures 4.7(a) and 4.7(b). For "MsUSA" sequence, we observe that all three transform coded sequences for step size of 9 have very nearly the same mean square error; however, the KLT(0.5) and DST on average require a few less entropy bits per frame then the DCT. For "Salesman" sequence the entropy bits and MSRE performance, for all three transforms, is almost identical.

Finally, visual comparison is performed on the reconstructed picture sequences to compare the picture quality of DCT, KLT(0.5) and DST coding algorithms. The subjective differences in picture quality are found to be very small. On careful observation for "MsUSA" sequence, DST coded sequence seems to be perceptionally superior to the DCT coded sequence (Figures 4.13(a) and 4.13(b)) by a small margin. Overall, for "MsUSA" sequence DST coding scheme is preferred both in terms of entropy bits required to code the picture as well as the reconstructed picture quality. For the "Salesman" sequence, visual quality differences are even smaller (Figures 4.14 (a) and 4.14 (b)) and no conclusion could be reached.

4.4 MULTIPLE-TRANSFORM APPROACH FOR MOTION-COMPENSATED CODING

A popular approach to motion-video compression is the block-matching motion-compensated coding scheme which estimates and compensates for motion of objects in a scene [21],[101], and then, encodes the motion-compensated frame-differential (MCFD) signals. Transform domain techniques are often employed for encoding the MCFD signals [2],[21],[102]-[103]. In this paper we investigate one possible improvement over the standard single transform coding approach; that is, different

transforms may be applied to the MCFD blocks within the same frame.

We first briefly describe the structure of a basic motion-compensated transform coding system. In a previous study [102] we have presented some coding results when one of the following transforms is applied to all the MCFD blocks -- DST, KLT, or DCT. Here, a best transform from a pre-decided set is selected for each MCFD block to maximize the coding performance. One of the important performance measures we employ is the number of bits used in coding. In order to estimate the actual bits used for each individual block, a real variable word length (VWL) code, rather than the statistical entropy bits in our previous study [102], is used to encode the quantized transform coefficients. However, as shown by our experiments the bit-rate results obtained using both measures, entropy and VWL, are found to be quite consistent.

In this study, we choose a few well-known orthogonal transforms such as KLT's (of different parameters), DST and DCT as our candidates. In selecting of transforms coding configurations from the above set of transforms several possibilities exist. It could be a two-transform coding system in which one out of two transforms is applied to a data block, or a 4-transform coding system that has 4 candidates for each block. It could be a 2-D transform applied to the whole block, or two separate 1-D transforms chosen for the horizontal and the vertical directions within a block. A few configurations have been tried. The selection of the transform for a particular MCFD block is decided either by the statistical characteristics of that block or by the coding efficiency -- number of bits required for coding that block. In the above schemes, the transform selection information may have to be transmitted to the receiver, which costs one to two bits per block.

Finally, the performance of various transform coding configurations is evaluated using

both statistical and visual criteria on a test sequence consisting of a typical head and shoulders view. The results suggest that the multi-transform scheme provides a slight improvement over the single transform coding schemes for coding MCFD signals. These results may also suggest the limitations of the orthogonal transform coding algorithms on the MCFD signals.

4.4.1 Entropy versus VWL Coding Bits

Up to now, in our studies, we have used entropy bits as one of the statistical criteria, for measuring performance of a scheme and comparing it to others. We now implement a variable word-length (VWL) code set, to compute the actual bits required in coding. This serves two purposes, first, it justifies our use of entropy bits as roughly the actual bits required, and second, it serves as a measure and allows us to use VWL bits in selection of transforms in *multi-transform* approach. The code-set for VWL coding is one-dimensional, and no statistically optimization was done for either of the two sequences. For the "MsUSA" sequence the actual VWL coding bits and the entropy bits are compared in Figure 4.10; we observe that, the VWL coding bits are roughly, only 10 percent larger than entropy bits. The results for "Salesman" sequence were similar, and are not being included here. We conclude that, the VWL coding bits are quite close to the entropy bits, and hence the entropy basis is a reasonable measure for comparing the performance of various schemes.

4.4.2 Block-Correlations and Transforms

In the previous chapter we find that the measurement of one-step correlation-coefficients in MCFD's along horizontal and vertical directions on a block basis, reveals considerable variations from block to block. The correlations are also found to

be considerably different in horizontal and vertical directions within a block, in many cases. This seems to suggest that a transform-coding scheme which takes into account individual block-correlations might be more efficient. Such a scheme, the non-separable 2D-KLT is computationally complex, and requires large-overhead for transmission of covariance matrix to the receiver. As a practical alternative, we limit ourselves to schemes that are much less complex and require a little overhead. Few such schemes are discussed in the next section.

4.4.3 Multiple Transform Approach

The *multiple-transform* approach, aims to exploit the wide variation in statistical nature of MCFD signal from block-to-block. In this scheme depending on the characteristics of the block being encoded, a selection is made, from a small set of pre-decided transforms. For every block encoded, some overhead is sent to the receiver to identify the transform selected. Many choices for transforms exist, and several configurations are possible. A few popular transforms are selected for investigations; and several configurations possible, are investigated. Several choices for the criteria to be used as the basis for transform-selection exist. As it is somewhat difficult to predict which ones would work better, many were tried and are listed here.

- The correlation-coefficients measured on a block basis are used to select an appropriate transform. From the pre-selected set, the transform with correlation closest to the, measured correlation of the block to be encoded, is selected. Decisions are made based on both, the horizontal and vertical correlation-coefficients of the block. Many ways to combine the two coefficients are possible.

- We compute the energy in an MCFD block in the spatial-domain, and the measured correlations in the block, and use them jointly as a switching criterion. This joint criterion, yields a robust basis for discriminating between blocks with differing complexity.

- We actually encode the MCFD block by use of all pre-selected transforms, and compute their VWL coding bits MCFD's reconstruction performance. The champion among possible candidates is then selected. The complexity of such a scheme is directly proportional to the number of candidates in the pre-selected set.

- We encode the MCFD block by use of all pre-selected transforms, and compute their VWL coding bits and MSRE performance. Now, the VWL bits and MSRE performance measures can jointly be used as a basis for deciding and selecting the champion among various candidates.

4.4.3.1 One Transform Selection per Block

In this configuration, we select a single transform for each MCFD block from the set of available choices. Sets containing 2 and 4 transform choices have been tried. For set with 2 transform choice, the DST and DCT transforms are selected as candidates; where as for set with 4 transform choice the KLT(-0.5), DST, KLT(0.5) and DCT are selected. Other choices for transforms can also be tried. Overhead required per block is 1 to 2 bits, corresponding to whether a set containing 2 or 4 transforms is selected. Either the VWL bits (best bit-rate) criterion or the correlation-coefficient based criterion are chosen as candidates, to permit selection of transforms for the block.

4.4.3.2 Correlation-Direction Based Transform Selection

In this configuration, we allow selection of transforms within a MCFD block. A set with 2 transforms is used, with the DST and DCT transforms, as the members. A simple scheme that allows selection of a transform pair for every block is considered. A different transform can be used for horizontal and vertical directions in a block; the first pair uses DST for horizontal direction and DCT for vertical direction, whereas the second pair uses DCT for horizontal and DST for vertical directions. An overhead of 1 bit per coded block is transmitted to identify the pair used. The choices in the transform pair are extended to 4, i.e., all 4 combinations of two transforms (DST and DCT) are valid candidates, the selection of combination is made and applied in horizontal and vertical directions. An overhead of 2 bits per coded block is transmitted to identify the pair used. A 2-transform coding scheme that allows a choice of transforms Computed correlations in horizontal and vertical directions of the MCFD block are used to select the transform pair chosen for encoding the block.

4.4.4 Simulation Results for Multiple Transform Scheme

As discussed, many configurations, number and choices of transforms to be included in the set, and various criterion make this study rather difficult. Out of many possible combinations, a few are chosen for statistical analysis; visual performance comparisons are made on even fewer schemes. The VWL coding bits/frame and MSRE's are used as measures in comparing performances. Some statistical performance results can be summarized as follows:

[1] For both image sequences, VWL bits/MSRE performance for DST/DCT based coding scheme is somewhat better then other combinations of transforms tried,

such as KLT(-0.25)/KLT(0.50) or DST/KLT(0.75) (Figures are not included).

[2] The difference in VWL performance of the best 4 transform switching scheme (using either VWL block bits or correlations as criteria), and the 2 transform switching scheme that uses correlation as the switching criteria is very little for the "Salesman" sequence, as shown in Figure 4.12(a). The difference for "MsUSA" sequence is relatively larger (Figure 4.11(a)).

[3] The MSRE performance both for "Salesman" and "MsUSA" sequences is nearly the same, irrespective of whether 2 or 4 transforms are chosen in the set, as shown in Figures 4.11(b) and 4.12(b).

[4] For the "Salesman" sequence, there is no clear winner on the basis of VWL bits (Figure 4.12(a)), when selection criteria that use correlations or VWL bits on a block basis are compared, and single transform selection configuration is used with choices possible among 4 transforms. For the "MsUSA" sequence, on the basis of VWL bits, the criteria that uses correlations does not perform as well as the one that uses VWL bits per block, as shown in Figure 4.11(a).

[5] The MSRE's, in coding both "Salesman" and "MsUSA" sequence, are higher when criterion based on VWL bits (best bit-rate) is used as compared to correlation based criterion, as shown in Figures 4.11(b) and 4.12(b).

[6] For both image sequences, VWL bits/MSRE performance when correlation direction based transform selection is made, i.e. different transforms are selected for horizontal and vertical directions within a block, is nearly similiar to that of a scheme which allows a choice of single transform for every block

(Figures are not included).

Various statistical measures used for comparing the performance of multi-transform schemes yield little additional information; as the differences are found marginal in many cases. Among choices in selection criteria, VWL bits per block criterion performs well on the basis of encoding-bits required per frame, whereas criterion based on correlations results in smaller MSRE's. Increasing the number of choice of transforms for the set used in transform selection did not always improve the VWL/MSRE performance. In some cases the savings in coding bits per frame is found to be roughly the same as the increase in overhead when the size of transform-set is increased. It implies that, for larger-size sets the law of diminishing returns may come into play. It also shows that in designing coding algorithms for MCFD signals, no significant benefit may result in distinguishing between more then 4 values of correlation-coefficients.

Some results allowing a visual comparison of coded images with single transform and multi-transform schemes for "MsUSA" and "Salesman" sequences are shown in Figures 4.13(b) and 4.14(b) respectively.

4.5 FILTER IN THE CODING LOOP

We investigate the effect of introducing a filter into the coding loop of the basic motion-compensated transform-coder. The filter is applied only to the pels originally classified as belonging to the moving-area of the reconstructed frame. The introduction of filter in the coding loop may cause progressive-effects in image-sequence coding because the reconstructed filtered current-frame is used for prediction and coding of the next frame, and so on. A simple low-pass filter is chosen,

and applied to the 3×3 window, centered over the pel to be filtered. The unnormalized coefficients for the filter are shown below.

$$\begin{matrix} 2 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 2 \end{matrix}$$

In other words, the value of each filtered-pel contains about one-quarter contribution from the original-pel, and three-quarters from the surrounding 8 pels of the unfiltered signal; the diagonal-pels weigh less than the adjacent pels.

In comparing the statistical results with and without the filter in coding loop, we find for "MsUSA" sequence, the statistical performance improves significantly with the filter. For example, from Figure 4.15(a) we find that on including the low-pass filter in the coding-loop, the number of moving blocks in the sequence decreases; according to Figure 4.15(b) we find a saving in VWL bits on the average, by 2500 bits per frame. Also, the MSRE's in uncompensable blocks, seem to be much lower when filter is included, than without it. With low-pass filter included in the coding loop, the statistical results for "Salesman" sequence appear to be conflicting to the results obtained for "MsUSA" sequence. For example, when low-pas filter is included in the coding loop, for the "Salesman" sequence, the number of moving blocks (Figure 4.16(a)) and the VWL coding-bits (Figure 4.16(b)) increase considerably as compared to when filter is excluded; Only the MSRE's appears to be lower (Figure 4.16(c)) when filter is included. This can perhaps be attributed to the fast movements, many sharp edges and non-uniform nature of background in "Salesman" sequence. The performance, when a *non-linear* filter is included in the coding loop, is also shown in Figures 4.16(a), 4.16(b) and 4.16(c). The results comparing the visual appearance of coded images with or without including low-pass filter in coding loop, are shown in Figures 4.17(a) and 4.17(b); the filtered images appear blurred as expected. On

examining the visual appearance of sequence "MsUSA" we find the blurriness is quite tolerable, and infact even improves the appearance because of smoothing of local noise in the coded sequence. Inclusion of low-pass filter causes "Salesman" sequence to appear blurry, i.e., the degradation is found to be distinctly disturbing.

4.6 DISCUSSION AND CONCLUSIONS

A brief review of schemes for low bit-rate image-sequence coding suggests some general directions that can be followed for for improving performance motion-compensated coding schemes. A motion-compensated transform coding-system is designed; it consists of a block motion-tracking algorithm for motion compensation, and a basic transform coding scheme in which only the uncompensable moving blocks are coded. Three orthogonal transforms are tested as possible candidates for this transform coding scheme. Their performance is found to be nearly equal in terms of mean square error and coding bit-rate. In a test, comparing the visual quality of reconstructed image-sequences, the DST coded sequence is found to be perceptionally preferable to the DCT or the KLT(0.5) coded sequences by a small margin.

Several configurations that use a multiple-transform approach for motion-compensated coding, are also investigated. This study is motivated by measurement of block-correlations and other statistics of the MCFD signal. A small set of orthogonal-transforms is chosen and a transform selected. Some overhead is required to convey the transform selection-information to the receiver. In comparing the performance and overhead considerations of various configurations, a two-transform selection scheme is found to be favorable. The visual improvements are small and obvious only on careful observation. For the schemes that allow choice of one out of four transforms, different transforms for horizontal and vertical directions, techniques

to minimize the transform-selection overhead can be investigated further. The overall improvements to the basic, single transform-coding scheme are found to be small, and so this study also shows a limit in performance of orthogonal-transforms for coding MCFD signal. The inclusion of low-pass filter in the coding-loop, seems somewhat desirable for "MsUSA" sequence, but proved undesirable for "Salesman" sequence.

Table 4.1 DCT Coefficients of Uncompensable MCFD Blocks
(a most active frame)

Mean Array (field 160: 113 uncompensable blocks)								
f	1	2	3	4	5	6	7	8
1	-0.7511	-0.1618	0.3906	-2.0517	-1.7976	-0.3853	-0.5342	-0.0579
2	1.7654	0.5437	0.1500	-0.6634	-0.5508	-0.0130	-0.5922	0.1330
3	0.8377	-0.3165	0.5437	1.0785	-0.4377	-0.1163	-0.6407	-0.0950
4	0.3016	-0.1416	0.6564	0.0300	0.2045	0.1166	-0.0939	-0.0706
5	-0.4878	-0.3617	0.3359	-0.0544	0.4126	0.0312	0.4727	-0.0593
6	-1.0794	0.2894	-0.4773	-0.0128	0.1082	0.2844	0.1465	-0.1144
7	-0.0848	-0.0273	0.0009	-0.2508	-0.4641	0.3219	-0.0680	-0.1888
8	-0.5593	0.5877	-0.3130	0.2247	0.3544	-0.1035	0.4980	0.1462

Normalised Variance Array (field 160: 113 uncompensable blocks)								
f	1	2	3	4	5	6	7	8
1	5.9200	6.0989	3.2691	3.7908	1.5566	0.9712	0.8980	0.4057
2	2.7244	2.2525	1.4560	1.1394	0.9577	0.5604	0.6440	0.3728
3	1.5557	0.8428	0.7091	1.0162	0.4590	0.2695	0.5351	0.3420
4	1.9865	1.8157	0.7884	0.5347	0.2216	0.4461	0.3146	0.3234
5	2.4000	0.8827	0.9435	0.6168	0.4140	0.2702	0.2093	0.1851
6	1.0735	0.7165	0.7334	0.4639	0.4027	0.3026	0.2434	0.2301
7	1.3933	1.0139	0.7316	0.3868	0.2239	0.2263	0.1927	0.3385
8	0.9652	0.6342	0.3739	0.5324	0.2470	0.2381	0.1823	0.2583

Table 4.2 DST Coefficients of Uncompensable MCFD Blocks
(a most active frame)

Mean Array (field 160: 113 uncompensable blocks)								
f	1	2	3	4	5	6	7	8
1	-0.8494	0.7455	0.3616	-2.0326	-1.5244	-0.7640	-0.5830	-0.1607
2	1.6188	0.7080	0.6544	-0.4189	-0.2534	-0.1302	-0.5242	0.1378
3	0.5682	-0.3345	1.0118	0.3803	-0.5854	-0.1829	-0.9058	-0.0816
4	0.7868	0.0024	1.0457	-0.1930	0.3772	-0.0016	-0.1786	-0.0125
5	-0.5014	-0.3916	0.3125	-0.1371	0.1396	-0.1655	0.2658	-0.0844
6	-0.5579	0.2772	-0.4661	-0.0944	-0.1326	0.3215	-0.0703	-0.0885
7	-0.1155	-0.0371	0.2429	-0.3486	-0.4901	0.2268	-0.1648	-0.2014
8	-0.4533	0.5259	-0.5733	0.3817	0.0968	0.0401	0.4552	0.1679

Table 4.2 (contd) ....

Normalized Variance Array (field 160: 113 uncompensable blocks)								
f	1	2	3	4	5	6	7	8
1	4.4928	4.3936	3.0112	4.1928	2.2340	1.5980	1.0318	0.5087
2	2.8227	2.4362	1.6465	1.0721	0.8930	0.6017	0.6348	0.3496
3	2.2211	1.3231	1.0620	0.8611	0.6411	0.4445	0.6389	0.3263
4	1.7080	1.4816	0.8700	0.5214	0.4329	0.6078	0.3451	0.3136
5	2.1532	0.7132	0.6874	0.6570	0.4334	0.3109	0.1813	0.1950
6	0.8858	0.7299	0.6532	0.4357	0.3803	0.3100	0.2455	0.2544
7	1.5051	1.1849	0.5960	0.3459	0.3186	0.2385	0.2030	0.3640
8	1.0099	0.8398	0.3953	0.3696	0.1878	0.2399	0.2036	0.2578

Table 4.3 KLT Coefficients (1st-order separable Markov model)
of Uncompensable MCFD Blocks (a most active frame)

Mean Array (field 160: 113 uncompensable blocks)								
f	1	2	3	4	5	6	7	8
1	-0.8447	0.3511	0.3077	-2.1571	-1.6581	-0.5515	-0.5123	-0.0868
2	1.7134	0.6521	0.5028	-0.5469	-0.4179	-0.0744	-0.5645	0.1415
3	0.6390	-0.3915	0.8319	0.6003	-0.6509	-0.1640	-0.7987	-0.0877
4	0.5551	-0.0659	0.8424	-0.1121	0.2369	0.0739	-0.1706	-0.0374
5	-0.5204	-0.3881	0.2785	-0.1131	0.2569	-0.0570	0.3752	-0.0707
6	-0.8181	0.2825	-0.5862	-0.0320	-0.0392	0.3102	0.0513	-0.1037
7	-0.0850	-0.0265	0.1001	-0.2940	-0.4890	0.2870	-0.1070	-0.1931
8	-0.5002	0.5546	-0.4814	0.3233	0.2401	-0.0413	0.4843	0.1542

Normalized Variance Array (field 160: 113 uncompensable blocks)								
f	1	2	3	4	5	6	7	8
1	5.0874	5.2036	3.3633	4.2659	1.9072	1.1965	0.9178	0.4404
2	2.8234	2.4387	1.6040	1.0794	0.9260	0.5654	0.6396	0.3586
3	1.8459	1.1322	0.9279	0.8280	0.5095	0.3449	0.5992	0.3323
4	1.8051	1.6343	0.8307	0.5250	0.2732	0.5160	0.3243	0.3209
5	2.3847	0.7618	0.7167	0.6643	0.4047	0.2747	0.1815	0.1846
6	0.9844	0.7159	0.6825	0.4454	0.4047	0.3030	0.2387	0.2398
7	1.4752	1.1067	0.6335	0.3440	0.2628	0.2277	0.1946	0.3502
8	1.0079	0.7516	0.3655	0.4318	0.2144	0.2367	0.1903	0.2584

Table 4.4 A Typical Small-magnitude Uncompensable MCFD Block

Luminance Value								
Pel	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	-3	1	1	1	-3	0	2	4
2	-3	-4	-2	-3	-4	-2	0	0
3	1	-2	1	1	-6	-2	-2	1
4	-1	0	2	-1	-5	-2	-1	0
5	1	1	6	-7	-3	0	0	2
6	-2	3	5	-7	-2	0	1	1
7	0	1	4	-2	-1	-4	0	4
8	-1	3	1	1	0	0	1	0

DCT Coefficients								
f	1	2	3	4	5	6	7	8
1	-3.1250	-0.1823	7.0788	-8.5508	-6.1250	-0.3165	3.6975	2.0157
2	-4.2498	-3.0436	-0.1373	-0.0868	2.0444	3.8128	0.5230	-3.8661
3	2.1396	-3.1629	-2.8258	-0.0551	2.6479	0.1538	-4.4383	-2.4323
4	2.3951	-1.9451	2.2627	-0.8871	-4.3577	-3.9442	1.1800	3.9650
5	3.1250	0.2986	-1.0079	0.2609	-1.3750	-0.3905	-0.8002	-1.5009
6	1.8539	0.9063	0.0613	-3.5423	2.2469	0.8746	-0.8841	-1.5058
7	2.2256	0.9820	-0.6883	0.3125	-1.7733	-0.2119	-0.1742	-2.0788
8	2.2534	1.0356	1.7305	-1.2521	1.8961	0.5419	2.1308	0.5562

DST Coefficients								
f	1	2	3	4	5	6	7	8
1	-5.8117	3.3884	7.4277	-7.2639	-5.6644	-2.0164	4.4692	2.3973
2	-4.5949	-2.6049	-3.1796	-1.1633	2.0487	4.8761	0.1845	-4.5020
3	0.0433	-2.0372	-0.3333	-3.3457	1.4990	-0.6667	-2.5443	-1.3085
4	-0.0338	-2.6580	2.9686	-0.3646	-3.4767	-3.2964	0.9939	2.4559
5	2.7907	-0.2548	0.7163	-1.0062	-0.6922	-0.4896	-1.3292	-1.4829
6	1.1908	1.1055	0.3333	-3.3084	1.7678	0.0000	-0.4230	-1.4139
7	2.8397	0.7756	0.7886	0.2046	-1.6548	0.0266	-0.6628	-2.3520
8	1.4589	1.2926	1.6482	-1.3439	2.2445	0.3275	2.6815	0.4695

Table 4.4 (contd) ....

KLT Coefficients								
f	1	2	3	4	5	6	7	8
1	-4.9642	1.9767	7.3121	-8.1112	-5.9884	-1.1017	4.4008	2.4076
2	-4.6065	-2.8313	-1.8943	-0.5785	2.2377	4.4016	0.3307	-4.3604
3	1.1284	-2.7690	-1.0101	-2.3494	1.6915	-0.1537	-3.3531	-1.7501
4	1.0363	-2.3025	2.8900	-0.7061	-3.8269	-3.5687	1.1671	3.1723
5	3.1681	-0.0687	0.0414	-0.3156	-1.1501	-0.3636	-1.1062	-1.5331
6	1.5624	1.1371	0.2972	-3.4405	2.0872	0.5083	-0.6543	-1.4451
7	2.5766	0.8689	0.1008	0.3345	-1.7718	-0.0968	-0.3751	-2.1926
8	1.7487	1.1811	1.8464	-1.2399	2.1312	0.4733	2.3604	0.5292

Table 4.5 A Typical Large-magnitude Uncompensable MCFD Block

Luminance Value								
Pel	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	11	11	6	4	1	-5	-7	-7
2	-12	12	9	3	-6	-12	10	24
3	-61	5	-3	-6	1	-7	-1	15
4	-20	4	-1	3	9	-12	2	13
5	29	20	17	18	-3	-17	1	8
6	-8	16	23	13	-18	-9	3	2
7	-21	1	13	-14	-17	0	5	1
8	-8	5	-2	-13	-8	3	5	-1

DCT Coefficients								
f	1	2	3	4	5	6	7	8
1	3.3750	-3.3354	5.6942	-49.0354	-20.3750	-12.5748	-14.6708	-1.2110
2	4.1953	-5.5480	-6.7334	-6.5708	15.8677	-11.2556	-9.6439	-0.0189
3	-10.3152	-0.5099	5.2060	19.0132	-12.4568	1.0757	9.2600	0.3883
4	23.8442	45.2894	9.8328	2.3448	6.5845	23.5822	9.2001	-1.7690
5	13.1250	25.3678	5.9144	25.8131	21.8750	6.1940	-0.4203	3.9661
6	-20.9101	-14.5573	-15.5632	4.4076	-8.8105	-9.0519	2.9232	2.8921
7	-10.2043	-1.9833	-14.7400	-1.8678	-6.8818	-6.5317	-3.4560	-7.2092
8	-3.3629	4.3065	-0.4023	3.1529	-1.1935	-2.7445	2.4051	0.2551

Table 4.5 (contd) ....

DST Coefficients								
f	1	2	3	4	5	6	7	8
1	5.7150	14.4446	10.8447	-48.2411	-11.0699	-22.9469	-18.0704	-5.2970
2	-0.0555	-14.2988	-11.4125	-8.4990	12.3913	-21.1771	-11.2175	-2.2554
3	-11.4016	-6.4397	8.1667	-5.5107	-22.8289	-5.8333	1.0728	-1.5709
4	22.4342	39.6480	13.8096	7.5705	18.0264	26.0051	9.2573	1.1351
5	7.1934	14.4573	8.0616	26.4912	19.1519	12.4637	4.2966	7.4316
6	-9.1715	-6.9487	-15.8333	2.4206	-10.6233	-5.8333	0.0482	1.8694
7	-3.9400	1.6391	-12.2831	2.6796	-9.5971	-5.0894	-6.0336	-6.9505
8	-2.9756	3.9695	-2.2779	5.2633	-2.1461	-1.3178	2.1589	0.5616

KLT Coefficients								
f	1	2	3	4	5	6	7	8
1	4.9861	6.1882	8.3362	-51.3482	-15.9592	-17.4072	-16.9253	-2.8207
2	1.8550	-10.9401	-8.9736	-7.8654	14.4152	-16.2816	-10.7133	-0.6396
3	-11.0324	-5.3244	5.9195	4.2474	-19.1605	-1.9778	5.0187	-0.2148
4	22.9414	42.7859	13.3229	6.4958	12.6228	24.7367	8.5369	-0.6039
5	9.8118	19.6193	7.4979	27.8740	20.7077	9.2785	1.6399	5.3927
6	-14.4720	-11.2642	-17.3919	2.7038	-10.3953	-7.9178	1.5919	2.3758
7	-6.2147	-0.0888	-14.5425	-0.1812	-8.6797	-6.2597	-4.6134	-7.1961
8	-3.2149	4.0818	-1.4220	4.2467	-1.5672	-2.1776	2.3319	0.3618

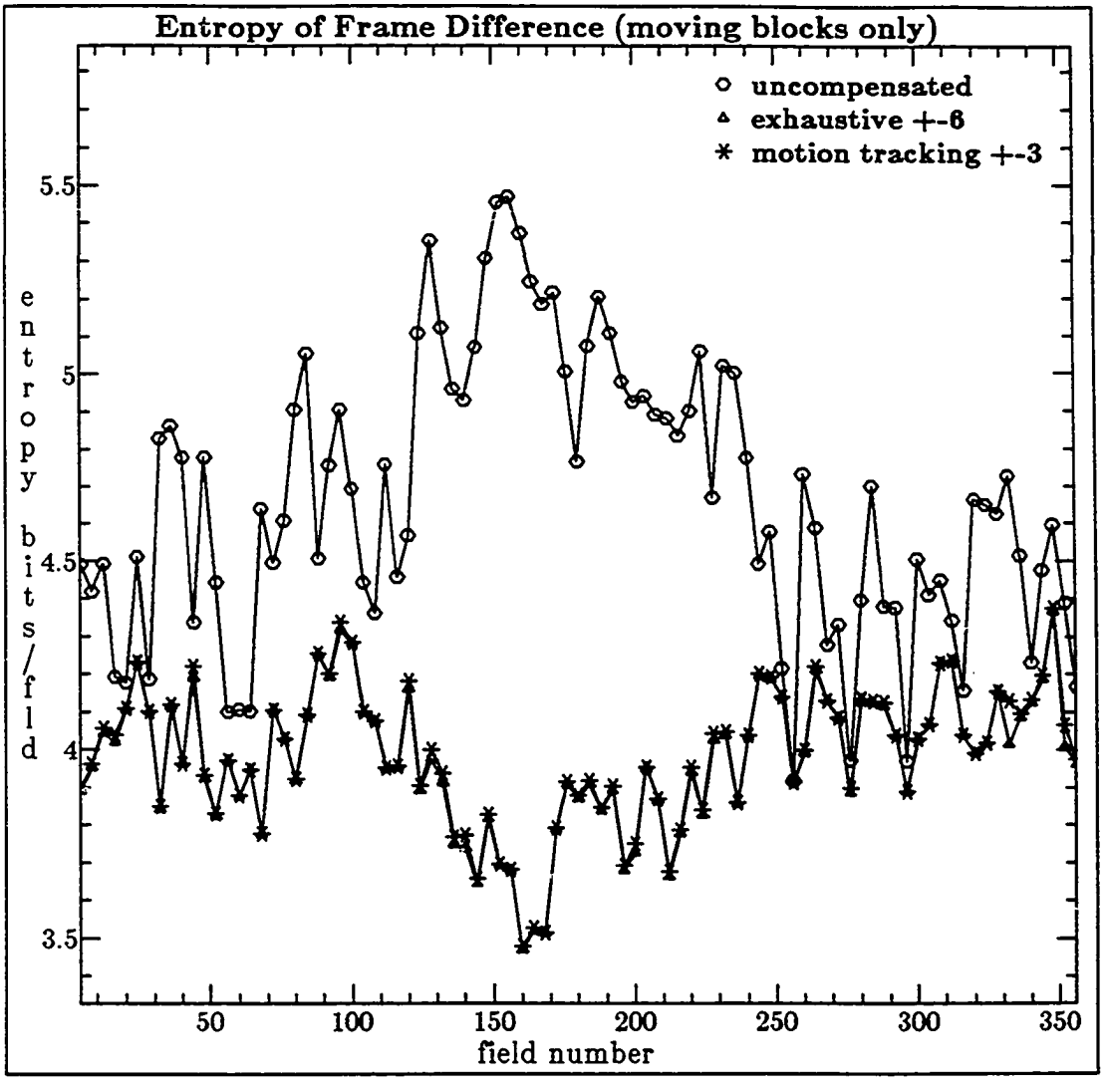


Figure 4.1 Performance comparison based on Entropy: "MsUSA" sequence

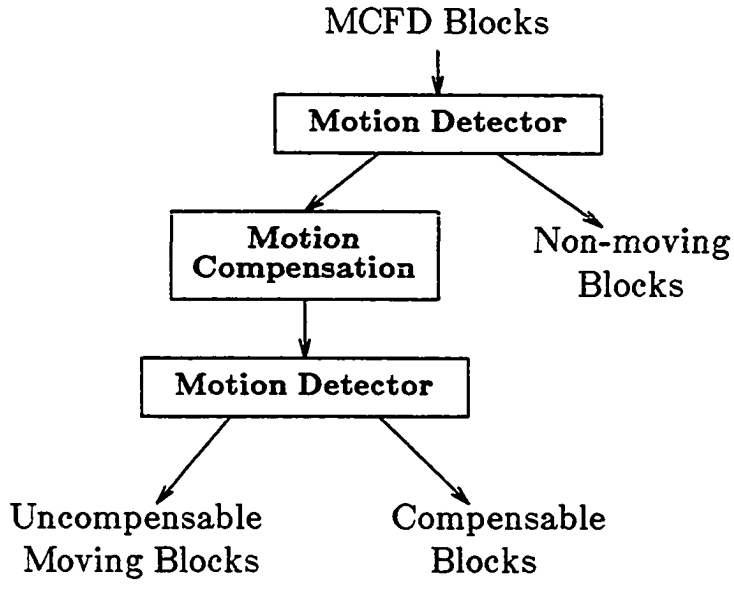


Figure 4.2 FD Block Types

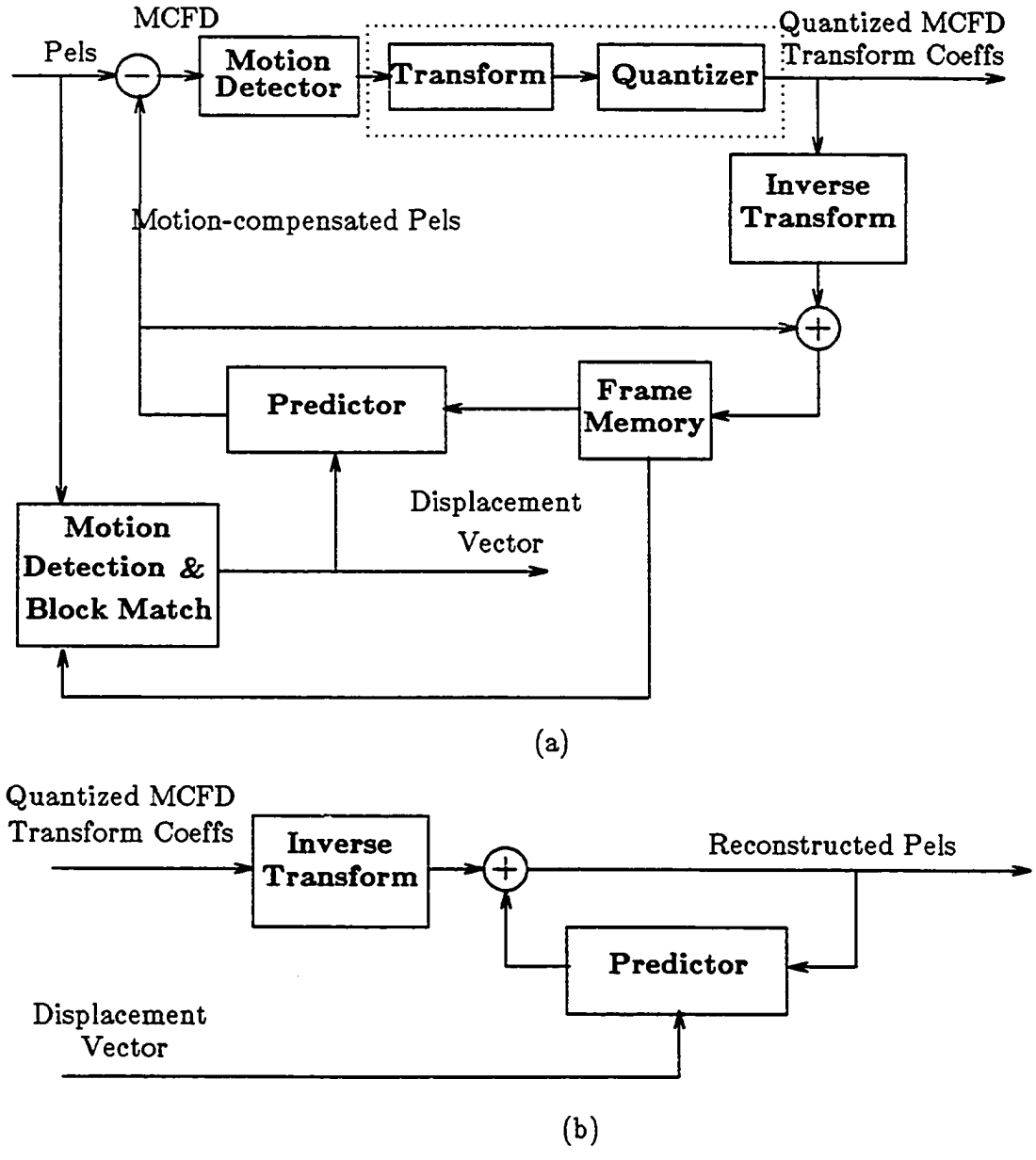


Figure 4.3 A Block-Matching Motion-Compensated Coding System

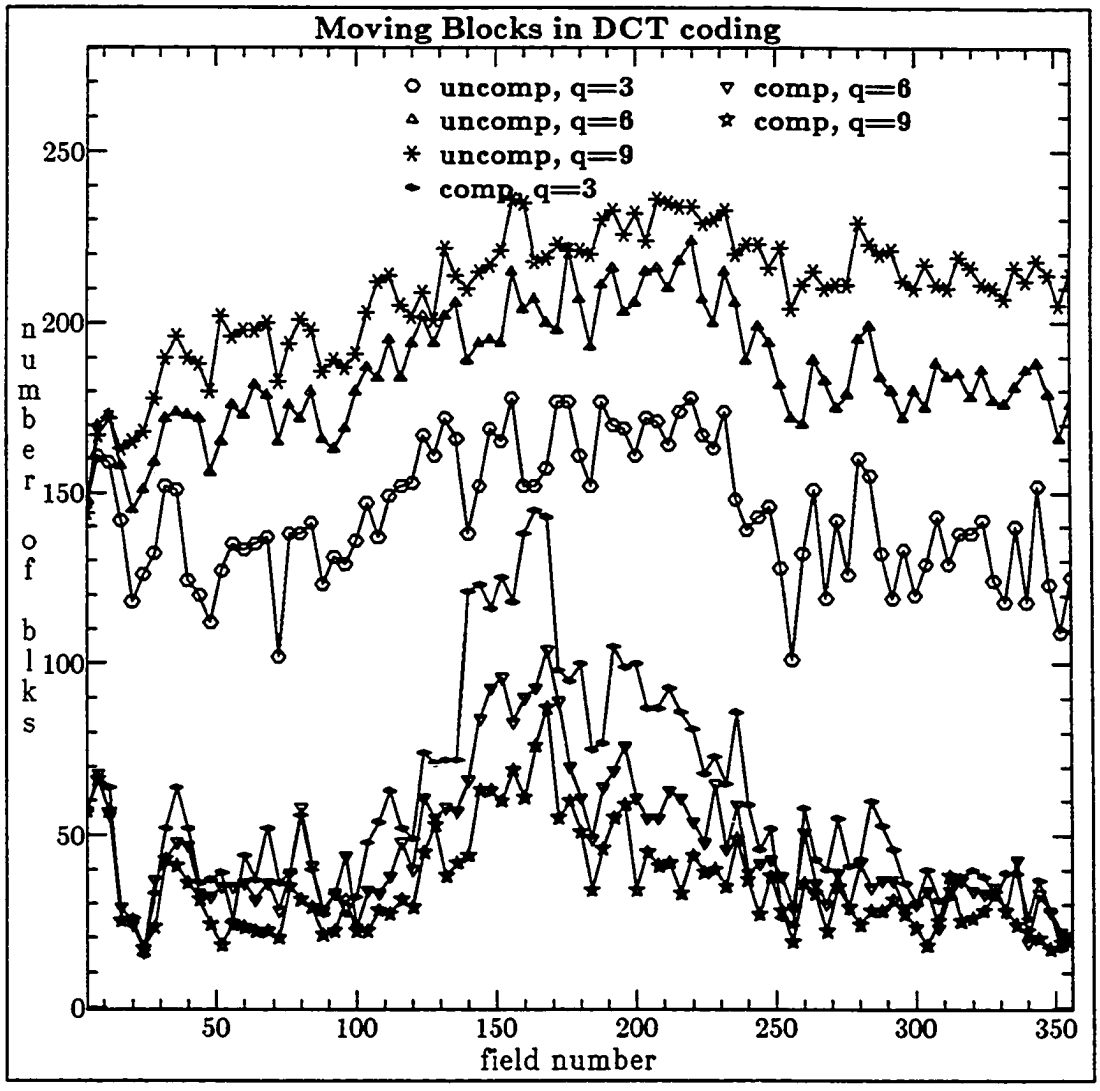


Figure 4.4(a) Motion Activity for Uncompensable and Compensable Blocks in DCT coding: "MsUSA" sequence

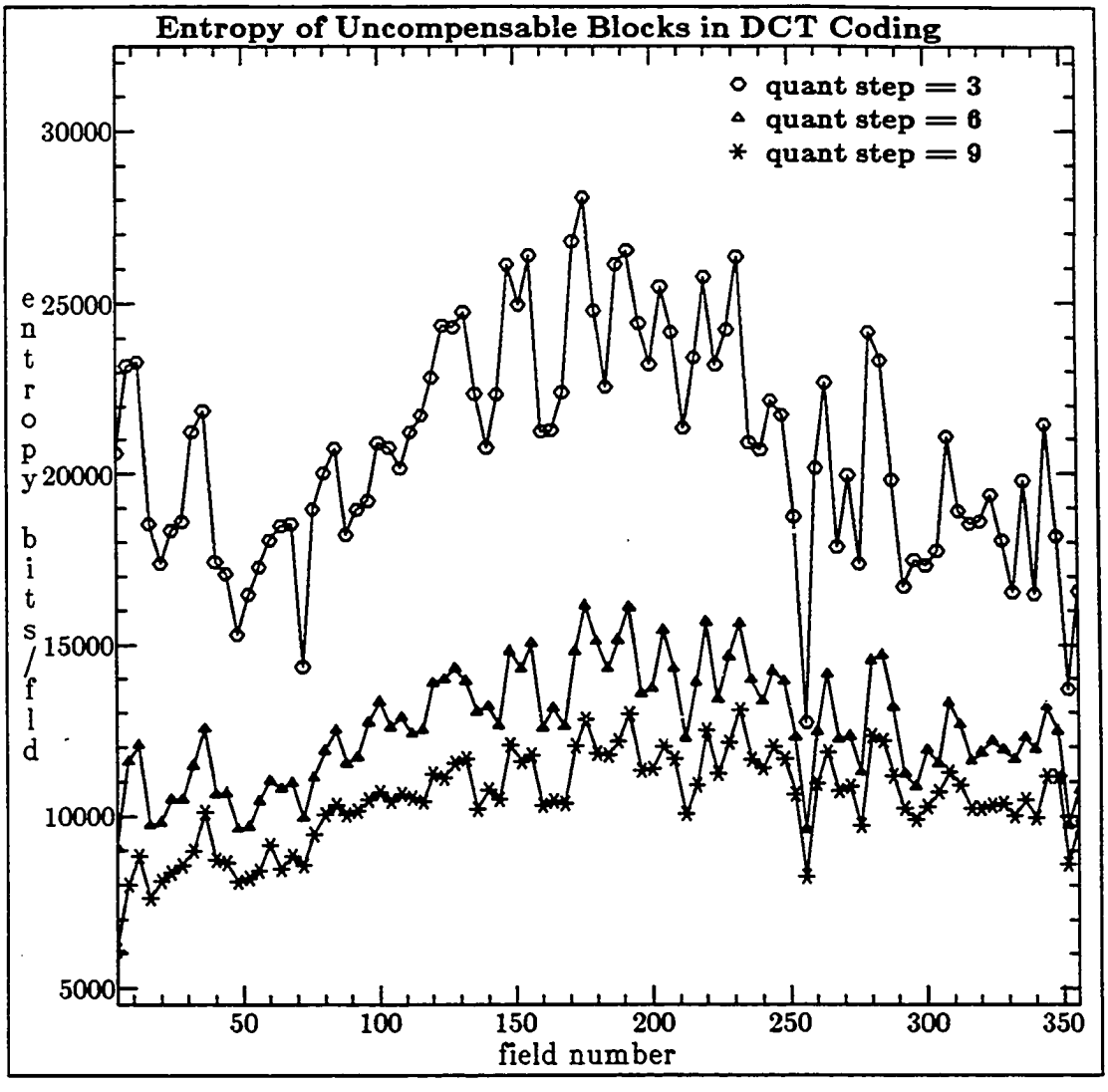


Figure 4.4(b) Transform coefficient step-size effect on Bit-rate in DCT coding: "MsUSA" sequence

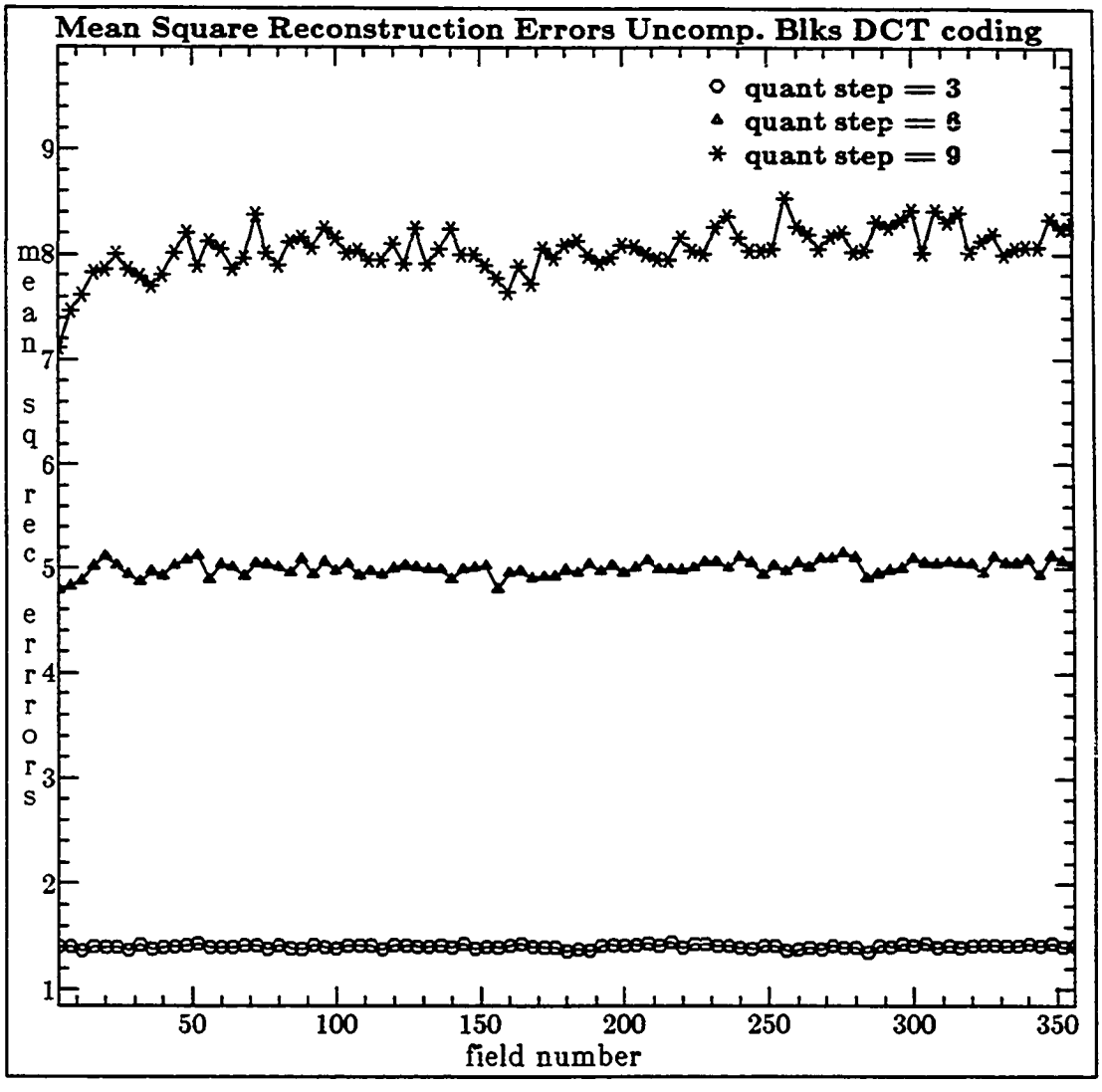


Figure 4.4(c) Transform coefficient step-size effect on Mean Square Reconstruction Errors in DCT coding: "MsUSA" sequence

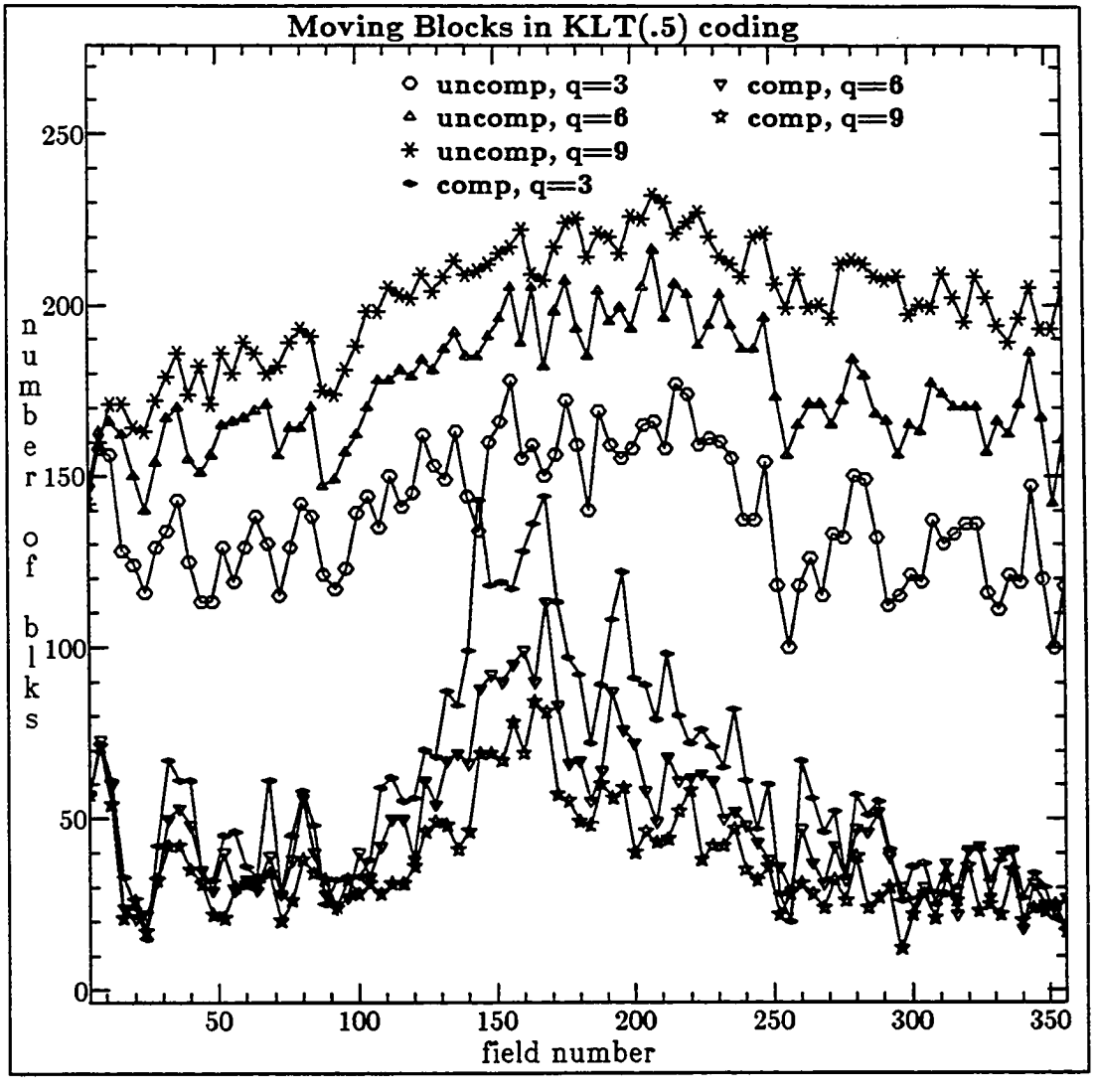


Figure 4.5(a) Motion Activity for Uncompensable and Compensable Blocks in KLT(0.5) coding: "MsUSA" sequence

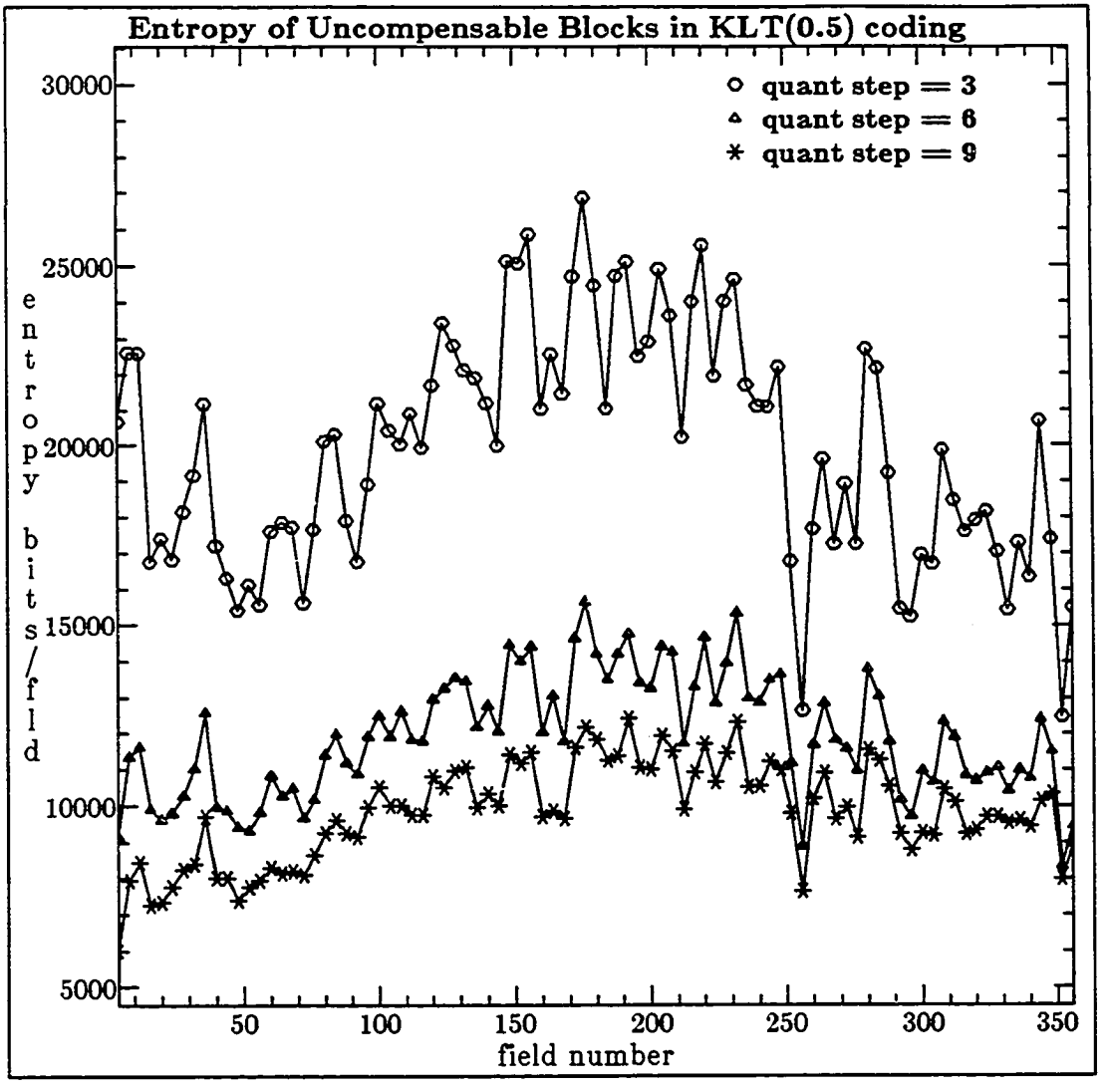


Figure 4.5(b) Transform coefficient step-size effect on Bit-rate in KLT(0.5) coding: "MsUSA" sequence

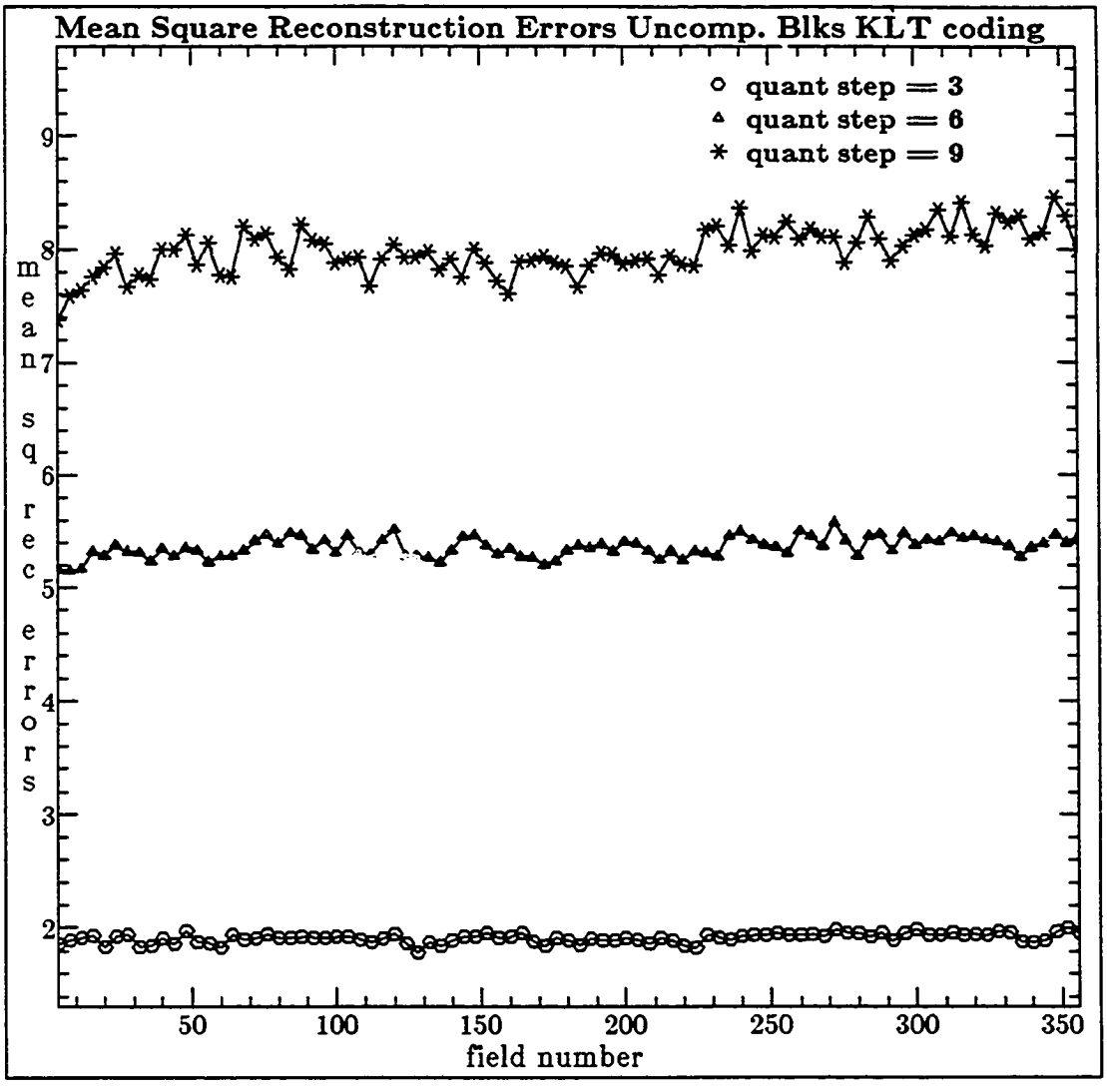


Figure 4.5(c) Transform coefficient step-size effect on Mean Square Reconstruction Errors in KLT(0.5) coding: "MsUSA" sequence

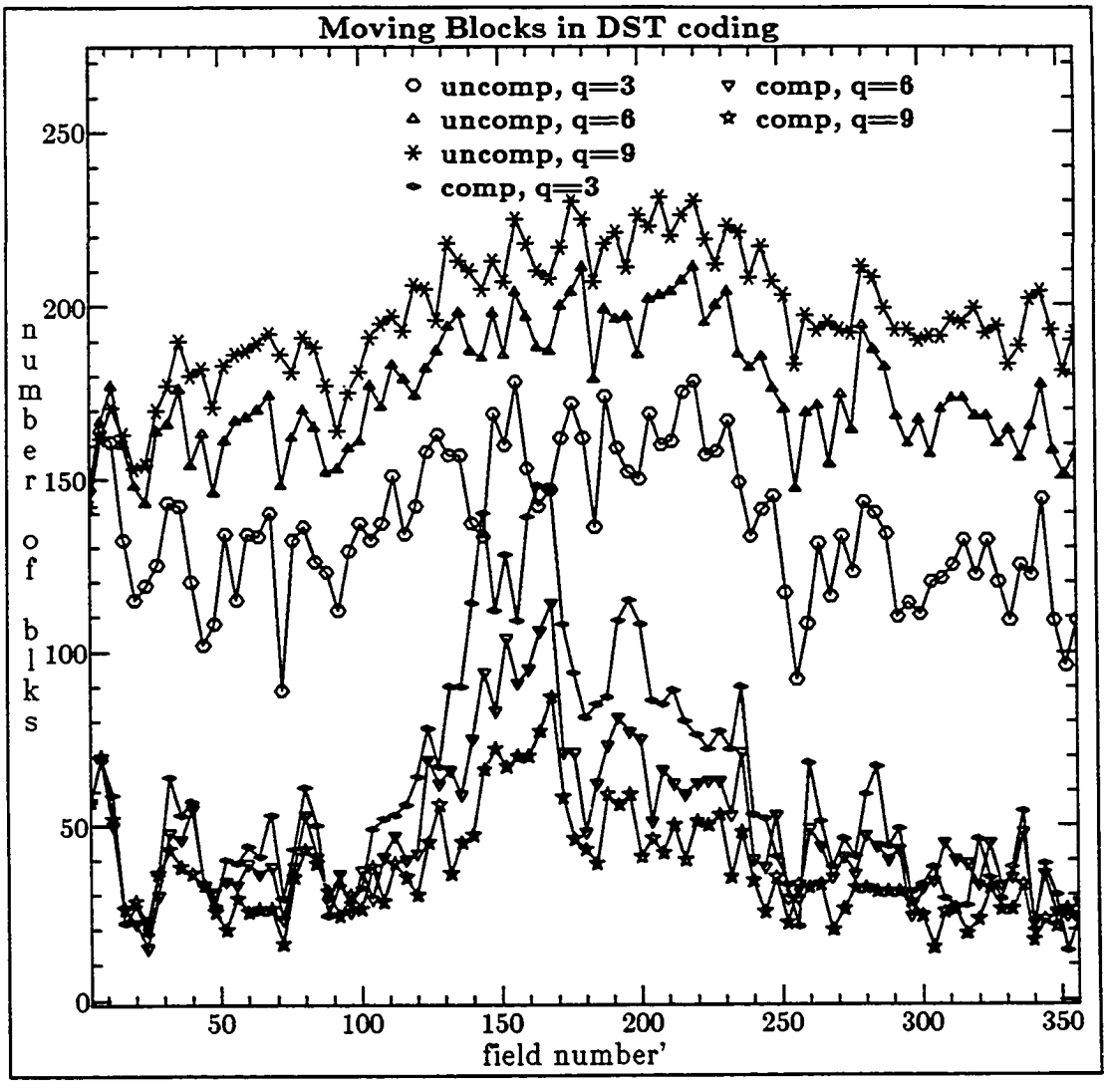


Figure 4.6(a) Motion Activity for Uncompensable and Compensable Blocks in DST coding: "MsUSA" sequence

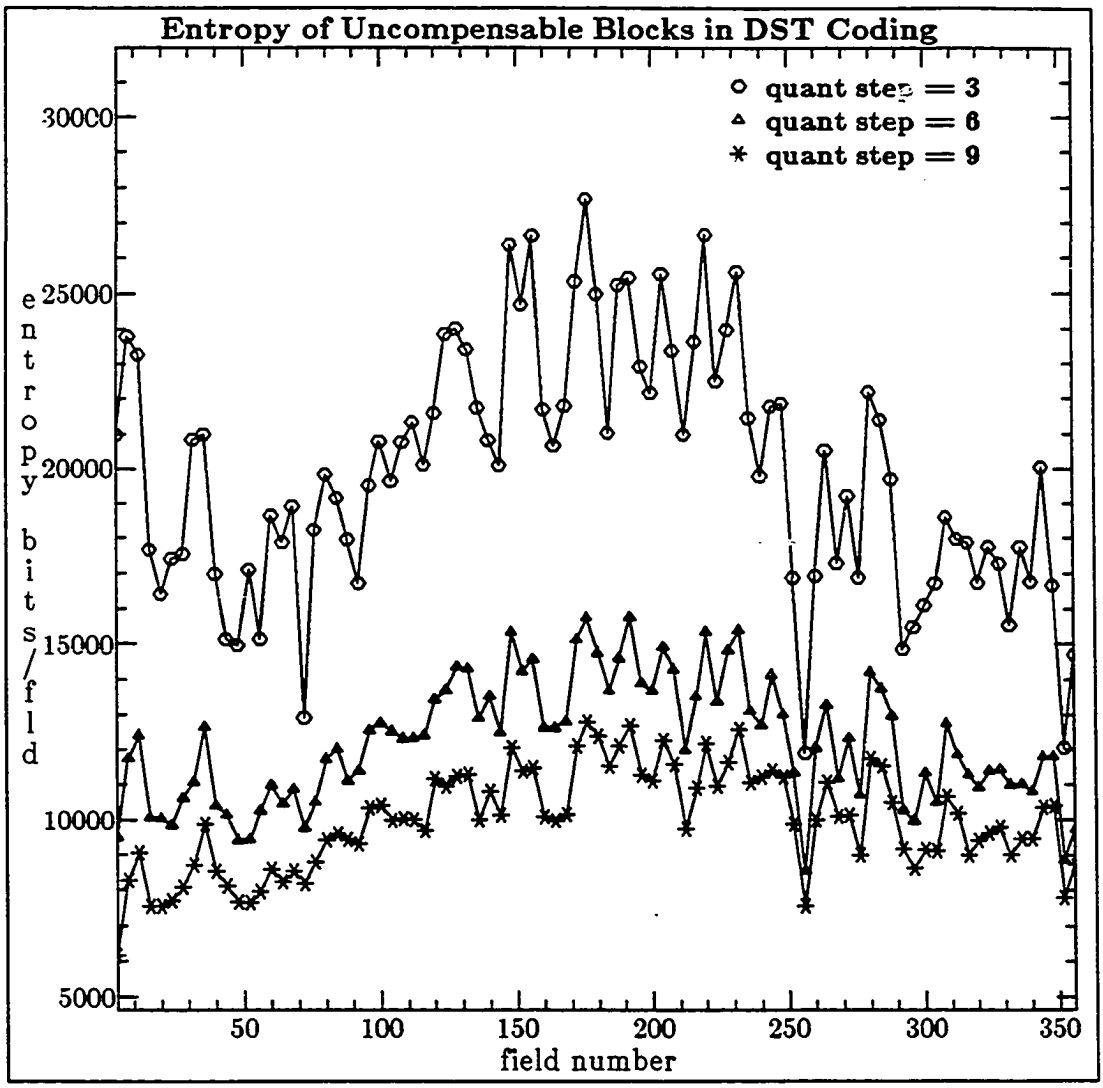


Figure 4.6(b) Transform coefficient step-size effect on Bit-rate in DST coding: "MsUSA" sequence

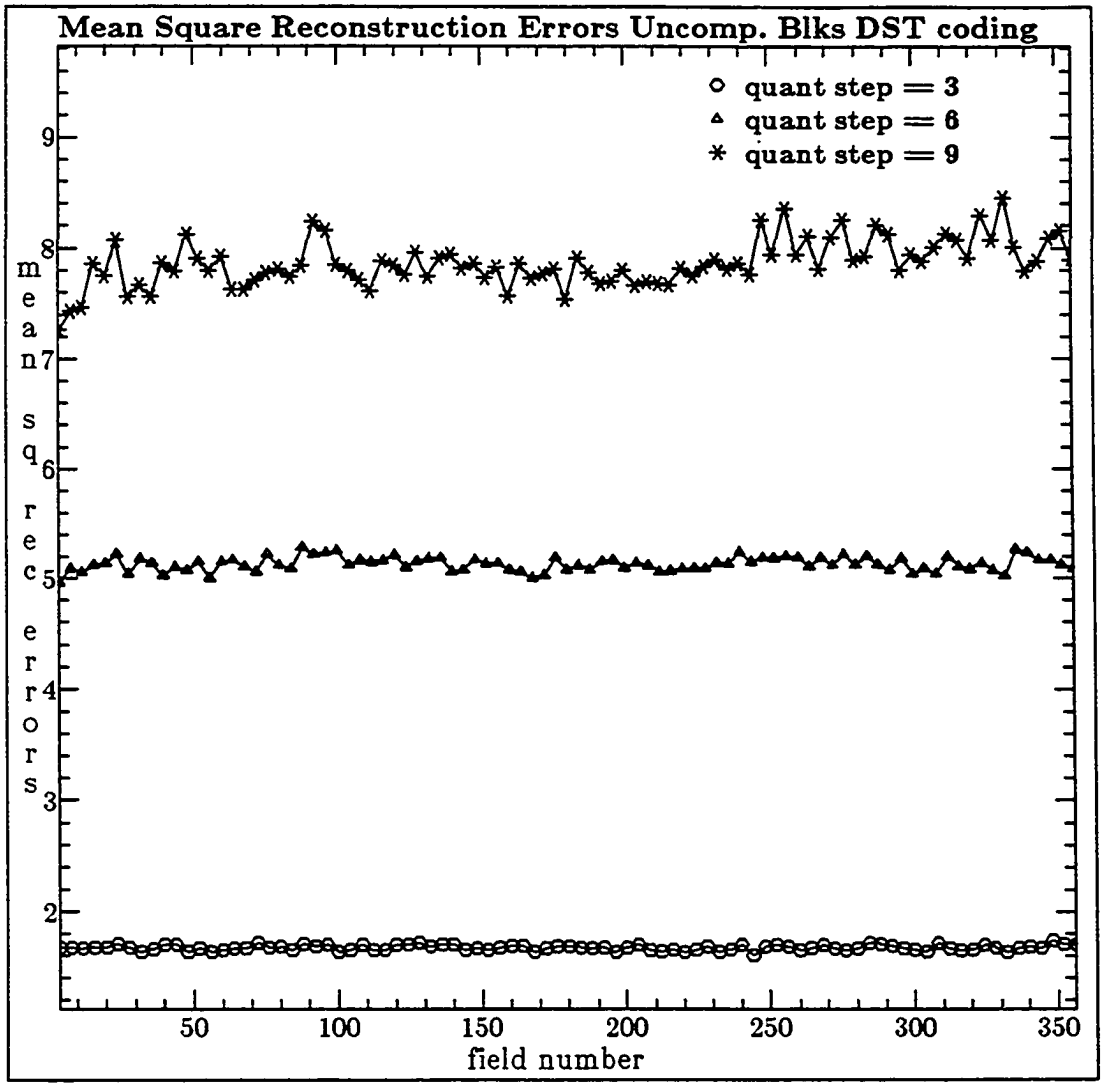


Figure 4.6(c) Transform coefficient step-size effect on Mean Square Reconstruction Errors in DST coding: "MsUSA" sequence

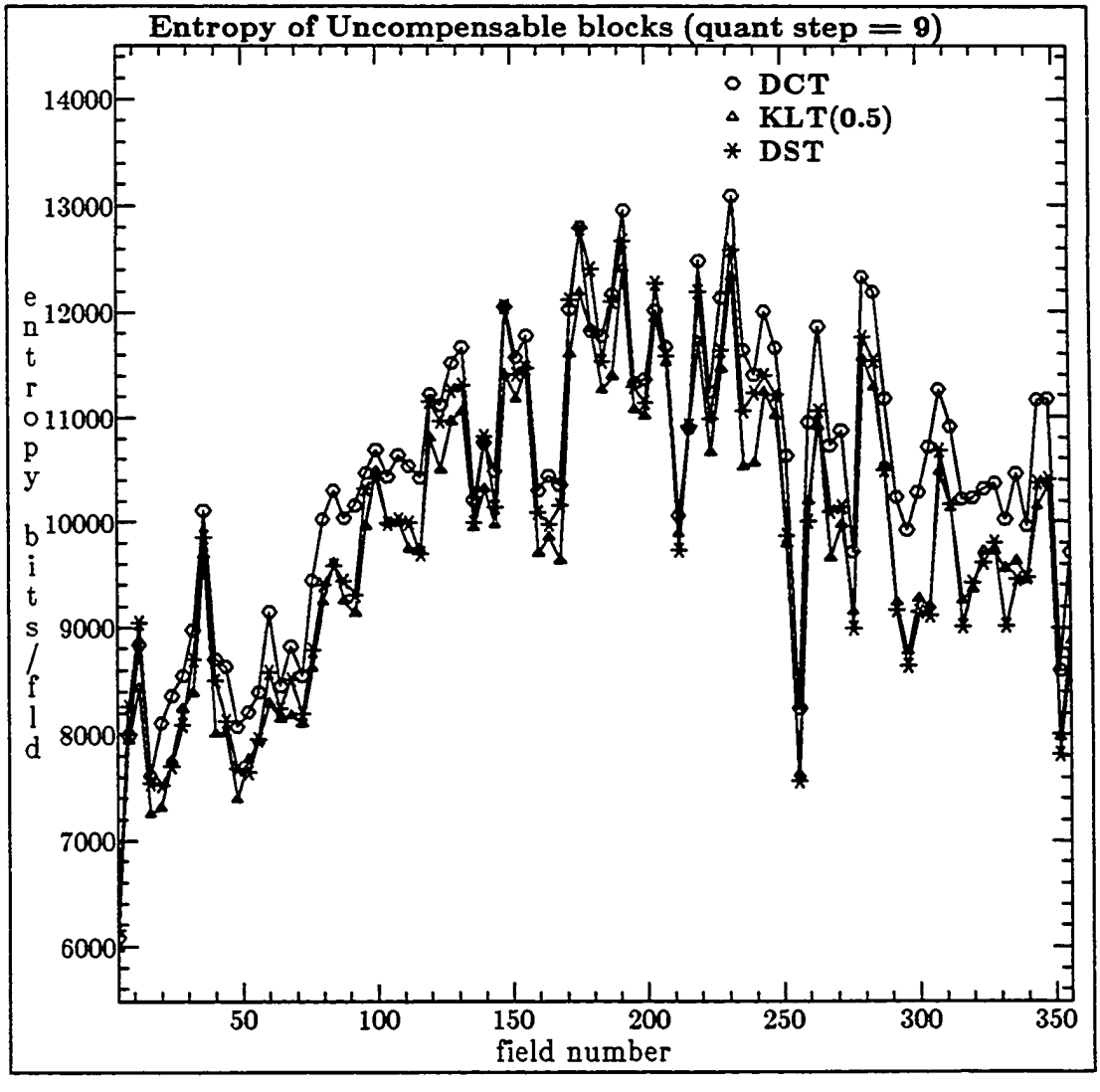


Figure 4.7(a) Bit-rate performance of various transform coding schemes, quant step = 9 : "MsUSA" sequence

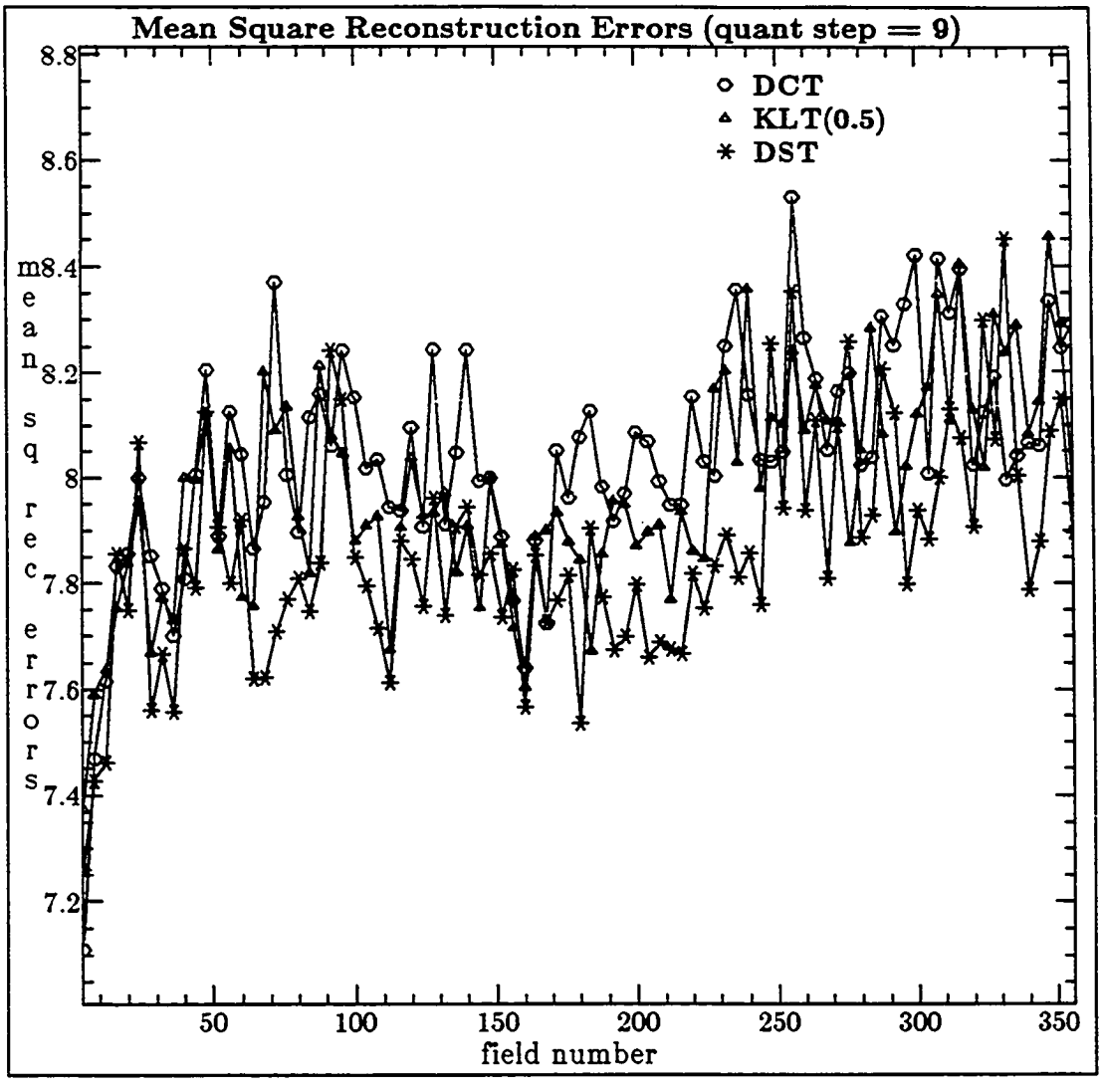


Figure 4.7(b) Mean Square Reconstruction Errors for various transform coding schemes, quant step = 9 : "MsUSA" sequence

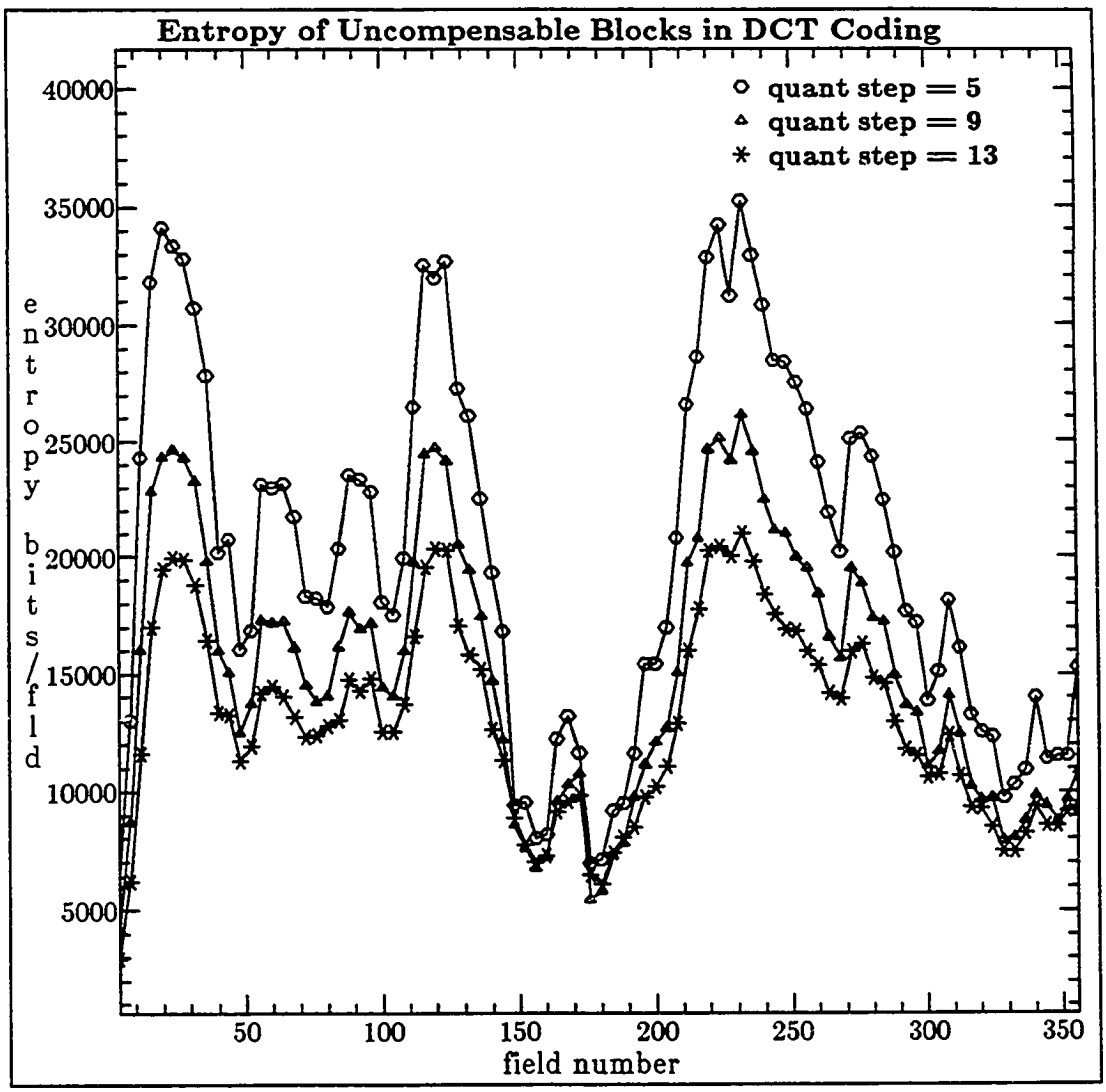


Figure 4.8(a) Transform coefficient step-size effect on Bit-rate in DCT coding: "Salesman" sequence

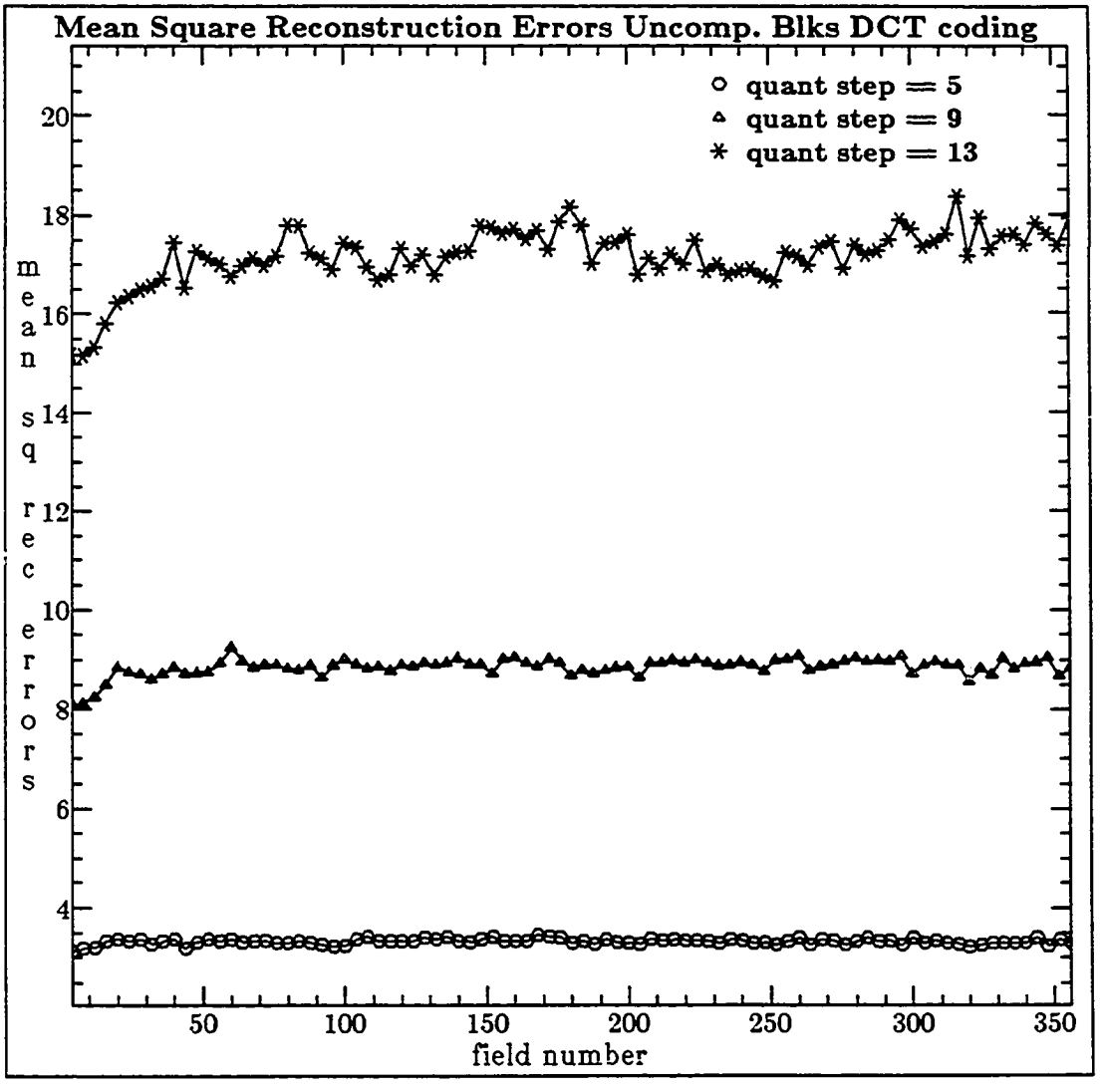


Figure 4.8(b) Transform coefficient step-size effect on Mean Square Reconstruction Errors in DCT coding: "Salesman" sequence

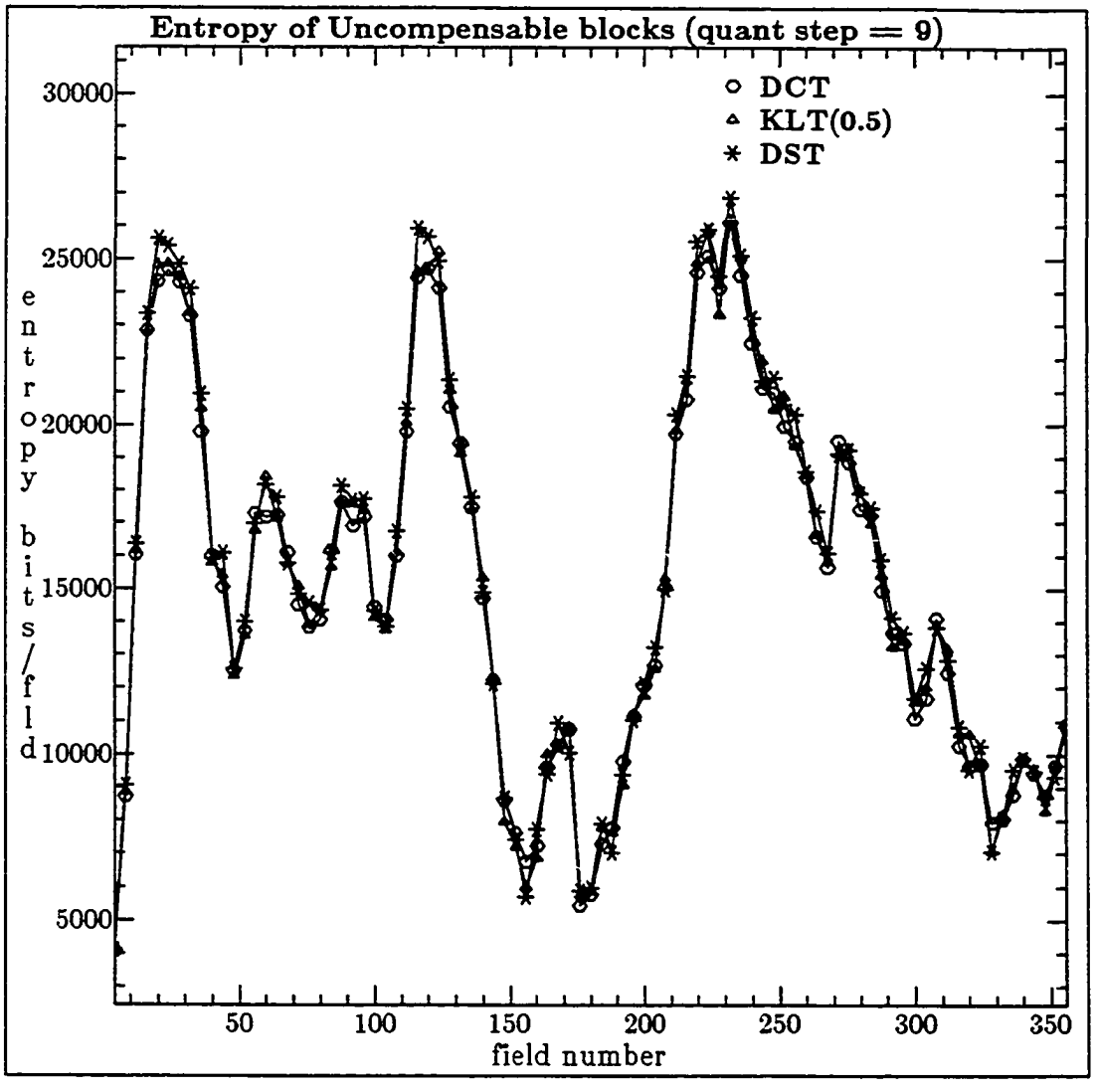


Figure 4.9(a) Bit-rate performance of various transform coding schemes, quant step = 9 : "Salesman" sequence

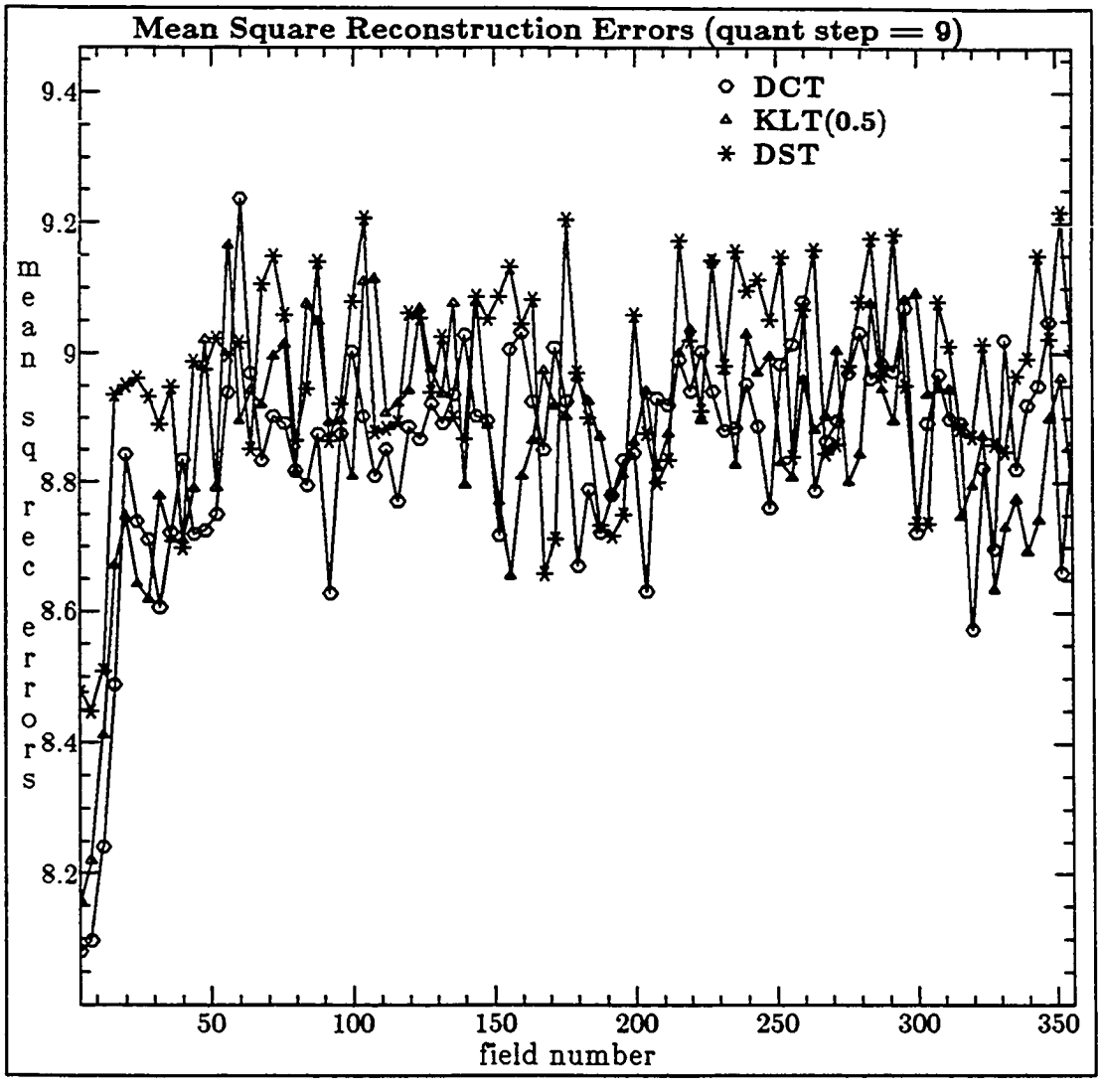


Figure 4.9(b) Mean Square Reconstruction Errors for various transform coding schemes, quant step = 9 : "Salesman" sequence

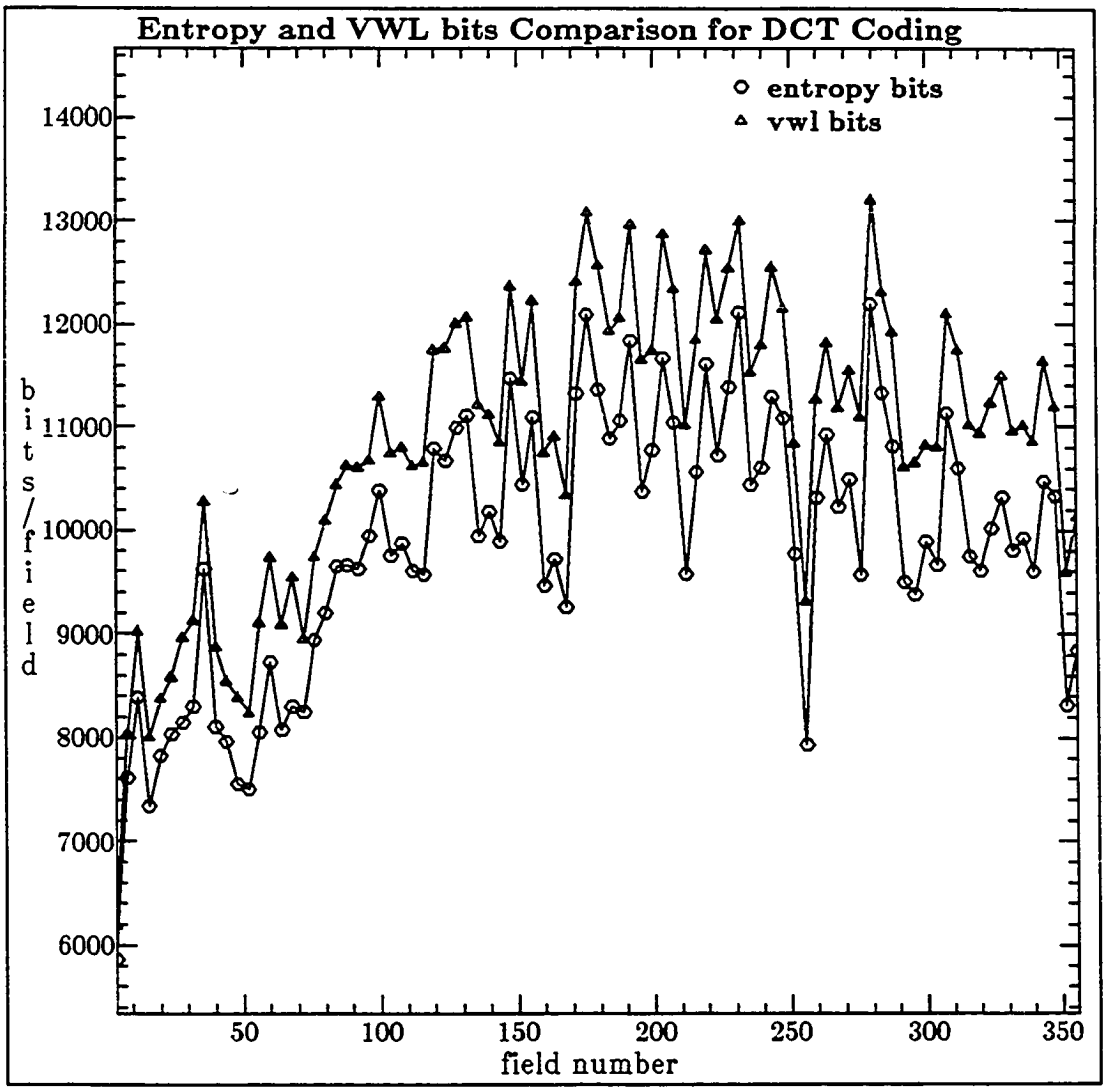


Figure 4.10 Entropy and VWL bits comparison in DCT Coding (quant. step 9) : "MsUSA" sequence.

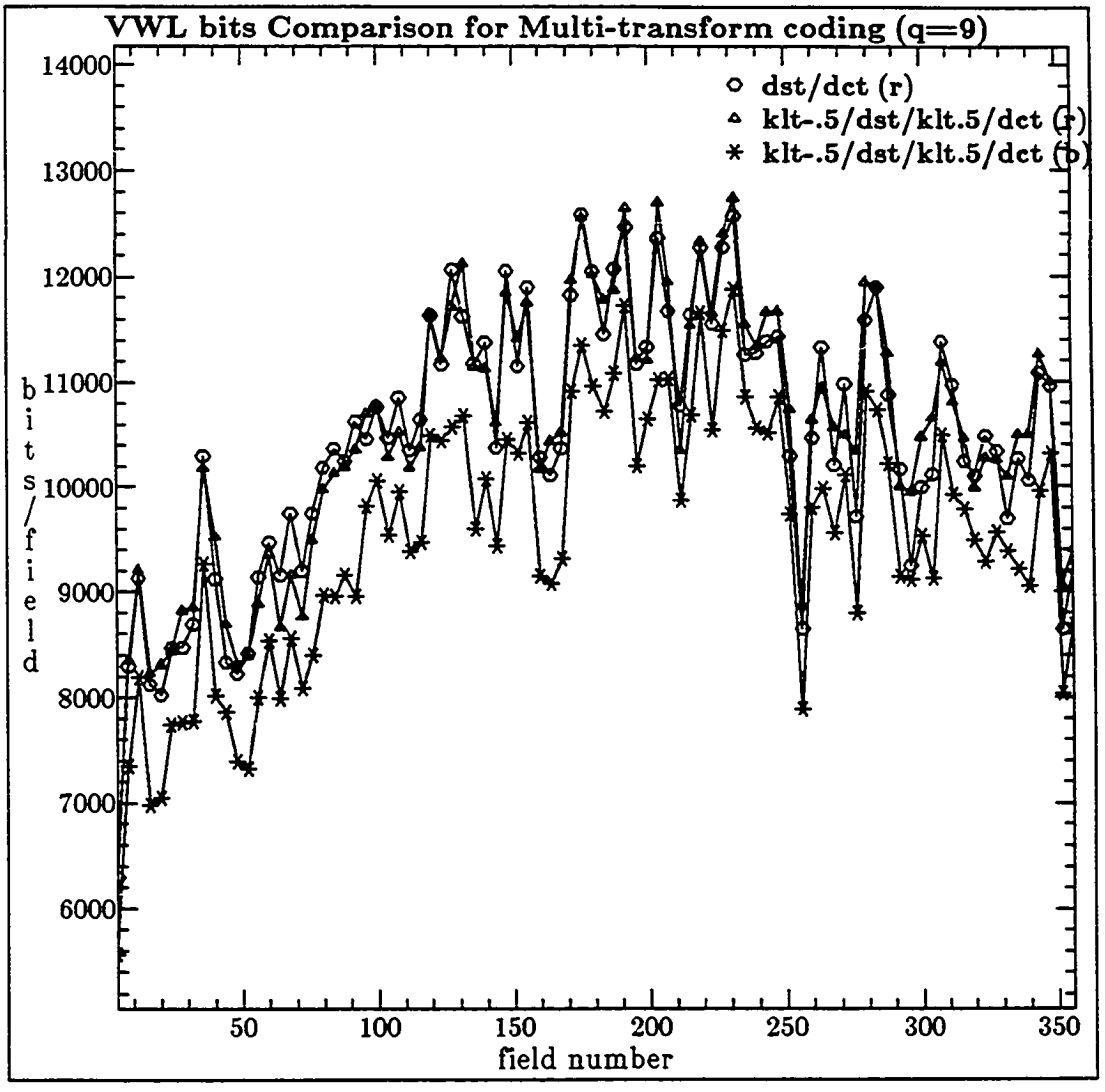


Figure 4.11(a) VWL bits comparison in multi-transform coding (choice of 2 or 4 transforms) on the basis of correlation (r) or best bit-rate (b), quant step = 9: "MsUSA" sequence

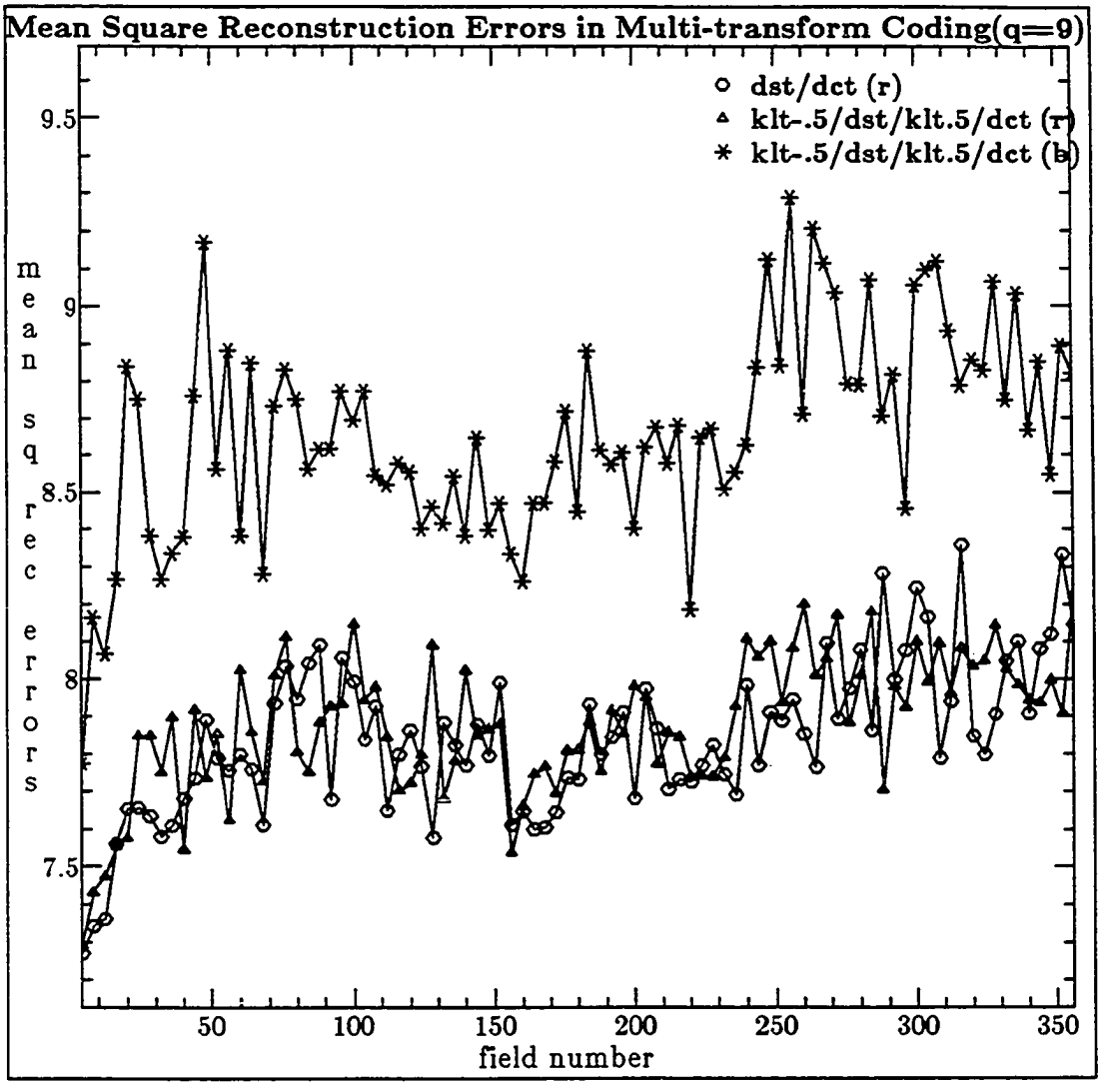


Figure 4.11(b) MSRE's comparison in multi-transform coding (choice of 2 or 4 transforms) on the basis of correlation (r) or best bit-rate (b), quant step = 9: "MsUSA" sequence

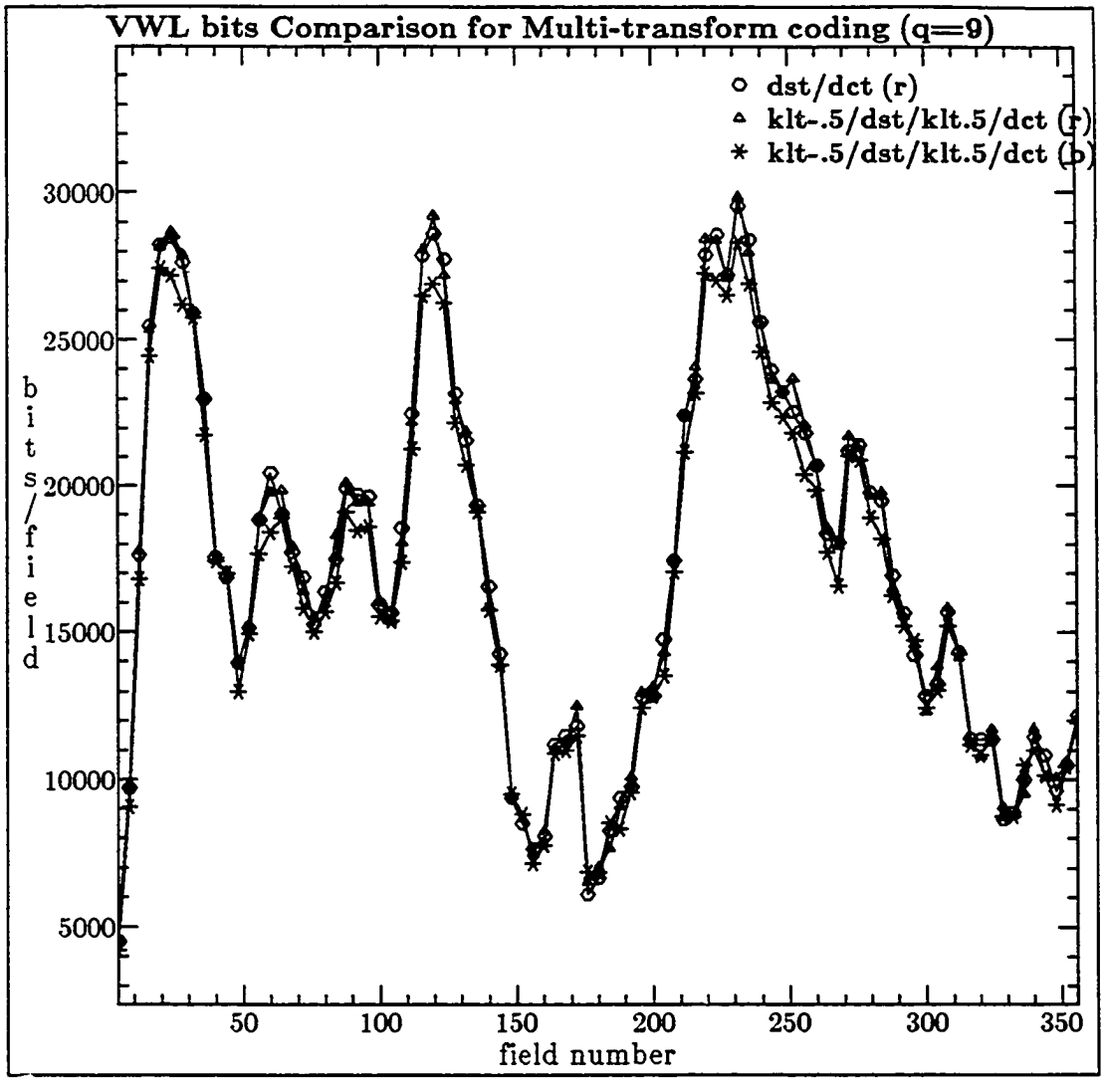


Figure 4.12(a) VWL bits comparison in multi-transform coding (choice of 2 or 4 transforms) on the basis of correlation (r) or best bit-rate (b), quant step = 9: "Salesman" sequence

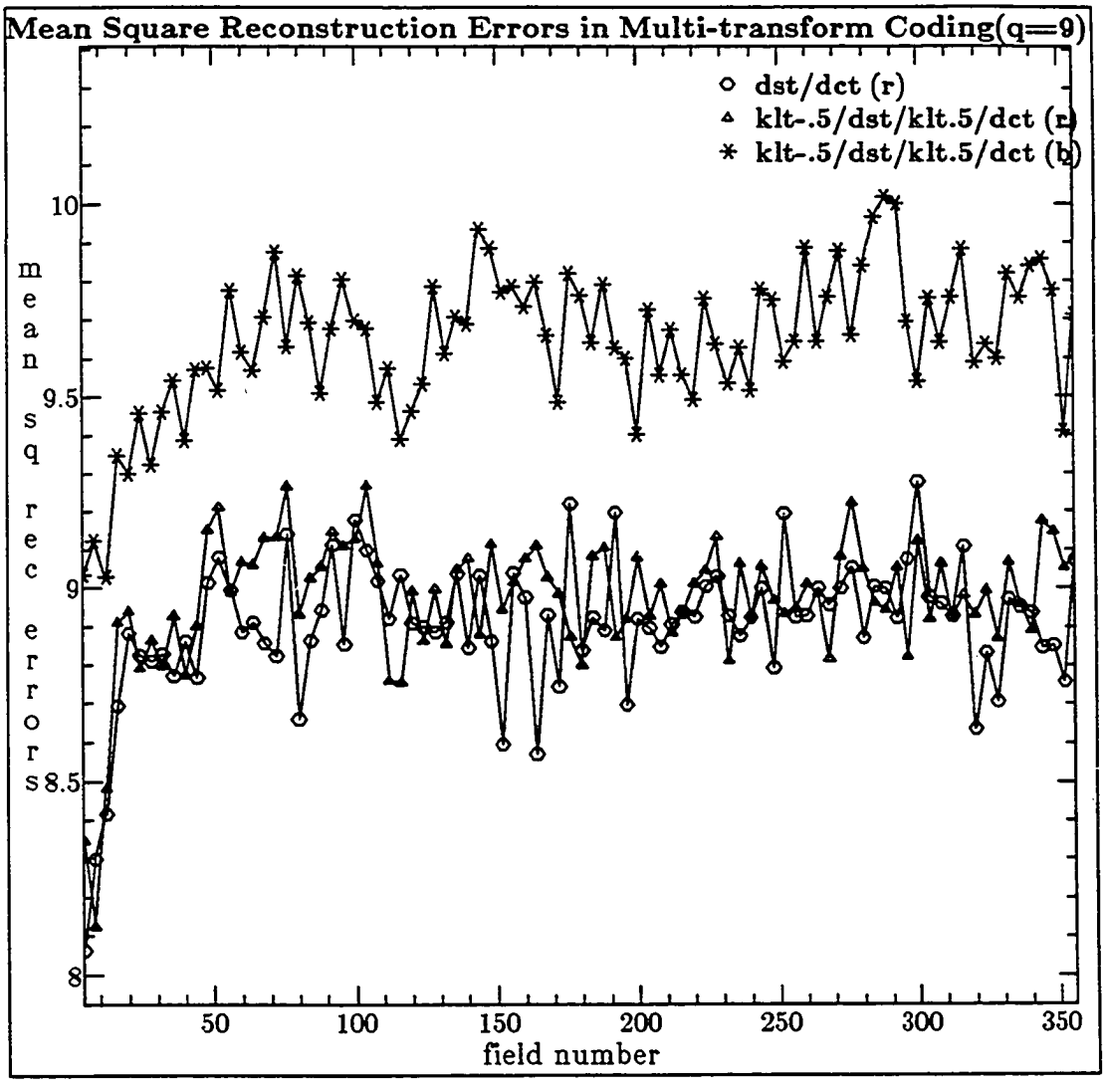


Figure 4.12(b) MSRE's comparison in multi-transform coding (choice of 2 or 4 transforms) on the basis of correlation (r) or best bit-rate (b), quant step = 9: "Salesman" sequence

Figure 4.13(a) Low resolution original, and DCT coded blk-size (8*8) q=9, images of "MsUSA" sequence

Figure 4.13(b) DCT/DST multi-transform coded blk-size (8*8) q=9, and DST coded blk-size (8*8) q=9, images of "MsUSA" sequence

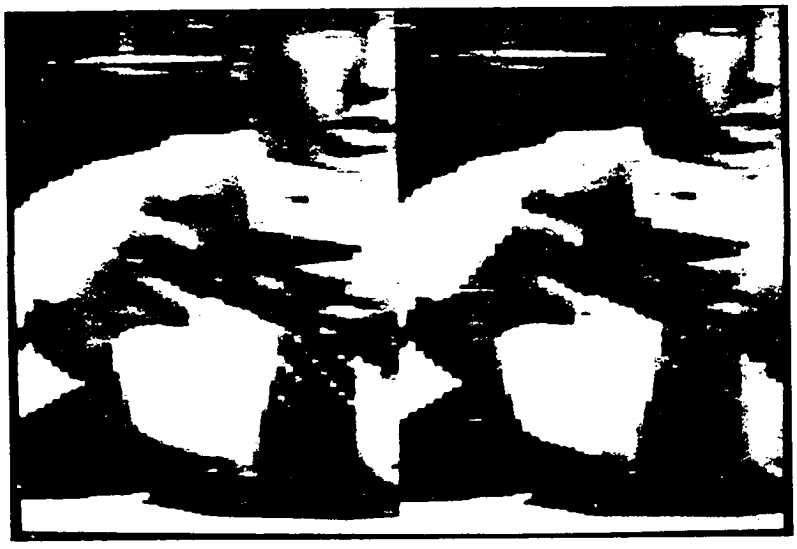


Figure 4.14(a) Low resolution original, and DCT coded blk-size (8*8) q=9, images of "Salesman" sequence

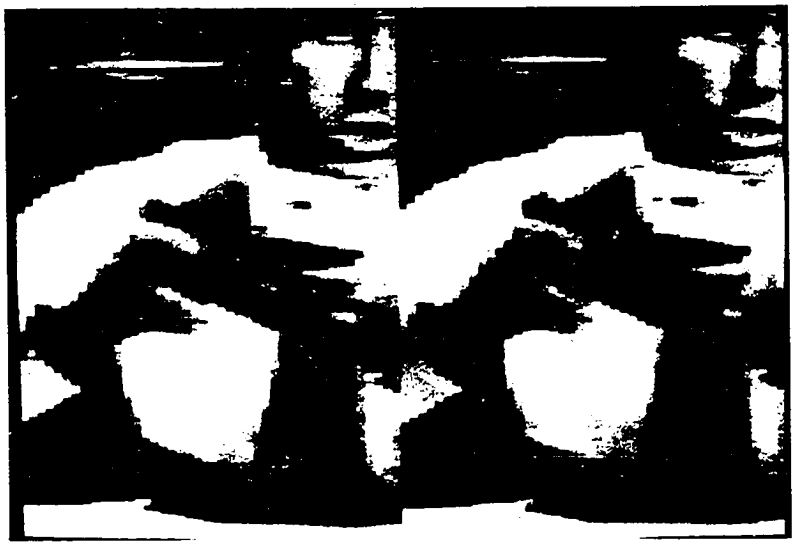


Figure 4.14(b) DCT/DST multi-transform coded blk-size (8*8) q=9, and DST coded blk-size (8*8) q=9, images of "Salesman" sequence

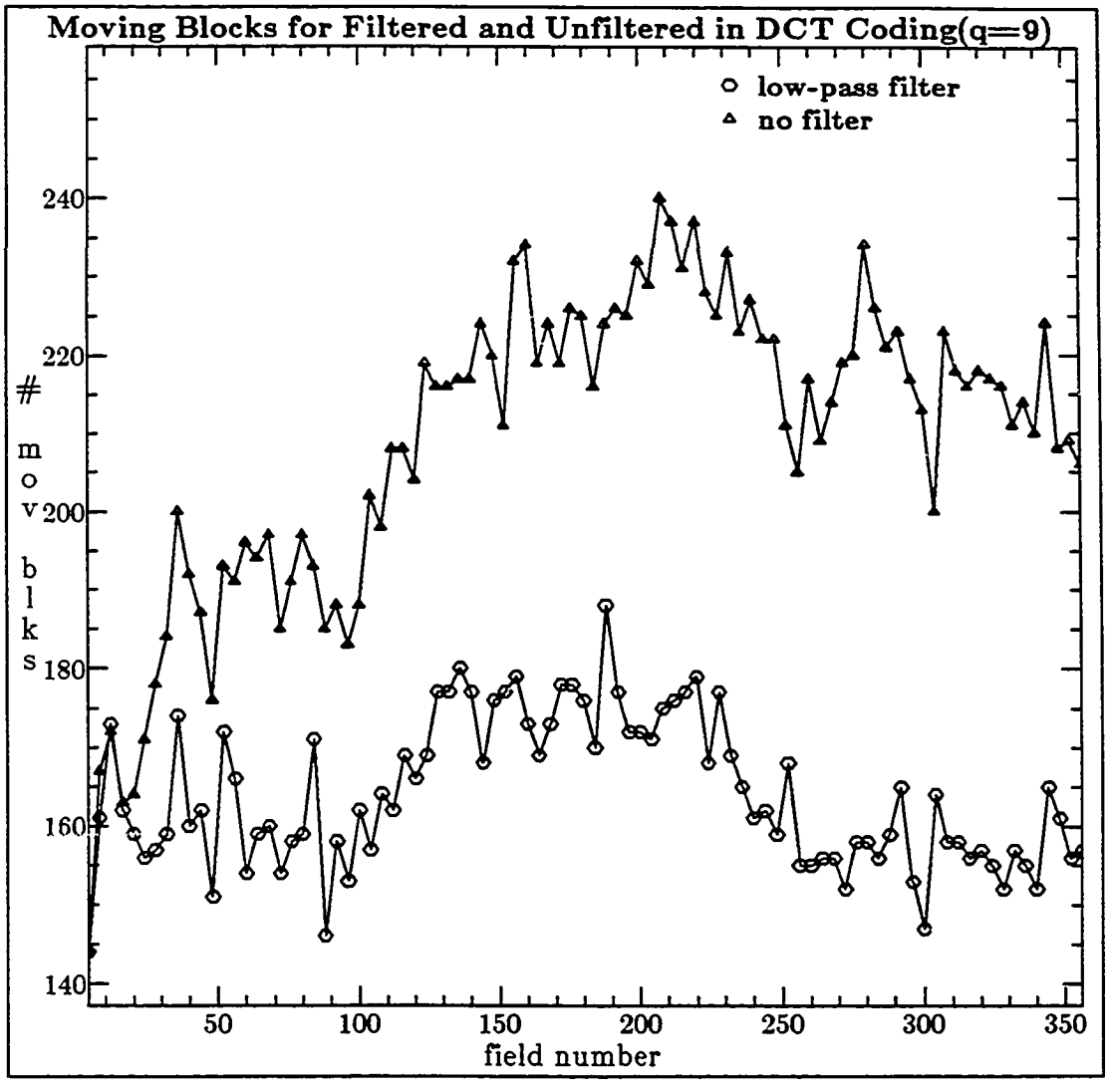


Figure 4.15(a) Moving Block Distribution in DCT Coding (quant. step 9)
with and without Filter : "MsUSA" sequence

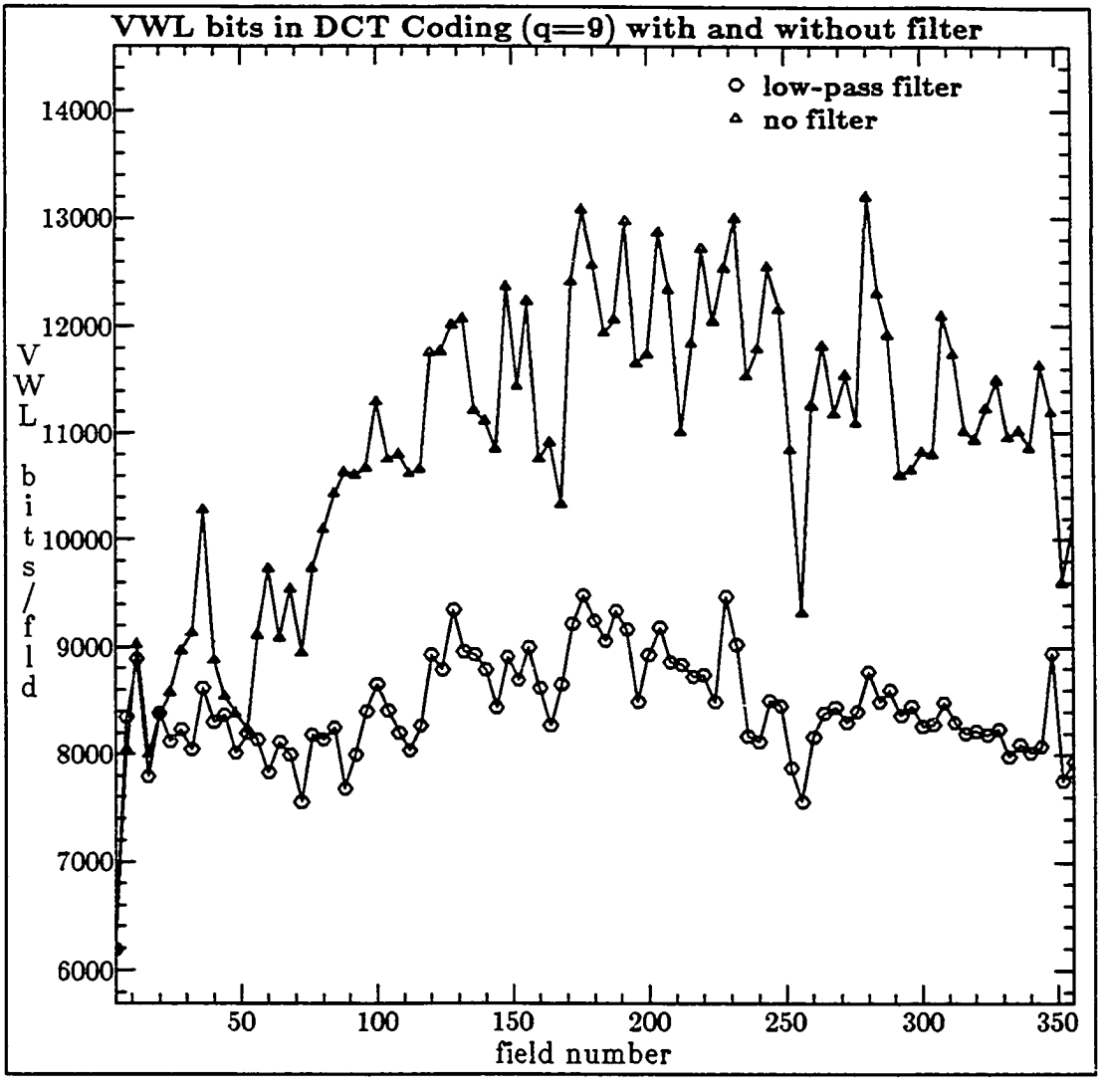


Figure 4.15(b) VWL Bits in DCT-coded Blocks (quant step 9) with and without Filter : "MsUSA" sequence

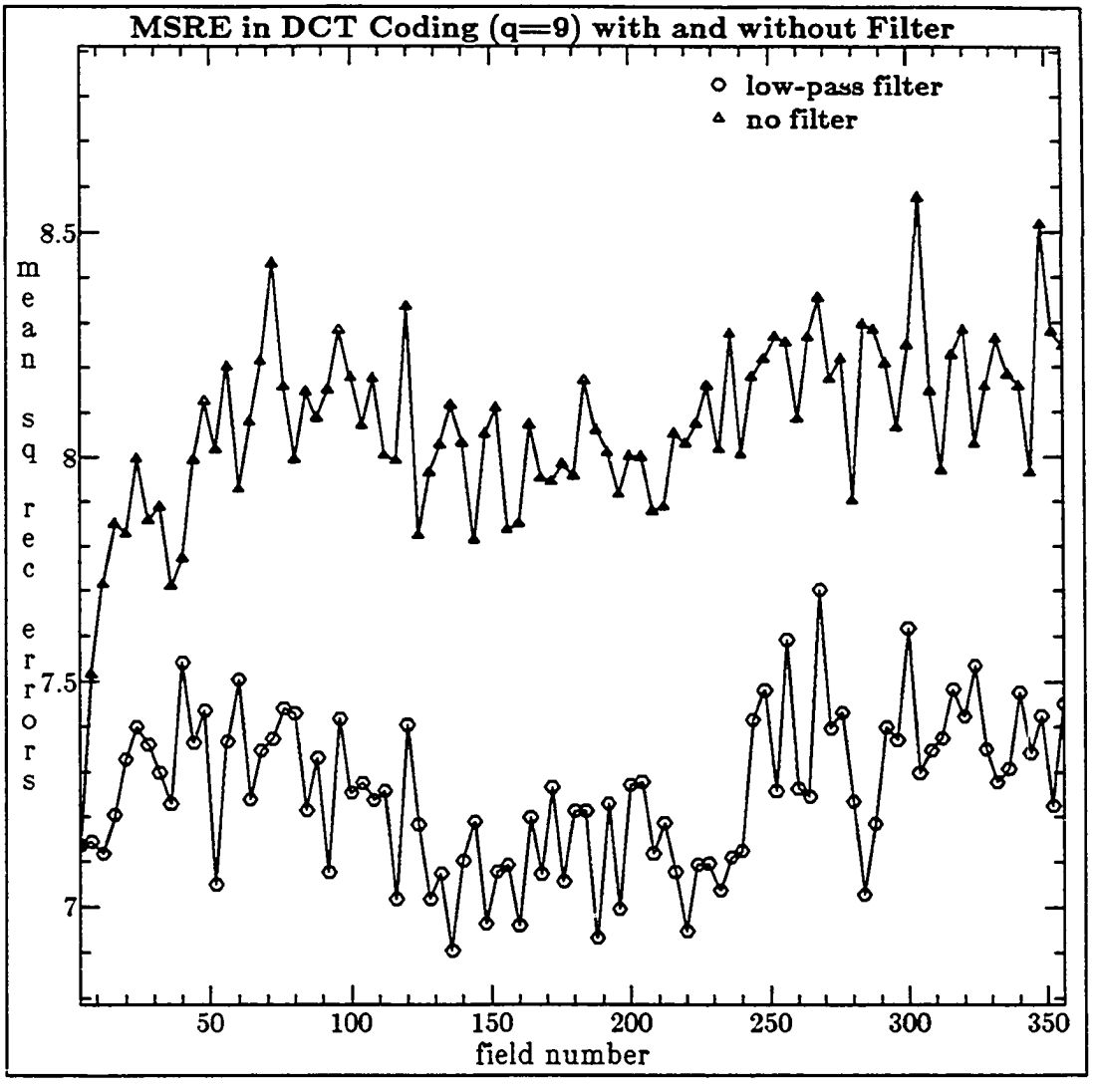


Figure 4.15(c) Mean Square Reconstruction Errors of Uncompensable blocks in DCT Coding (quant. step 9) with and without filter : "MsUSA" sequence

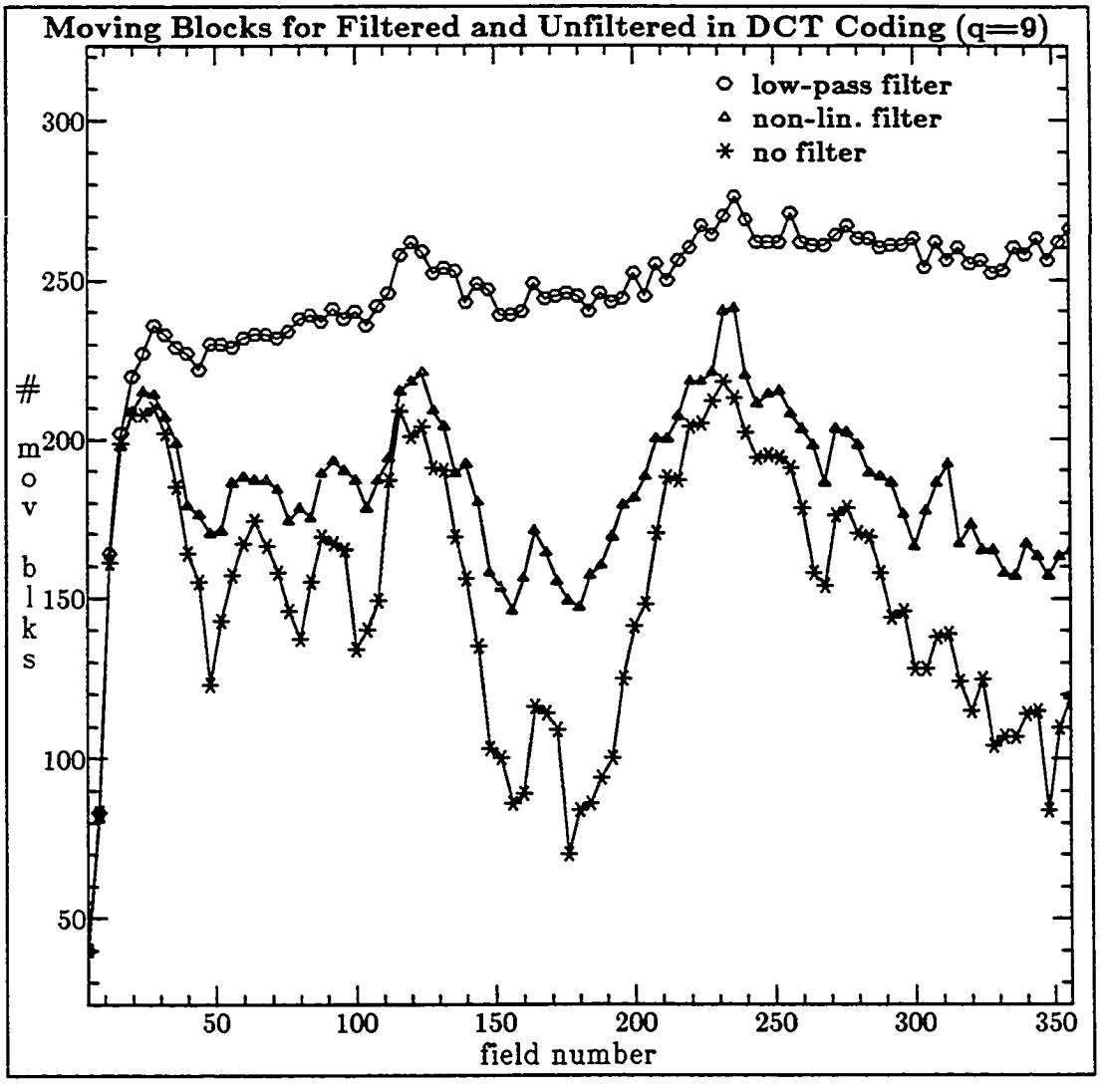


Figure 4.16(a) Moving Block Distribution in DCT Coding (quant. step 9)
with and without Filter : "Salesman" sequence

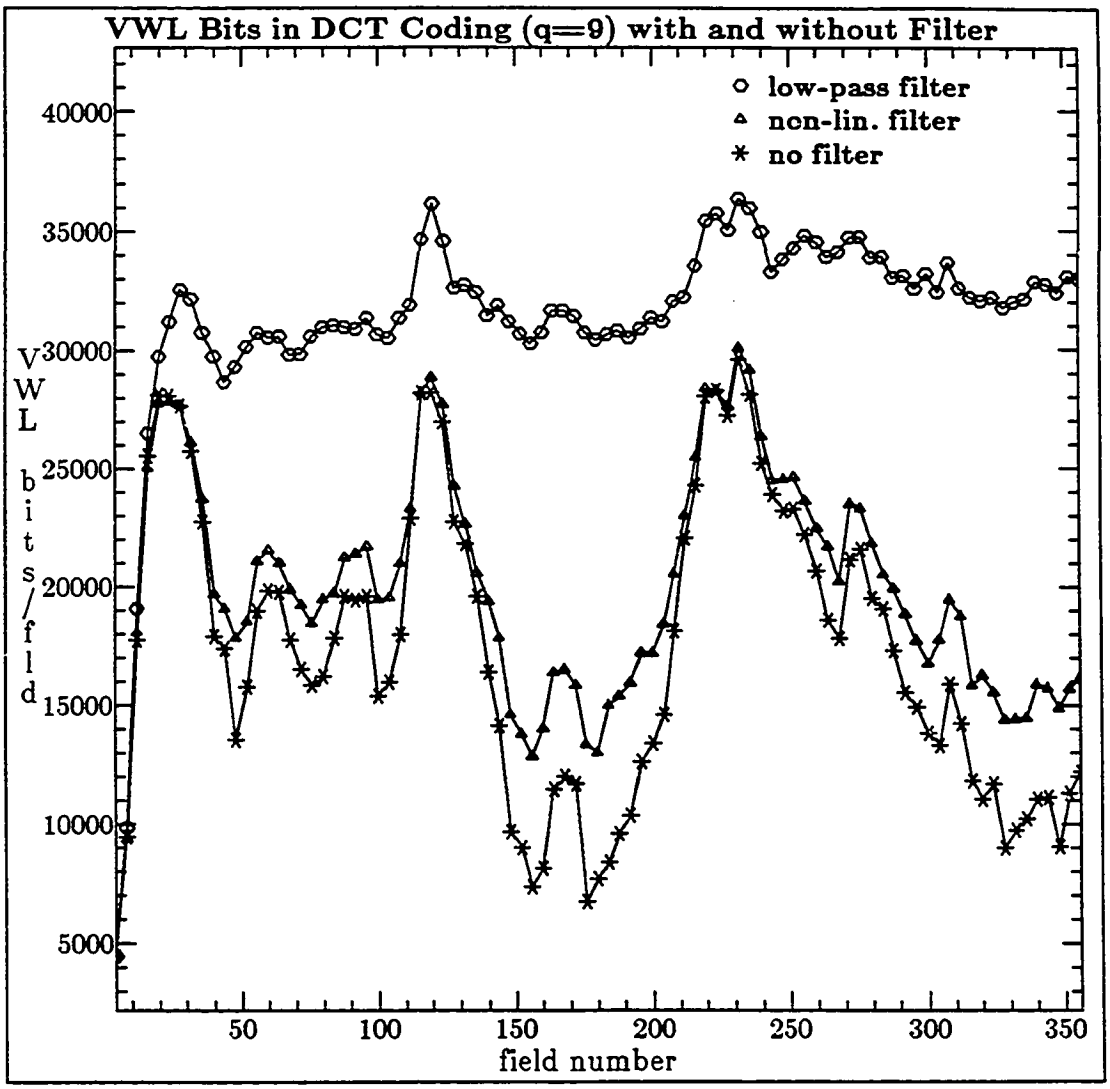


Figure 4.16(b) VWL Bits in DCT-coded Blocks (quant step 9) with and without Filter : "Salesman" sequence

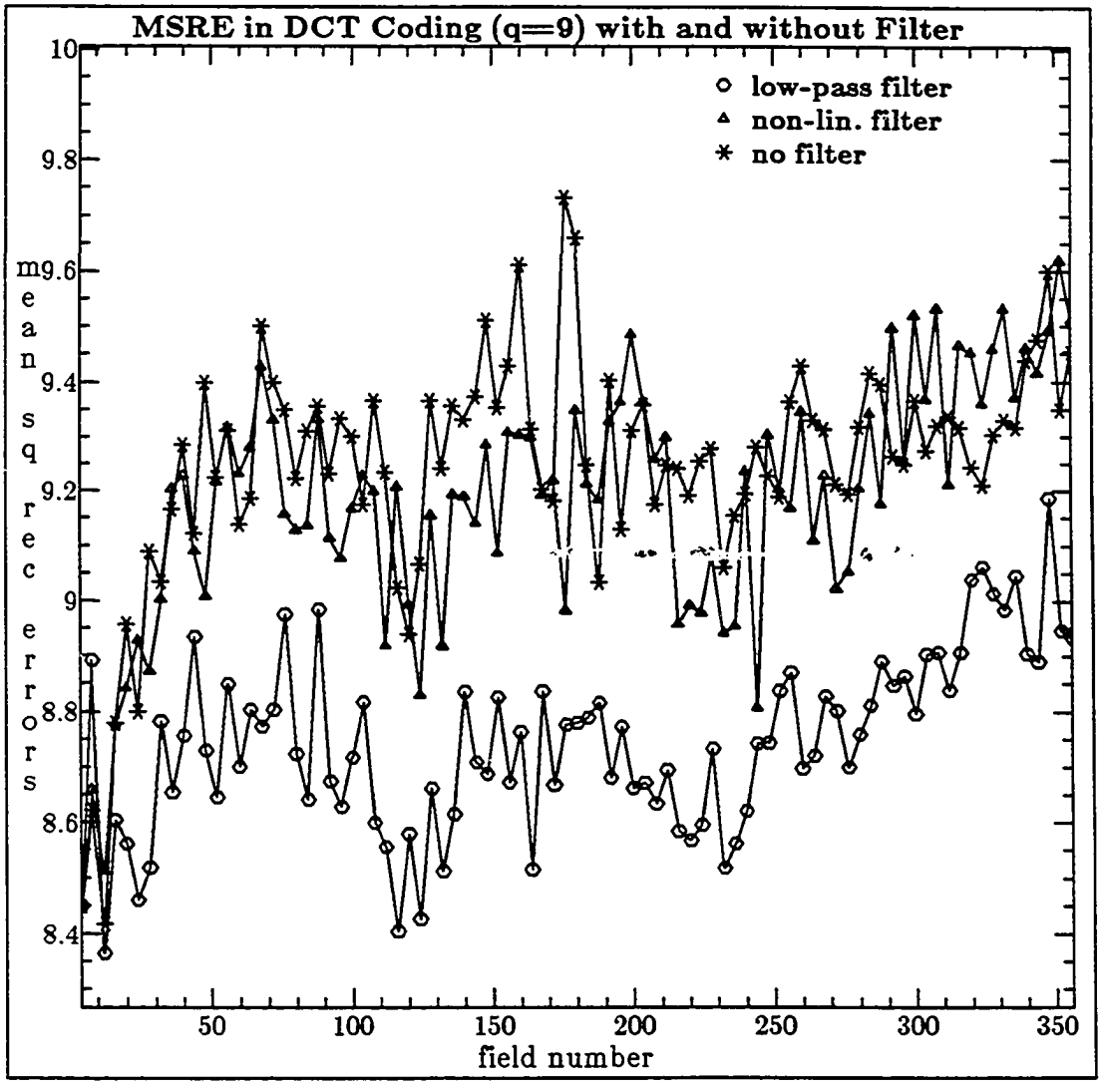


Figure 4.16(c) Mean Square Reconstruction Errors of Uncompensable blocks in DCT Coding (quant. step 9) with and without filter : "Salesman" sequence

Figure 4.17(a) DCT coded blk-size (8*8) q=9, without and with a low-pass filter in the coding loop, images from "MsUSA" sequence

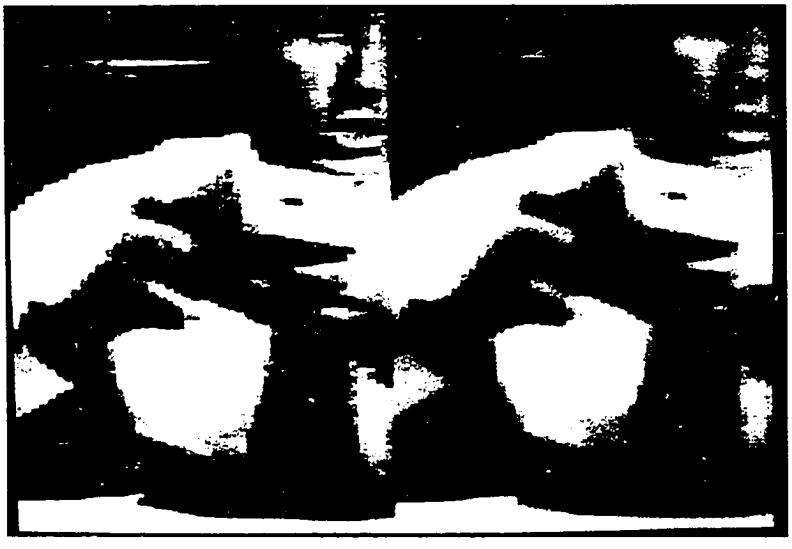


Figure 4.17(b) DCT coded blk-size (8*8) q=9, without and with a low-pass filter in the coding loop, images from "Salesman" sequence

CHAPTER 5: VARIABLE BLOCK-SIZE MOTION-COMPENSATED CODING

5.1 INTRODUCTION

In this chapter we compare several algorithms for motion-compensated interframe coding when either a fixed block-size or a variable block-size block-matching technique is employed for motion estimation. In the first part of the chapter, we describe a variable block-size motion estimation scheme that improves the motion estimation performance by subdividing blocks of images into sub-blocks and estimating their movements. In the second part, the variable block-size technique is applied to encode the motion-compensated interframe differential pels in both the transform domain and the pel-domain coding. Our simulation results demonstrate that both transform domain and pel-domain coding algorithms benefit from the variable block-size (VBS) approach, and a pel-domain algorithm performs almost as well as a DST algorithm when the VBS approach is used.

Several studies [Chapter 3],[21] and [101] have shown that the block-matching motion-compensation technique significantly reduces the bits required for inter-frame video encoding. This technique first partitions an image frame (or field) into fixed-size blocks and estimates the displacements (motion vectors) for the moving blocks. Then, only the differences (so-called *motion-compensated frame differences (MCFD)*) between the current frame and the translated previous frame in the moving blocks are coded and transmitted. In general, the motion even in video-conferencing scenes is not purely translational and may result in large MCFD's when large block-sizes are

used. Therefore, the use of smaller blocks improves the performance of motion compensation. This, however, also increases the computational complexity and the overhead of transmitting the increased number of motion vectors to the receiver. As a compromise between high motion-vector overhead and good motion-compensation performance, smaller block-sizes can be used only when the motion compensation on larger block-size does not perform very well. This is the basic idea of our proposed variable block-size motion-compensation algorithm. In coding the motion-compensated differential signals, the smaller block-size motion compensation is likely to produce less bits as compared to codes generated by using a larger block-size motion estimation. The bit saving due to a more accurate motion compensation may, however, be offset by the overhead required for sending the extra motion vectors. Thus we investigate the effectiveness of various coding algorithms on real video sequences for two purposes:

(1) to see the advantages offered by a variable block-size motion estimator in the real coding environment.

(2) to compare the transform-domain coding schemes with the pel-domain coding schemes.

5.2 CODING SYSTEM, MOTION, AND FRAME-DIFFERENCES

Before further discussing details of the variable block-size motion compensation algorithm, we again briefly describe the overall coding system used in this Chapter. Typically, there are three basic elements in an interframe encoding system: (1) a *motion detector* which detects the moving blocks, (2) a *displacement estimator* which estimates the displacement vectors of moving blocks, and (3) a *data compression*

algorithm which encodes the inter-frame differences after motion-compensation. The function of a motion detector is to classify a block to be *moving* or *non-moving* before any motion estimation can be done for that block. Our motion detector here is the same as the one used in [6] and [7]. A pel at location (x,y) in frame k is called *moving* if $|s_k(x,y) - s_{k-1}(x,y)| > T_0$, where $s_k(x,y)$ and $s_{k-1}(x,y)$ represent pels in frames k and k-1, respectively. A block is called *moving* if the number of moving pels in that block is greater than or equal to N_0. For "MsUSA" sequence the parameters, T_0=3 and N_0=10, and for "Salesman" sequence T_0=6 and N_0=7, work adequately for blocks of size 8×8 and 8-bit pels in the range 0-255.

The goal of a block-displacement estimator is to find the best match of a block from frame k in a suitable *match area* in the previous frame k-1. A block-matching algorithm which tracks the motion between neighboring frames has a low computational complexity and a good performance as discussed in Chapters 3, 4, and references [101],[102]. The algorithm employed here uses the displacement vector of the identically located block in the previous frame as the initial match location for the current-frame block and corrects the initial estimate by a small exhaustive search around it. In order to decide if the best match has been found, the mean absolute (frame) difference (MAD) matching criterion is adopted here for its simplicity and good performance. The MAD value also tells us how well a match has been made so that a sub-block matching may or may not be needed.

The frame differences after motion compensation (MCFD) have to be encoded and transmitted to the receiver if their values are significant. We follow the principle discussed in previous Chapter to initially classify the image blocks into the following three types.

Type 1: *nonmoving blocks* -- The frame-difference values are mostly small and hence this type of block is identified by the motion detector as non-moving.

Type 2: *compensable moving blocks* -- These blocks are originally identified as moving by the motion detector, but after motion compensation the values in these blocks are found to be below the motion activity threshold. For such blocks only the motion vectors need to be transmitted to the receiver.

Type 3: *uncompensable moving blocks* -- These blocks are identified as moving blocks and cannot be well-compensated by the block-matching motion compensation. Both the motion vectors and the coded MCFD values are transmitted for reconstruction of these blocks. The Type 3 blocks are examined further by the variable block-size motion compensator as described in the next section.

5.3 VARIABLE BLOCK-SIZE MOTION ESTIMATION

The selection of block size in motion compensation involves a trade-off in the quality of motion compensation and the overhead of transmitting the motion vectors. A small block-size offers the advantage of achieving accurate motion estimates because the motion is likely to be more uniform across the small block, and therefore complicated movements (other than translational) can be better compensated. On the other hand, the disadvantages associated with small block-sizes are:

(1) the displacement estimate is more sensitive to noise, and

(2) the motion-vector overhead is larger.

For large block-sizes, the above advantages and disadvantages are reversed. The variable block-size motion compensation algorithm provides a reasonable compromise as it allows some flexibility in improving motion estimates for parts of the hard-to-

compensate blocks while it also tries to keep the overhead small.

The basic variable block-size (VBS) motion-compensation scheme is as follows. We first compensate all the moving blocks (identified by the motion detector) with the regular block-size, and then the motion-compensated frame difference (MCFD) blocks are classified (by the motion detector again) into the compensable (Type 2) or the uncompensable (Type 3) blocks. The uncompensable blocks are partitioned into sub-blocks, and the sub-block motion compensation algorithm is applied to them. After the sub-block motion compensation, the sub-block MCFD's are used to reconstruct the blocks back together from the sub-blocks. The block motion detector is applied to the VBS motion-compensated blocks to judge the performance of the sub-block motion compensation and categorizes them into the following subgroups:

Type 3a: *sub-block compensable moving blocks* -- With sub-block motion compensation these blocks are now classified as compensable blocks.

Type 3b: *sub-block uncompensable moving blocks* -- Even with sub-block motion compensation these blocks are still above the block motion-activity threshold, i.e. not well compensated.

A block diagram which illustrates the classification of MCFD blocks is shown in Figure 5.1. In our experiments, the regular block size is chosen to be 8×8, and the sub-block size to be 4×4.

In order to investigate the effectiveness of this variable block-size motion compensation algorithm, we apply it to a real video sequence. The original "MsUSA" and "Salesman" image-sequences are digitized from NTSC signals sampled at twice the color subcarrier frequency and 8 bits per pel; and converted to component form consisting of a luminance signal Y and chrominance signals U and V. A line of

luminance contains 368 pels, and there are 240 such lines per field. Since our application is low bit-rate video-conferencing, we further reduce the spatial resolution by subsampling in the horizontal direction by a factor of 2. The image field now consists of 184 *pels* $\times$ 240 *lines*. This is converted into block format with block size of 8$\times$8. Each field now contains 23 Y blocks in the horizontal direction and 30 Y blocks in the vertical direction, and will be referred to as a "frame" in the terms such as *frame difference* used in this chapter. Performance of motion compensation algorithms is investigated at a temporal subsampling rate of 4:1, i.e., 15 fields (also referred to here as frames) per second. The displacements in the high-motion areas in the sequence still do not exceed 5 pels. This is typical of most movements in a video-conferencing environment.

In this part of computer simulations, we assume the coded pictures are perfectly reconstructed (i.e. no coding errors) so that only the effect of variable block-size motion compensation is demonstrated. As shown by Fig. 5.2(a), the fixed-size motion compensation on the 8$\times$8 moving blocks can significantly reduce the number of blocks that actually require coding. The sub-block motion compensation (on 4$\times$4 blocks) further reduces the number of such blocks by about 15%. Next, we observe from Fig. 5.2(b) that the mean square values of the motion-compensated frame-differences decreases by about 30% when the 4$\times$4 sub-block motion compensation is used after the 8$\times$8 block motion compensation. Thus the VBS motion compensation technique, which subdivides the 8$\times$8 uncompensable blocks into 4$\times$4 sub-blocks for motion compensation, could be effective in reducing bits in encoding the MCFD pels.

Fig. 5.3(a) shows the block diagram of an interframe encoding system with the fixed-size motion compensation and block-coding. Fig. 5.3(b) shows an interframe encoding

system with the VBS motion-compensation and block-coding. In Fig. 5.3(c) we allow the VBS motion-compensation followed by sub-block coding. The data compression and decompression in all coding systems are handled by either a transform-domain or a pel-domain coding technique. These techniques with or without the VBS approach are now discussed.

5.4 TRANSFORM DOMAIN CODING WITH AND WITHOUT VARIABLE BLOCK SIZE MOTION-COMPENSATION

Transform coding is a popular technique in both intraframe and interframe picture compression [2] Although the Discrete Cosine Transform (DCT) has been a favored coding technique for compressing the highly correlated signals in intraframe coding, we have found that the Discrete Sine Transform (DST) results in a slightly better reconstructed picture quality at approximately the same bit-rate as required for DCT coding, in compressing the low-correlated MCFD signals. Therefore, DST is chosen for our simulations.

We use the basic coder structures of Figures 5.3(a), 5.3(b) and 5.3(c), and insert the transform compression algorithms into the data compression block. In the data compression block, transform of 8×8 or 4×4 uncompensable blocks, is taken and then a uniform midtread quantizer is applied to the dominant coefficients which are selected by the following rules:

(1) coefficients are significant if greater than a threshold (= twice the quantization step),

(2) the three highest frequency significant coefficients are discarded for 8×8 blocks whereas only one highest frequency significant coefficient is discarded for 4×4 blocks,

and

(3) the four lowest frequency coefficients are retained even if they are insignificant for 8×8 blocks whereas at least the 3 lowest frequency coefficients are retained for 4×4 blocks. In addition, to make final entropy bits closer to reality every coded transform coefficient assumes at least 1.1 bits.

We consider three schemes.

Scheme 1: Quantized-DST Coefficients in 8×8 blocks with fixed block-size 8×8 motion-compensation -- we quantize the selected coefficients of the Type 3 blocks and then entropy-code them.

Scheme 2: Quantized-DST Coefficients in 8×8 blocks with VBS motion-compensation -- we quantize the selected coefficients of the sub-block uncompensable 8×8 block (Type 3b) and then entropy-code them.

Scheme 3: Quantized-DST Coefficients in 4×4 sub-blocks with VBS motion-compensation -- we quantize selected coefficients of the 4×4 active uncompensable sub-block and then entropy-code them. The sub-blocks on which coding needs to be done are identified by a sub-block activity detector, which is similar to the motion detector in Section 1 with size 4×4, $T_0=3$ while $N_0=4$.

Fig. 5.3(d) illustrates this active/inactive segmentation process.

We measure the entropy bit-rate, the mean square reconstruction error (MSRE), and the number of uncompensable blocks of this DST coder for the three schemes. The results for "MsUSA" sequence are shown in Fig 5.4. The number of uncompensable blocks (Fig. 5.4(a)), i.e., the Type 3 blocks in the fixed block-size case and the Type 3b in the VBS case, is about 25% less with VBS motion-compensation. From Fig 5.4(b)

we find that the number of active sub-blocks is roughly two thirds of the total number of sub-blocks for the VBS motion-compensation case. Fig. 5.4(c) shows that the entropy bits per field for the Scheme 2 (VBS motion-compensation) is, about 20% lower on the average (roughly 2,250 bits per frame) than that of the Scheme 1 (fixed block motion-compensation). Furthermore, Scheme 3 (VBS motion-compensation and coding) is, about 12% lower in terms of entropy-bits than Scheme 2. The mean square coding errors for the uncompensable blocks (Type 3) are roughly the same for all schemes (Fig. 5.4(d)). The visual quality of all coded sequences are also found to be very similar, with slight preference to sequences obtained using Schemes 2 and 3.

For "Salesman" sequence the entropy results for DST coding are shown in Fig 5.6(a), Schemes 1 and 2 are compared; only during periods of high activity the Scheme 2, with sub-block motion-compensation and block coding saves about 2000 bits as compared to Scheme 1, that uses block motion-compensation and block-coding. From Figure 5.6(b), we see that Scheme 2 has lesser MSRE's during periods of low activity, where as the errors are about the same as that of Scheme 1 during periods of high activity.

For "MsUSA" the VBS motion-compensated coder of Scheme 2 can reduce the transform codes by 2,250 bits per frame on the average, but, it requires extra motion-vectors, which may cost about 1,000 to 2,000 bits per frame if the differential motion vectors (between the 8×8 motion-vector and the 4×4 motion-vectors in that block) are transmitted. The VBS motion-compensated coder of Scheme 3 reduces the bit rate on the average by another 1,100 bits per frame as compared to coder of Scheme 2, but, requires another overhead (200 to 500 bits per frame) for transmitting the sub-block (activity) information. For "Salesman" sequence the improvement of Scheme 2 as

compared to Scheme 1 is even lower. In any case, Scheme 2 shows slight visual improvement over Scheme 1.

Overall, the VBS Schemes 2 and 3 provide, both statistically and visually, somewhat better performance as compared to the ordinary fixed block-size Scheme 1; however these improvements are small.

5.5 PEL DOMAIN CODING WITH AND WITHOUT VARIABLE BLOCK-SIZE MOTION-COMPENSATION

As an alternative to transform domain coding approach, we consider the pel domain coding of MCFD signals. In the previous Chapter we saw that the significant transform coefficients in a MCFD block are usually widely spread and do not have a regular distribution pattern. In addition, it is often necessary to retain one third or more of the transform coefficients in a block to obtain a reasonable quality reconstruction. The distribution of MCFD pels in a block (in the pel domain) do not appear to be much worse. Typically, one half or less of the MCFD pels are significant and spread out in a block. This motivates us to test pel-domain compression algorithms and compare their performance to the transform domain algorithms.

We still use the basic coder structures of Figures 5.3(a), 5.3(b) and 5.3(c), with pel-domain compression algorithms inserted into the data compression block. Three simple pel-domain coding schemes are considered.

Scheme 1: Quantized-MCFD in 8×8 blocks with fixed block-size 8×8 motion-compensation -- we simply coarse-quantize (using a midtread) quantizer all the pels in the Type 3 blocks and then calculate the entropy of the quantized pels. Similar to that in the DST coefficient quantization case, every quantized pel assumes at least 1.1

bit.

Scheme 2: Quantized-MCFD in 8×8 blocks with VBS motion-compensation -- every pel of the sub-block uncompensable 8×8 block (Type 3b) is quantized and entropy-coded.

Scheme 3: Quantized-MCFD in 4×4 sub-blocks with VBS motion-compensation -- every pel of the 4×4 active uncompensable sub-block is quantized and entropy-coded. The sub-blocks on which coding needs to be done are identified by the same sub-block activity detector as discussed in Scheme 3 for transform coding.

Again, we compare the above three schemes based on the following criteria:

(1) number of coded MCFD pels.

(2) entropy of the coded MCFD pels.

(3) mean square reconstruction errors, MSRE (of Type 3 blocks).

The results for "MsUSA" sequence are shown in Figures 5.5(a), 5.5(b) and 5.5(c), respectively. A quantization step-size of 11 is used. In terms of both the entropy bits (Figure 5.5(b)) and the number of coded pels (Figure 5.5(a)), Scheme 3 outperforms the other two. The MSRE's (Fig. 5.5(c)) of Scheme 3 are slightly higher than that of Scheme 2 due to the few large-magnitude MCFD pels classified as inactive sub-blocks by our active/inactive detector. The quality of reconstructed pictures is not noticeably different using any of the three pel-domain approaches.

The results for "Salesman" sequence are shown in Figures 5.7(a), 5.7(b) and 5.7(c). A quantization step-size of 9 is used. In Figure 5.7(a) we observe that for Scheme 2, the number of coded pels are midway between that for Scheme 1 and Scheme 3. The entropy performance results of Figure 5.7(b) show a significant improvement in going

from Scheme 1 to Scheme 2; however there is very little difference in performance between Scheme 2 and Scheme 3. This can be attributed to the fact that, with 8×8 block-sizes, motion can not be compensated very well for "Salesman" sequence, therefore reducing block-size to 4×4 for motion-compensation in Scheme 2 results in a significant improvement. However, the distribution of difficult MCFD pels is random and scattered over the 8×8 block, and no significant improvement in performance is obtained in going to smaller block-sizes in Scheme 3. Also, on the basis of MSRE's for the three schemes we again note that Scheme 2 performs better than Scheme 1 and Scheme 3. The MSRE's for the Scheme 3 were found to be worst; this is so because the active/inactive segmentation, treats some sub-blocks with significant MCFD's as inactive, as few large MCFD's may be distributed evenly in the block. For this sequence we show a comparison of entropy bits for DST and pel-domain coding scheme in Figure 5.8(a); both the schemes use a step-size of 9. We note that, in using Scheme 1 for transform-domain and pel-domain coding, during low to moderate activity periods both schemes require nearly the same number of bits; only during periods of very high activity transform-domain scheme requires about 5000 bits less than transform-domain coding. From Figure 5.8(b) we see that, pel-domain coding had only two third of MSRE as that compared to transform domain coding.

5.6 CONCLUSION

Several block-matching motion-compensated coding schemes are compared with and without the VBS motion compensation. First, we find that both pel-domain and transform domain coding algorithms benefit from the VBS approach. For "MsUSA" sequence, a pel-domain and transform-domain schemes are found to be comparable on the statistical basis, e.g., pel-domain approach (Scheme 3) has performance

comparable to that of the DST-approach (Scheme 2) when both the schemes use VBS motion-compensation. The pel-domain scheme for "MsUSA" sequence with a coarse-quantizer (step-size = 11) results in nearly the same MSRE's as the DST based transform-coding scheme with a relatively finer quantizer (step-size = 9). As far as the coded image-quality is concerned, the DST based Schemes 2 and 3 (with VBS motion-compensation) are preferred to either the fixed block-size DST Scheme 1 or any of the pel-domain Schemes 1, 2 or 3. Although for "MsUSA" sequence the quantization noise is more visible in the pel-domain coded sequences as compared to the transform-domain coded sequences, the pel-domain schemes can perhaps be improved with relative ease as compared to the transform-domain approach.

In transform-domain coding of "Salesman" sequence comparisons between Schemes 1 and 2, show that Scheme 2 performs slightly better than Scheme 1 in active areas. The overall differences are rather small however. Further, in pel-domain coding of this sequence, the differences between entropy coding-bits for Schemes 2 and 3 are also rather small; the Scheme 2 outperforms Scheme 3 in terms of MSRE's however. We also compare the pel-domain and transform-domain approaches when Scheme 1 is used; we find pel-domain approach results in a low overall MSRE as compared to the transform-domain approach, and only during periods of very high activity transform-domain approach performs distinctly better in terms of entropy bits.

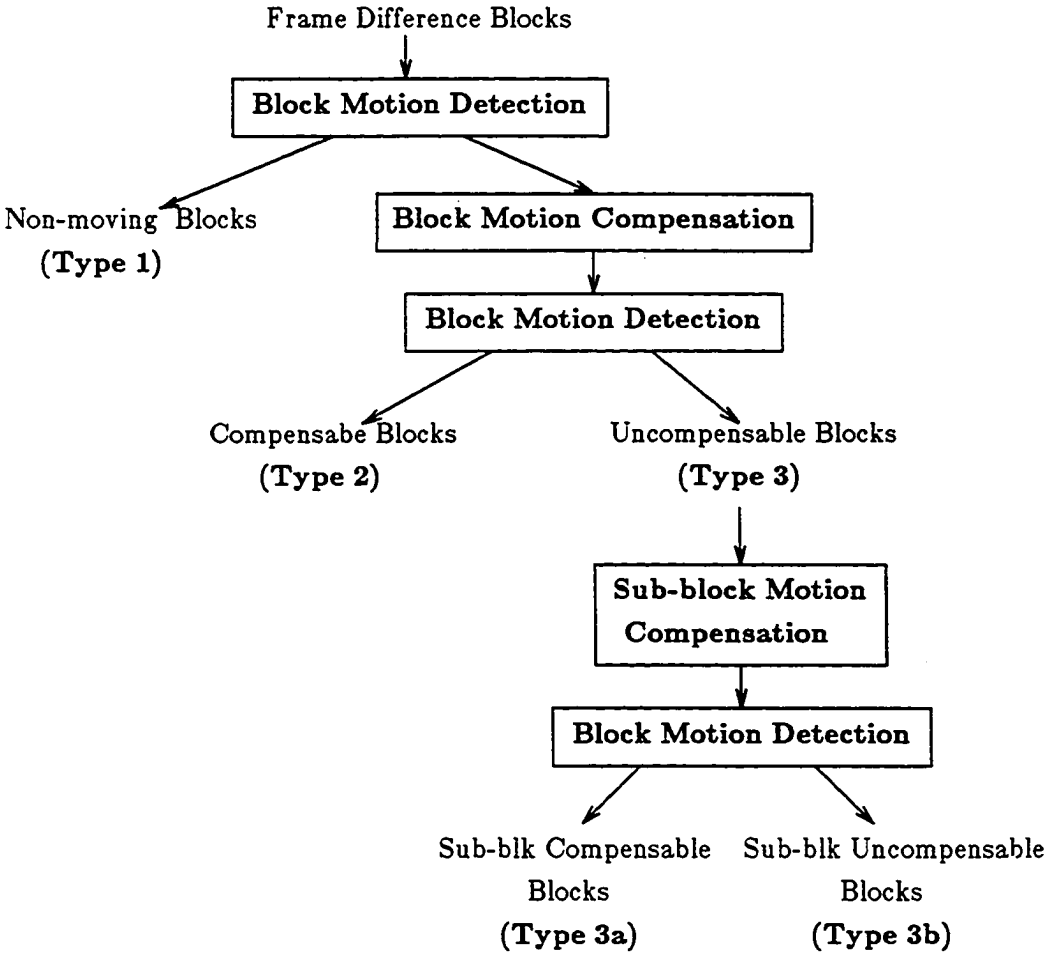


Figure 5.1 FD Block Classification for Variable-block Size Scheme

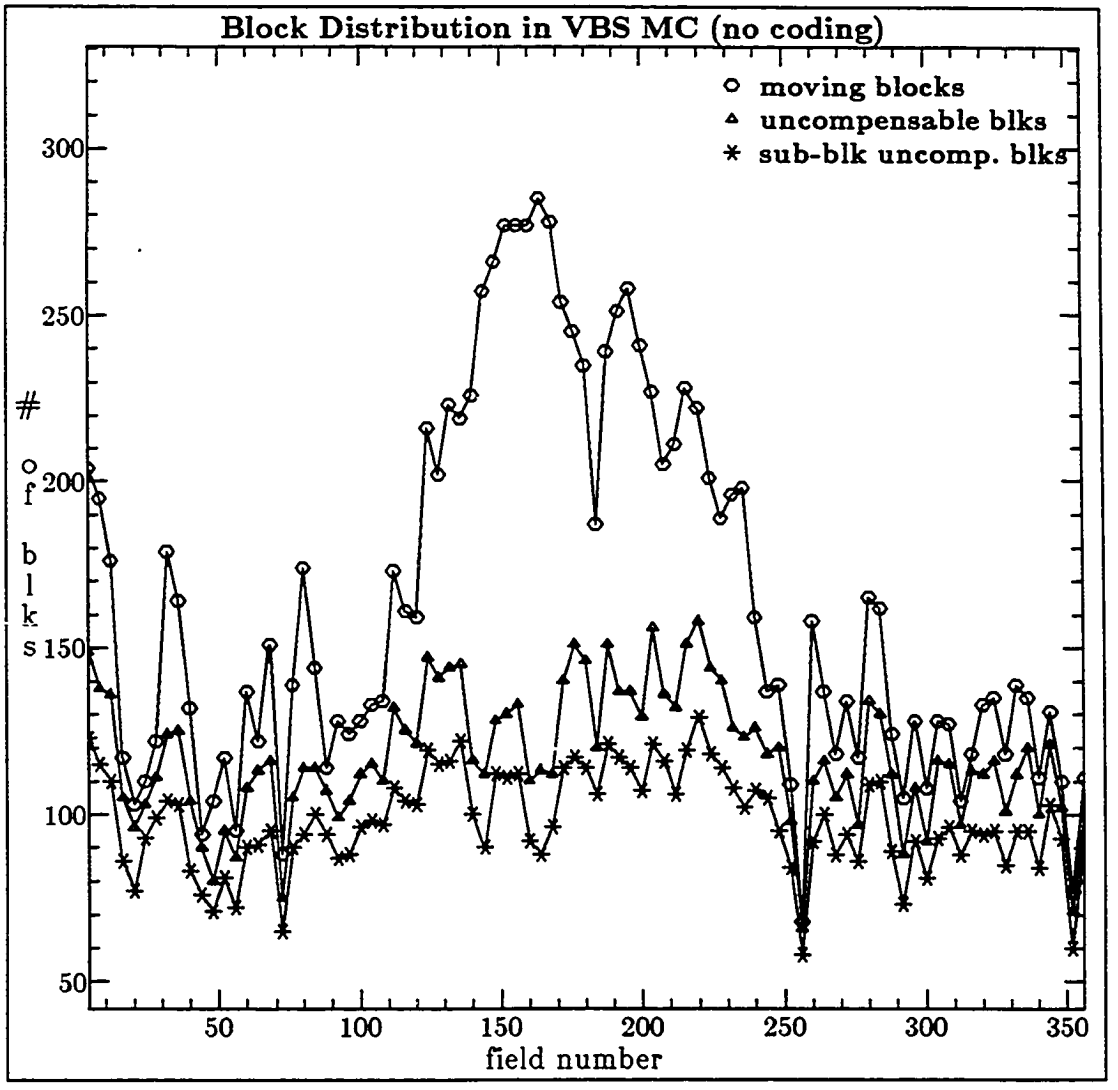


Figure 5.2(a) Frame Difference Block Distribution in Variable Block-size Motion Compensation (no coding): "MsUSA" sequence

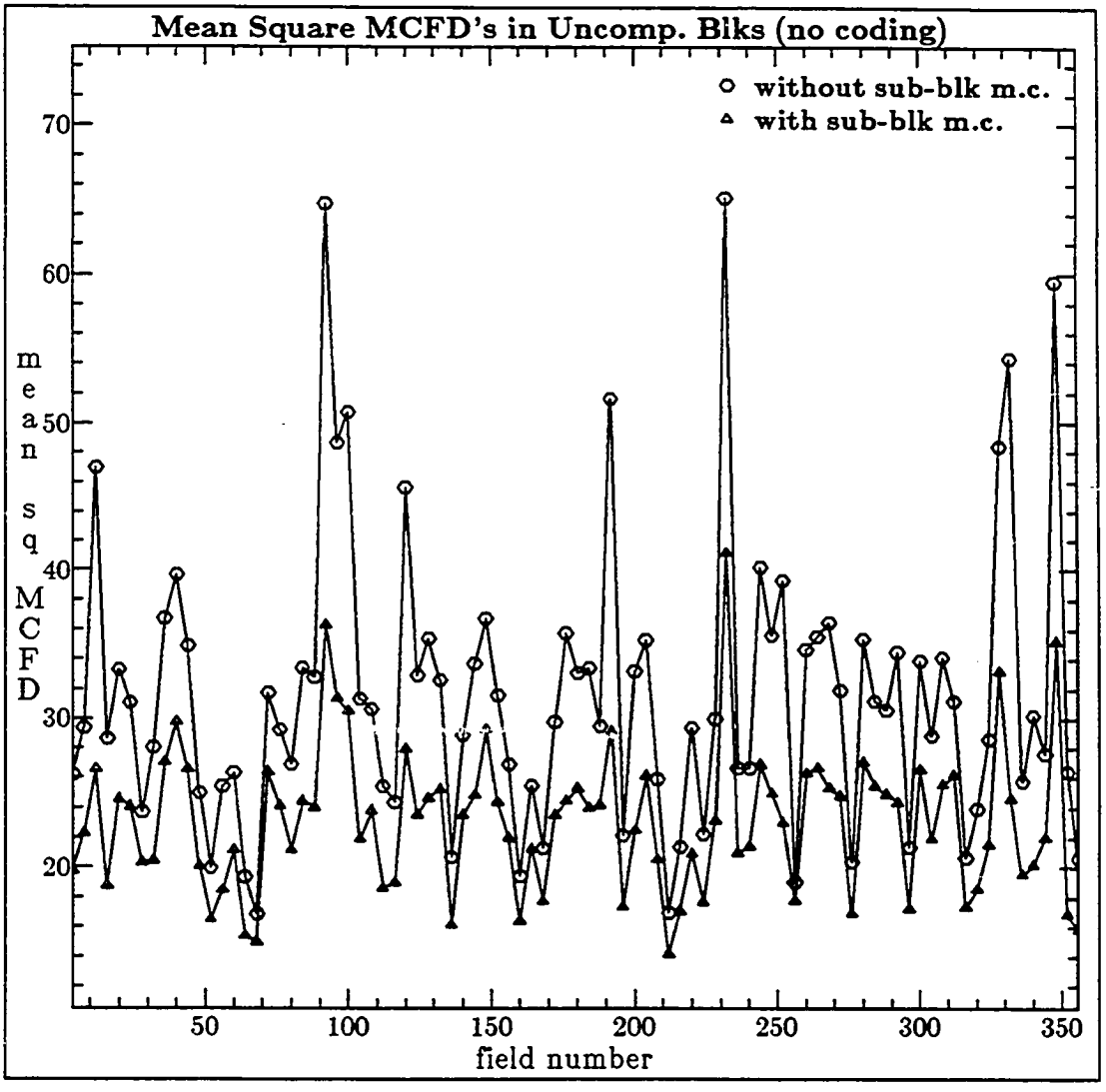


Figure 5.2(b) Mean Square values in MCFD Blocks (no coding) with and without sub-block motion-compensation: "MsUSA" sequence

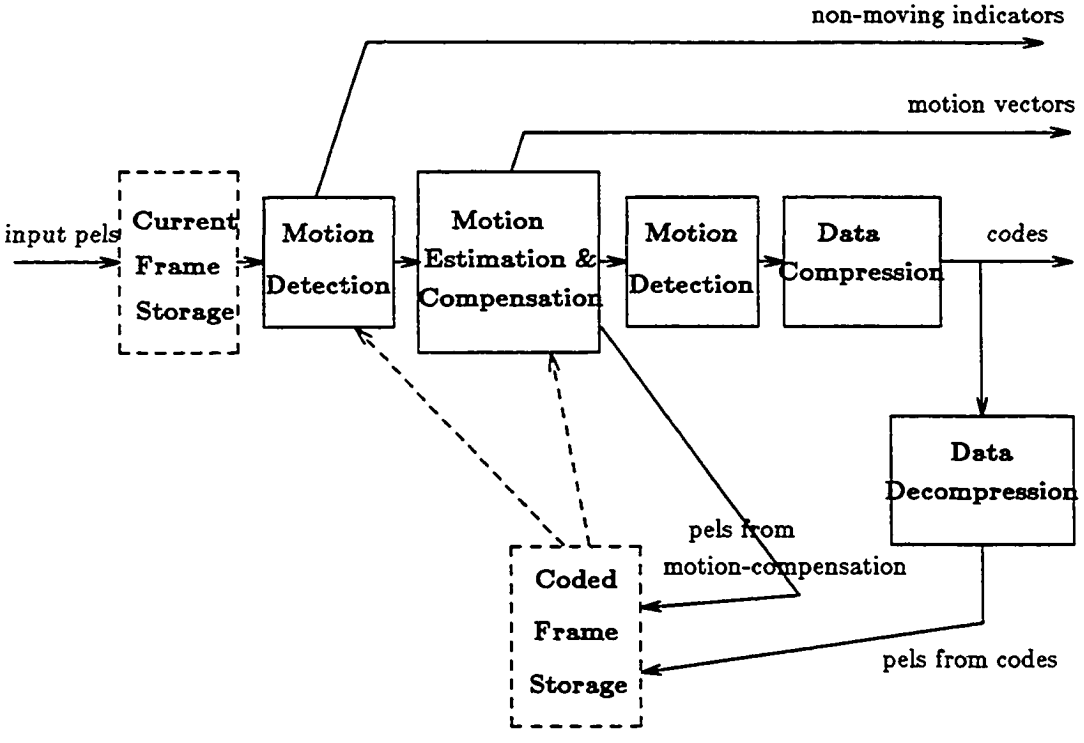


Figure 5.3(a) Interframe Coding system with fixed Block-size Motion-compensation

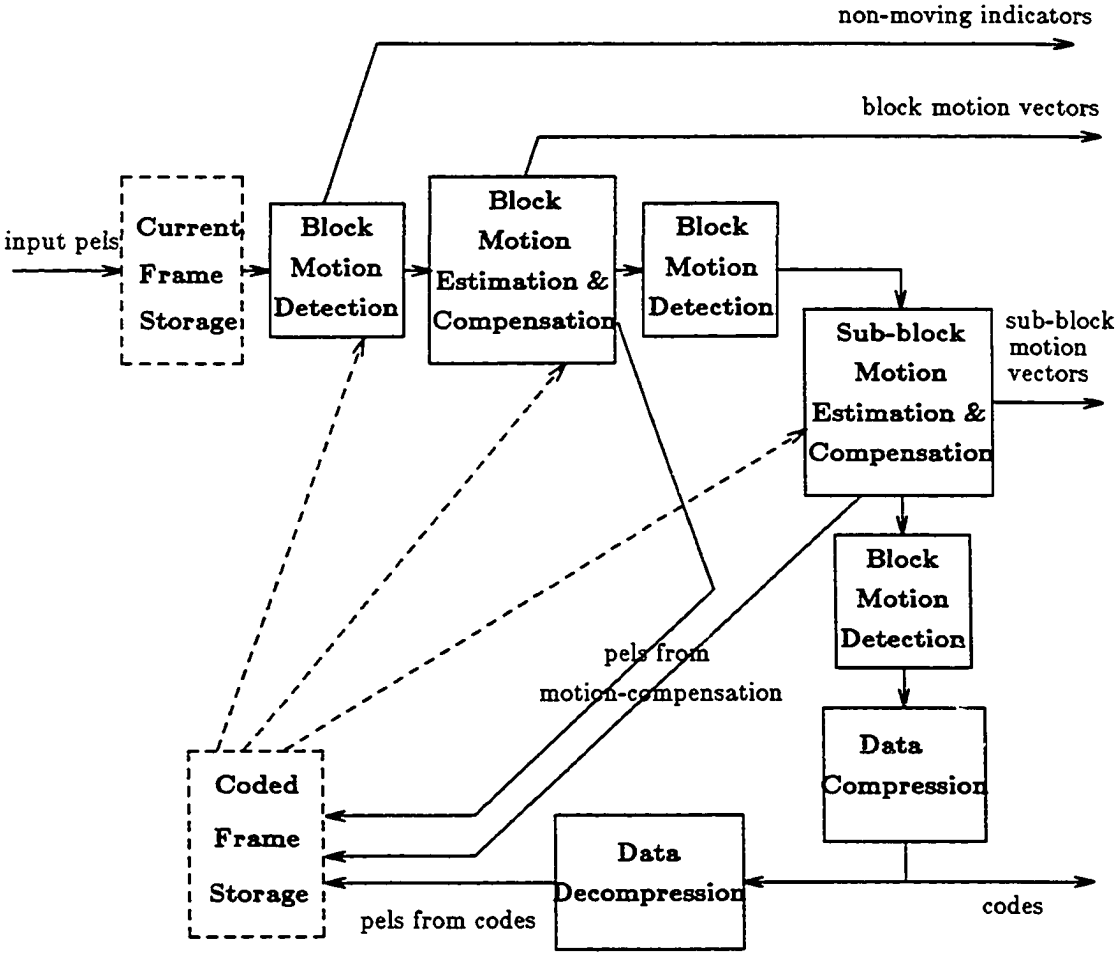


Figure 5.3(b) Interframe Coding system with variable Block-size Motion-compensation

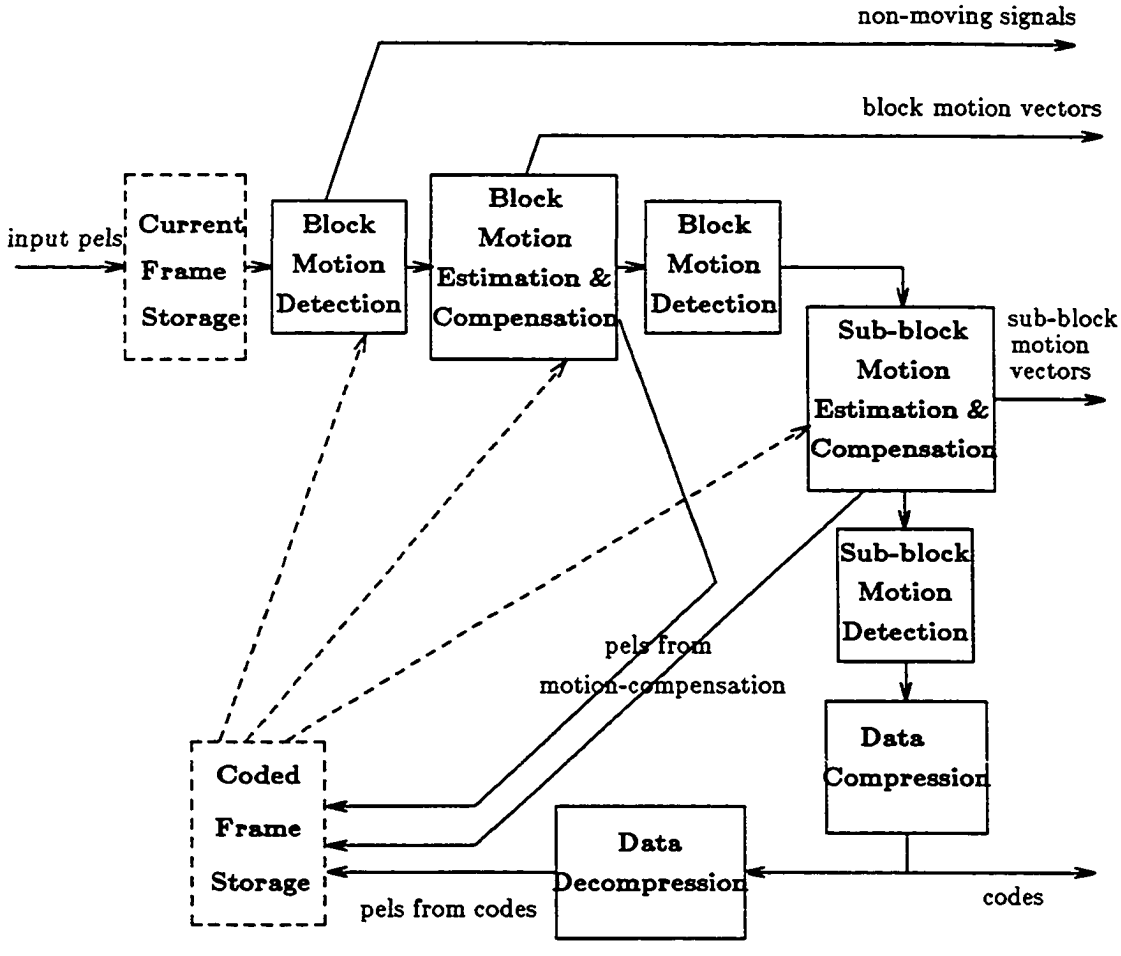


Figure 5.3(c) Interframe Coding system with variable Block-size Motion-compensation and Coding

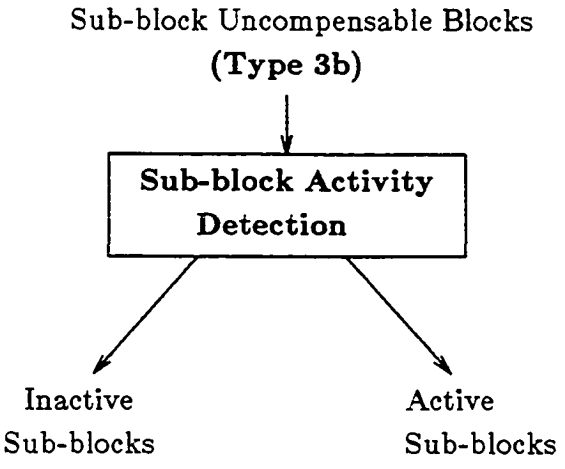


Figure 5.3(d) Sub-block Classification for Coding

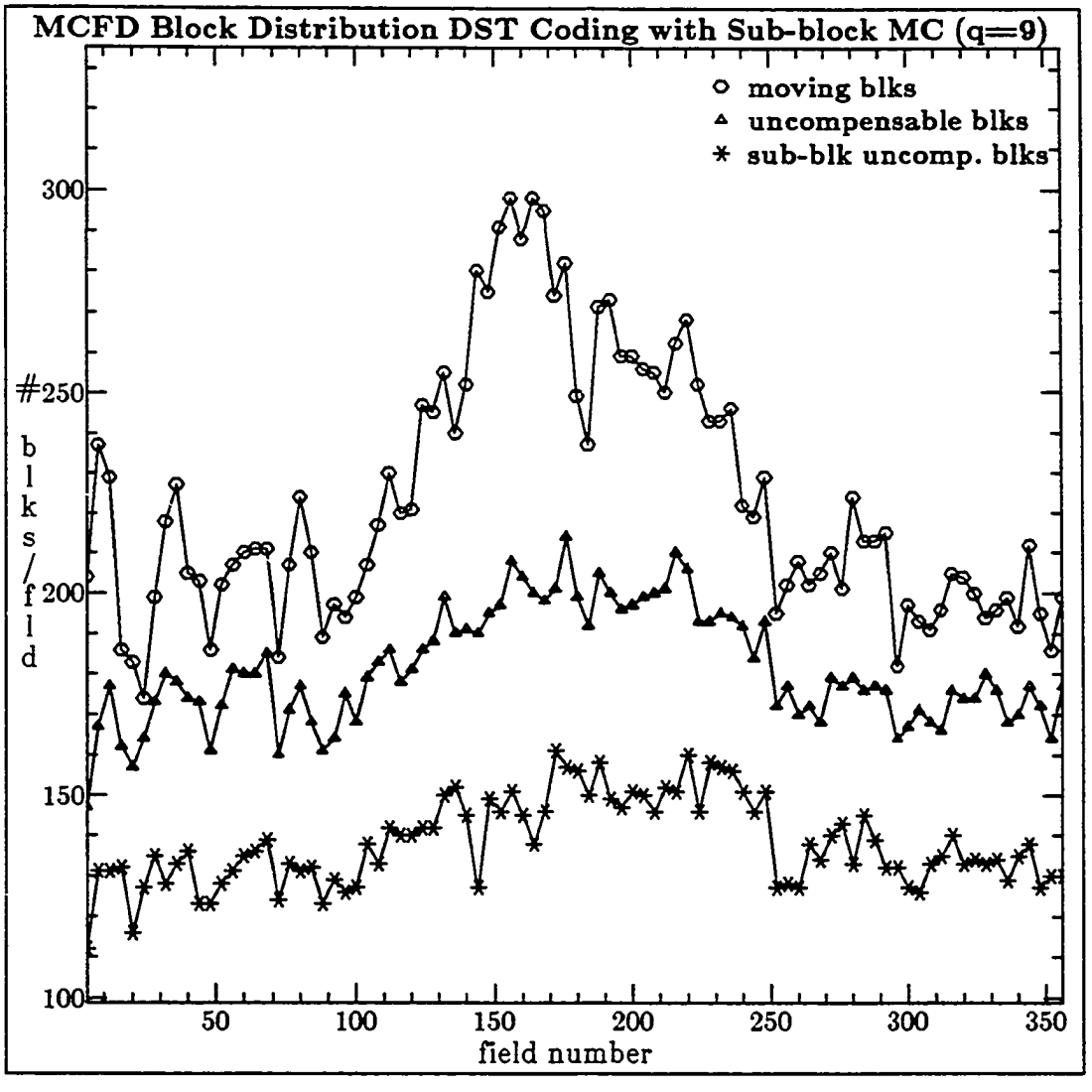


Figure 5.4(a) Frame Difference Block Distribution in DST Coding with Variable Block size Motion Compensation (quant step 9): "MsUSA" sequence

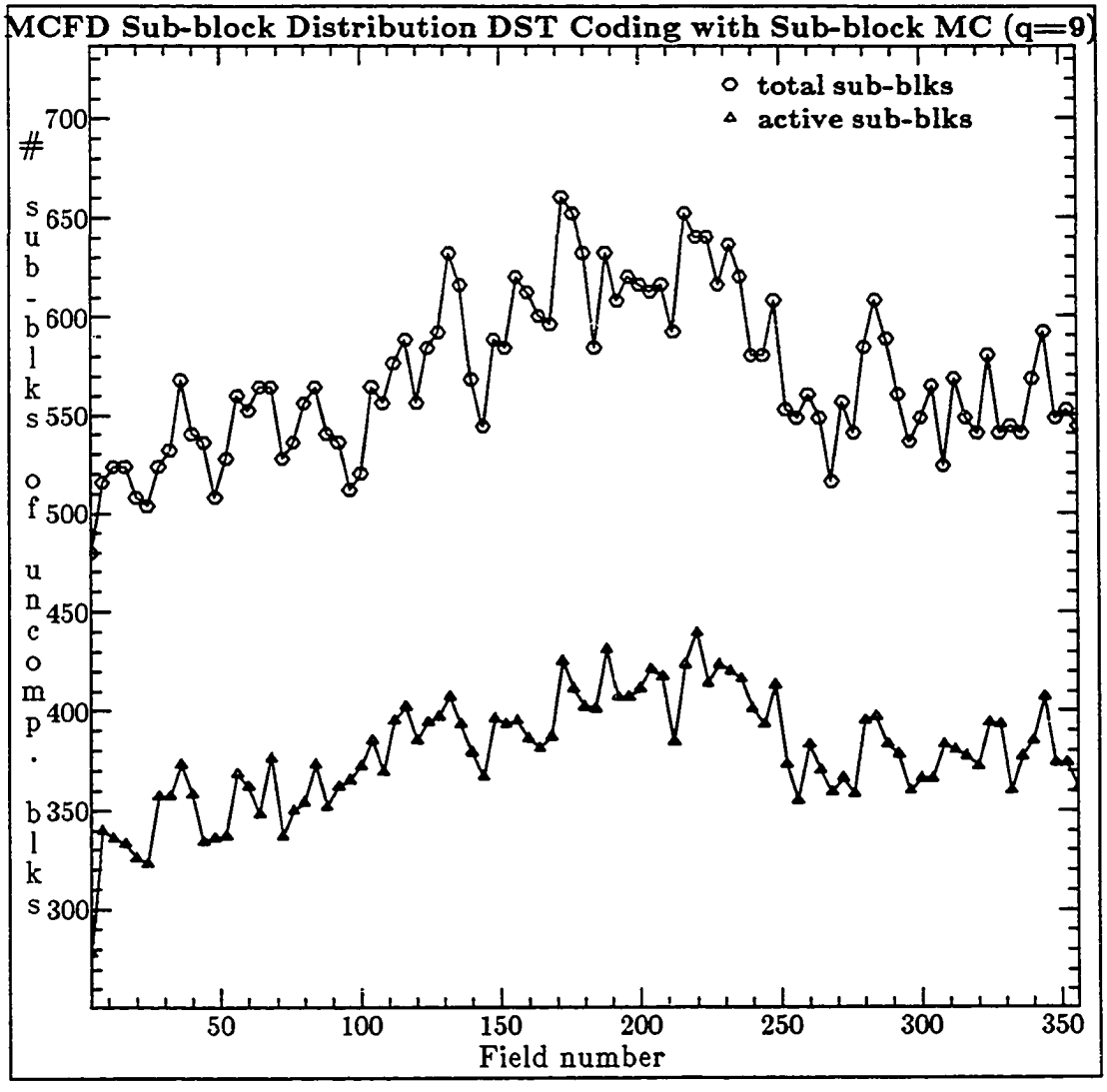


Figure 5.4(b) Frame Difference Sub-Block Distribution in DST Coding with Variable Block size Motion Compensation (quant. step 9): "MsUSA" sequence

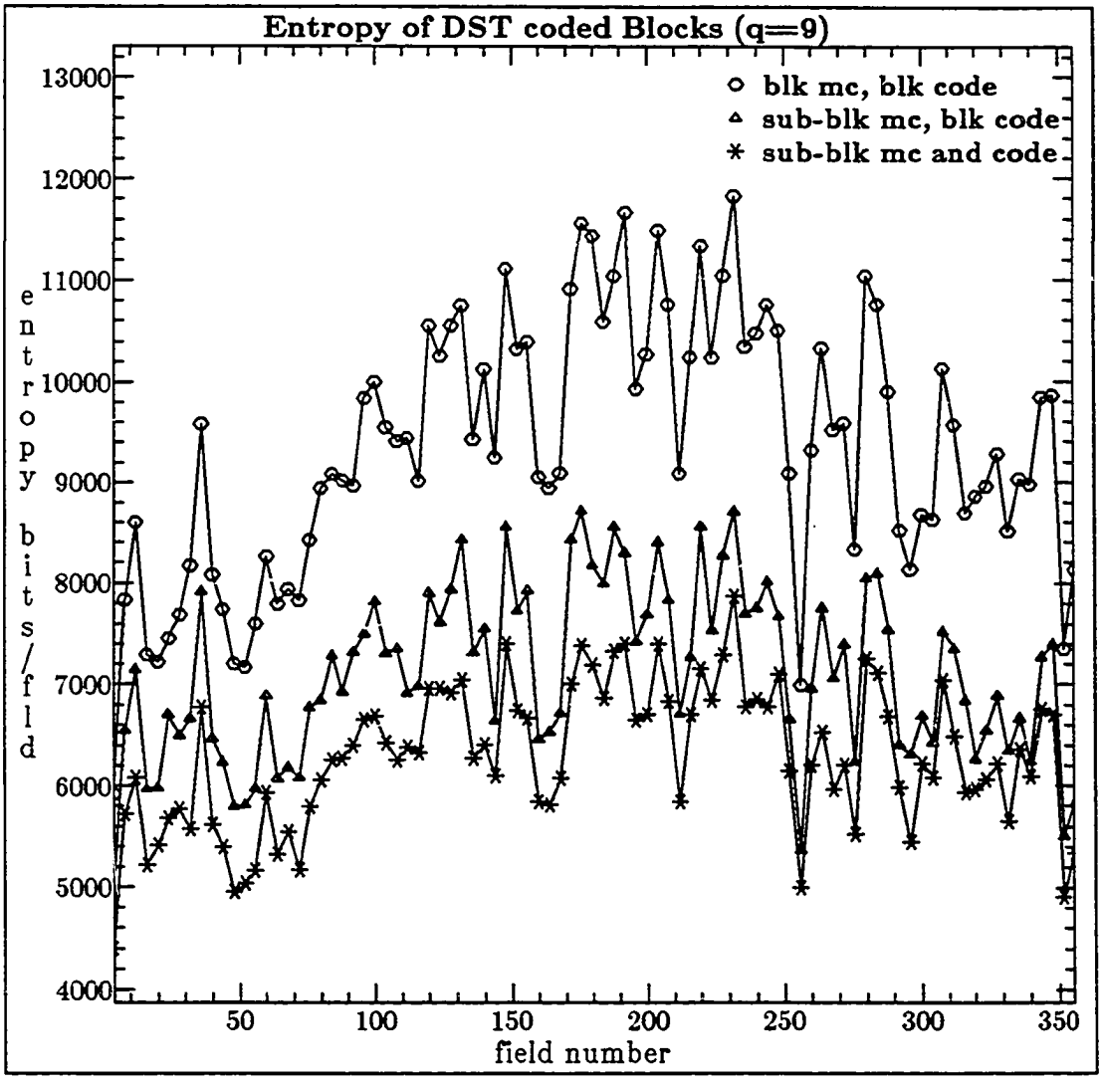


Figure 5.4(c) Entropy of DST-coded Blocks (quant step 9) : "MsUSA" sequence

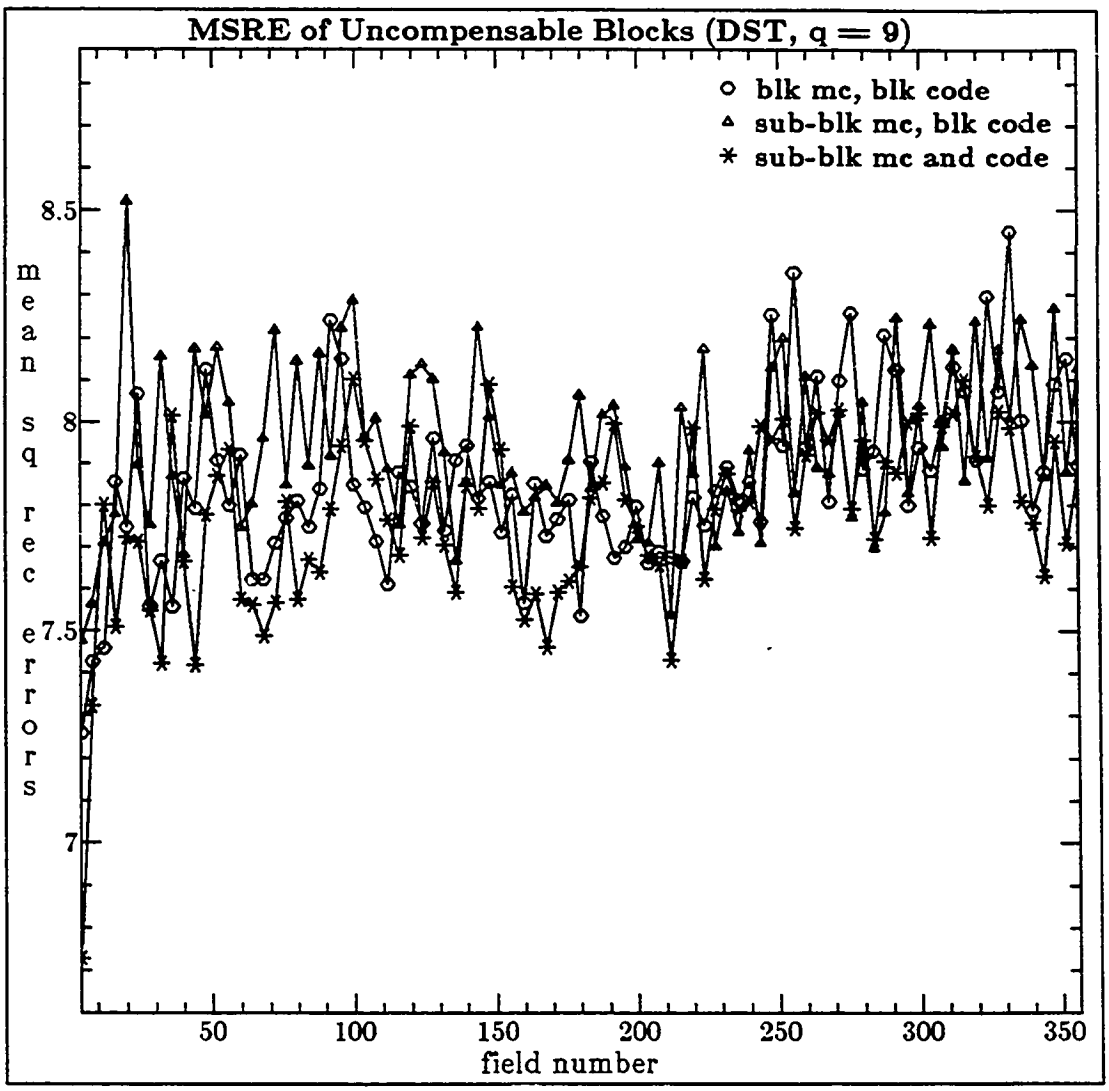


Figure 5.4(d) Mean Square Reconstruction Errors of Uncompensable blocks in DST Coding (quant. step 9) : "MsUSA" sequence

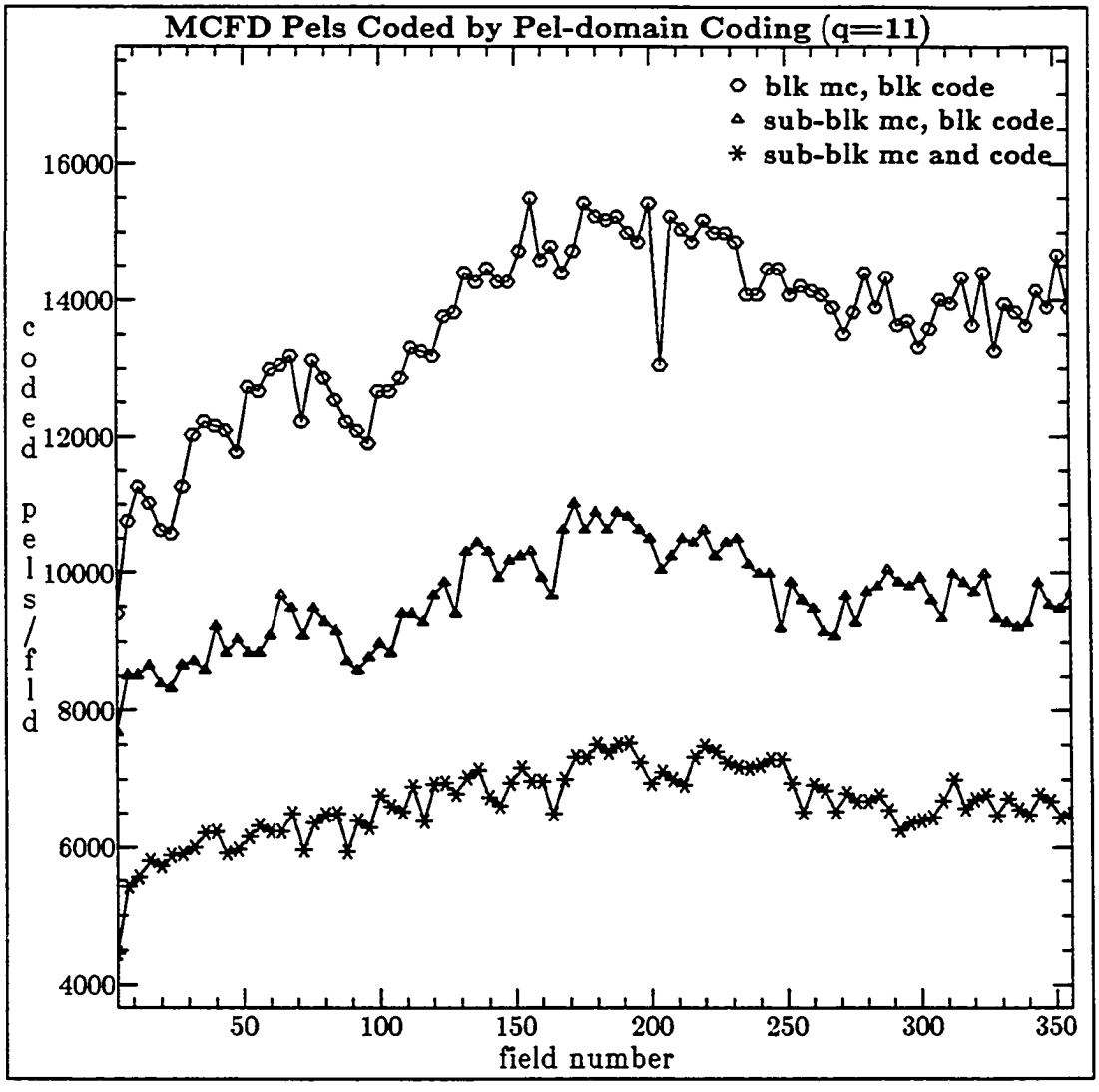


Figure 5.5(a) MCFD Pels Coded by Pel-domain Coding Algorithms
(quant. step 11) : "MsUSA" sequence

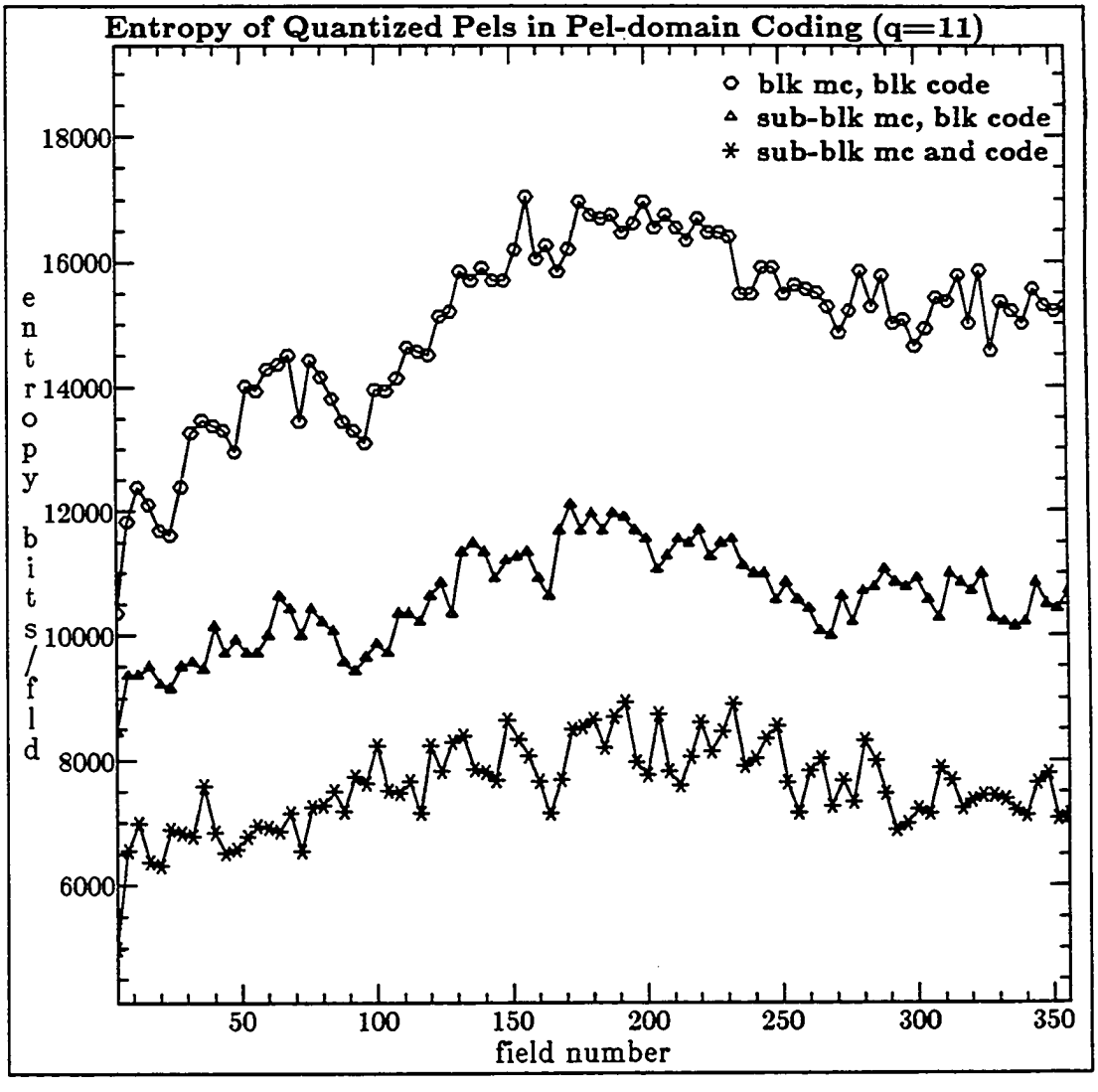


Figure 5.5(b) Entropy of Quantized Pels in Pel-domain Coding
(quant step 11) : "MsUSA" sequence

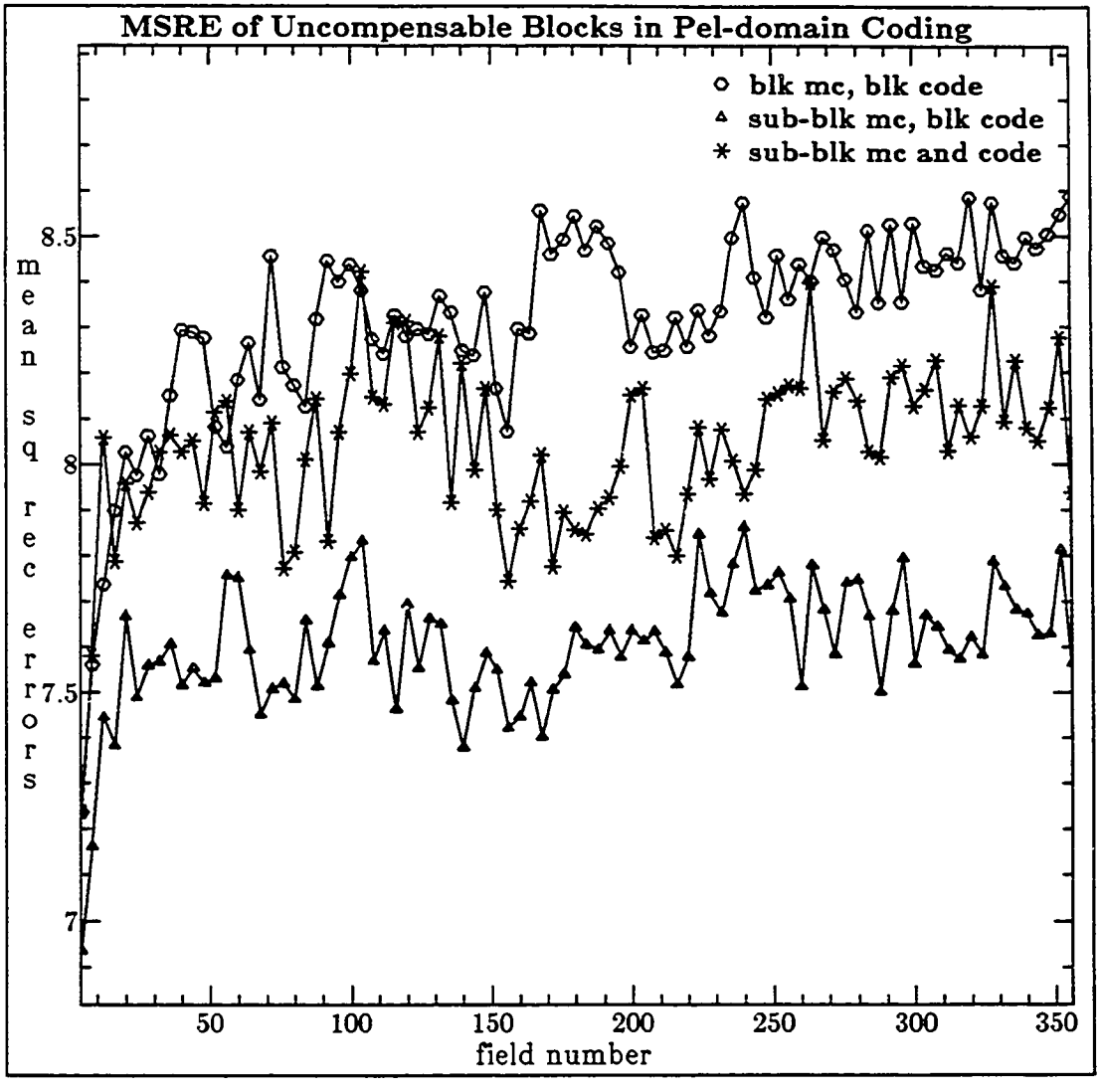


Figure 5.5(c) Mean Square Reconstruction Errors of Uncompensable blocks in Pel-domain Coding (quant. step 11) : "MsUSA" sequence

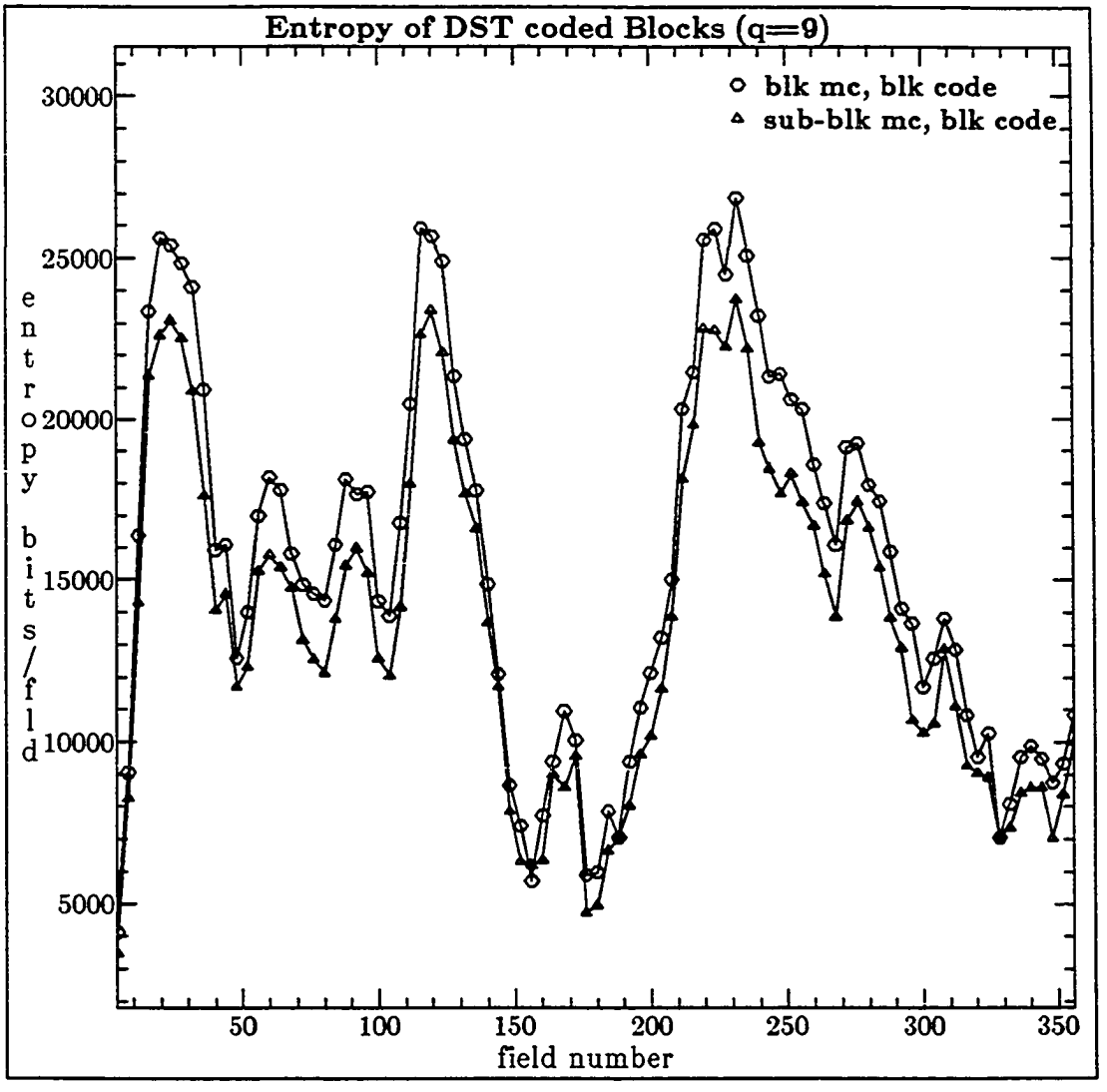


Figure 5.6(a) Entropy of DST-coded Blocks (quant step 9) : "Salesman" sequence

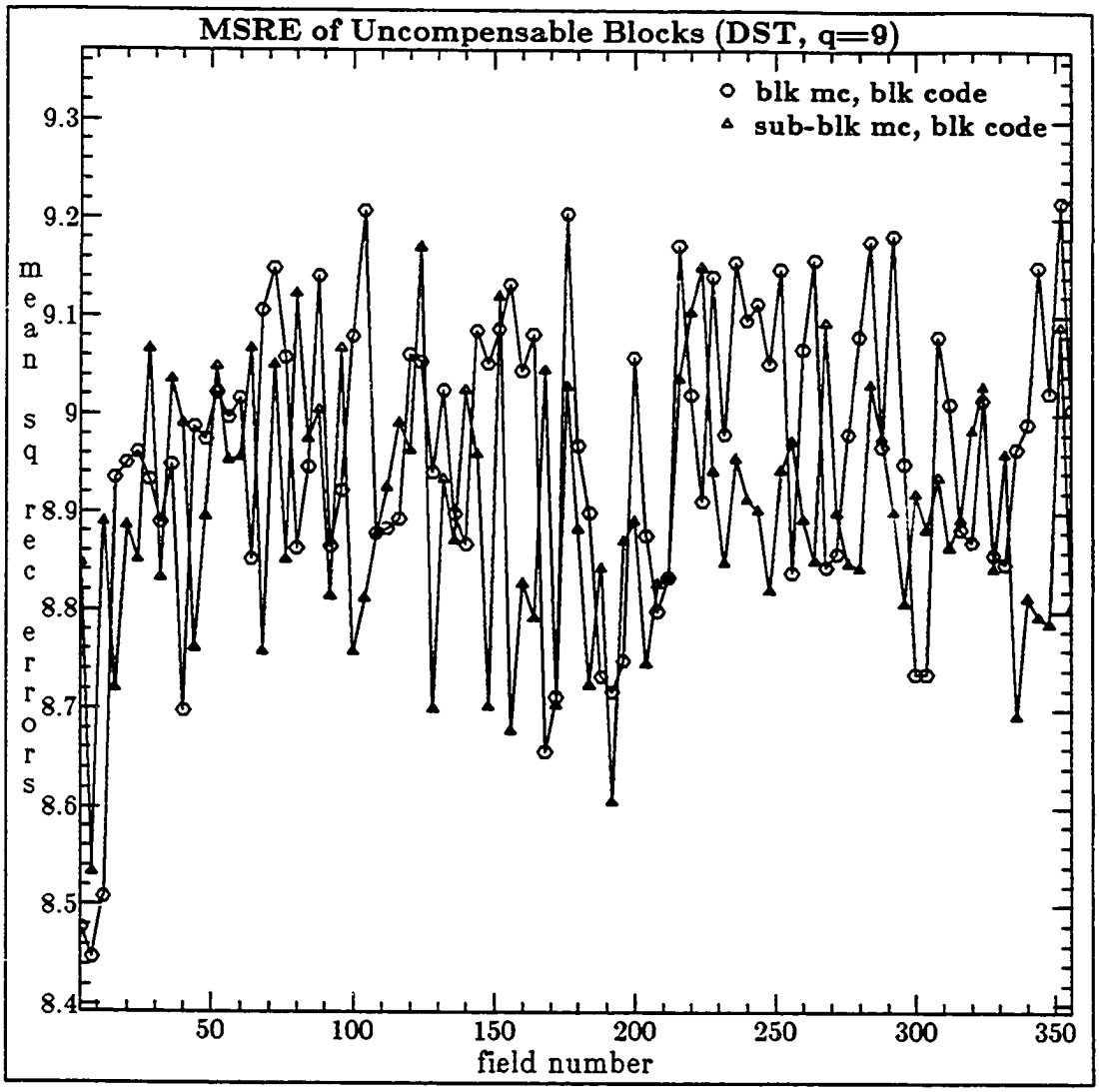


Figure 5.6(b) Mean Square Reconstruction Errors of Uncompensable blocks in DST Coding (quant. step 9) : "Salesman" sequence

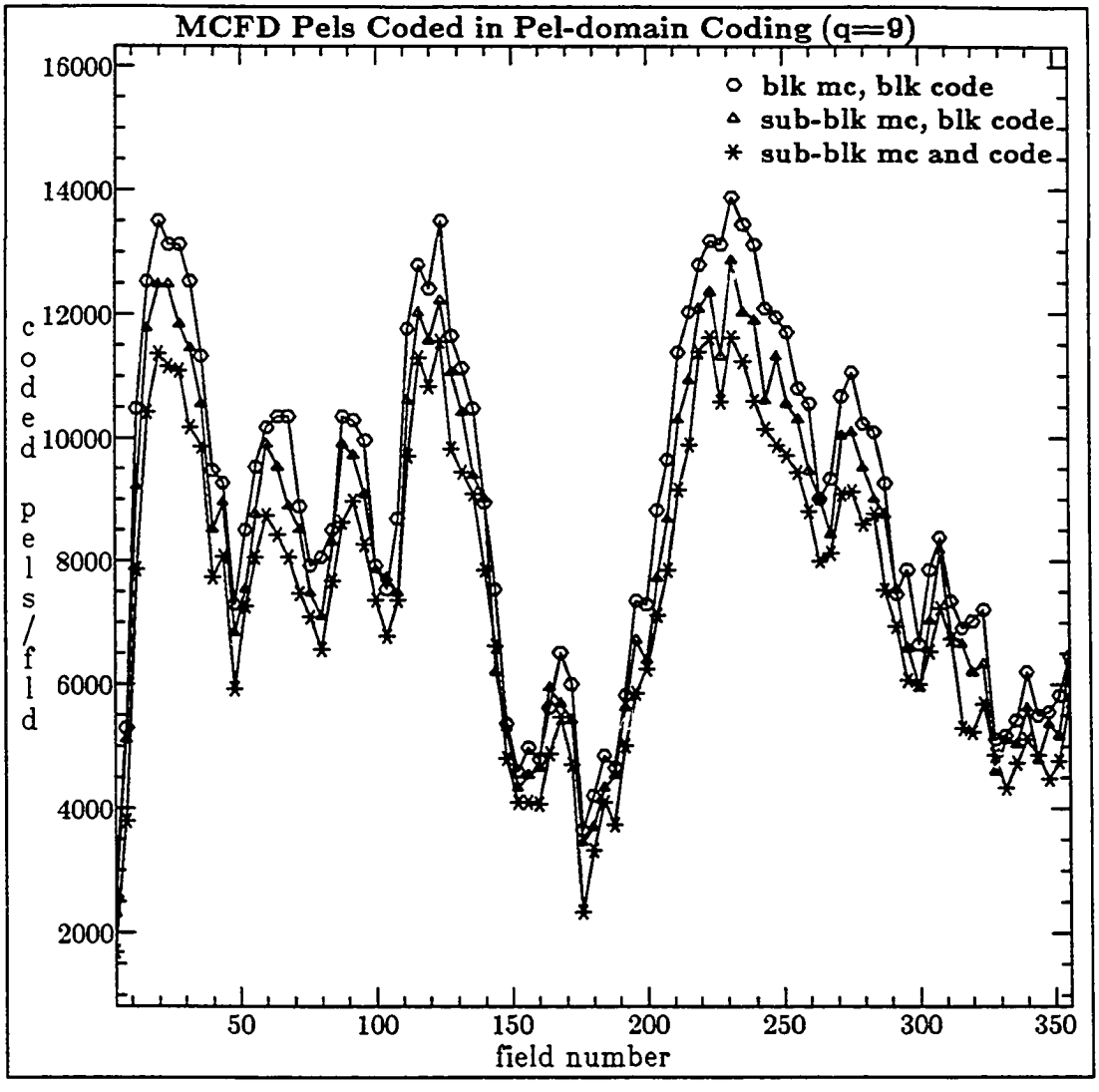


Figure 5.7(a) MCFD Pels Coded by Pel-domain Coding Algorithms
(quant. step 9) : "Salesman" sequence

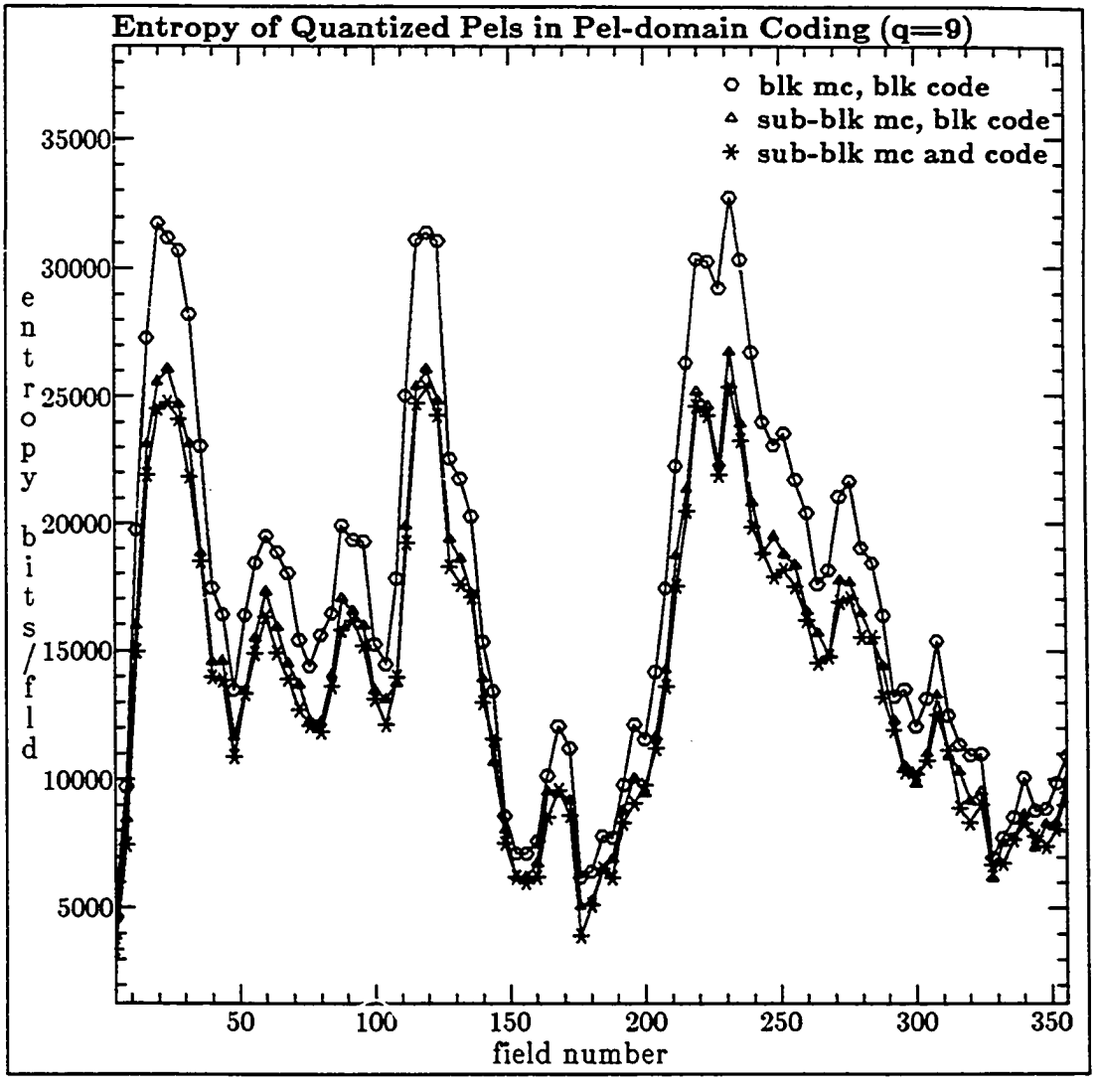


Figure 5.7(b) Entropy of Quantized Pels in Pel-domain Coding
(quant step 9) : "Salesman" sequence

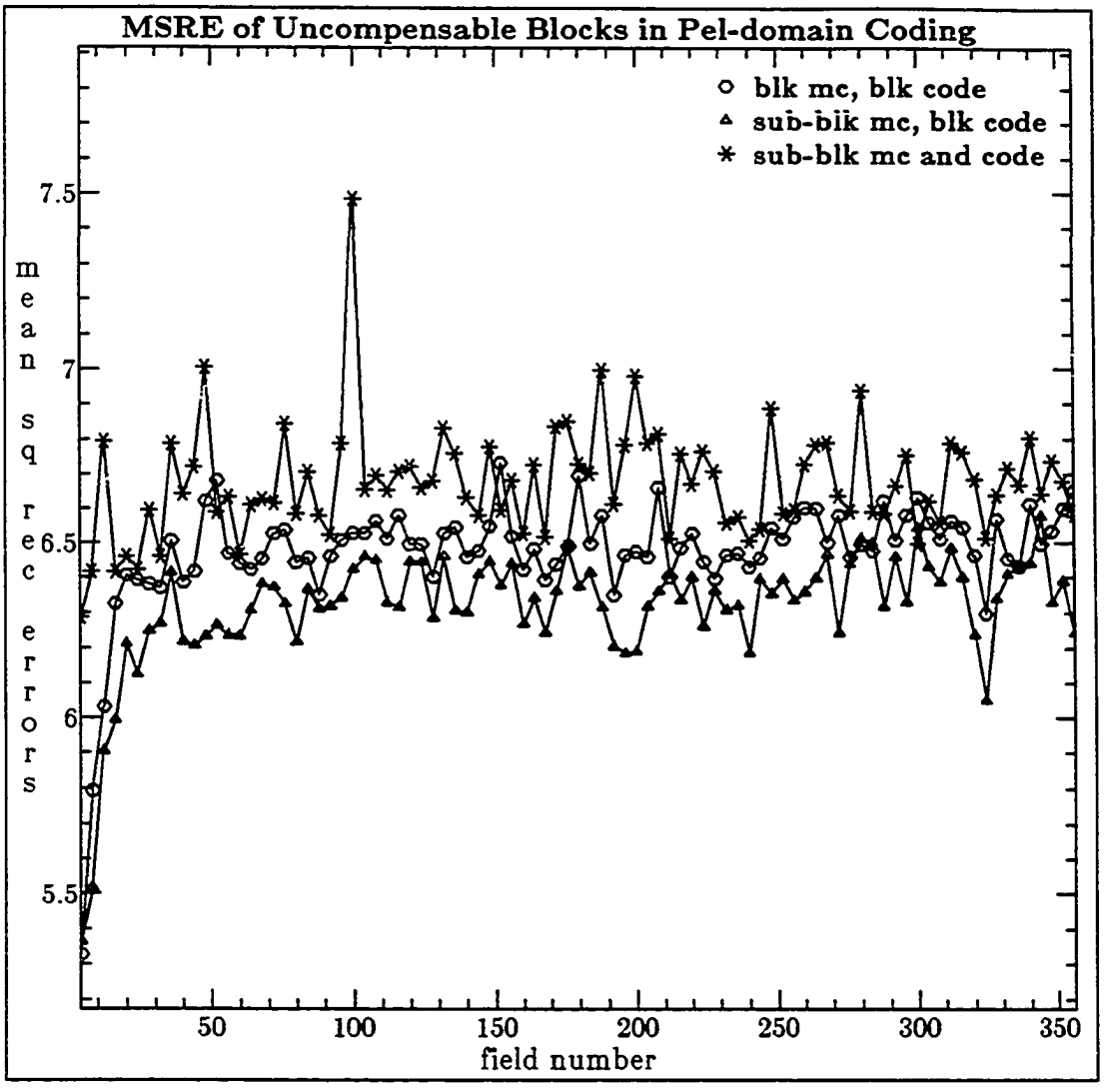


Figure 5.7(c) Mean Square Reconstruction Errors of Uncompensable blocks in Pel-domain Coding (quant. step 9) : "Salesman" sequence

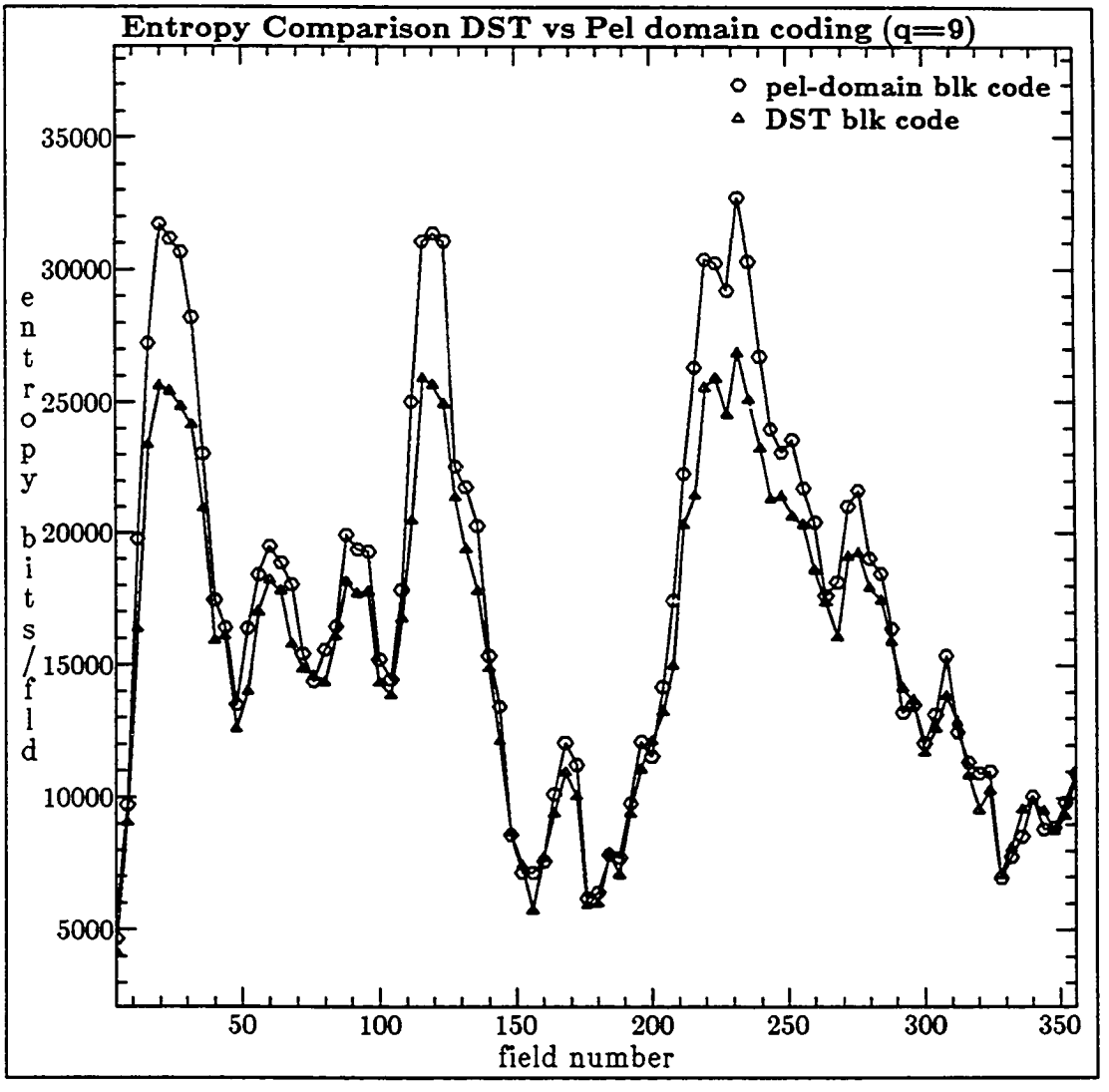


Figure 5.8(a) Entropy comparison for Pel-domain versus DST coding (quant step 9) : "Salesman" sequence

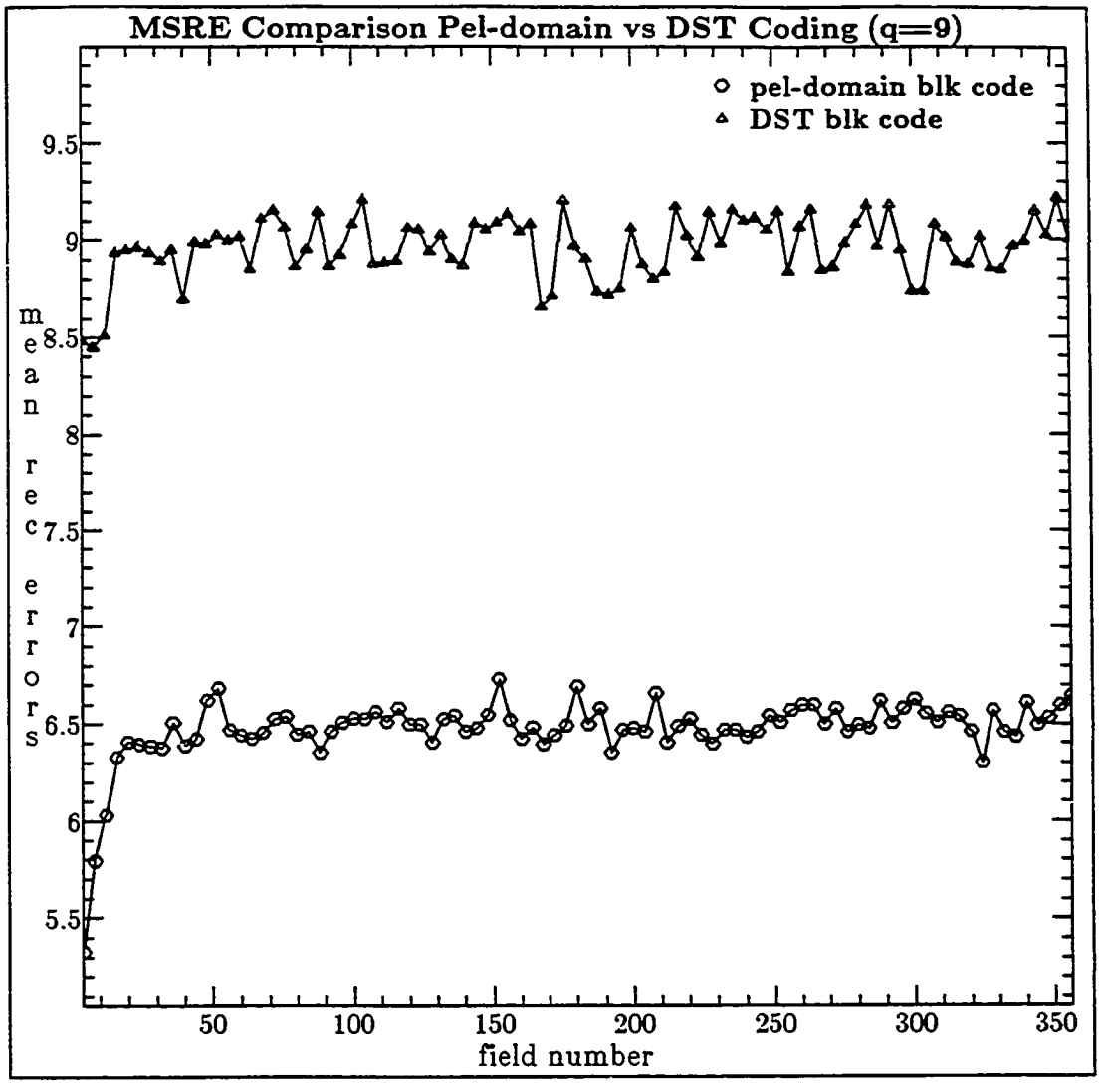


Figure 5.8(b) Mean Square Reconstruction Errors of Uncomp. blocks for Pel-domain and DST Coding (quant. step 9) : "Salesman" sequence

Figure 5.9(a) Low resolution original, and DST coded blk-size (MC 8*8, coding 8*8) q=9, images of sequence "MsUSA"

Figure 5.9(b) DST blk-size (MC VBS, coding 8*8) q=9, and DST blk-size (MC VBS, coding 4*4) q=9, images of sequence "MsUSA"

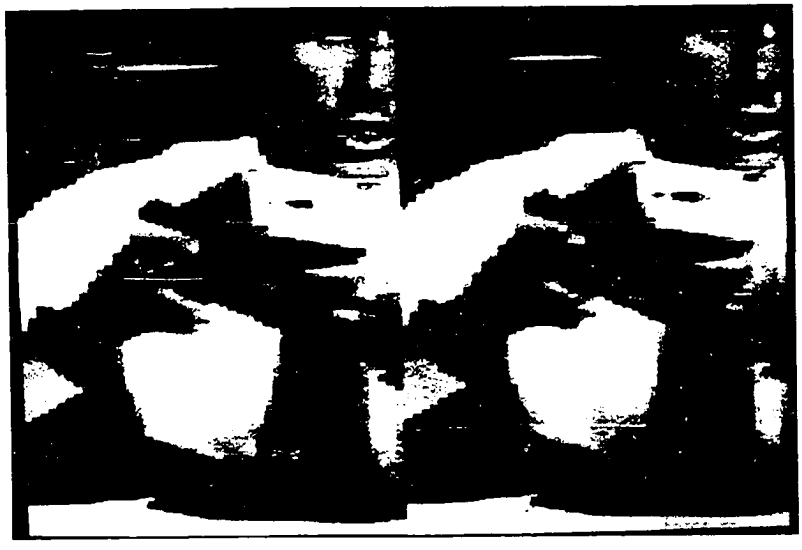


Figure 5.10(a) Low resolution original, and DST coded blk-size (MC 8*8, coding 8*8) q=9, images of sequence "Salesman"

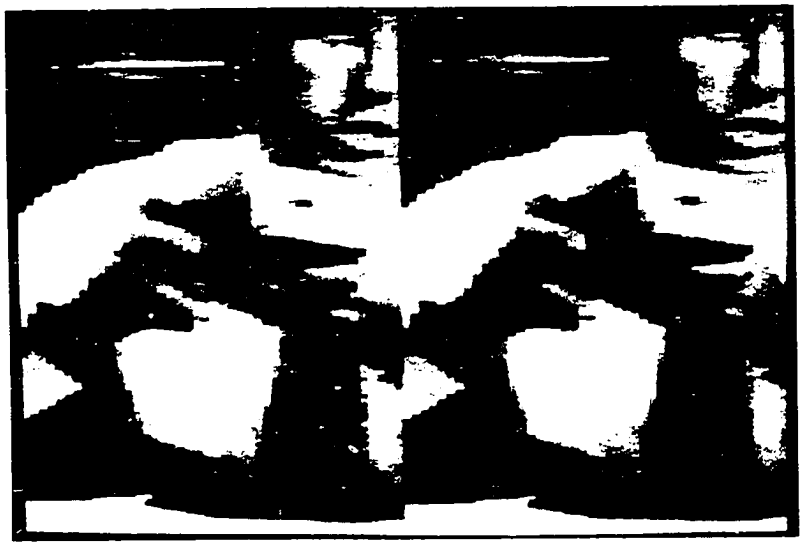


Figure 5.10(b) DST blk-size (MC VBS, coding 8*8) q=9, and DST blk-size (MC VBS, coding 4*4) q=9, images of sequence "Salesman"

CHAPTER 6: CLUSTER/TRANSFORM HYBRID CODING

6.1 INTRODUCTION

In the previous chapter we introduced *variable block-size* (VBS) scheme for motion-compensation and coding. This approach provides adaptivity and improves performance, by sub-dividing blocks in to sub-blocks, and spending extra effort to encode difficult sub-blocks. The performance of both transform-domain and pel-domain coding algorithms improve with the VBS technique, and in some cases the performance of pel-domain algorithms may approach that of transform-domain algorithms, for coding MCFD blocks. The pel-domain techniques can perhaps represent sharp transitions and edges in a scene efficiently, whereas transform-domain techniques are efficient for encoding slowly varying signal. A scheme combining both pel and transform-domain techniques can therefore be proposed, such that it retains the efficiency of transform-domain technique while providing a improved visual representation of sharp-transitions and edges. In this chapter we introduce such an approach, called the *Cluster/Transform* Hybrid-coding scheme. In Chapter 4, we briefly reviewed a hybrid coding algorithm [107] that combines DCT Transform coding with Scalar quantization of pulsive pels, of MCFD blocks. The study [107], is reported to be effective as it represents the highly pulsive pels of MCFD blocks better, and improves the visual quality of reconstructed image-sequence by reducing the so-called "mosquito" effects. Our *Cluster/Transform* coding scheme, while it has similar goals and agrees in principle that pulsive MCFD signal can be better represented in pel-domain while residual-signal can be transform coded, argues that in an MCFD-block, pulsive pels usually appear as clusters of small-size rather than as single-pels and should be dealt in this manner for better representation. The

reason for clustering of pulsive pels can be explained from the characteristics of motion and its compensation by block-matching. Whenever the motion of objects in a scene is not purely-translatory or causes background areas to be uncovered, block-matching motion-compensation approach causes only parts of blocks to get well-compensated. This is so, because only a few pels belonging to a current-frame block, may find a corresponding match in the displaced previous frame, causing groups of many, unmatched, significant-pels (or clusters) to appear.

6.2 MOTIVATION

In Chapters 3, 4 and 5, we point that, considerable differences may exist in the statistical nature of various MCFD blocks, and to exploit these differences we introduce the concepts of *Multi-transform* and *Variable Block-size* (VBS) coding. A way to classify MCFD blocks into various categories is also presented in Chapter 3, and consists of classifying one-step horizontal and vertical correlation-coefficients of the blocks into 3 categories. Other schemes [71],[108] coefficient power-distribution in the transform-domain as a way of classifying blocks to be transform-coded into various categories. In classifying MCFD blocks, such that they reflect the coding complexity, a criterion based on block-correlation coefficients and energy of block in spatial-domain is devised. The coding complexity of an MCFD block is dependent on both the correlations and block energy, for example, it is relatively easier to encode a block with high correlations and containing significant energy than a block with low-correlations and comparable energy, and the chosen criterion should discriminate between the two situations. For our purposes, a simpler criterion such as the VWL bits required in transform-coding of MCFD blocks suffices to classify the blocks, as it represents the actual coding complexity that a typical transform-coder (of Chapter 4)

faces in coding the MCFD block. Based on VWL coding bits, MCFD blocks are classified into three categories, blocks that require roughly, 1 bit/pel or less (*category 1*), 1-2 bit/pel (*category 2*), more than 2 bits/pel (*category 3*) to encode. This simplified, three category classification is effective in demonstrating the coding complexity of various blocks, as well as the maximum amount of expected improvement in overall performance, if blocks belonging to a specific category are encoded by an alternate coding scheme. The results of classification for "MsUSA" and "Salesman" sequences are shown in Figures 6.1 and 6.2.

In Figure 6.1 we see that out of all the uncompensable MCFD blocks that are transform coded, on the average, about 65 percent of blocks belong to *category 1*, about 28 percent belong to *category 2* and only about 7 percent belong to *category 3*. It implies that the sequence "MsUSA" contains only a few difficult blocks, most blocks can be coded reasonably well by spending only few bits per block and improvement in bit-rate performance as compared to transform coding would be difficult. Fig 6.2 shows the results of the classification for the "Salesman" sequence, of all the uncompensable MCFD blocks that are transform coded, on the average, only about 12 percent of blocks belong to *category 1*, about 40 percent of blocks belong to *category 2*, and about 48 percent of blocks belong to *category 3*. It implies that, this sequence, contains many difficult blocks and alternate coding schemes may be potentially more beneficial here. Therefore, in our proposed *hybrid scheme*, we wish to encode these difficult blocks more efficiently.

6.3 VECTOR, SCALAR, AND TRANSFORM CODING

We compare *vector, scalar* and *transform* coding techniques briefly, in order to understand where each technique is better suited, keeping coding of MCFD signal in

mind. In *vector quantization* (VQ) coding, pels are encoded as vectors (or blocks), and the coding process consists of performing a search through a pre-generated codebook to obtain the closest representation of a source-vector of pels, from the codebook. A statistical criterion, such as, mean square error (MSE) is computed between the source-vector and all the candidate vectors in the codebook, and minimum value of this criteria, indicates the closest representation (in statistical sense) of the source-vector, available from the codebook. An index specifying the location of representation-vector is sent to the receiver, and can be used to perform a lookup at the receiver, which has an identical codebook. Efficient coding is possible, as only the index, which takes only a few bits, is transmitted and can be used to obtain the representation-vector. The codebook used in encoding is generated before-hand and holds the key to the performance that can be expected from a VQ scheme. Some characterstics and limitations of VQ based scheme when used for MCFD coding, can be listed here.

— Images selected in training-set for generating code-book may influence performance.

— The match criteria such as MSE used in search, may not be adequate to provide good visual representation.

— For VQ to be efficient size of vector should be large, which in turn requires the codebook size to be large, to allow adequate representations.

— For large-size codebooks, search for finding closest representation may become computationally complex.

— Even though VQ with multiple-codebooks may improve the search complexity and visual performance, overhead involved for this scheme may be excessive.

— In coding sequence of images, the progressive effects due to mis-representation of vectors, because of limitations of matching criteria or lack of appropriate vectors in codebook, may be significant.

In *scalar-quantization* (SQ) coding, each pel is encoded independent of others by using a quantizer having fixed number of levels. This approach is generally less efficient, but if the neighboring pels are rather weakly-correlated, then SQ coding may be necessary for accurate representation. In this case, VQ based coding scheme, even though it does not require a high-correlation in adjacent pels of the source-vector, may not work well. The reason can be attributed to the limitations introduced by the codebook design procedures; A very large codebook that includes possible representation vectors (many combinations of pels) is required for coding the weakly-correlated MCFD signal.

In *transform* coding, a block of pels is converted via transformation into a block of coefficients, such that most of the energy of pels in the block is packed in to only a few low-frequency coefficients. These coefficients containing significant energy are quantized and transmitted while others are discarded. Some features can be stated here.

— Transform coding works well when input signal has high correlation, and only then significant amount of energy is packed into few low frequency coefficients.

— At low bit-rates significant amount of edge information may be lost due to discarding or coarse quantization of many coefficients.

— Adaptive transform coding schemes improve performance but also require overhead.

— In coding pulsive MCFD signal, significant coefficients are not usually concentrated in low frequency region (Chapter 4), but are often distributed over the entire block, requiring many bits.

6.4 CLUSTER CODING

The algorithms for image-compression that operate on more then 1 pel at-a-time are generally referred to as *block* or *vector* coding algorithms, typical examples are *VQ* and *transform* coding. The entire coding process can be thought of first, considering a group of pels as a single entity (called block or vector), and next, finding an appropriate representation for this entity. As discussed earlier, in *VQ*, this representation (quantization) is also a vector, where as in *transform* coding this is a scalar. In our *cluster* coding approach we also consider a group of pels (cluster) as a single entity, and the representation of each pels in the entity is scalar, even though some common information regarding the cluster can be extracted and used in coding. In terms of coding-complexity, *cluster* coding approach resembles *VQ*, i.e., encoder is far more complex then the decoder. We list some features of cluster-coding algorithm to facilitate its comparison with *VQ* and *transform* coding.

— Variable cluster sizes and shapes (patterns), small size clusters to encode very difficult pels, larger sizes otherwise.

— Variable placement of clusters, address of cluster location is sent as overhead.

— Search for grouping of pels into clusters may be complex, if many cluster sizes and shapes are allowed.

— Ease of retaining or discarding information, as number of clusters, cluster size and shape, magnitude of pels in cluster can be used.

In this scheme, adjacent pels do not have to be very similar (scalar nature), if they do have something in common they can be more efficiently coded (vector nature). We now formally discuss the details regarding clusters and their representation.

6.4.1 What is a Cluster?

We apply definitions obtained from region analysis techniques [109] to define a cluster. The region analysis techniques are used to group together the pixels in an image that share the same values of a feature. The regions produced by region segmentation typically have several properties.

(a) Such regions are mutually exclusive -- that is no pixel belongs to more than one region

(b) They are usually mutually exhaustive -- that is each pixel belongs to some region

(c) Each region consists of single cluster of contiguous pixels, it is simply connected.

(d) Each region satisfies some predicate, which usually indicates uniformity in the desired features.

(e) No two adjacent regions satisfy the same set of predicates -- that is, no two adjacent regions appear to be the same.

Region-segmentation techniques are attractive for a number of reasons. For one,

there are usually far fewer regions, so region segmentation is also a form of data-compression. Moreover, regions are group of pels with presumably the same semantic interpretation, so they are conventional units for later stages of image understanding. We apply above properties of regions, in a loose manner, to define a cluster. In short, clusters will be groups of connected pels, they are mutually exclusive, and have some common predicate such as they are all pulsive pels or all pels in cluster have the same sign, but may not be mutually exhaustive.

6.4.2 Cluster Sizes and Shapes

The sizes and shapes (patterns) selected may depend on the specific application that *cluster extraction* is being performed. In the present context of coding MCFD signal, formatted into 8×8 size blocks, statistical studies on blocks show that cluster of sizes with 3 to 7 pels offer reasonable compromise between quality of representation, bits for coding and addressing overhead. Clusters of 3 basic sizes, containing 3, 4 and 6 pels are chosen to form standard shapes, to be used in cluster-extraction. For 3 pel clusters shapes of 1×3 and 3×1, where as 4 pel clusters with shapes of 1×4, 2×2 and 4×1 pels, and 6 pel clusters with shapes of 2×3 and 3×2 pels are selected. Figure 6.3 shows all the seven standard cluster shapes used, these shapes can be located anywhere with in the MCFD block. A 3-bit overhead is sufficient to uniquely identify the shape being used. It is necessary to point out that the shapes used are found suitable for seeking "difficult" pels when block size of 8×8 is used. If the blocks are larger in size or/and statistics of "difficult" pels require larger clusters, more pels can be used.

6.4.3 Extracting Clusters

With the cluster sizes and shapes selected, we propose to extract clusters from blocks. Again, here we assume, MCFD blocks of size 8×8 are available and contain "difficult" pels in various shapes and sizes. The *cluster-extraction* algorithm must try to obtain an overall solution regarding cluster shapes and locations. Some restrictions can be placed, for example in coding MCFD blocks we may want to include only the "difficult" pels in the clusters, and the maximum number of clusters allowed per block is 4. This is decided by estimating the number of bits that can probably be spent for encoding difficult pels in the block. Usually we expect difficult MCFD blocks to contain more clusters, then other MCFD blocks. For any block, if number of clusters extracted exceeds 4, then the most significant clusters, based on size, shape and other constraints can be retained, and others discarded.

An MCFD block is scanned to find significant pels, a pel is called significant if its magnitude is greater than 10. These significant pels are grouped together, by placing a cluster shape on it and determining whether the number of significant pels in a cluster agree with pre-decided, minimum number of significant pels required for that shape. Many candidates, for significant clusters of a block can be extracted by repeating the process in different parts of the block, always taking into account past decisions made. To find the optimum-solution for clusters, given the selected shapes a binary-tree search structure of Fig 6.4 is followed. Large clusters (with size 6) are extracted first, since there are two possible shapes, two initial solutions corresponding to the left and right branches of the binary-tree is obtained. At the next level, search is continued with shapes of 1×4 and 4×1 pels, and is followed by search with shape of 2×2 pels. At this point, in the tree 4 partial solutions for cluster search through the

block, are available. At the final level in the tree, search progresses with cluster shapes of size 1×3 and 3*times*1, resulting in a total of 8 independent-solutions per block. Even though 8 solutions are obtained, they may not necessarily be unique, and so we discard the repeated solutions and thus retain only the unique ones. Among the set of unique solutions, some may contain more then 4 clusters. Energy in each such cluster, and for every solution is computed, and clusters ordered based on energy. The clusters with relatively little energy are discarded at the onsets. If for any solution the number of clusters is still larger then 4, then the 4 most significant clusters are retained and rest discarded. The next step consists of selecting the best solution for cluster placement from the set of remaining solutions. Average energy per cluster in a solution, and a measure of coding difficulty of cluster are used to come up with the best solution for cluster placement for the MCFD block. A 1 bit overhead per MCFD block indicates prescence/absence of clusters. A 2 bit overhead per MCFD block, having clusters, that can be represented by cluster coding is required for specifying the number of clusters. We discuss the representation of clusters extracted, next.

6.4.4 Representing Clusters

To represent a cluster that is placed anywhere within the MCFD block, both addresses and quantized values of pels are necessary. Since "difficult pels" are grouped into clusters of one of the standard shapes an address per cluster is required. For every cluster, we specify the address of the top left hand corner pel belonging to the cluster located in a 8×8 MCFD block. This, together with the shape of the cluster can identify pels of a cluster, belonging to the MCFD block. For addressing purposes, 6 bits per cluster are sufficient to uniquely identify the location of the cluster in the 8×8 block. In reality, this overhead can be reduced further, for example, the cluster

location scan be defined with respect to each other, resulting in some savings in addressing bits. For this study, possible ways of minimizing the overhead are not investigated.

For quantizing the clusters a basic three bit quantizer with output levels of ± 5, ± 14, ± 25, and ± 40 is designed. The quantizer design was based on the statistics of differential signals [26], and characteristics of MCFD signals. For every cluster the sign of pels was noted. If the sign of all pels is same, a "mode" bits is *set*, which signifies that only 2 bits will be required to encode each pel in the cluster. The "sign" bit decides whether the positive or negative half of the quantizer is to be used. For larger size clusters if one pel was of different size then others, its magnitude is inspected, if small the cluster is coded with 2 bits/pel. If the "mode" bit is *clear* 3 bits/pel are required, and shows that MCFD pels with significant magnitudes but opposite signs appear within the same clusters, and the sign bit is disregarded. Even though the significant pel threshold (absolute value) is 10, the smallest level (absolute value) allowed is 5; it is possible that, in a 6 pel cluster only 4 or 5 pels have absolute values larger then the threshold, rest of them can be encoded with the smallest available level. Occasionally a significant error in representation of single pel in a cluster may occur.

The smallest cluster size is defined to be 3 pels, causing isolated single significant pels to be ignored in cluster formation. Such pels may also occur because of other reasons. In forming clusters, due to absence of all possible shapes of given size, in which significant pels can appear, "extra" pels may be left out. Significant single pels can also result due to errors in quantization of large clusters. Various "leftover" or "extra" pels discussed above will be referred to as *scalar* pels. In our scheme, for every block

that is cluster-coded, up to 2 such pels are retained and quantized by the 3 bit/pel quantizer. From the original MCFD block, the quantized cluster representations are subtracted out to obtain *residual* MCFD block. This residual block is scanned and significant isolated pels detected. A 1 bit indicator if *set* shows prescence of any such pels. If such pels are found, another 1 bit indicator is used to show whether 1 or 2 such pels have been allowed. Each *scalar* pel is represented by 6 bit address and 3 bit quantized value. The represented *scalar* pels are subtracted from the *residual* MCFD block.

6.5 A CLUSTER/TRANSFORM HYBRID CODING ALGORITHM

We combine cluster-coding concepts presented so far, along with the basic motion-compensated transform coding(of Chapter 4), to develop a hybrid-coding algorithm. A complete frame-difference (FD) block classification diagram, including the blocks to be *cluster* coded is shown in Figure 6.5. A motion detector first classifies FD blocks into nonmoving and moving blocks. The moving blocks are compensated, tested and classified into compensable and uncompensable blocks, as before (Chapters 4 and 5). The uncompensable MCFD blocks are tested for prescence of significant clusters. If such clusters are not present, the 8×8 block is transform coded. If significant clusters are found the process of cluster extraction, and cluster representation are followed. The cluster representations are differenced from the original MCFD block, to obtain the *residual* block. On the residual block either a single transform with block-size 4×4 or that of size 8×8 is applied, based on the characteristics of the residual block. For instance, if a single 4×4 area in MCFD block containing over 75 percent of the energy of residual block exists, than a 4×4 transform is applied, and 5 bits are sent to identify the location of 4×4 area in 8×8 block. If no such area can be found, transform

of size 8×8 is selected. The transform coefficients are scanned, selected, quantized and transmitted following algorithm similar to the one discussed in Chapter 4.

In Figure 6.6 we present the detailed flow-diagram of the scheme used for *cluster/transform* hybrid coding, that can be explained by the following algorithm.

Step 1: Check uncompensable block for prescence of *significant* clusters. If no clusters exist go to Step 11.

Step 2: Use a search procedure to identify valid clusters. Find the number, type and locations of valid clusters.

Step 3: For each cluster check if sign of all significant pels in the clusters is identical. If so set mode bit and go to Step 4.1, else reset mode-bit, and go to Step 4.2.

Step 4.1: Sign bit of cluster indicates if positive or negative half of the 3 bit/pel quantizer are used. Thus cluster-pels are quantized using only 2 bits/pel. Go to Step 5.

Step 4.2: Quantize cluster-pels with a 3 bits/pel quantizer.

Step 5: Determine if there are any *significant* isolated pels, that need to be represented as scalar-pels. If not, go to Step 7.

Step 6: Quantize scalar-pels by a 3 bit/pel quantizer.

Step 7: Detect if *residual* MCFD block after cluster and scalar-pel removal can be coded by a single 4×4 transform. If not, go to Step 9.

Step 8: Locate a significant 4×4 area in the 8×8 block. Apply 4×4 transform to this area. Scan, select, quantize, and transmit coefficients. Inverse transform to obtain pel-domain representation of 4×4 area of MCFD block. Go to Step 10.

Step 9: Use transform size 8×8 to encode the *residual* MCFD block. Scan, select, quantize, and transmit coefficients. Inverse transform to obtain *residual* MCFD block in pel-domain.

Step 10: Reconstruct the MCFD block by combining *clusters*, *scalar pels* and *transform* representations. Go to Step 12.

Step 11: Use transform size 8×8 to encode the MCFD block. Scan, select, quantize, and transmit coefficients. Inverse transform to obtain MCFD block represented, in pel-domain.

Step 12: Reconstruct the pixel-block by combining the represented MCFD block with the prediction block.

6.6 PERFORMANCE

We have explained the cluster coding and stated an algorithm for coding MCFD blocks, that combines the *cluster* and *transform* coding techniques. We now, apply this algorithm on image sequences "MsUSA" and "Salesman" to evaluate its performance. Some statistical performance results shown in Figures 6.7 and 6.8, consist of VWL coding bits and MSRE's, and are compared for the *hybrid* and *DCT* only coding algorithms. For the hybrid coding, the VWL bits in Figures 6.7(a) and 6.8(a) include all overhead bits required in the *cluster* coding and selection of

cluster/transform mode. The overhead involved in *cluster* coding part has not been optimized for this study. Both for "MsUSA" and "Salesman" sequences, hybrid coding technique represents improvements in coding bits as compared to the DCT only scheme, even though the differences are small. The parameters for cluster part of the hybrid coding algorithm need to be optimized to improve the performance. In terms of MSRE of uncompensable MCFD blocks, hybrid coding does not seem to perform as well as the DCT only scheme. The difference in MSRE performance is small for "MsUSA" sequence, but is larger for the "Salesman" sequence. Thus cluster coding followed by block-size selectable transform coding contributes to the increased MSRE's. Earlier, in comparing Figures 6.1 and 6.2, we showed that MCFD signal of "Salesman" sequence contains many more difficult blocks than that of "MsUSA" sequence. Block-statistics in hybrid coding of "MsUSA" show that, on the average only about 10 percent of all uncompensable MCFD blocks are coded by the cluster coding part, whereas similar statistics for "Salesman" sequence show that about 40 percent of the uncompensable MCFD blocks are coded by the cluster-coding part.

The coded images of "MsUSA" comparing DCT transform coding and hybrid coding are shown in Figures 6.9(a) and 6.9(b), similiar comparisons for the "Salesman" sequence are made in Figures 6.10(a) and 6.10(b). Single images such as these are insufficient to show details in picture quality obtained by various coding schemes, and to allow any comparisons, they are only presented to show general picture quality. On comparing the visual quality of coded image-sequences, we note that both for "MsUSA" and for "Salesman" sequences the hybrid coding technique results in a superior performance as compared to DCT only coding scheme. The improvements in coded image-sequence quality for "MsUSA" sequence are small and observable only on close inspection of details such as facial expressions. The improvements for the

"Salesman" sequence are more obvious and can be noticed in the reproduction of sharp transitions and edges in the scene, such as shoulders, chair, box etc.

6.7 DISCUSSION

In this chapter we have introduced the *cluster* coding approach, and shown its application in *hybrid* coding scheme for encoding MCFD signal. In *cluster* coding the so-called "difficult" pels of a MCFD block are represented as clusters. Some standard shapes for clusters are defined, and a *search* algorithm is followed, to arrive at a several solutions. The solution, best from coding point of view is selected, and clusters are quantized. The residual signal obtained after removal of quantized-clusters from the MCFD-block is transform coded. For transform-coding a block-size of 8×8 or 4×4 can be selected. The coefficients are scanned, quantized and transmitted. The MCFD signal is reconstructed by recombining the quantized cluster and transform representations.

In the statistical sense, improvement in *hybrid* coding as compared to the DCT coding is small, both for "MsUSA" and the "Salesman" sequence; the hybrid coding algorithm can perhaps be optimized to improve its statistical performance. In comparing the quality of coded image sequences, we observe that for both sequences *hybrid* coding algorithm outperforms the DCT coding algorithm perceptually. The improvement is relatively small for "MsUSA" sequence, but is more noticeable for "Salesman" sequence. The *hybrid* coding algorithm seems well suited for coding MCFD's as it automatically categorizes the information in a MCFD block in order of importance. Also, the overall bit-rate can be controlled with relative ease as many choices for retaining the clusters or/and transform-domain information exist, for e.g., if bit-rate is required to be cut-down, only the large-clusters and some transform-domain

information can be sent; further, the transform size can also be selected if necessary. This is in sharp contrast to simple transform-domain (DCT only) schemes, in which quantization step-size is the only parameter that is often controlled to cut down on the bit-rate, resulting in degradation in visual appearance of the coded image-sequence. Thus, *hybrid* coding, even though more complex than the simple transform coding schemes, appears to offer a sensible way to control bit-rate/visual performance of a image-sequence coding-system.

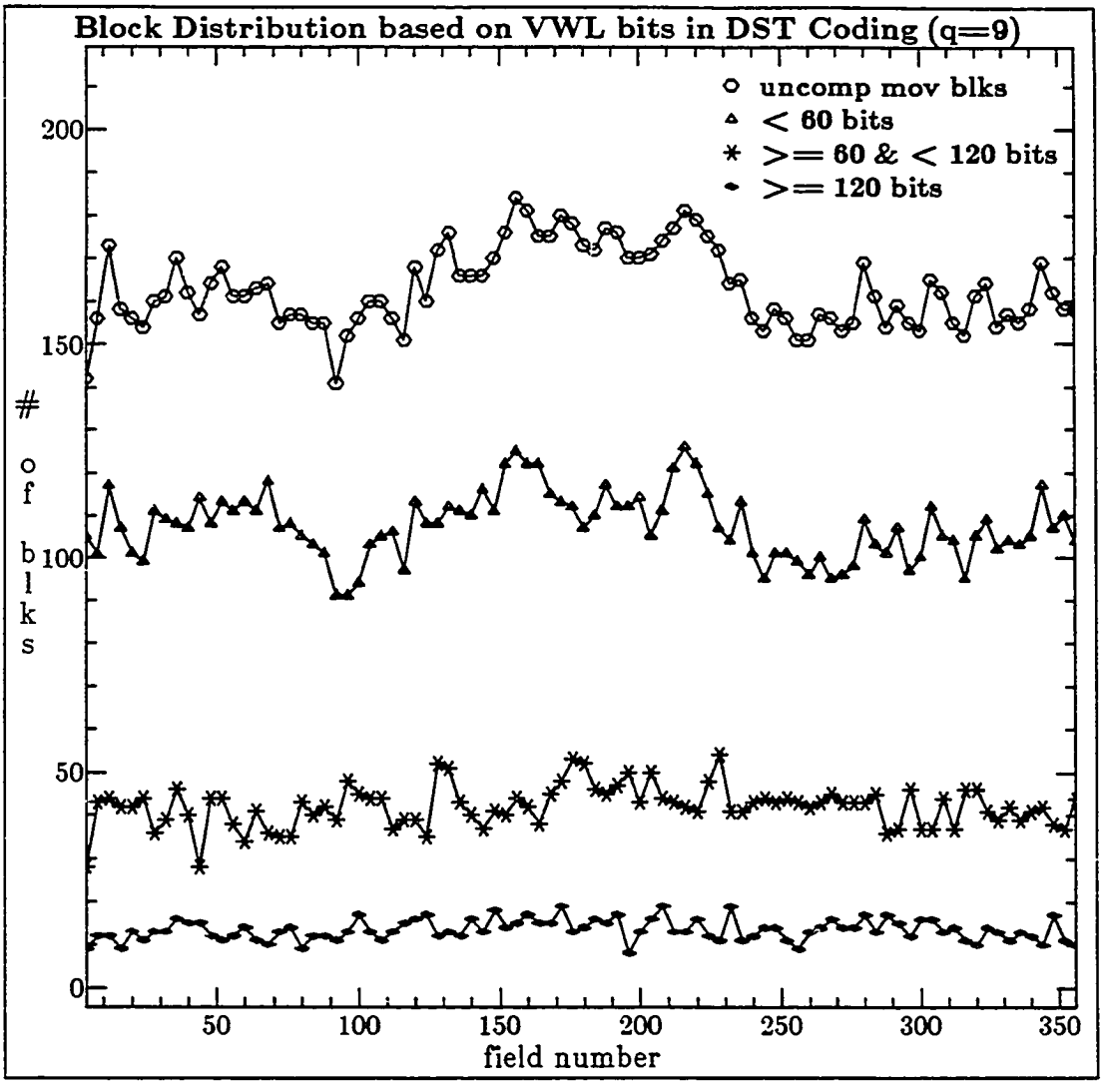


Figure 6.1 Block Distribution based on VWL bits in DST coding
(quant. step = 9) : "MsUSA" sequence

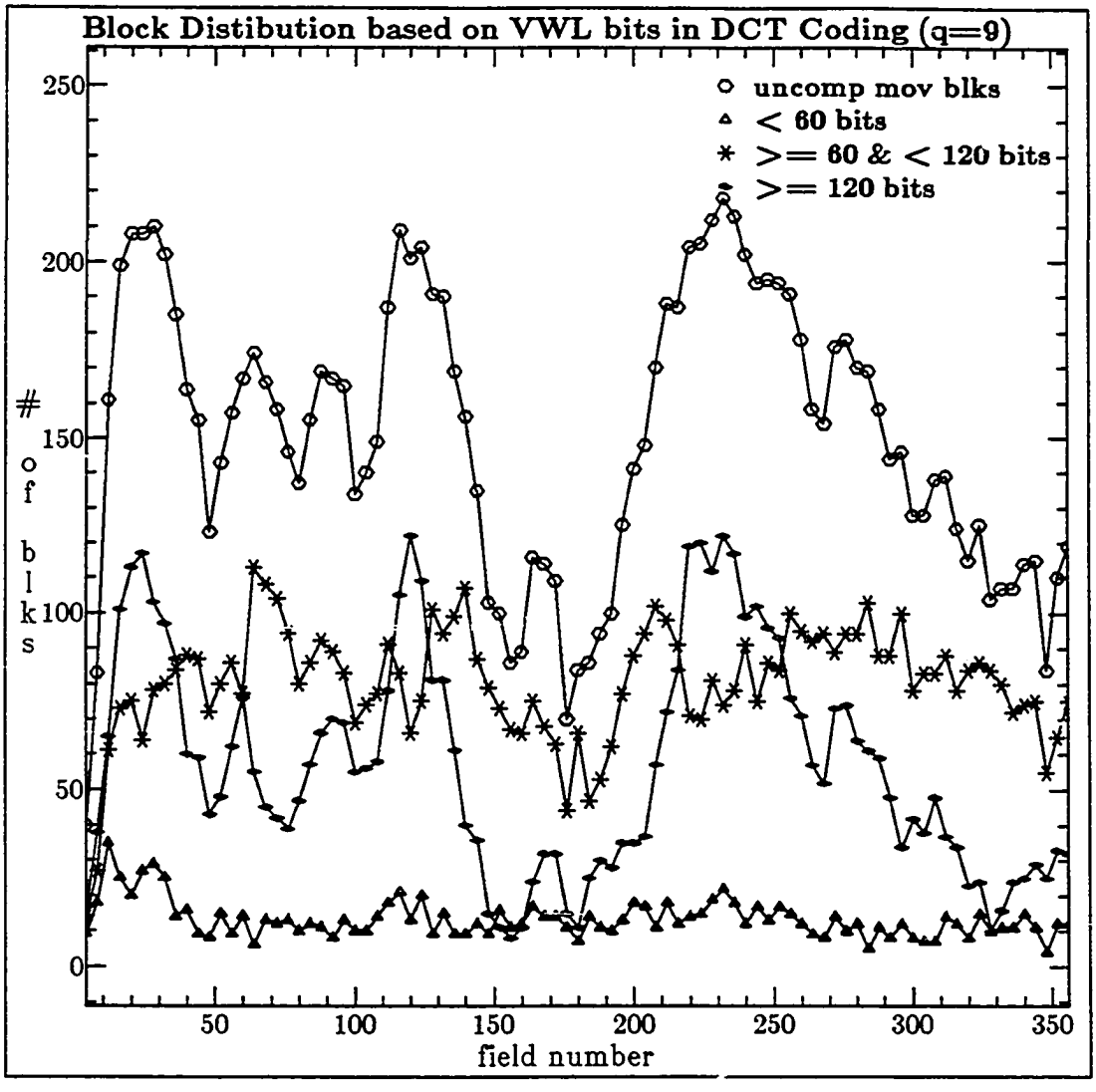


Figure 6.2 Block Distribution based on VWL bits in DCT coding (quant. step = 9) : "Salesman" sequence

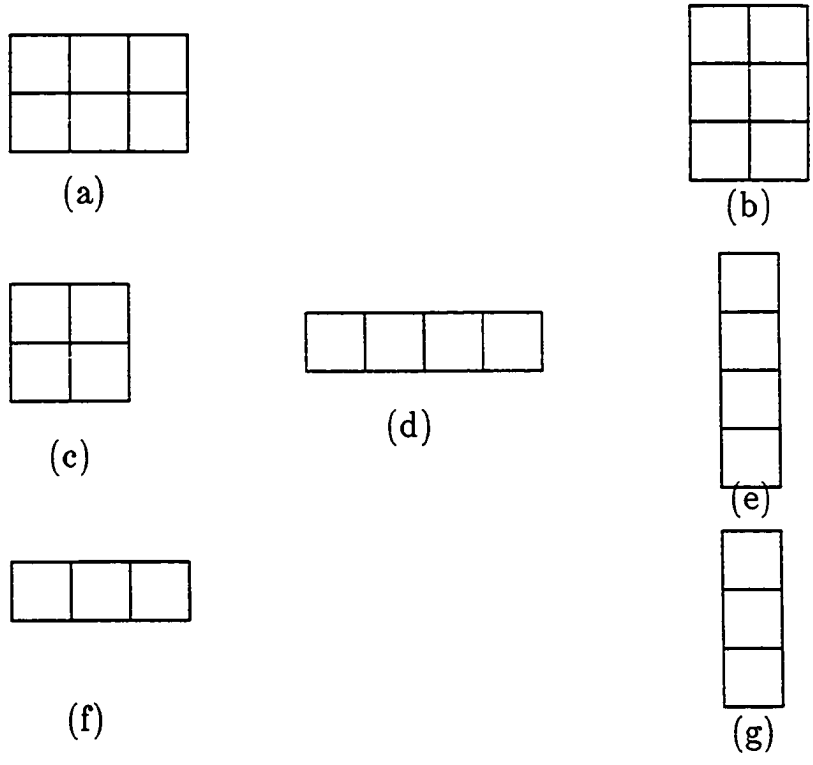


Figure 6.3 Standard Shapes used in Cluster Extraction

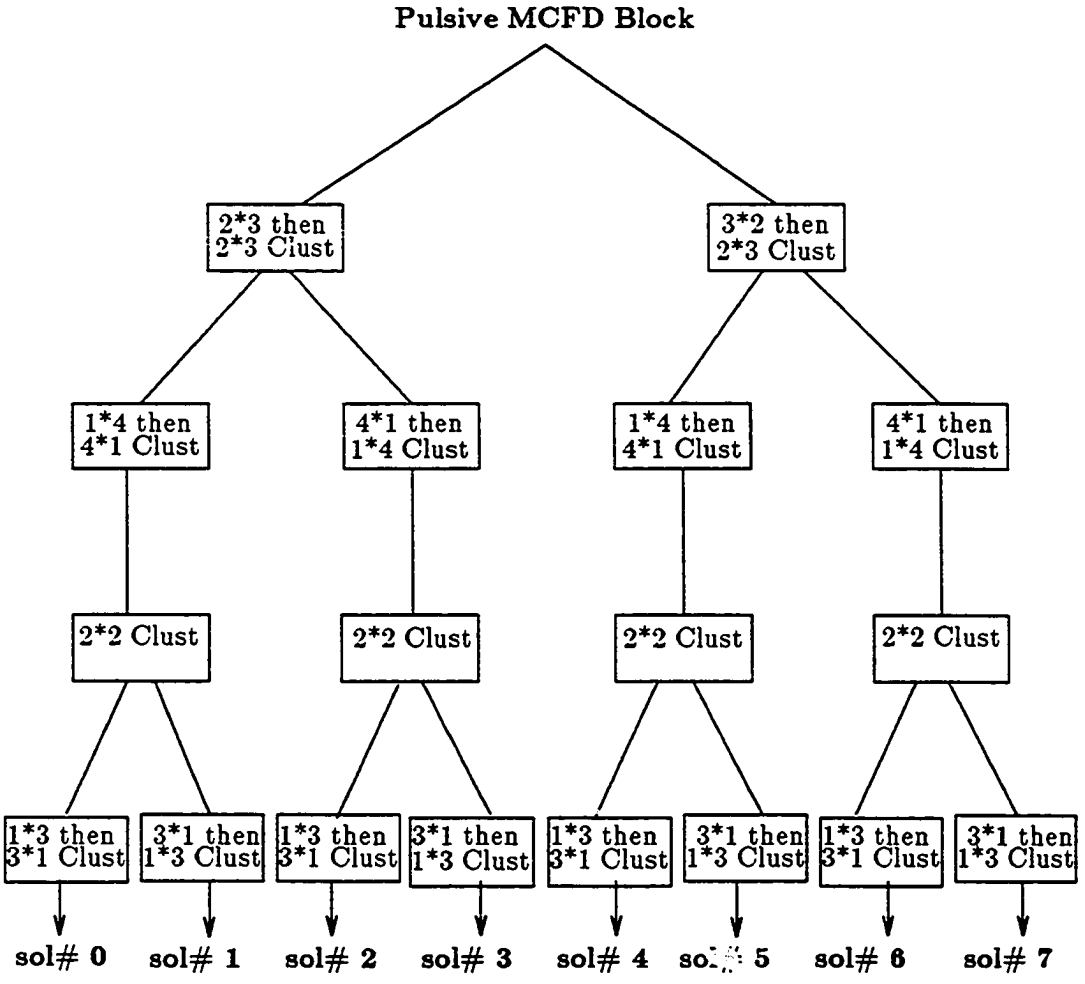


Figure 6.4 Cluster search order and candidates for optimum solution

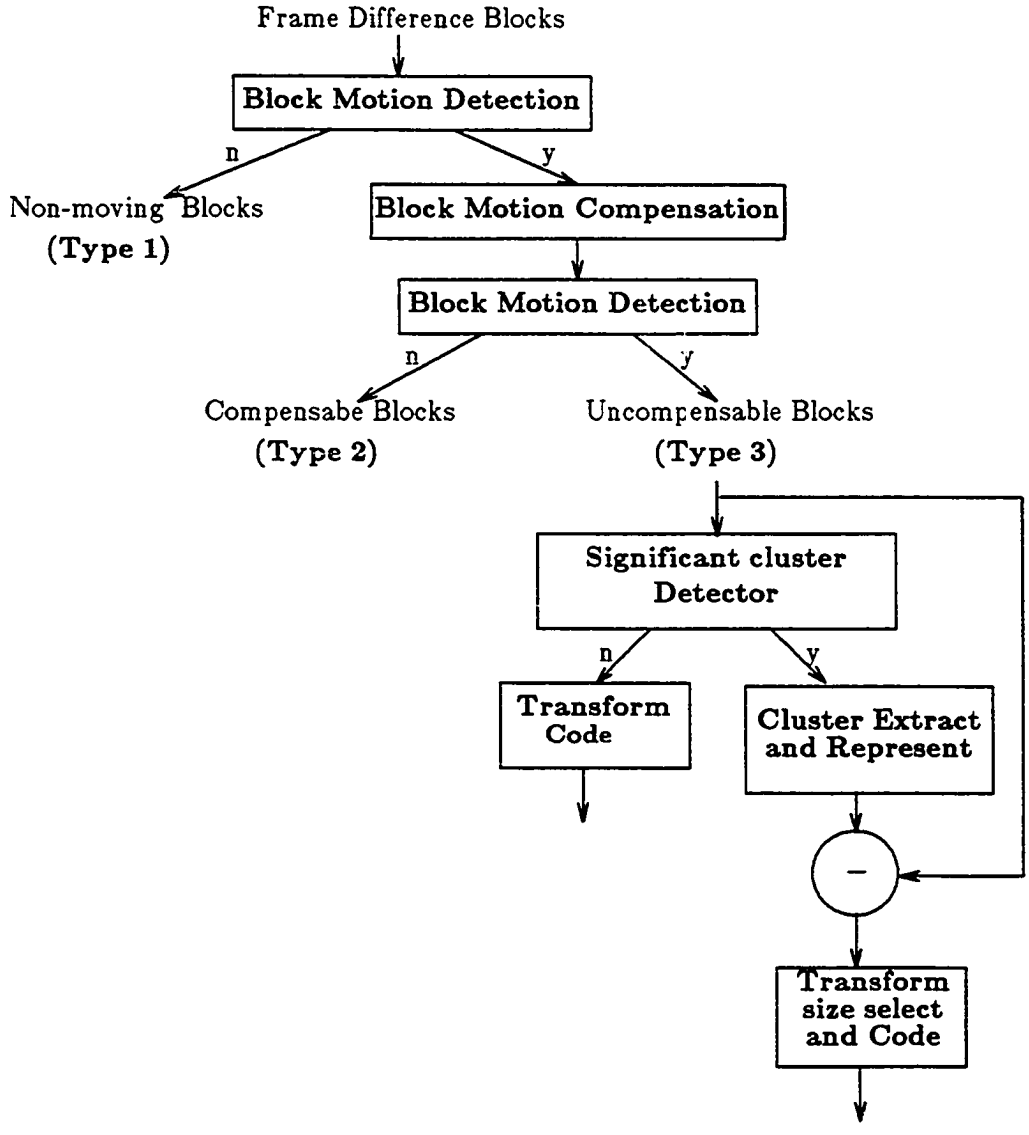


Figure 6.5 FD Block Classification for Hybrid (cluster/transform) Coding

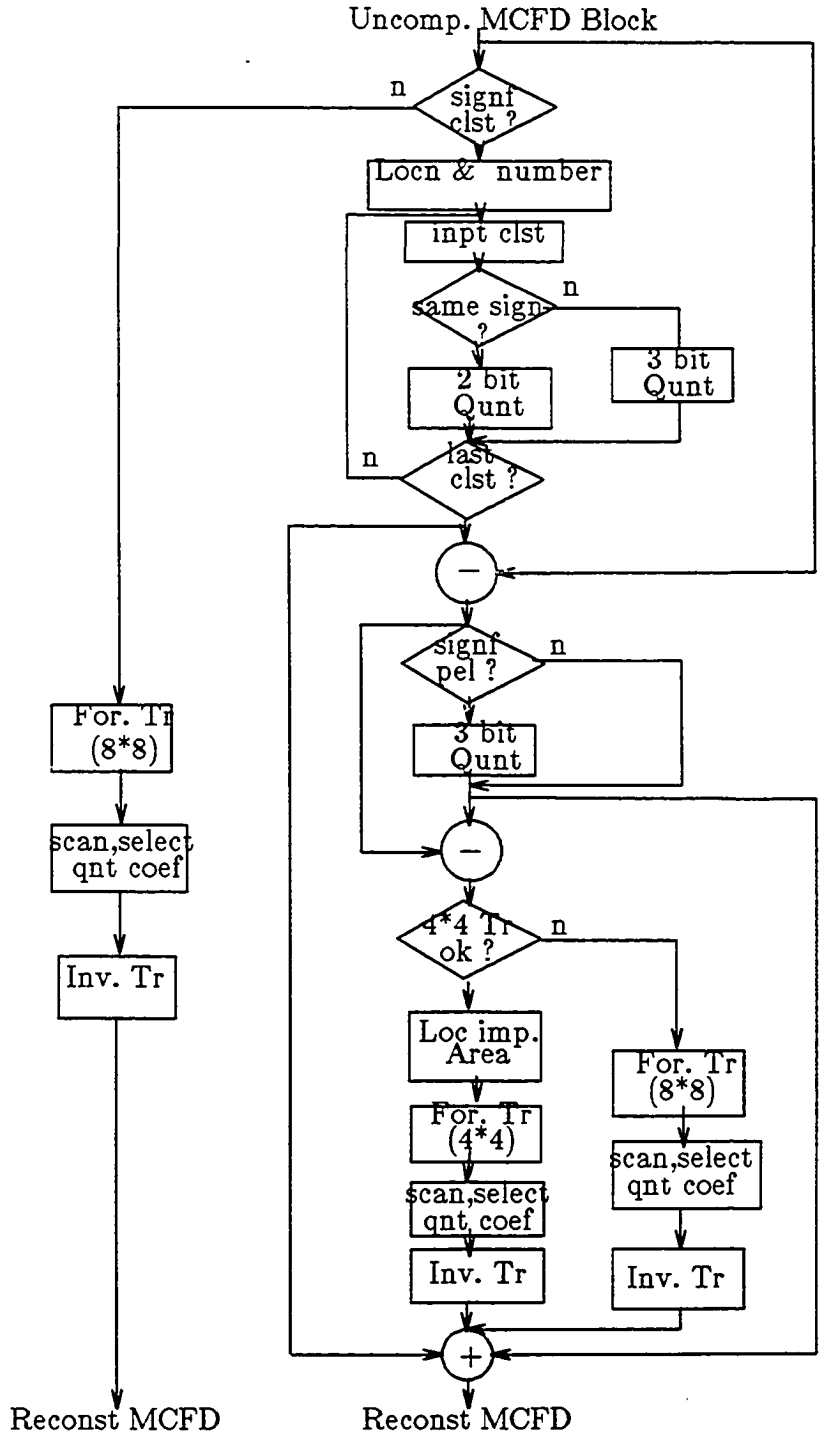


Figure 6.6 Detailed Flow-Diagram of Hybrid (cluster/transform) coding Algorithm

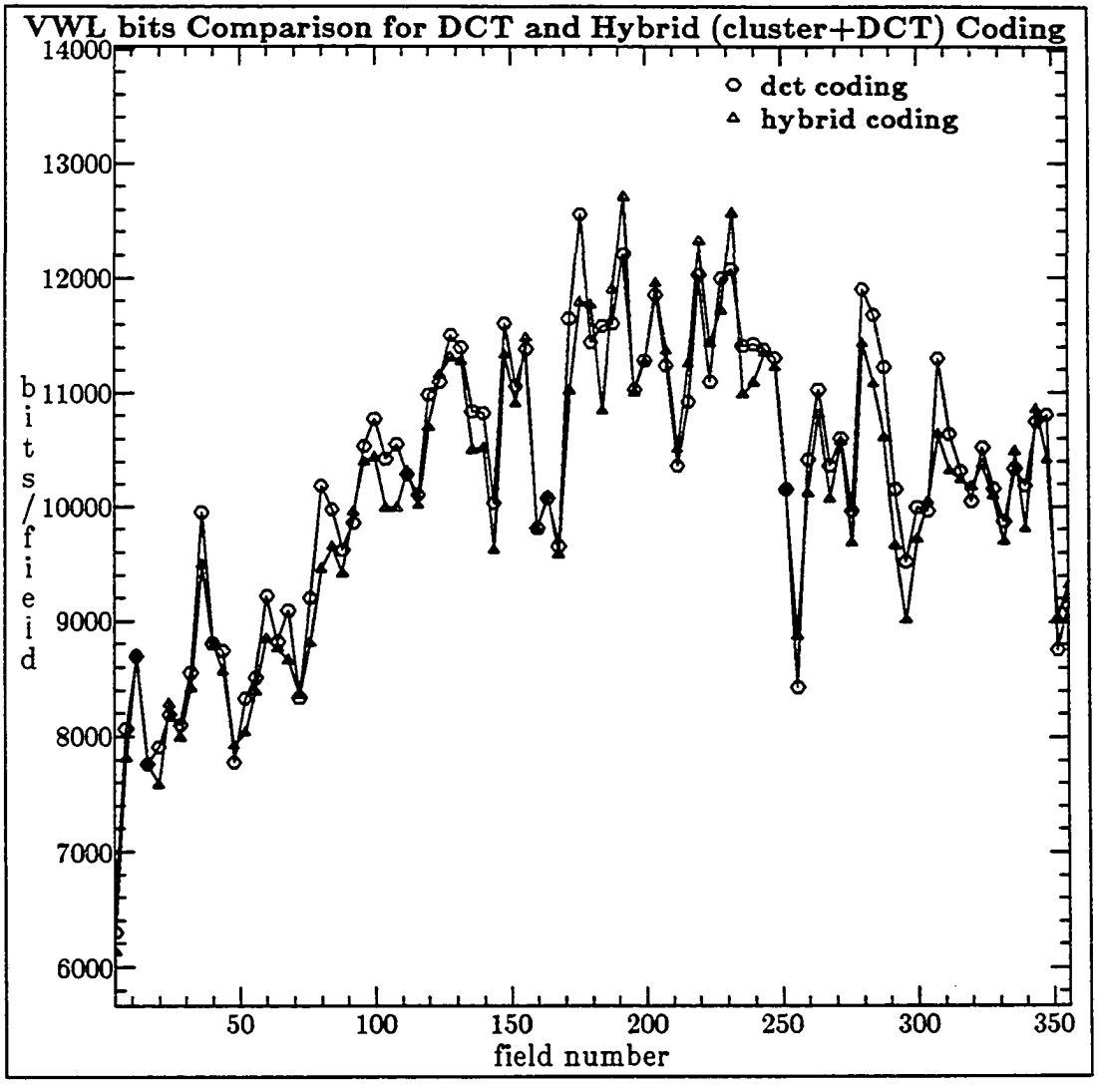


Figure 6.7(a) VWL bits comparison for DCT, and Hybrid (cluster+DCT) Coding, qnt step=9 for DCT part: "MsUSA" sequence

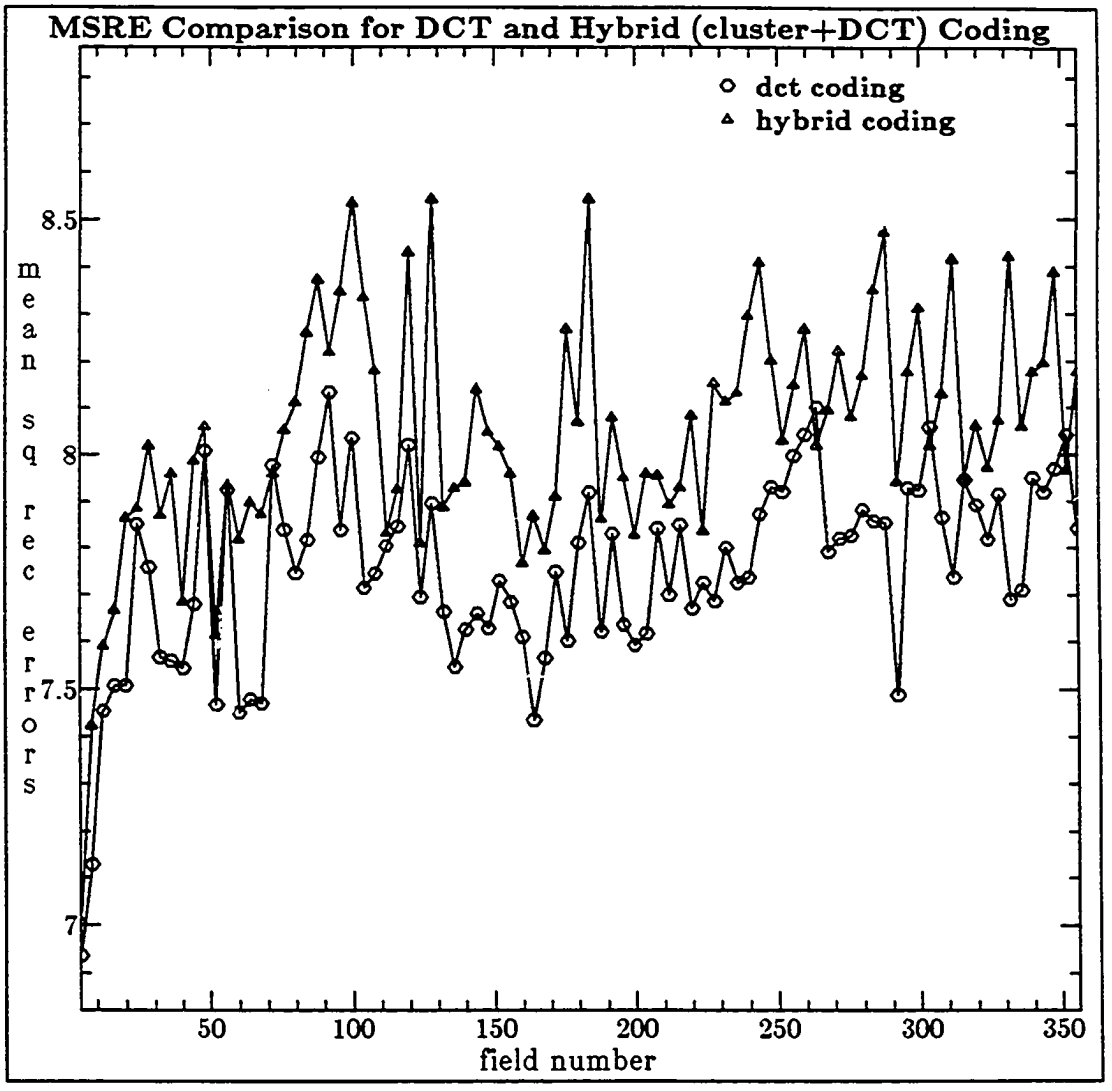


Figure 6.7(b) Mean Square Reconstruction Error comparison for DCT, and Hybrid (cluster+DCT) Coding, qnt step=9 for DCT part: "MsUSA" sequence

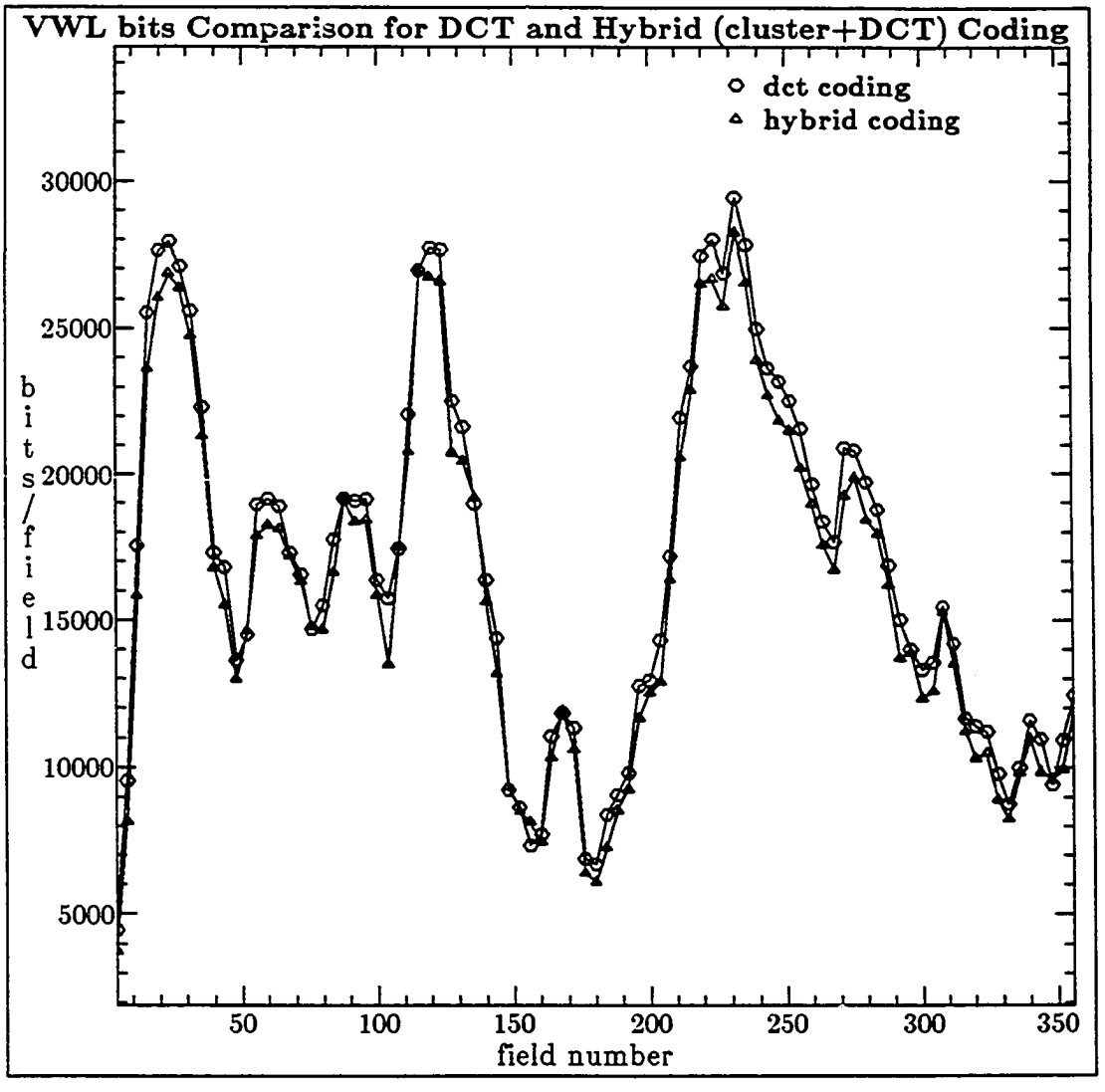


Figure 6.8(a) VWL bits comparison for DCT, and Hybrid (cluster+DCT) Coding, qnt step=9 for DCT part: "Salesman" sequence

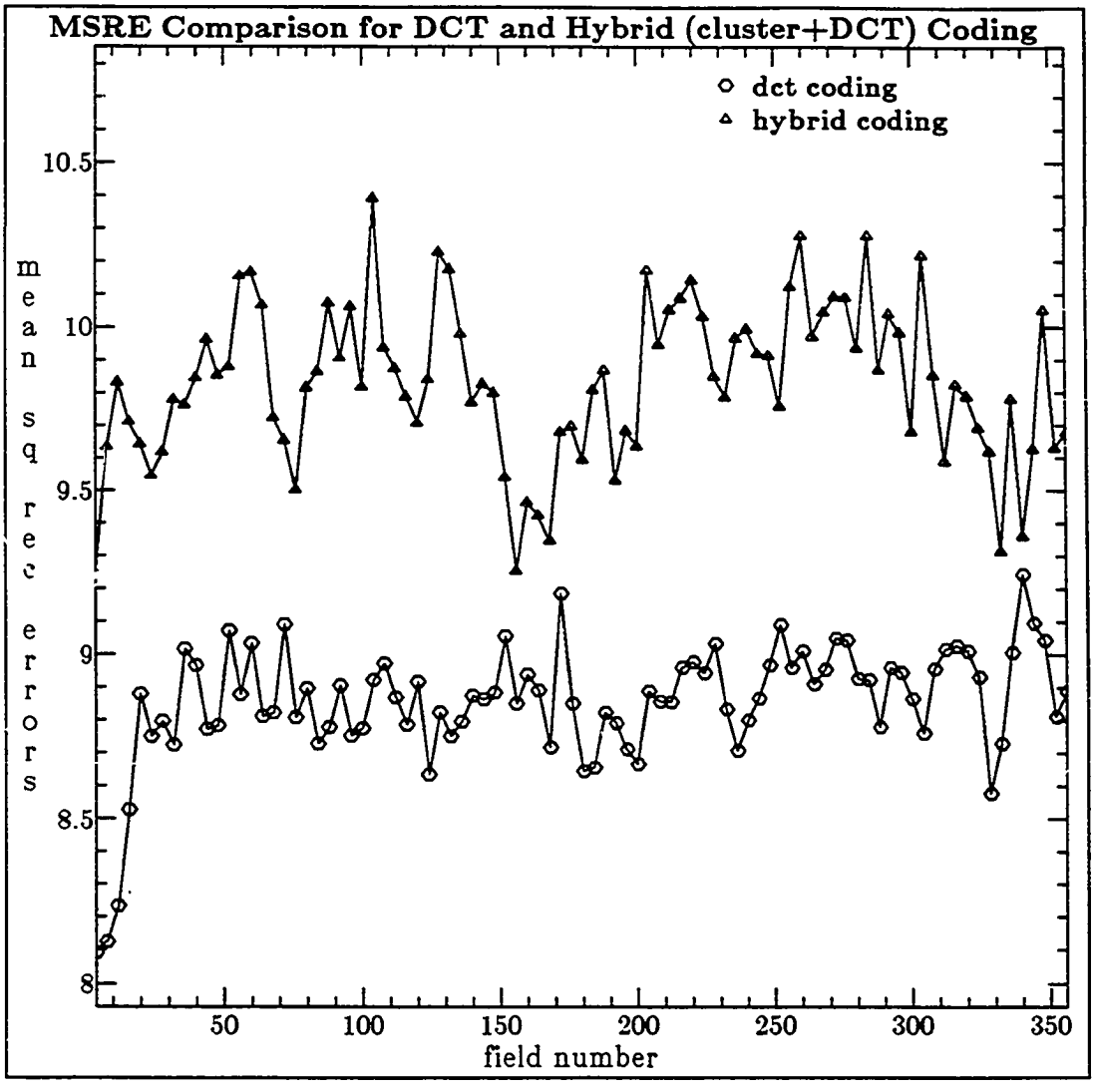


Figure 6.8(b) Mean Square Reconstruction Error comparison for DCT, and Hybrid (cluster+DCT) Coding, qnt step=9 for DCT part: "Salesman" sequence

Figure 6.9(a) Image of sequence "MsUSA" coded using DCT based coding Algorithm

Figure 6.9(b) Image of sequence "MsUSA" coded using Hybrid coding Algorithm

Figure 6.10(a) Image of sequence "Salesman" coded using DCT based coding Algorithm

Figure 6.10(b) Image of sequence "Salesman" coded using Hybrid coding Algorithm

CHAPTER 7: A COMPLETE MOTION-COMPENSATED CODER

7.1 INTRODUCTION

In the previous chapters we have discussed various schemes and investigated several potential algorithms for use in efficient video-coding systems for low bit-rate applications. The basic structure chosen for such a coder is that of block-processing and is again briefly reviewed as follows. Each image frame is partitioned into blocks of fixed size, and then the movement of each block in the frame is estimated as compared to the previous frame, the blocks are compensated for movement to obtain a translated frame which is differenced with previous frame to obtain MCFD image. The significant blocks of this MCFD image are encoded by a block-coding algorithm, e.g., transform coding, and transmitted to the receiver. A motion-vector is transmitted for these blocks; the receiver reconstructs the frame by first obtaining the prediction block using the motion vector, and adds back the decoded MCFD block. The insignificant-blocks are replenished using the previous frame and thus, a complete-frame is reconstructed.

Thus far our emphasis has been to compare the performance of various algorithms for low bit-rate coder under the condition of a fixed quantization step-size. We have judged algorithms on the basis of their performance in coding MCFD luminance signal. In this chapter, we first provide some results for a complete coder that uses a fixed quantization step-size. In this coder chrominance-bits and various overhead bits are included to specify bit-rate for a complete fixed coder. Next, we describe a complete motion-compensated coder that can operate in the bit-rate range of 80-320 Kbits/sec. A desired bit-rate for coder operation can be selected, and a buffer

feedback control attempts to regulate the coder behavior for the specified bit-rate.

7.2 A FIXED STEP-SIZE CODER

A fixed step-size motion-compensated interframe coder briefly outlined above is simulated. Previously we have concentrated on finding the best strategy for luminance. Here we discuss bits required per frame for encoding both luminance and chrominance signals as well as the motion related overhead information that needs to be transmitted to the receiver. A motion-detector is used to classify each block of every frame to be moving or non-moving. Such a motion-detector operates on blocks of differential signal between two frames and classifies a block as non-moving or moving. For the blocks classified as moving, a displacement estimate is computed. Both the motion-detection and motion-vector bits form an overhead, that is transmitted to the receiver. The motion-detection information is sent for each block whereas displacement information is sent only for the moving blocks.

7.2.1 Performance of Coder

We discuss performance results for a complete-coder when a fixed step-size is used. Previously, bits required for emcoding the boundary blocks were excluded from computations; we now include these bits in our computations. The total bit-rate includes bits needed to encode luminance signal, chrominance signals, and motion-related overhead. We now compute the overall VWL/MSRE performance of the complete coder that uses quantization step sizes of 9 and 13 for the "MsUSA" and "Salesman" sequences.

For image sequence "MsUSA", the quantization step-size of 9 results in average data rate of about 250 Kbit/s; with step size of 13, the average data rate is found to be 150

Kbit/s approximately. Encoding of luminance signal requires on the average about 11 Kbits/sec; whereas coding of chrominance signals requires roughly one-third of the amount of bits required for luminance signal. Typically, the motion-related requires about 2 Kbit/frame as overhead. On the basis of MSRE of uncompensable blocks, increasing the step size from 9 to 13 results in increasing of MSRE roughly by 1.5 times.

The results on image sequence "Salesman" are somewhat more involved. At step size of 9, the total coding bit-rate estimated for this sequence is 300 Kbit/s. Surprisingly enough going to step size of 13, did not result in any overall reduction in the bit-rate. This can be explained on the basis that the progressive effects in salesman sequence last for a longer term. The coding of frames in active part of sequence, with coarse quantization step, results in a progressive loss in performance due to the feedback effects; there are several areas of sudden motion and adopting a coarse quantization-step throughout the sequence prevents acceptable (below the threshold) reconstruction of many blocks. In "MsUSA" sequence many blocks can be motion-compensated and do not require coding, and many others do not have to be progressively encoded because they satisfy the threshold criterion of acceptability; however in "Salesman" sequence number of such compensable blocks is far fewer. Overall the luminance signal requires most of the bit-rate, the chrominance signal requires only about one eighth the bit-rate of luminance, and there are slightly fewer motion-overhead bits required as compared to "MsUSA" sequence.

7.3 A COMPLETE CODER WITH BUFFER FEEDBACK CONTROL

In controlling the bit-rate of a variable bit-rate scheme many techniques can be employed. In the low bit-rate coding environment, the status of buffer fullness is often

used to control the step size of the quantizer, and thus regulate the rate of flow of bits into the buffer. The buffer can be defined for a line of blocks, a group of line of blocks, or a frame; thus the buffer-types can be: a block-line buffer, group of block-lines buffer, or a frame-buffer. Depending on the type of buffer selected the quantizer step-size can be altered to respond to buffer status once every block-line, a few block-lines, or a frame. As the buffer type decides how often the input to buffer is to be regulated, it also suggests the size of the buffer, e.g. a smaller buffer may suffice if the step-size is regulated once every group of blocks, whereas a larger buffer-size is necessary if step-size is regulated only once every frame. The maximum buffer-size is also dependent on the over-all coder delays encountered. For a real-time implementation the delays should be no more then the time required to transmit couple of frames.

The choice of buffer-control strategy may alter the visual perception of artifacts or degradations that usually appear in low bit-rate coded image sequences, thence altering the the spatio-temporal performance of a coding scheme. For example, if quantizer step size is regulated every block-line or group of block-lines then the effect of quantization with different step sizes may be visible within different areas of a reconstructed frame, causing spatial non-uniformity. This strategy may be beneficial on the other hand, in providing better regulation of buffer contents. If step size is allowed to vary on a frame basis, the spatial uniformity in visual perception may be preserved; non-uniformity may occur in perception along the temporal axis. This approach may also have some difficulty in regulating buffer-contents in areas of sudden, high, temporal activity. Some factors affecting the overall performance of a buffer feedback-control coding scheme have been discussed. For a general purpose coder designed to operate in a bit-rate range, many other factors may also effect the

performance significantly. We now discuss some requirements from a buffer feedback control scheme where buffer quantization-step is to be regulated only once every frame.

7.3.1 Desirable Attributes of Buffer Feedback Control

[1] The initial step (base step) size should be a function of desired bit-rate, not just the smallest available step size.

[2] The buffer control strategy should use some hysteresis to permit the switching of quantization step size. For a coder where decision on quantization levels are made only once every frame and buffer of limited size is allowed, it may be important to use hysteresis to control the reduction of step size rather then the increase in step size. Reduction of step size should therefore be permitted when buffer status is at the closest desirable safe level.

[3] Changes in step sizes should not permit instability, this may occur if step sizes of coder bounces back and forth between two levels. This phenomenon called 'ringing' can cause visual artifacts in the coded sequence due to periodicity in switching quantization step-sizes.

[4] If possible, the buffer feedback control should not allow large swings in quantization step size from one frame to next, and therefore permit only a gradual change in visual quality of the coded sequence.

[5] The final (buffer = 100% full) step size should also be a function of desired bit-rate.

7.3.2 Buffer-size and Status

We now provide some details regarding selection of buffer size and calculation of buffer status. Our flexible codec is allowed to operate at the selected bit-rate in range of 80 to 320 Kbit/s; since this range is rather wide one size buffer may not be adequate because of practical reasons. In order to gain some knowledge regarding the buffer-size that would be appropriate, the buffer size is not kept fixed, but rather, dependent on the given bit-rate. This size is selected to be 10 percent of the desired bit-rate. If frame rate of 15 frames/sec is allowed, the buffer is large enough to accommodate 1.5 frames of average size for a given bit-rate. The buffer size is also in agreement with limitations in size due to the delay restrictions for a practical system. Based on the status of buffer fullness the step-size of the quantizer is updated once every frame, if necessary. The complete buffer status is computed per frame by,

$$BF(n) = BF(n-1) + [C(n) - AV] \tag{7.1}$$

$BF(n)$ is the buffer status at frame n.

$BF(n-1)$ is the buffer status at frame $n-1$.

AV is average number of bits required to code any frame and is calculated from, $AV = \frac{bitrate}{15}$.

$C(n)$ is number of bits required to encode frame n.

7.3.3 Buffer Fullness & Control (BFC) Algorithm

The size of the buffer depends on the bit-rate selected; in particular this size is kept at about 10 percent of the desired bit-rate. This means that at 15 frames/sec frame-rate the buffer is large enough to accommodate 1.5 frames of average size. This

buffer-size also seems consistent with the maximum overall delay (<250 msec approx.) allowed in a practical system to keep synchronization between voice and video.

The BFC algorithm selects only odd quantization steps. Finer quantization is not necessary as the step size is chosen only once every frame, and since the decisions are made less frequently, the buffer usually demands a rather rapid update. The fullness of buffer is computed every frame, and based on the status of buffer at previous frame and other considerations, an appropriate step-size for quantizer is selected at the current frame.

We relate the quantization step size 'q' to the percentage buffer status 'PBF' according to the following equation:

$$q = \begin{cases} init\_step & \text{if } PBF < 20\% \\ init\_step + \left\lfloor \frac{PBF}{10} - 1 \right\rfloor^{\#} \times 2 & \text{if } 20\% \leq PBF < 70\% \\ init\_step + \left\lfloor \frac{PBF}{10} - 1 \right\rfloor^{\#} \times 2 + 2 & \text{if } 70\% \leq PBF < 100\% \\ final\_step & \text{if } PBF = 100\% \end{cases} \tag{7.2}$$

The quantization step-size 'q' is calculated by eqn. (7.2), may have to be modified according to the *buffer fullness-controlling* algorithm. We now specify this algorithm, which along with the computed step-size, regulates the overall rate at which contents of buffer are changed.

+ In switching from the current value of step-size to the next higher value, small amount of hysteresis (2 out of 10 percent of buf contents) is introduced.

$\lfloor x \rfloor$ denotes odd integer smaller then x.

Therefore, if buffer contents are governed by hysteresis range, previous step-size is employed, otherwise the computed step-size is employed.

+ In switching from the current value of step-size to a lower value, hysteresis introduced is somewhat more significant, partly because decisions on step-size are made only once every frame, and therefore large swings in buffer status may be expected. Since decision is made based on past buffer contents and activity in the current frame is not known, a conservative approach is followed.

+ When the step-size is increased, less bits go into the buffer, whereas decreasing the step-size usually increases the flow of bits, causing sudden filling of the buffer. Using a step-size lower than the previous step-size, depending on the activity in the sequence, may cause a swing of as much as 40-50 percent in the buffer status.

+ If the buffer is less than 10 percent full the step-size computed according to eqn (7.1) is used. If buffer is more than 10 percent full coding is continued with the step-size used previously until it drives the buffer to less than 10 percent full, and step size is then decreased.

+ The decrease in step-size even when buffer is less than 10 percent full is clipped, a large sudden downward swing in step-size is not allowed, for e.g., step-size is not allowed to suddenly decrease from 17 to 7, but to an intermediate value such as 11. This prevents unstable swinging between extreme values.

+ An init_step of 13, and final_step of 41 proved to be good choices for coding sequence "MsUSA" at 100 Kb/s and "Salesman" at 200 Kb/s. An init_step of 7,

and final_step of 35 work reasonably well for sequence "MsUSA" at 200 Kb/s and "Salesman" at 300 Kb/s.

7.3.4 Performance of Algorithm

For the "MsUSA" sequence the performance of the coder with buffer feedback-control was quite smooth. The temporal activity for this sequence increases or decreases rather gradually to prevent cyclic switching (ringing) between levels. Hysteresis ensures somewhat smooth transition in buffer-status. An override mechanism can be used to cancel hysteresis when buffer fullness is relatively high. Using the specified algorithm, step-sizes are also found to vary relatively smoothly. Switching to lower step-size is allowed only when some stability in bit-rate has been achieved, and provided that buffer is not too full. During area of high temporal activity a coarser step size is used, the algorithm attempts to maintain a constant step-size over that part of the sequence; ringing is prevented in these instances by resorting to hysteresis. Even though measures for clipping are incorporated, generally the algorithm and the nature of motion in the sequence prevents very sharp transitions in step-sizes to take place. Some statistical results for "MsUSA" sequence comparing the performance of buffer-feedback control coder at bit-rates of 100 and 200 Kb/s are shown in Figures 7.3 and 7.4. For 100 and 200 Kb/s overall transmission, Figure 7.3(a) shows the contents of buffer in terms of percentage fullness, whereas Figure 7.3(b) shows the step size chosen at each frame; Figure 7.4(a) shows the total bits required at every frame, while Fig7.4(b) shows the MSRE's.

The performance of buffer-feedback control algorithm on "Salesman" sequence was computed at bit-rates of 200 Kbit/s and 300 Kb/s, and is shown in Figures 7.5 and 7.6. For this sequence the initial and final step-sizes used at 200 and 300 Kbits/s are

similiar to that used for "MsUSA" sequence at 100 and 200 Kbits/s, respectively. The behavior of the algorithm for this sequence is found to be somewhat different; with buffer feedback-control once every frame, sharp changes due to sudden temporal activity in various parts of the sequence causes sharp fluctuations in the buffer status. Quick updates of step-size are necessary to avoid buffer overflow. The clipping part of the algorithm goes into effect, and modifies the step-sizes so as to limit the large transitions in the buffer status. In addition, hysteresis also goes into effect when buffer fullness is less than a preset threshold, to stabilize the performance, i.e. prevent 'ringing', and ensure somewhat smooth operation. Even despite various measures to regulate the buffer contents it gets about three quarters or more full (Figure 7.5(a)) during periods of high temporal-activity. Overall buffer-status undergoes far wider fluctuations than that for At 200 and 300 Kbits/s, for the "Salesman" sequence the effect of step-size variation is shown in Figure 7.5(b); the total bit-rate and MSRE's performance is shown in Figures 7.6(a) and Figure 7.6(b) respectively.

7.4 DISCUSSION

A fixed step-size coder with step-sizes of 9 and 13 is simulated, and allowed comparisons of bit-rates and MSRE's performance. A variable step-size coder, with buffer feedback-control is found to have many desirable attributes required of such a coder, including that of a stable operation. The visual performance of algorithm sequence is found to be satisfactory at 100 Kbit/s, and good at 200 Kb/s. The performance for the "Salesman" sequence is found to be good at 300 Kbits/s, but is less then satisfactory at 200 Kbits/s.

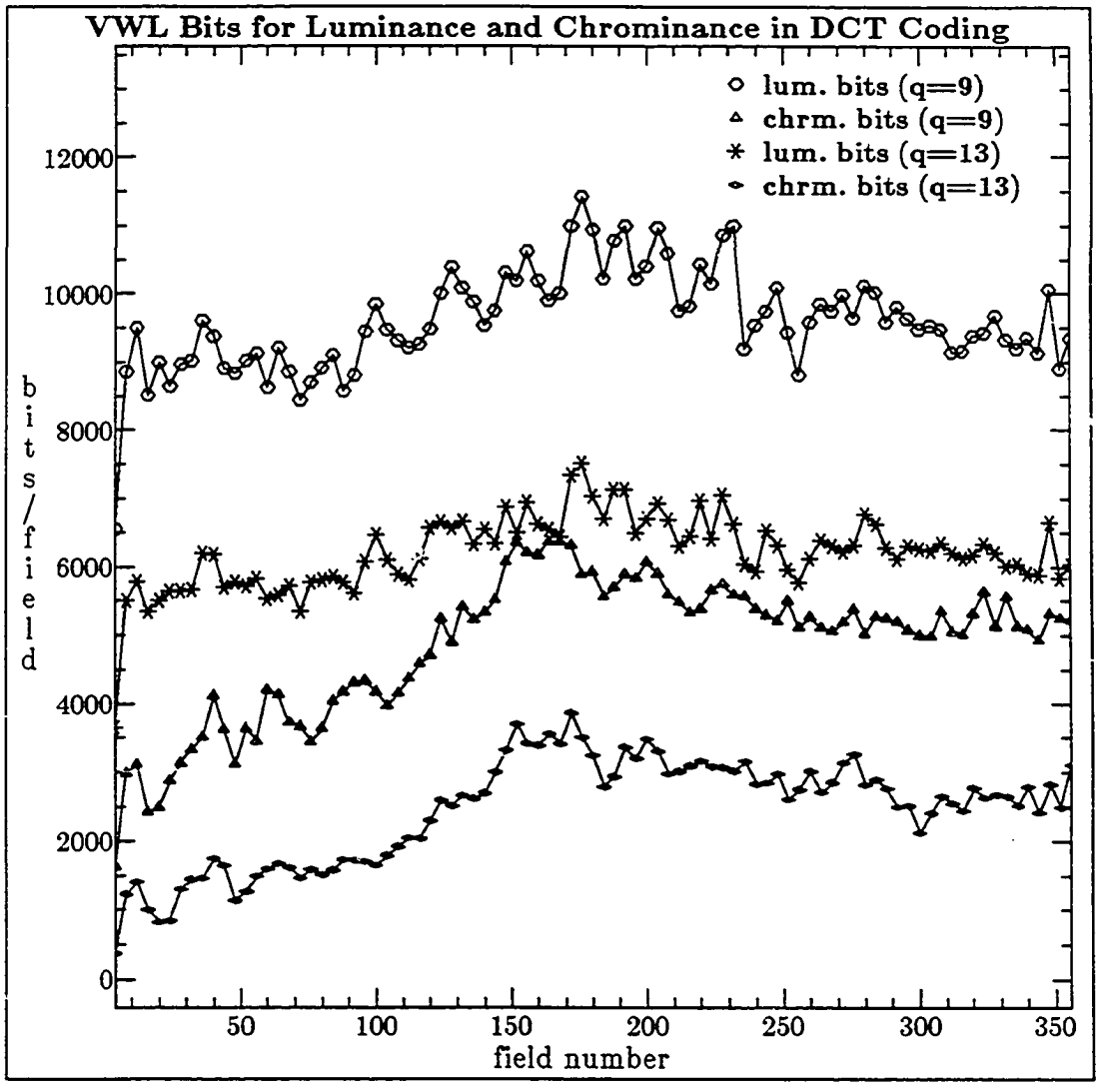


Figure 7.1(a) Luminance and Chrominance Bits for the uncompensable blocks in DCT-coding (q=9 & q=13) : "MsUSA" sequence.

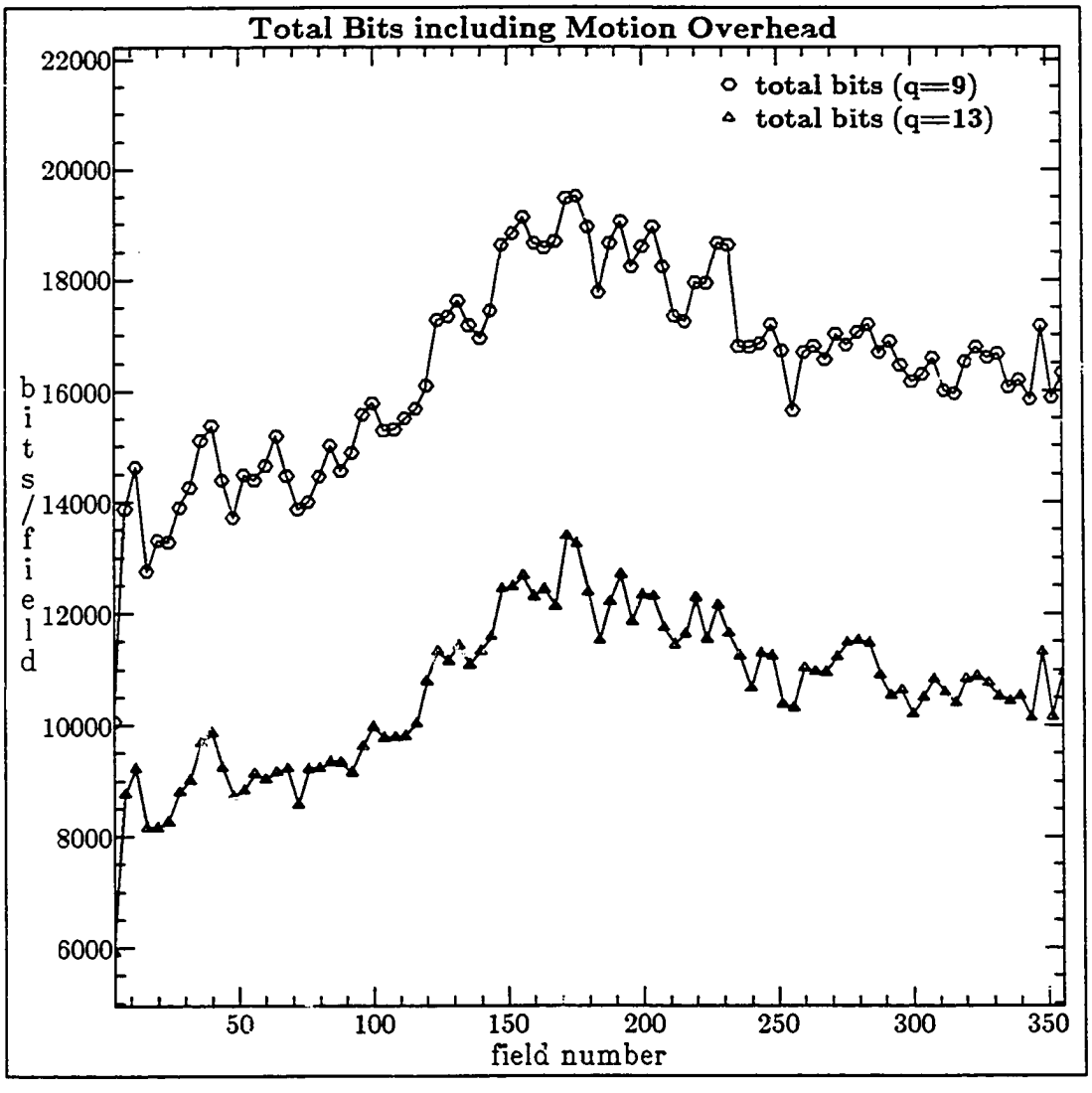


Figure 7.1(b) Total Bits including Motion overhead in DCT-coding quant steps (q=9 & q=13) : "MsUSA" sequence.

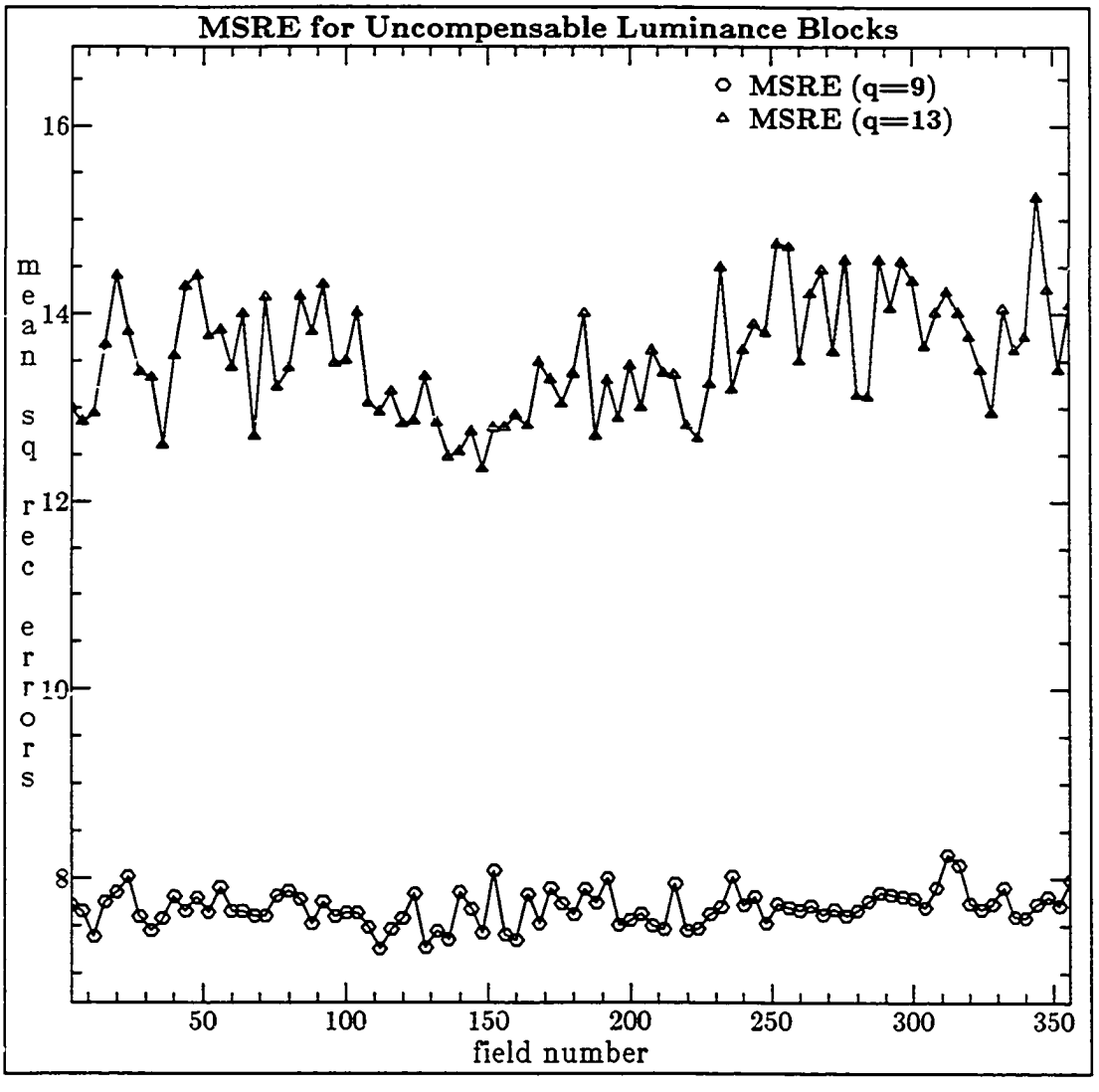


Figure 7.1(c) Mean Square Reconstruction Errors in DCT-coding quant steps (q=9 & q=13) : "MsUSA" sequence.

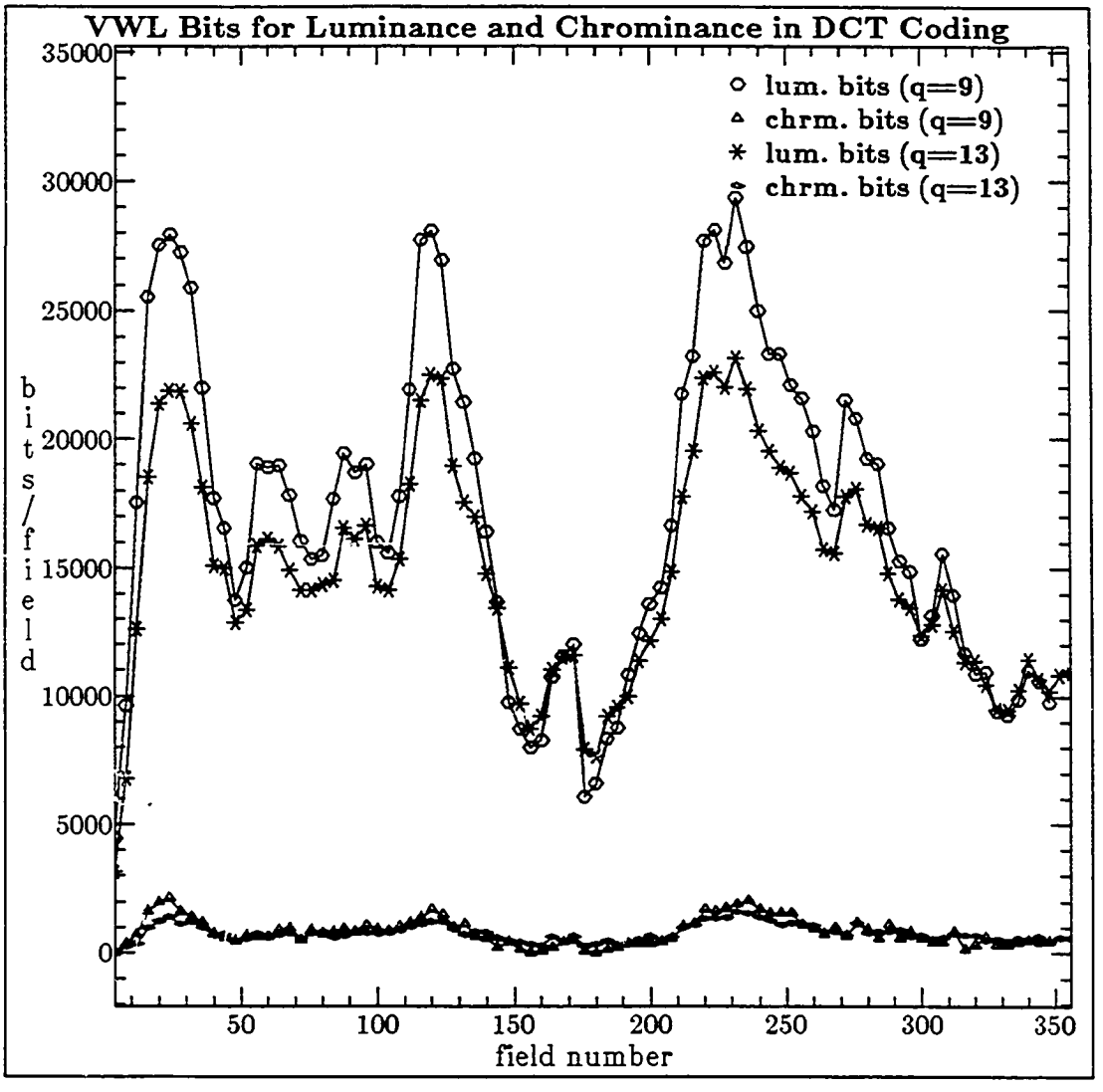


Figure 7.2(a) Luminance and Chrominance Bits of uncompensable blocks in DCT-coding (q=9 & q=13) : "Salesman" sequence.

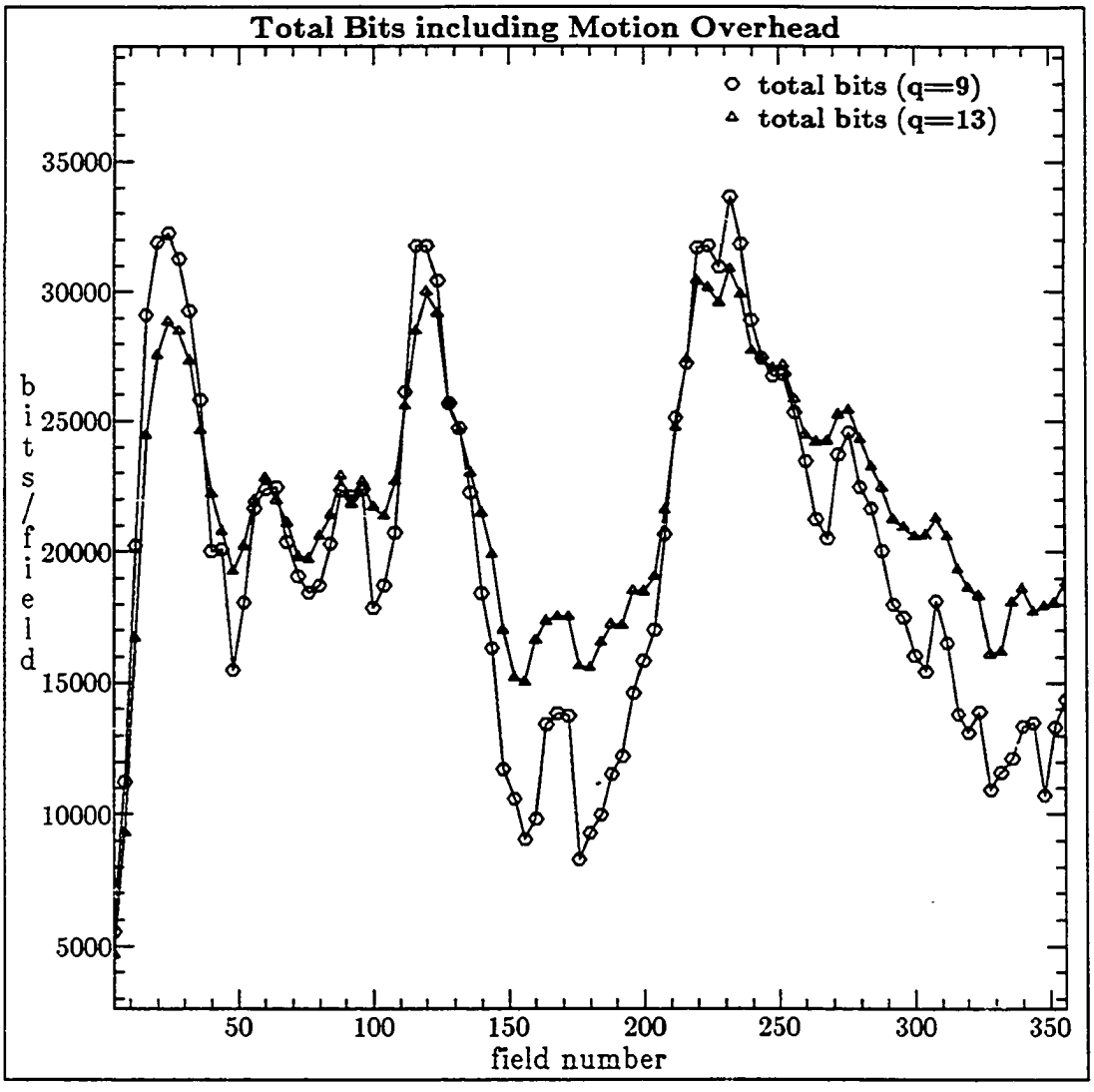


Figure 7.2(b) Total Bits including Motion overhead in DCT-coding quant steps (q=9 & q=13) : "Salesman" sequence.

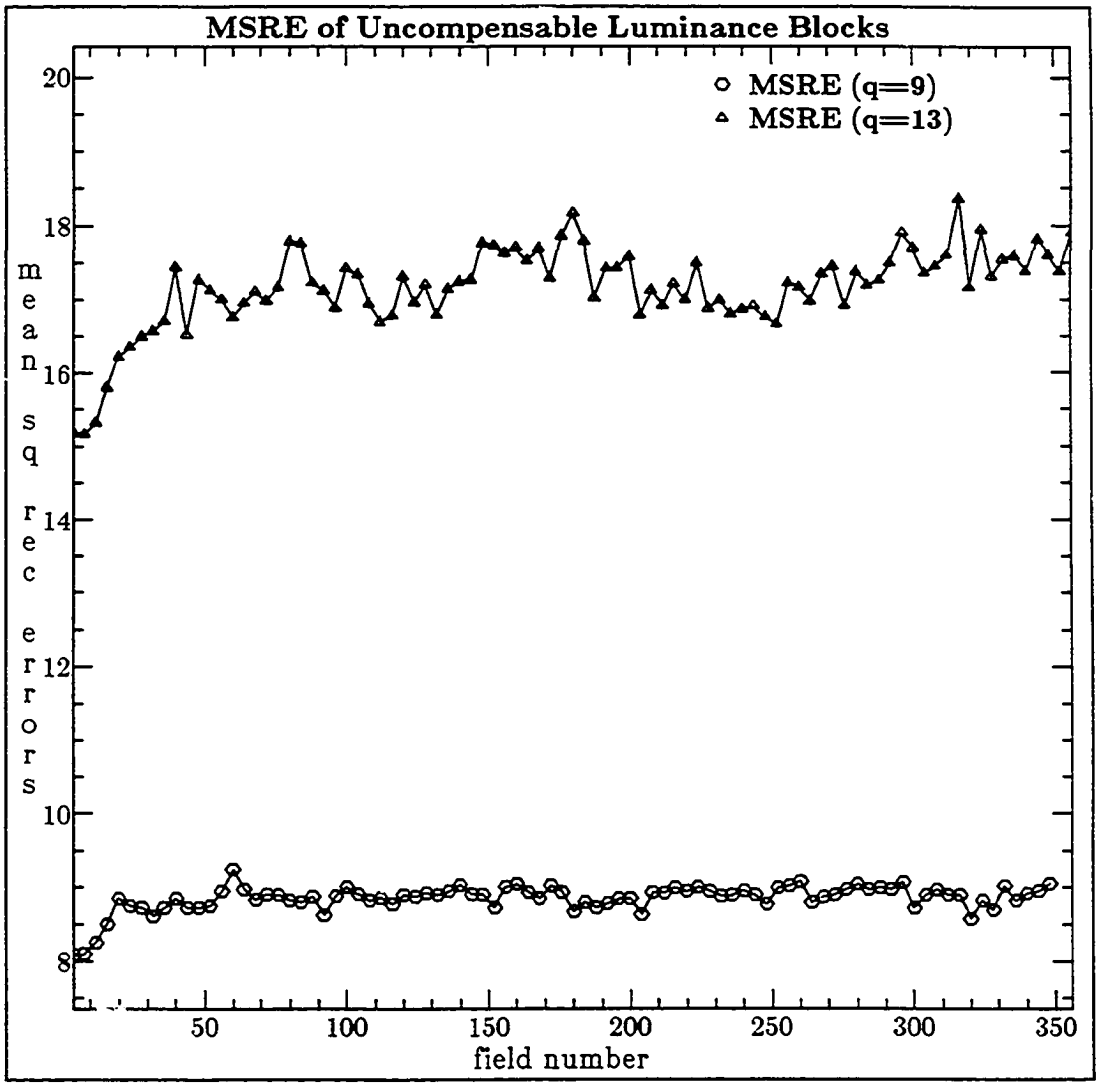


Figure 7.2(c) Mean Square Reconstruction Errors in DCT-coding quant steps (q=9 & q=13) : "Salesman" sequence.

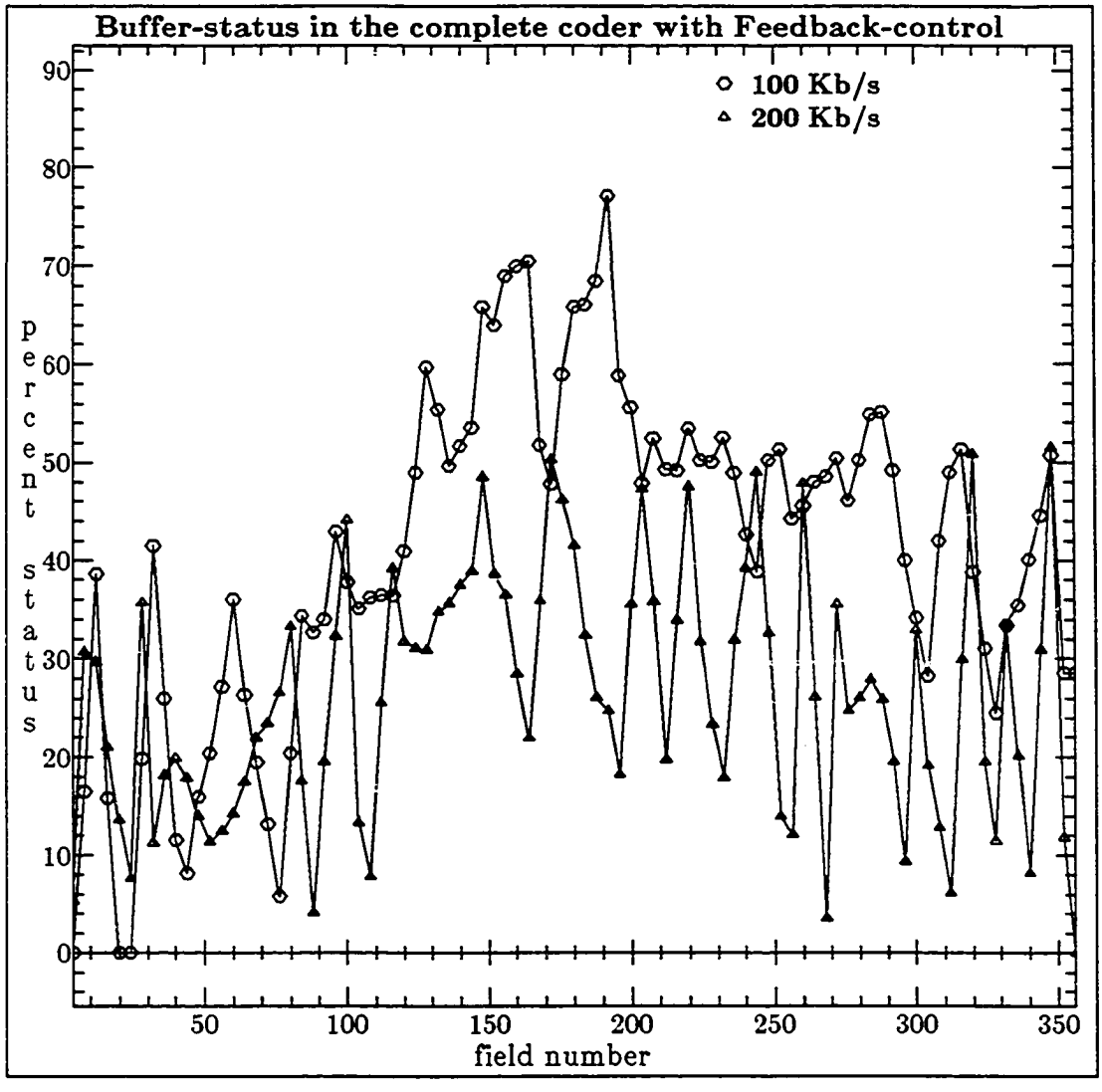


Figure 7.3(a) Variation in Percentage Buffer-status for complete coder with feedback-control, DCT-coding : "MsUSA" sequence.

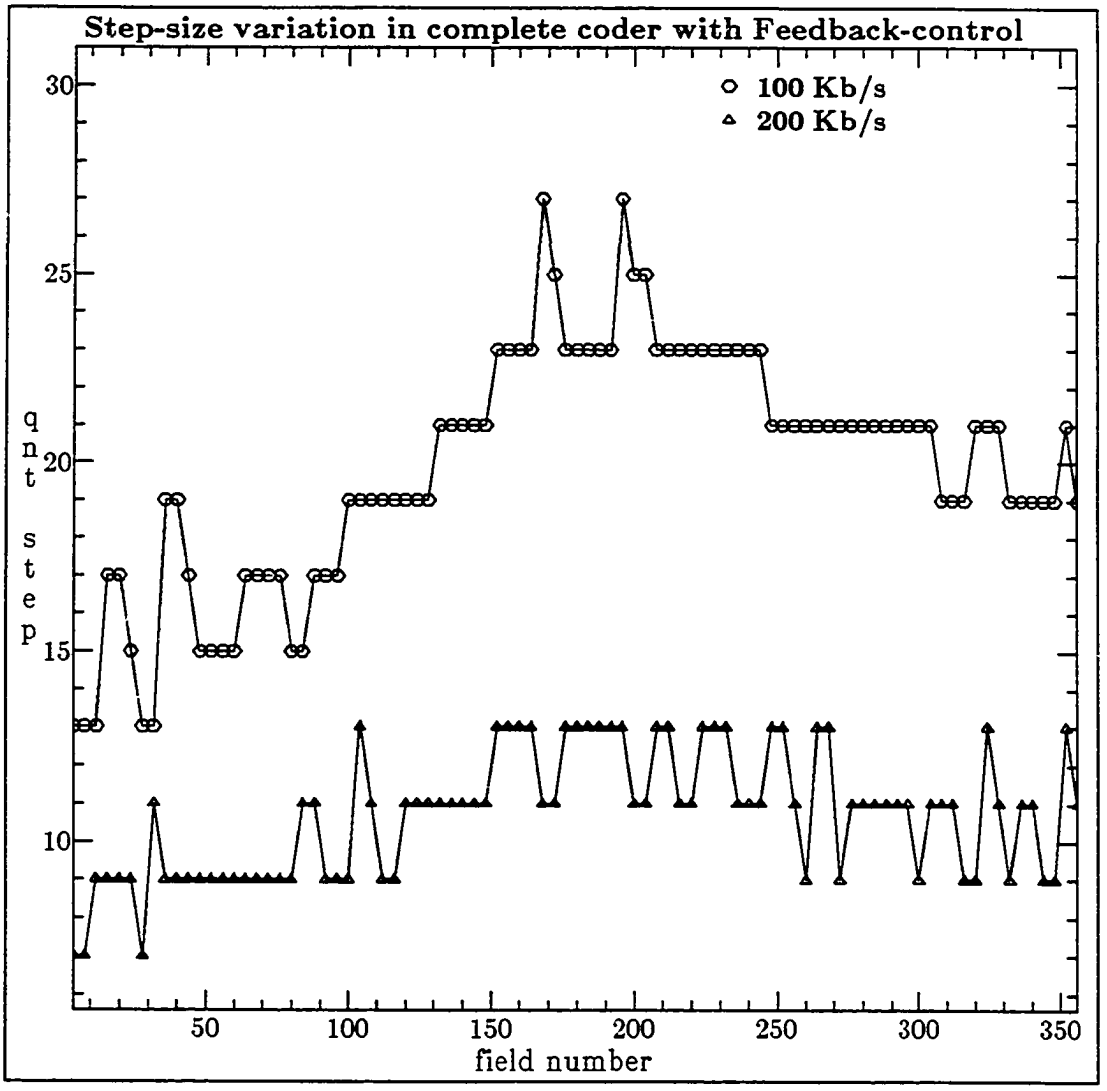


Figure 7.3(b) Variation in Step-size for the complete coder with feedback-control, DCT-coding : "MsUSA" sequence.

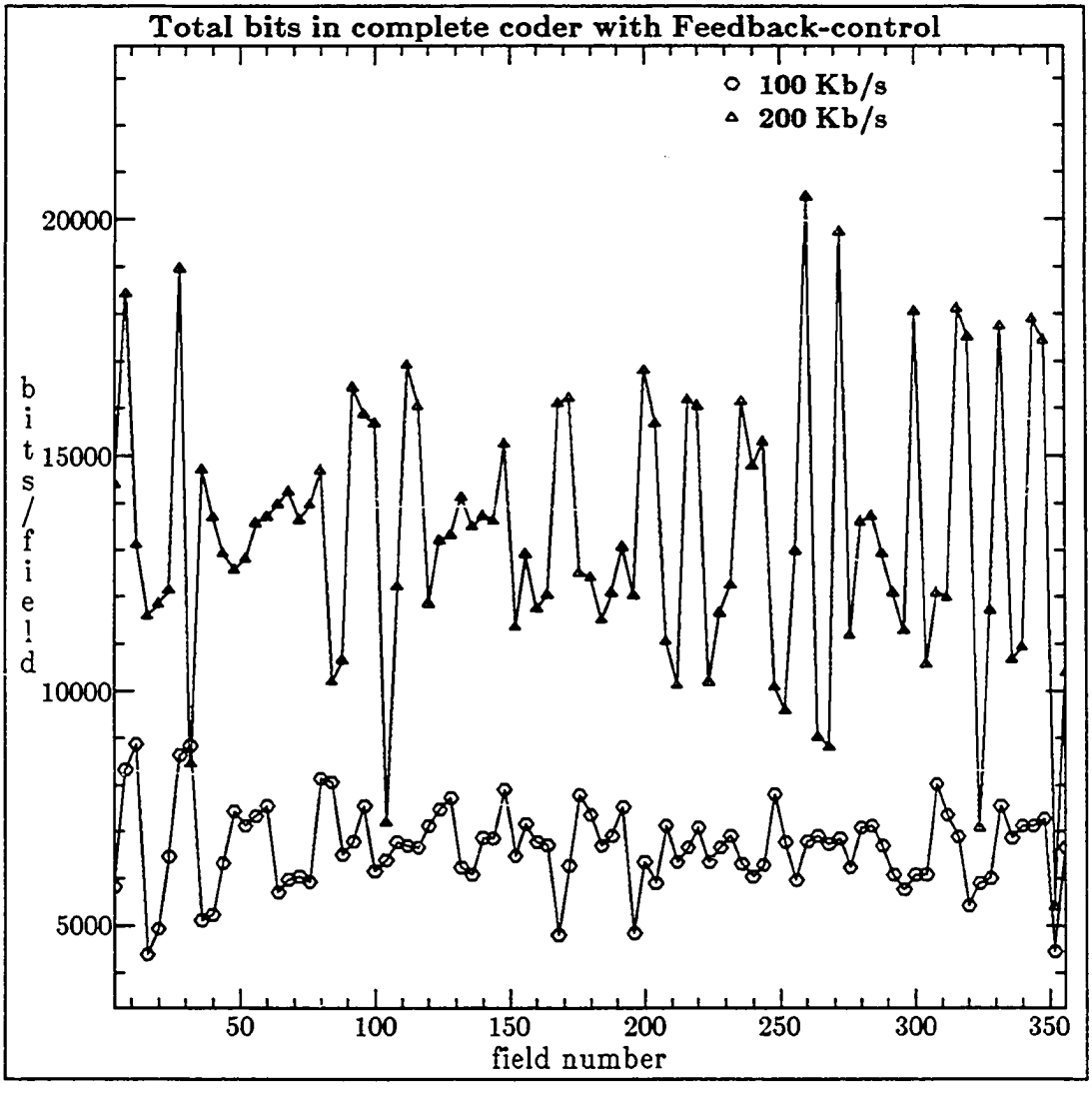


Figure 7.4(a) Total bits (including motion overhead) for a complete coder with feedback-control, DCT-coding : "MsUSA" sequence.

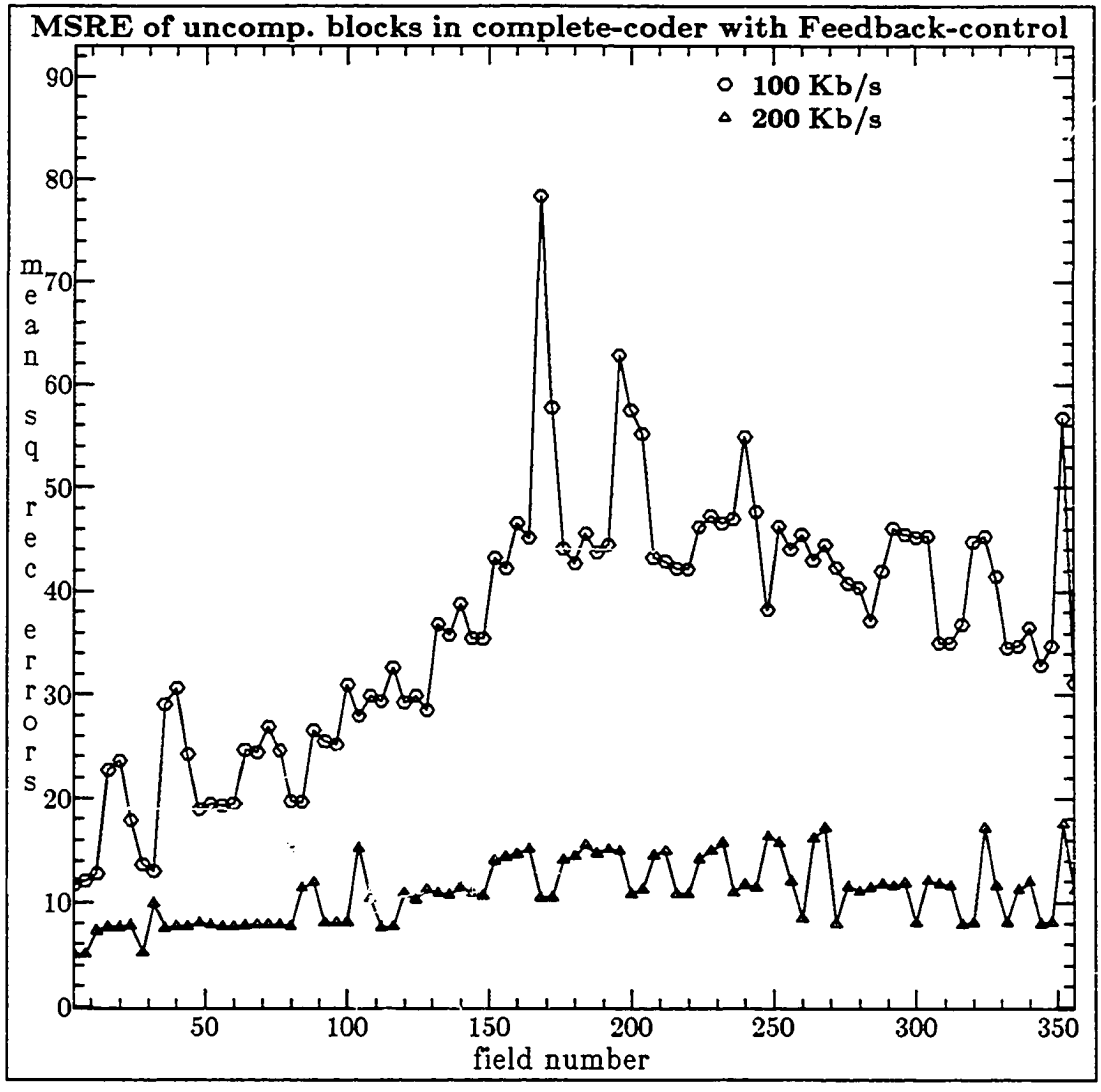


Figure 7.4(b) Variation in Step-size for the complete coder with feedback-control, DCT-coding : "MsUSA" sequence.

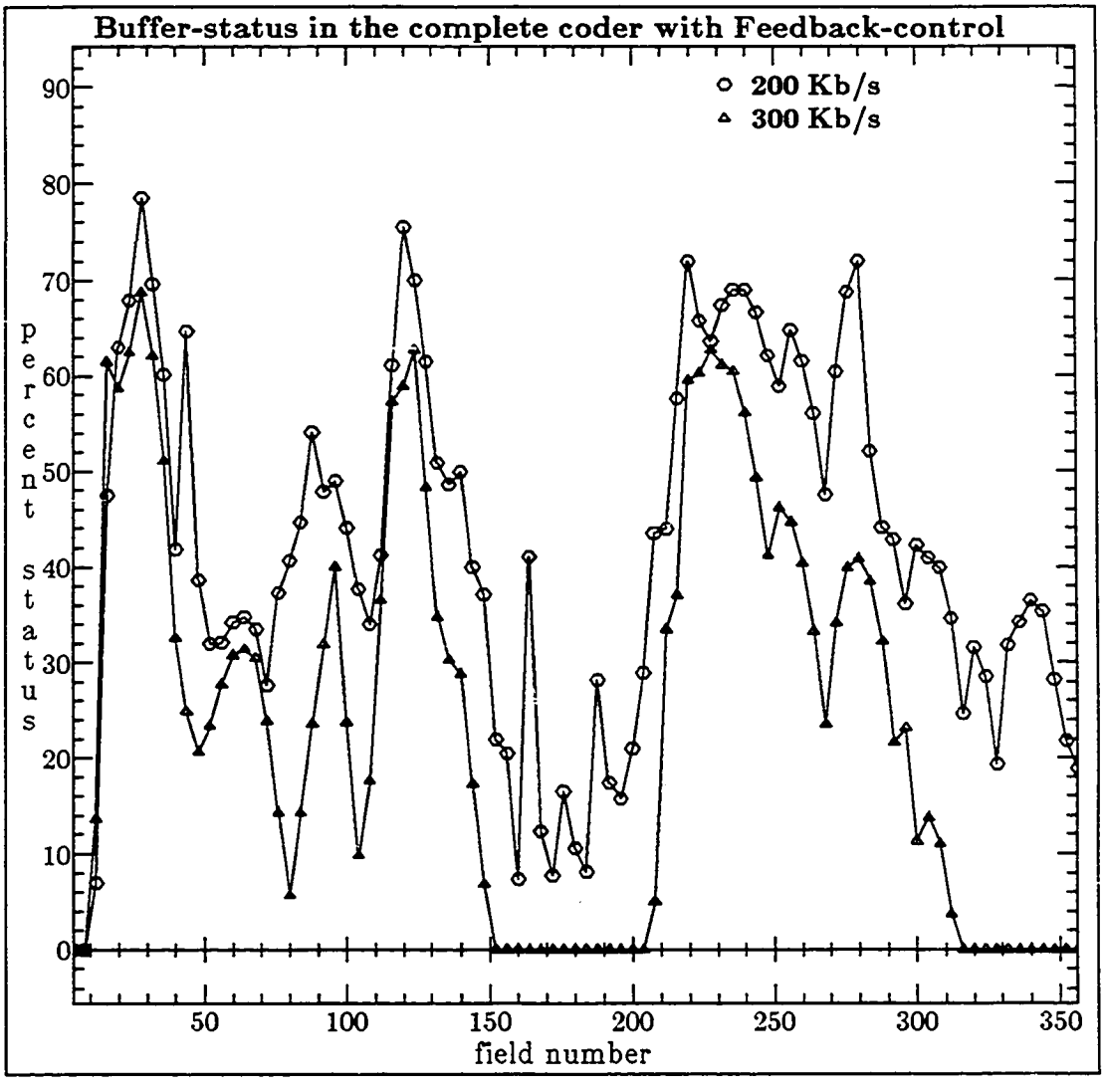


Figure 7.5(a) Variation in Percentage Buffer-status for complete coder with feedback-control, DCT-coding : "Salesman" sequence.

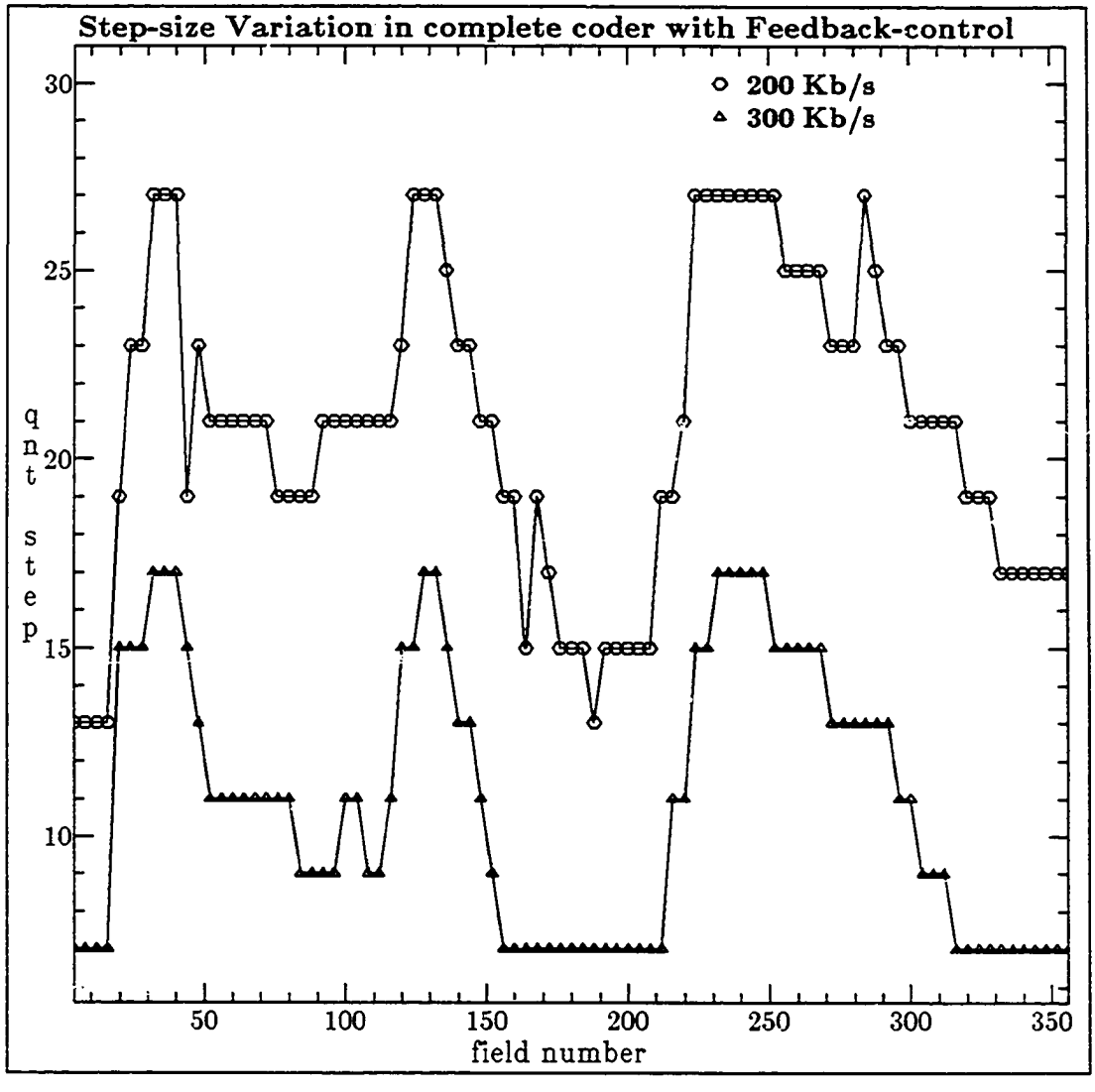


Figure 7.5(b) Variation in Step-size for the complete coder with feedback-control, DCT-coding : "Salesman" sequence.

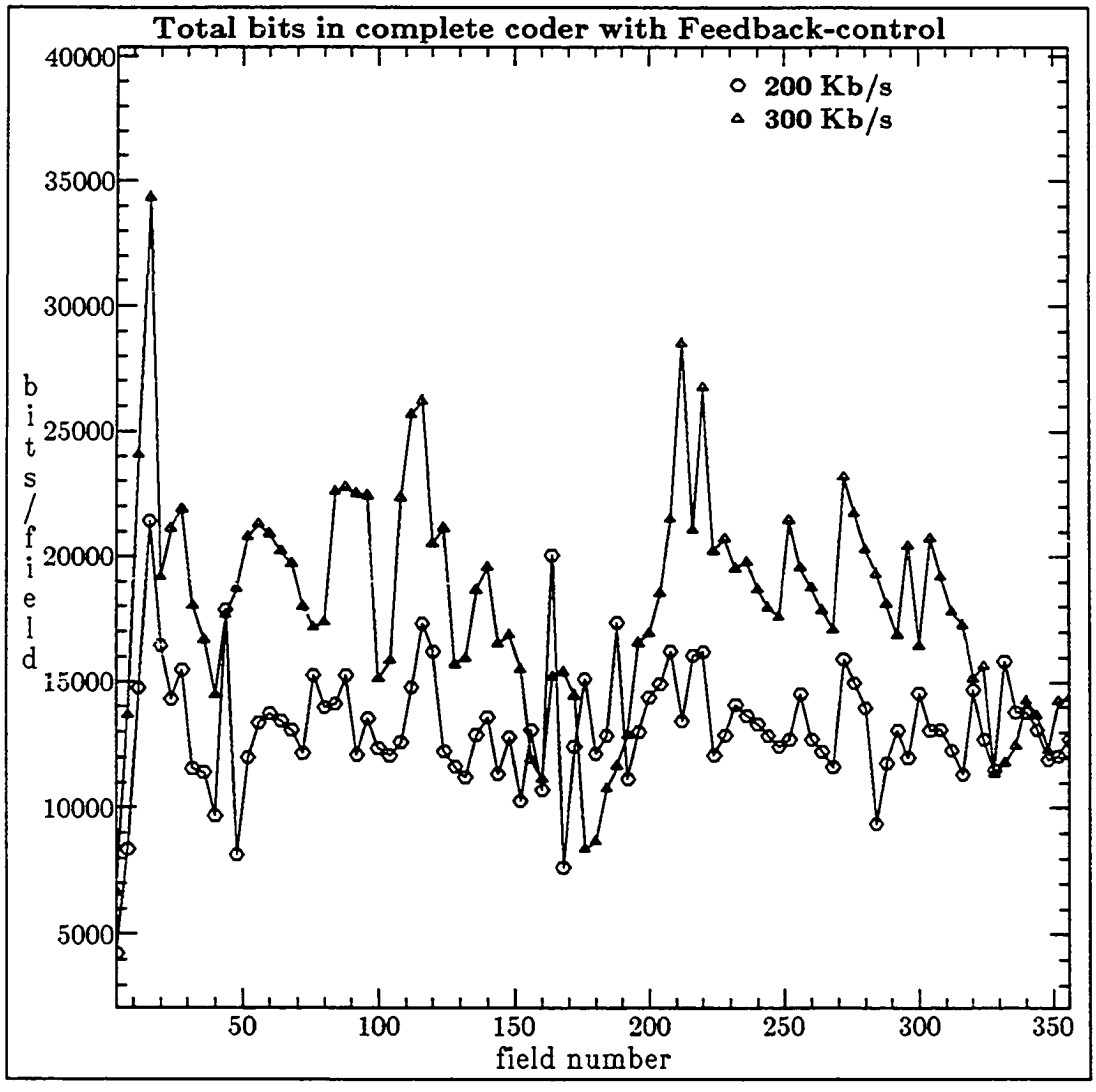


Figure 7.6(a) Total bits (including motion overhead) for a complete coder with feedback-control, DCT-coding : "Salesman" sequence.

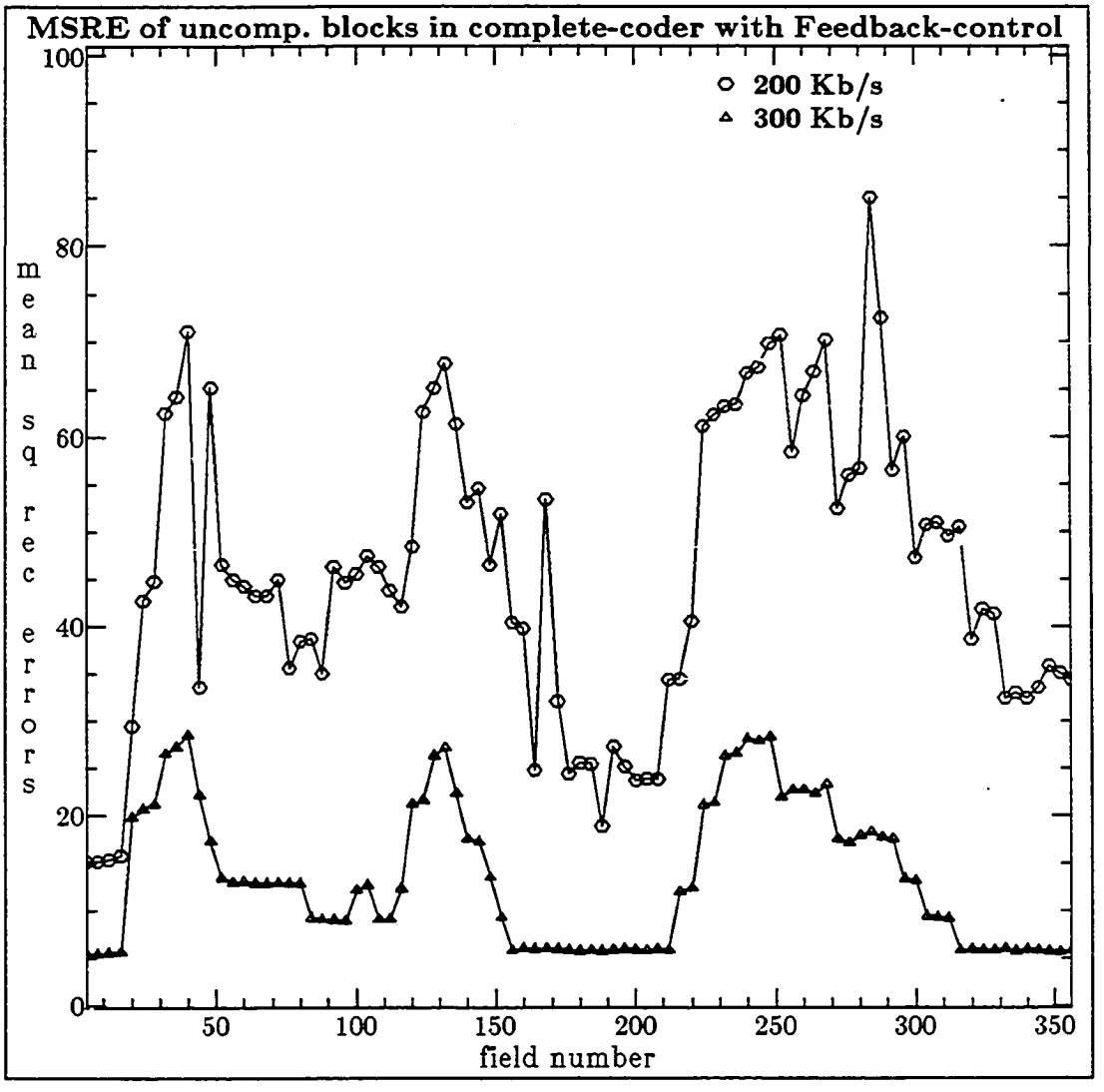


Figure 7.6(b) Variation in Step-size for the complete coder with feedback-control, DCT-coding : "Salesman" sequence.

Figure 7.7(a) Image from sequence "MsUSA" coded at 100 Kb/sec

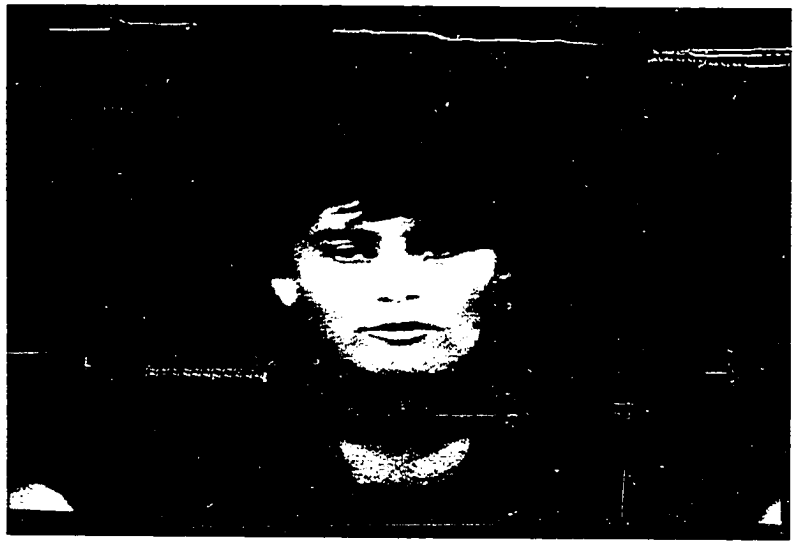


Figure 7.7(b) Image from sequence "MsUSA" coded at 200 Kb/sec

Figure 7.8(a) Image from sequence "Salesman" coded at 200 Kb/sec

Figure 7.8(b) Image from sequence "Salesman" coded at 300 Kb/sec

CHAPTER 8: DISCUSSION AND DIRECTIONS

8.1 DISCUSSION

In order to measure the performance of any scheme, many criteria may be taken into account. In intraframe image coding, several statistical measures such as, entropy bits, mean square error (MSE) or weighted mean square error (MSWE), signal to noise ratio (SNR) etc. are often used in comparing performance of various systems. Since the ultimate receiver of the reconstructed image is really a human being, criteria based on human visual system have been used. There is a general lack of agreement regarding performance measures and how exactly to use them. The problem is even more complicated when evaluating the performance of a coded image-sequence, as a single measure such as overall MSE or SNR is usually unsatisfactory, as it does not tell us regarding the performance on individual parts of the sequence. In addition, poor performance on one part of the sequence may appear more visually degrading, then on another part of sequence, due to amount and characteristics of motion, and the quality of motion-compensation. In any case, in order to compare the performance of different schemes, we have resorted to detailed statistical performance measures in each frame, and overall visual performance evaluation on the entire sequence. Several other factors affecting the performance can be listed. The performance of an algorithm may also depend on the characteristics of test image-sequence. Among the two image-sequences employed in our simulations, we find considerable differences in the performance of motion-estimation algorithms on each, due to the widely differing amount and activity of motion in each sequence. Parameters such as motion detection thresholds, block-activity measures, coefficient selection and quantization thresholds, filter in coding loop, effect the performance

indirectly; in a closed-loop operation, the reason for poor performance of a scheme may be difficult to isolate because of interactions among various parameters.

Our original goal in this thesis has been to provide improvements to enable efficient coding of motion-video for low bit-rate applications. The term "low bit-rate" in present context means bit-rates of about 50 to few hundred Kbit/s. Of course, some of the schemes with appropriate modifications can also be applied to improve the performance of higher bit-rate systems. Towards satisfying our goal, we have presented, in previous chapters, several improvements to the performance of basic motion-compensated coding scheme. We briefly list some of the contributions of this thesis, and then we review some salient features of each.

- The *orthogonal search* to reduce complexity of computations in motion-compensation by block-matching. Reduction of motion overhead, and tradeoffs in block-sizes, range of search, and number of samples used in matching per block.

- *Statistical properties* such as spatial and temporal correlation, block correlation-histogram, variance and other measurements on compensated and uncompensated motion-compensated frame differential (MCFD) signals of real video-sequences.

- Performance of motion-compensated *transform coding* system with DCT, DST or KLT(0.5) transforms. A *multi-transform* approach to exploit wide variation in statistical nature of MCFD signal from block-to-block.

- A *variable block-size* approach to improve performance via adaptivity, its application to motion-compensation and coding, and its performance evaluation

in transform-domain and pel-domain coding systems.

- A *cluster-coding* approach to deal with "difficult" MCFD pels, its application to a cluster/transform hybrid motion-compensated coder, and its performance compared to simple transform-domain coding technique.

- A complete *fixed step-size* coder, and a complete *buffer-feedback* (step-size controlled) coder, and their statistical, and visual performance on video-sequences.

Motion-compensation by block-matching is explored. An efficient search-technique, called *orthogonal search* is proposed. This technique reduces the computations necessary in block-matching. Other ways of simplifying block-matching, and reducing the overhead transmitted to the receiver, when block-matching is part of a coding system, are investigated. These consist of, dependent search techniques, reducing samples for matching, choice of block sizes and matching criteria. Among many variations possible in dependent search, schemes that use motion estimate from spatially adjacent-block or temporally adjacent block as the initial estimate, are found practical. Entropy performance, in choosing various block-sizes, or selecting a variable-block approach are indicated.

Statistical measurements on uncompensated and compensated frame-differential signals indicate the *properties* of these signals. These properties when analyzed yield insight into the nature of these signals. Among many properties investigated are, entropies, variances, spatial correlations per frame, temporal correlations between adjacent frames, and spatial block-correlation histograms. While many statistical properties of MCFD signals are found to be similar in image test-sequences "MsUSA"

and "Salesman", there were, however some significant differences. Overall, indications of complexity of each sequence and some insight into possible compression techniques suitable for each sequence is obtained.

Several *transform coding* approach to motion-compensated frame-differential coding are investigated. The effectiveness of various orthogonal transforms such as the DCT, the DST or the KLT(0.5) is explored. Statistical measures show considerable variations in characteristics of individual MCFD blocks. To exploit these variations, a scheme that uses *multiple-transforms* for coding is introduced. In this scheme, depending on the characteristics of the block being coded, a selection is made, from a small set of pre-decided transforms. Choices for various possible configurations are investigated. The transform selection information is sent as overhead to the receiver.

A *variable-block* approach is investigated for motion-compensation and coding of MCFD's. In this approach MCFD blocks are divided into sub-blocks, if necessary. Two configurations are tested; the first, uses blocks for motion-estimation and sub-blocks for coding, where as the second, uses sub-blocks for motion-estimation and coding. Algorithms for coding MCFD blocks by the transform-domain and the pel-domain coding techniques are discussed. Simulation results demonstrate that both transform-domain and pel-domain algorithms benefit from the variable block-size approach.

A *cluster coding* method is introduced. To obtain a good solution in *extracting* significant-clusters from a MCFD block a search procedure is followed. If too many clusters are found then cluster size, shape or magnitude information can be used to discard relatively less-significant ones. A cluster/transform hybrid coding scheme is then proposed, tnat operates in either the transform mode or the hybrid mode

depending on characteristics of the MCFD block. In the hybrid mode, clusters are quantized in pel-domain by a scalar quantizer, whereas, the residual signal is transform coded and quantized. By this method the clusters of significant pels in MCFD blocks, having sharp pulsive-transitions are efficiently encoded as compared to coding them by transform domain approach. Simulation results demonstrate improvement in visual performance of reconstructed sequences.

A *complete-coder* with pre-fixed step-size computes the overall performance including all overheads. A strategy useful for controlling the buffer status is briefly outlined. A complete-coder based on this strategy of buffer feedback-control allowing fixed bit-rate operation is simulated. The results on image-sequences demonstrate the performance of algorithm expected at various bit-rates,

8.2 TRENDS, APPLICATIONS, AND DIRECTIONS

In our discussions so far, we have not considered the effect of channel errors. In any practical situation such errors are expected, and are often bursty in nature. From practical point of view it is therfore important that, a video-codec should be somewhat less error-sensitive and be able to recover from errors rapidly. Errors in single bits can usually be corrected by means of FEC codes, which themselves require extra channel capacity.

Due to recent advances in hardware and VLSI, it is currently feasible to build many of the systems discussed in this thesis. The problem of standards for video-coders therefore comes into play; international standards meetings such as the CCITT, are working towards developing standards for $N\times 384$ Kbit/s, and $M\times 64$ Kbit/s video-codecs. Compatibility standards for a 384 Kbit/s codec are currently being finalized.

At 384 Kb/s these codecs are expected to carry typical video-conferencing scenes, such as, head and shoulder view of person/group of persons engaged in active conversation. They are also expected to include techniques to transmit high quality text or graphics images. Another standards committee, currently active in standards field is the ISDN. Typical ISDN services at 128 Kb/s are expected to include video transmission capability. As a result efforts to develop video-codecs, that operate around 100 Kb/s may be in progress. Another application of interest is for domestic consumer-market; there is considerable interest in development of inexpensive coders that can operate at 56/64 Kbit/s rates. Such coders are aimed towards use in single user, head and shoulder type video-conferencing application.

An approach actively persued to make motion video-services a reality, is the packet-video concept. In packet-video application, video is expected to be sent as packets of predetermined length, which depending on statistics of network, may allow higher instantaneous data-rates. Some advance knowledge regarding the statistics of packet generation, congestion and flow-control on the network may be helpful in designing such a packet-video system. Various important considerations for such a scheme are: fixed vs variable packet size, packet delay due to network traffic, priority of packets, effect of partial packet recovery, algorithms to compensate for packet loss, may have ultimate effect on the performance. Standards organizations are currently considering some of the issues listed above.

8.3 PROSPECTS AND PROBLEMS

We now discuss some prospects and problems in image coding, and try to predict some general directions in which research in image-processing field is expected to continue. Extensions of schemes that exist currently will probably continue in near

future, with special emphasis on applications. Algorithms will perhaps get more adaptive, as knowledge based techniques find their application and hardware becomes even faster and denser. A list of open problems and possible extensions of this thesis are listed below.

8.3.1 Hybrid Motion-compensation

In this thesis, we have investigated block-matching as the technique for motion-estimation. It however, may require significant overhead for transmission of motion-vectors. in very low-bit rate applications. Differential techniques on the other hand, are pixel based and may not require this overhead. They may however be more sensitive to input noise, lack of luminance gradients, coding noise etc. A technique that combines the block-matching and the differential techniques may be robust and efficient.

8.3.2 Hybrid Cluster/Transform Coding

We introduced the concept of cluster coding, it seems to be a promising technique for coding difficult areas of an image. This approach can perhaps be combined with region segmentation and other artificial intelligence approaches to improve the performance of image classification or coding algorithms.

8.3.3 Motion Compensated Interpolation

In low bit-rate coding, it is often necessary to resort to temporal subsampling to cut down the number of frames that need to be transmitted. For regenerating the image-sequence, interpolation of missing frames is required. One possible technique, that can improve the quality of motion rendition uses motion-compensated

interpolation. Even though many such schemes have been investigated in the past, and seem promising it has been difficult to get good performance.

8.3.4 Non-orthogonal Transforms & Model-based Approach

To exploit the properties of the human-visual system, transforms that match human perception models and characteristics are being investigated. These are non-othogonal transforms and may be better suited to properties of the eye. Some researchers believe that image modeling is the key to improving the performance of coding algorithms. For example, in image-sequence processing, model the person based on some parameters collected and update only that part of the model that changes.

8.3.5 Post-processing

Post-processing deals with cosmetically improving the appearance of the coded images, by use of properties of human visual-system. Efforts are underway to better model and understand the human visual-system, and use its properties to improve the visual perception of coded images. Several researchers are working towards this goal, but the results have been very subjective so far.

REFERENCES

[1] A. N. Netravali and B. G. Haskell, "Digital Pictures - Representation and Compression," *Plenum Press,* 1988.

[2] R. J. Clarke, "Transform Coding of images," *Academic Press,* 1985.

[3] K. R. Rao and R. Srinivasan, eds., "Teleconferencing," *Van Nostrand Reinhold,* 1985.

[4] T. S. Huang, ed., "Image Sequence Analysis," *Springer-Verlag,* 1979.

[5] T. S. Huang, ed., "Image Sequence Processing and Dynamic Scene Analysis," *Springer-Verlag,* 1983.

[6] N. S. Jayant and P. Noll, "Digital Coding of Waveforms: Principles and Applications to Speech and Video," *Prentice Hall,* 1984.

[7] R. C. Gonzalez and P. Wintz, "Digital Image Processing," *Addison-Wesley,* 1977.

[8] N. Ahmed and K. R. Rao, "Orthogonal Transforms for Digital Signal Processing," *Springer-Verlag,* 1975.

[9] W. K. Pratt, "Digital Image Processing," *John-Wiley,* 1978.

[10] W. K. Pratt, Ed., "Image Transmission Techniques," *Academic Press,* 1979.

[11] H. Taub and D. L. Schilling, "Principles of Communication Systems," *McGraw-Hill,* 1971.

[12] A. V. Oppenheim and R. W. Schafer, "Digital Signal Processing," *Prentice-Hall,* 1979.

[13] D. Dudgeon and R. M. Mersereau, "Multidimensional Signal Processing," *Prentice-Hall,* 1984.

[14] A. Papoulis, "Probability, Random Variables and Stochastic Processes" *McGraw-Hill,* 1959.

[15] A. Habibi, "Survey of Adaptive Image Coding Techniques," *IEEE Trans. Commun.,* Vol. COM-25, pp. 1275-1284, Nov. 1977.

[16] J. O. Limb, C. B. Rubenstein and J. E. Thompson, "Digital Coding of Color Video Signals - A Review," *IEEE Trans. Commun.,* Vol. COM-25, pp. 1349-1385, Nov. 1977.

[17] A. N. Netravali and J. O. Limb, "Picture Coding: A Review," *Proc. of the IEEE,* Vol. 68, pp. 366-406, Mar. 1980.

[18] B. G. Haskell and R. Steele, "Audio and Video Bit-Rate Reduction," *Proc. of the IEEE,* Vol. 69, pp. 252-262, Feb. 1981.

[19] A. K. Jain, "Image Data Compression: A Review," *Proc. of the IEEE,* Vol. 69, pp. 349-384, Mar. 1981.

[20] A. K. Jain, "Advances in Mathematical Models for Image Processing," *Proc. of the IEEE,* Vol. 69, pp. 502-528, May 1981.

[21] H. G. Musmann, P. Pirsch and H. J. Grallert, "Advances in Picture Coding," *Proc. of the IEEE,* Vol. 73, pp. 523-548, April 1985.

[22] J. Max, "Quantizing for Minimum Distortion," *IRE Trans. Inform. Theory,* vol. IT-6, pp. 7-12, Mar. 1960.

[23] J. Makhoul, "Linear Prediction: A Tutorial Review," *Proc. of the IEEE,* Vol. 63, pp. 561-580, Apr. 1975.

[24] D. L. Schilling, N. Scheinberg, J. Garodnick, "Video Encoding using Adaptive Delta Modulator," *IEEE Trans. Commun.,* Vol. COM-22, pp. 1682-1689, Nov. 1978.

[25] J. Barba, N. Scheinberg and D. L. Schilling, "A Modified Adaptive Delta Modulator," *IEEE Trans. Commun.,* Vol. COM-29, pp. 1769-1786, Dec. 1981.

[26] R. C. Brainard and A. Puri, "Compact Coder for Component Color Television" to appear in *IEEE Trans. Commun.*.

[27] J. C. Candy, M. A. Franke, B. G. Haskell and F. W. Mounts, "Transmitting Television as Clusters of Frame-to-Frame Differences," *B.S.T.J.*, pp. 1889-1917, Jul-Aug. 1971.

[28] B. G. Haskell, "Entropy Measurements for Nonadaptive and Adaptive Frame to Frame, Linear Predictive Coding of Video-telephone Signals", *B.S.T.J.*, pp. 1155-1173, July 1975.

[29] B. G. Haskell and R. L. Schmidt, "A Low Bit-Rate interframe coder for Video-telephone," *B.S.T.J.*, pp. 1475-1495, Oct. 1975.

[30] B. G. Haskell and J. O. Limb, "Predictive Video Encoding Using Measured Subject Velocity," U. S. Patent 3632865, Jan. 1972.

[31] F. Rocca, "Television Bandwidth Compression Utilizing Frame-to-Frame Correlation and Movement Compensation," in *Picture Bandwidth Compression*, T. S. Huang and O. J. Tretiak, Eds. NY:Gorden and Breach, 1972, pp. 675-693.

[32] J. E. Cunningham, "Frame Correction Coding," in *Picture Bandwidth Compression*, T. S. Huang and O. J. Tretiak, Eds. NY:Gorden and Breach, 1972, pp. 623-653.

[33] J. W. Woods and T. S. Huang, "Picture Bandwidth Compression by Linear Transformation and Block Quantization," in *Picture Bandwidth Compression*, T. S. Huang and O. J. Tretiak, Eds. NY:Gorden and Breach, 1972, pp. 555-573.

[34] D. J. Connor and J. O. Limb, "Properties of Frame-Difference Signals Generated by Moving Images," *IEEE Trans. Commun.*, Vol. COM-22, pp. 1564-1575, Oct. 1974.

[35] M. Miyahara, "Analysis of Perception of Motion in Television Signals and its Application to Bandwidth Compression," *IEEE Trans. Commun.*, Vol. COM-23, pp. 761-766, Jul. 1975.

[36] B. Widrow and J. McCool, "A Comparison of Adaptive Algorithms Based on the Methods of Steepest Descent and Random Search," *IEEE Trans. Ant. Prop*, Vol. AP-24, pp. 615-637, Sept. 1976.

[37] J. O. Limb and J. A. Murphy, "Estimating the Velocity of Moving Images in Television Signals," *Computer Graphics and Image Processing*, Vol. 4, 1975, pp. 311-327.

[38] S. Brofferio and F. Rocca, "Interframe Redundancy Reduction of Video Signals Generated by Translating Objects," *IEEE Trans. Commun.*, Vol. COM-25, pp. 448-455, Apr. 1977.

[39] C. Cafforio and F. Rocca, "Tracking Moving Objects in Television Images," *Signal Processing*, Vol. 1, 1979, pp. 133-140.

[40] C. Cafforio and F. Rocca, "Methods for Measuring Small Displacements for Television Images," *IEEE Trans. Info. Theory*, Vol. IT-22, pp. 573-579, Sept. 1979.

[41] C. Cafforio and F. Rocca, "The Differential Method for Image Motion Estimation," in *Image Sequence Processing and Dynamic Scene Analysis*, T. S. Huang Ed., Berlin, Germany: Springer Verlag, 1983, pp. 76-103.

[42] A. N. Netravali and J. D. Robbins, "Motion-Compensated Television Coding: Part I," *B.S.T.J.*, pp. 631-669, March 1979.

[43] K. A. Prabhu and A. N. Netravali, "Motion-Compensated Component Color Coding," *IEEE Trans. Commun.*, Vol. COM-30, pp. 2519-2527, Dec. 1982.

[44] S. Sabri, K. Cuffing and B. Prasada, "Coding of Video Signals at 50Kb/s Using Motion Compensation Techniques," *Proc. IEEE Mil. Commn. Conf.*, Washington, D.C., Nov. 1983, pp. 809-816.

[45] R. Paquin and E. Dubois, "A Spatio-Temporal Gradient Method estimating the Displacement Field in Time-Varying Imagery," *Computer Vision, Graphics and Image Processing*, Vol. 21, 1983, pp. 205-221.

[46] D. R. Walker and K. R. Rao, "Improved Pel-Recursive Motion-Estimation," *IEEE Trans. Commun.*, Vol. COM-32, pp. 1128-1134, Oct. 1984.

[47] F. May and W. Wolf, "Picture Coding with Motion Analysis for Low Rate Transmission," in *Proc. Int. Commun. Conf.*, Philadelphia, PA, June 1982, pp. 2G.7.1-7.5.

[48] H. C. Bergmann, "Displacement Estimation Based on Correlation of Image Segments," *IEEE Proc. Int. Conf. Electronic Image Process.*, York, England, July 1982, pp. 215-219.

[49] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, Vol. COM-29, pp. 1799-1808, Dec. 1981.

[50] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in *Proc. Nat. Telecommun. Conf.*, New Orleans, LA, Nov. 29-Dec. 3, 1981, pp. G5.3.1-G5.3.5.

[51] Y. Ninomiya and Y. Ohtsuka, "A Motion-Compensated Interframe Coding Scheme for Television Pictures," *IEEE Trans. Commun.*, Vol. COM-30, pp. 201-211, Jan. 1982.

[52] T. Ishiguro and K. Iinuma, "Television Banwidth Compression Transmission by Motion-Compensated Interframe Coding," *IEEE Comm. Mag.*, pp. 24-30, Nov. 1982.

[53] T. Koga, A. Hirano, K. Iinuma, Y. Iijima, and T. Ishiguro, "A 1.5 Mb/s Interframe Codec with Motion Compensation," *Proc. Int. Commun. Conf.*, Boston, MA, June 1983.

[54] K. Matsuda, T. Tsuda, T. Ito, and S. Maki, "A New Motion Compensation Coding Scheme for Video Conferences," in *Proc. Int. Commun. Conf.*, Amsterdam, Holland, May 1984, pp. 234-237.

[55] R. Srinivasan and K. R. Rao, "Predictive Coding Based on Efficient Motion Estimation," *IEEE Trans. Commun.*, Vol. COM-33, pp. 888-896, Aug. 1985.

[56] S. Kappagantula and K. R. Rao, "Motion Compensated Predictive Interframe Coding," *IEEE Trans. Commun.*, Vol. COM-33, pp. 1011-1015, Sept. 1985.

[57] C. D. Bowling and R. A. Jones, "Motion Compensated Image Coding with a Combined Maximum A Posteriori and Regression Analysis," *IEEE Trans. Commun.*, Vol. COM-33, pp. 844-857, Aug. 1985.

[58] R. Lenz, A. Gerhard, "Estimation of Motion Parameters in TV-Scenes using the Auto- and Cross-Subtraction-Function," *Picture Coding Symposium*, Rennes, France, 1984.

[59] M. Bierling, "A Differential Displacement Estimation Algorithm With Improved Stability," *2nd Int. Tech. Symp. on Optical and Electro-Optical Applied Science and Engineering (SPIE Conf.B594 Image Coding),* Cannes, France, Dec. 1985.

[60] D. N. Hein, "Video Compression Using Conditional Replenishment and Motion Prediction," *PhD Dissertation,* Kansas State University, Manhattan, Kansas, 1981.

[61] D. N. Hein and N. Ahmed, "Video Compression Using Conditional Replenishment and Motion Prediction," *IEEE Trans. Electromagn. Compat.,* Vol. EMC-26, pp. 134-142, Aug. 1984.

[62] J. J. Y. Huang and P. M. Schultheiss, "Block Quantization of Correlated Gaussian Random Variables," *IEEE Trans. Commun. Systems,* Vol. CS-11, pp. 280-296, Sept. 1963.

[63] W. D. Ray and R. M. Driver, "Further Decomposition of the Karhunen-Loeve Series Representation of a Stationary Random Process," *IEEE Trans. Inform. Theory,* Vol. IT-11, pp. 663-668, Nov. 1970.

[64] P. A. Wintz, "Transform Picture Coding" *Proc. IEEE Trans. Comput.,* Vol. C-23, pp. 90-93, Jan. 1974.

[65] T. Natarajan and N. Ahmed, "On Interframe Transform Coding," *IEEE Trans. Commun.,* Vol. COM-25, pp. 1323-1329, Nov. 1977.

[66] J. A. Roese, W. K. Pratt and G. S. Robinson, "Interframe Cosine Transform Image Coding," *IEEE Trans. Commun.,* Vol. COM-25, pp. 1329-1339, Nov. 1977.

[67] A. Habibi, "An Adaptive Strategy for Hybrid Image Coding," *IEEE Trans. Commun.,* Vol. COM-29, pp. 1736-1740, Dec. 1981.

[68] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform," *IEEE Trans. Comput.,* Vol. C-23, pp. 90-93, Jan. 1974.

[69] T. R. Natarajan and N. Ahmed, "On Interframe Transform Coding," *IEEE Trans. Commun.,* Vol. COM-25, pp. 1323-1329, Nov 1977.

[70] J. A. Roese W. K. Pratt and G. S. Robinson, "Interframe Cosine Transform Image Coding," *IEEE Trans. Commun.,* Vol. COM-25, pp. 1329-1339, Nov 1977.

[71] W. H. Chen and C. H. Smith, "Adaptive Coding of Monochrome and Color Images," *IEEE Trans. Commun.,* Vol. COM-25, pp. 1285-1292, Nov 1977.

[72] A. K. Jain, "A Sinusoidal Family of Unitary Transforms," *IEEE Trans. Pat. Anal. Machine Intell.,* Vol. PAMI-1, pp. 356-365, Oct 1979.

[73] K. N. Ngan, "Adaptive Transform Coding of Video Signals," *IEE Proc.,* Vol. 129, Pt. F, pp. 28-40, Feb. 1982.

[74] R. C. Reininger and J. D. Gibson, "Distributions of the Two-Dimensional DCT Coefficients for Images," *IEEE Trans. Commun.*, Vol. COM-31, pp. 835-839, July 1983.

[75] W. H. Chen and W. K. Pratt, "Scene Adaptive Coder," *IEEE Trans. Commun.*, Vol. COM-32, pp. 225-232, March 1984.

[76] W. A. Pearlman and P. Jakatdar, "The Effectiveness and Efficiency of Hybrid Transform/DPCM Interframe Image Coding," *IEEE Trans. on Commun.*, Vol. COM-32, pp. 832-838, July 1984.

[77] R. J. Clarke, "Relation between the Karhunen Loeve and Cosine Transforms," *IEE Proc.*, Vol. 128, Pt. F, pp. 359-360, Nov. 1981.

[78] A. Z. Meiri and E. Yudilevich, "A Pinned Sine Transform Image Coder," *IEEE Trans. on Commun.*, Vol. COM-29, pp. 1728-1735, Dec. 1981.

[79] E. J. K. Bisherurwa and F. P. Coakley, "New Fast Discrete Sine Transform with Offset," *Electronics Letters,* Vol. 17, pp. 803-805, Oct. 15, 1981.

[80] R. J. Clarke, "Performance of Karhunen-Loeve and Discrete Cosine Transforms for Data having Widely Varying Values of Intersample Correlation Coefficient," *Electronics Letters,* Vol. 19, pp. 251-253, Mar. 31, 1983.

[81] R. J. Clarke, "Application of Sine Transform in Image Processing," *Electronics Letters,* Vol. 19, pp. 490-491, June 23, 1983.

[82] W. K. Cham, D. Allott and R. J . Clarke, "Block Classification Image Coding with Combined Transforms," *Proc. Int. Conf. Acc. Speech Signal Processing,* pp. 29.10.1-29.10.4, Dec. 1984.

[83] R. J. Clarke, "Relation between the Karhunen Loeve and Sine Transforms," *Electronics Letters,* Vol. 20, pp. 12-13, Jan. 5, 1984.

[84] E. Dubois and S. Sabri, "Noise Reduction in Image Sequences Using Motion Compensated Temporal Filtering," *IEEE Trans. on Commun.*, Vol. COM-32, pp. 826-831, July 1984.

[85] P. Santago, and S. A. Rajala, "Techniques for Video Compression by Projection onto Convex Sets," *Proc. SPIE 28th Annual Symposium on Optics and Electro-Optics,* San Diego, CA, Aug. 1984, pp. 432-435.

[86] M. R. Civanlar and P. Santago, "An Improved Transform Coder for Image Sequences Using Attributes of Difference Pictures," in *Proc. Int. Conf. Acc. Speech Signal Processing,* pp. 10.2.1-10.2.4, Tampa, Fl, Mar. 1985.

[87] P. Santago, and S. A. Rajala, "New Techniques for Combined Pixel and Frequency Domain Image Sequence Coding," *Proc. IEEE Mil. Commn. Conf.*, Nov. 1985, pp. 624-627.

[88] S. Ericsson, "Motion-Compensated Hybrid Coding at 50 kb/s," in *Proc. Int. Conf. Acc. Speech Signal Processing,* pp. 10.8.1-10.8.4, Tampa, Fl, Mar. 1985.

[89] S. Ericsson, "Fixed and Adaptive Predictors for Hybrid Predictive/Transform coding," *IEEE Trans. Commun.,* Vol. COM-33, pp. 1291-1302, Dec. 1985.

[90] A. Furukawa, T. Koga and K. Niwa, "Coding Efficiency Analysis for Motion-Compensated Interframe DPCM with Transform Coding," in *Proc. IEEE Globecom,* pp. 689-693, New Orleans, Louisiana, Dec. 1985.

[91] B. G. Haskell and H. M. Hang, "Comparison of Discrete Cosine Transform Vector Quantization of Medical Imagery," in *Proc. SPIE, Application of Optical Instr. in Med. and Pic. Arch.* and Commn. Sys. for Med. Application," Vol. 626 pp. 399-408, Newport Beach, CA, Feb. 1986.

[92] S. Carlsson and C. Reillo, "Contour Based Representation of the Displacement Field for Motion-Compensated Image Coding," in *Proc. Int. Conf. Acc. Speech Signal Processing,* pp. 161-164, Tokyo, Japan, Apr. 1986.

[93] M. Ohta, T. Mochizuki and T. Koga, "An Adaptive Hybrid Coding with Motion Compensation and DCT," in *Picture Coding Symposium,* pp. 83-84, Tokyo, Japan, Apr. 1986.

[94] M. Kaneko, Y. Hatori, A. Koike and H. Yamamoto, "Improvement of Transform Coding Algorithm for Motion Compensated Frame-to-Frame Difference Signals," in *Proc. IEEE Globecom,* pp. 276-280, Houston, Texas, Dec. 1986.

[95] M. Ohta and T. Koga, "Adaptive VWL Coding of Transform Coefficients for Subprimary Rate Video Transmission," in *Proc. IEEE Globecom,* pp. 271-275, Houston, Texas, Dec. 1986.

[96] A. N. Netravali and J. D. Robbins, "Motion-Adaptive Interpolation of Television Frames," *Picture Coding Symposium 1981,* Montreal, Canada, June 1981, pp 12.2.

[97] H. C. Bergmann, "Motion Adaptive Frame Interpolation," in *Proc. Int. Zurich Seminar on Digital Communications,* Zurich, Switzerland, Mar. 1984, pp. 57-61

[98] A. Furukawa, T. Koga and K. Iinuma, "Motion-Adaptive Interpolation for Video-Conferencing Pictures," *Proc. Int. Conf. Commn.,* Amsterdam, Holland, May 1984, pp. 707-710.

[99] B. Girod, R. Thoma, "Motion-Compensating Field Interpolation from Interlaced and Non-interlaced Grids," *2nd Int. Tech. Symp. on Optical and Electro-Optical Applied Science and Engineering (SPIE Conf.B594 Image Coding),* Cannes, France, Dec. 1985.

[100] M. Bierling, R. Thoma, "Motion-Compensating Field Interpolation Using a Heirarchically Structured Displacement Estimator," *Signal Processing,* Vol. 11, 1986, pp. 387-404.

[101] A. Puri, H. M. Hang and D. L. Schilling, "An Efficient Block-Matching Algorithm for Motion Compensated Coding," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* pp.25.4.1-25.4.4, Dallas, Texas, April 1987.

[102] A. Puri, H. M. Hang and D. L. Schilling, "Motion-Compensated Transform Coding based on Block Motion Tracking Algorithm," *Proc. IEEE Int. Conf. Commun.,* pp.5.3.1-5.3.5, Seattle, Wash., June 1987.

[103] A. Puri, H.-M. Hang, and D. L. Schilling, "Interframe coding with Variable Block-size Motion-compensation," *Proc. IEEE Global Commun. Conf.,* pp. , Tokyo, Japan, Nov. 1987.

[104] K. Matsuda, T. Tsuda, T. Itoh and O. Kawai, "A Study of a Vector Quantizer for Sub-rate Video Codec," in *Proc. IEEE Globecom,* pp. 261-265, Houston, Texas, Dec. 1986.

[105] Y. Kato, N. Mukawa, S. Okubo, H. Hashimoto and H. Yasuda "Discrete Cosine Transform Coding for Video-conferencing using Vector and Scalar Quantization," in *Proc. IEEE Globecom,* pp. 266-270, Houston, Texas, Dec. 1986.

[106] Y. Kato, N. Mukawa and S. Okubo, "A Motion Picture Coding Algorithm Using Adaptive DCT Encoding Based on Coefficient Power Distribution," *IEEE Jrnl. on Selected Areas in Commun.,* Vol. SAC-5, pp. 1090-1099, Aug. 1987.

[107] M. Kaneko, Y. Hatori and A. Koike, "Improvements of Transform Coding Algorithm for Motion-Compensated Interframe Prediction Errors - DCT/SQ Coding," *IEEE Jrnl. on Selected Areas in Commun.,* Vol. SAC-5, pp. 1068-1078, Aug. 1987.

[108] T. Koga and M. Ohta, "Entropy Coding for a Hybrid Scheme with Motion Compensation in Subprimary Rate Video Transmission," *IEEE Jrnl. on Selected Areas in Commun.,* Vol. SAC-5, pp. 1166-1174, Aug. 1987.

[109] P. R. Cohen and E. A. Feigenbaum, Ed., "The Handbook of Artificial Intelligence," vol III, (Region Analysis, p 225), *William Kaufmann, Inc.* 1981