PREDICTIVE DECODING FOR DELAY REDUCTION IN VIDEO COMMUNICATIONS

by

Yue-Meng Chen
B.Eng in Electronics Engineering, Zhejiang University, P.R.China

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

In the School
of
Engineering Science

SIMON FRASER UNIVERSITY

Fall 2007

APPROVAL

Name:	**Yue-Meng Chen**
Degree:	**Master of Applied Science**
Title of Thesis:	**Predictive decoding for delay reduction in video communications**

Examining Committee:

Chair: **Dr. Atousa HajShirMohammadi**
Lecturer, School of Engineering Science

Dr. Ivan Bajic
Senior Supervisor
Assistant Professor, School of Engineering Science

Dr. Jie Liang
Supervisor
Assistant Professor, School of Engineering Science

Dr. Jiangchuan Liu
Examiner
Assistant Professor, School of Computing Science

Date Defended/Approved: Nov 22, 2007

ABSTRACT

Low delay is critically important for interactive video communication. Unpredictable delays and bursty traffic in today's networks may significantly degrade the performance of interactive video services. This thesis presents several predictive decoding techniques for delay reduction. The basic idea is to predict future video frames from past video data, and display them before they arrive at the decoder. Inevitably, this will reduce the quality of the displayed frames somewhat, but it will also enable the user to choose the proper trade-off between quality and delay.

The frame prediction module was implemented as an add-on to the XviD version of MPEG-4 video decoder, and tested on a variety of standard sequences. The performance highly depends on the characteristics of the sequence, such as motion intensity, frame rate, resolution, etc. Our results indicate that in most cases, it is possible to reduce the perceived end-to-end communication delay by about 100 ms while maintaining reasonable video quality.

Keywords: Communication delay, Motion Segmentation, Temporal prediction, Variable-block-size motion prediction

DEDICATION

To my parents and my beloved wife

ACKNOWLEDGEMENTS

This thesis would not have been possible without the support and guidance of many people.

First, I would like to thank my supervisor Dr. Ivan Bajic, for giving opportunity to work with him, for guiding me toward interesting projects and for his involvement and interest in my progress. I also learn a lot from his methodical and critical approach to research problems. I would like to thank Dr. Jie Liang for motivating me to learn more about multimedia through his source coding class, talks, and constructive feedback. I would also thank Dr. Jiangchuan Liu for the patience to read this thesis and contributing to its content. I would like to thank Dr. Atousa HajShirMohammadi for chairing the defence.

I would also thank to my fellow graduate students in multimedia communication research group at Simon Fraser University.

As this work comes closer to completion, I start to look forward to some good research work in my PhD program, that will hopefully be both useful and intellectually pleasing. Therefore, I would see this dissertation as a valuable learning process, and the start of my effort toward PhD research.

TABLE OF CONTENTS

Approval ii

Abstract iii

Dedication iv

Acknowledgements v

Table of Contents vi

List of Figures viii

List of Tables x

Chapter 1 Introduction 1

1.1 Motivation 1
1.1.1 Interactive Video Applications 1
1.1.2 Impairments Introduced by the Network 4
1.1.3 Impact of Delay on Video Performance 5
1.1.4 Questions 6
1.2 Approach 6
1.2.1 Error Concealment Techniques 7
1.2.2 Predictive Decoding 8
1.3 Thesis Contribution and Outline 9

Chapter 2 Predictive Decoder Architecture and Design 12

2.1 Frame Prediction Module 14
2.1.1 Add-on Module to Standard Video Decoders 17
2.2 Prediction Module Architecture 18
2.3 Frame Prediction Procedure 20

Chapter 3 Adaptive Frame Prediction Module 24

3.1 System and Interface Aspects 24
3.2 Motion Segmentation 25
3.2.1 Motivations 25
3.2.2 Clustering Threshold Estimator 28
3.2.3 Seed Block Pattern 30
3.2.4 Region Growing Criterion 32
3.2.5 Motion Region Smoothing 34
3.2.6 The Processing Steps 35
3.3 Variable-block-size Motion Prediction 37
3.4 Temporal Prediction 42
3.5 Post-processing for Overlapped Areas 43
3.6 Post-processing for Empty Areas 45

Chapter 4 Implementation and Smulation Rsults..50
4.1 Test Sequences..50
4.2 The Assessment of Motion Segmentation 51
4.3 Assessment Methodology...54
4.3.1 Method 0: Zero Motion Vector..54
4.3.2 Method 1: Motion Extension...54
4.3.3 Method 2: Motion Segmentation, VMF, Sequential Frame Synthesis...56
4.3.4 Method 3: VBS Motion Prediction and Region Smoothing 56
4.4 The Overall Assessment of Frame Prediction..................................57
4.4.1 The Objective Quality 57
4.4.2 The Subjective Quality..71
4.5 The Complexity Analysis...77

Chapter 5 Conclusion and Future Directions 82
5.1 Possibilities for Improving Prediction Performance...........................83
5.2 The Frame Prediction in H.264/AVC...85
5.3 Comprehensive Video Quality Assessments 86

Reference List 89

LIST OF FIGURES

Figure 1.1: The Teleconference System 2
Figure 1.2: The network impairments to video teleconference 4
Figure 1.3: Perceived delay reduction by frame prediction. 9
Figure 2.1: Jitter buffer for video packets. 13
Figure 2.2: Video receiver system diagram 15
Figure 2.3: Prediction module in the standard MPEG-4 decoder. 17
Figure 2.4: Internal architecture of prediction module 19
Figure 2.5: The error propagation of Motion Prediction. 21
Figure 2.6: Sequential frame prediction mechanism. 22
Figure 3.1: System and interface aspects 25
Figure 3.2: Diagram of motion segmentation 28
Figure 3.3: The seed pattern 31
Figure 3.4: The region growing directions 32
Figure 3.5: Typical Motion Vector Noise 34
Figure 3.6: VMF-based MV filtering 35
Figure 3.7: The system diagram of Variable-Block-Size Motion Prediction 37
Figure 3.8: The motion vector prediction for 4x4 sub-blocks 40
Figure 3.9: MV prediction for overlapped areas 41
Figure 3.10: A preliminary version of the future frame 42
Figure 3.11: Boundary matching for overlapped areas. 44
Figure 3.12: Filling thin empty areas. 46
Figure 3.13: Filling thick empty areas 47
Figure 3.14: Illustration of empty area post-processing where [top left] is the intermediate frame after post-processing of overlapped areas; [top right] is the frame after filling thin vertical empty areas; [bottom left] is the frame after filling thin horizontal empty areas; [bottom right] is the final frame after filling thick empty areas. 49
Figure 4.1: The motion segmentation result: *Garden* 52

Figure 4.2: The motion segmentation result: *Coastguard* 52
Figure 4.3: The motion segmentation result: *Tennis* 53
Figure 4.4: The motion segmentation result: *Football* 53
Figure 4.5: MV inversion followed by motion extension. 55
Figure 4.6: Prediction performance in PSNR (dB) – High Motion Level 58
Figure 4.7: Prediction performance in PSNR (dB) – Medium Motion Level.........59
Figure 4.8: Prediction performance in PSNR (dB) – Low Motion Level..............59
Figure 4.9: Prediction performance VS. Frame Rate – High Motion Level..........61
Figure 4.10: Prediction performance VS. Frame Rate – Medium Motion Level 61
Figure 4.11: Prediction performance VS. Frame Rate – Low Motion Level.........62
Figure 4.12:The PSNR of frame prediction vs. The motion level – *Carphone* 64
Figure 4.13:The PSNR vs. The motion level – *Foreman* 64
Figure 4.14:The PSNR vs. The motion level – *Mother & Daughter* 65
Figure 4.15:The PSNR vs. The motion level – *Miss America* 65
Figure 4.16: Prediction performance vs. Motion Density: *Foreman*, Left: Same bit-rate, Right: Same encoded PSNR. 69
Figure 4.17: Prediction performance vs. Motion Density: *Carphone*, Left: Same bit-rate, Right: Same encoded PSNR. 69
Figure 4.18: Prediction performance vs. Motion Density: *Mother & Daughter*, Left: Same bit-rate, Right: Same encoded PSNR. 70
Figure 4.19: Prediction performance vs. Motion Density: *Miss America*, Left: Same bit-rate, Right: Same encoded PSNR. 70
Figure 4.20: Visual comparison for frame prediction assessment. [Top left] is the frame without prediction, and other figures show the same frame produced when predicting three frames ahead by three prediction methods. [Top right]: Method 1, [Bottom left]: Method-2, [Bottom right]: Method-3 72
Figure 4.21: Quality of frame prediction (*Foreman*)..73
Figure 4.22: Quality of frame prediction (*Flower Garden*). 74
Figure 4.23: Quality of frame prediction (*Mother & Daughter*)............................75
Figure 4.24: MIPS Dissipation of frame prediction (*High Motion*).......................79
Figure 4.25: MIPS Dissipation of frame prediction (*Medium Motion*). 80
Figure 4.26: MIPS Dissipation of frame prediction (*Low Motion*)........................80

LIST OF TABLES

Table 1.1: Several ping RTTs from Vancouver. 5
Table 4.1: Test sequences. 51
Table 4.2: Bit rates needed to maintain the same video quality. 68
Table 4.3: Decoding and prediction time. 77

CHAPTER 1
INTRODUCTION

1.1 Motivation

1.1.1 Interactive Video Applications

As one of the greatest inventions of the 20th century, the Internet and its applications have expanded significantly in the past decades to become the basis for personal, economic, and political advancement. Internet applications have evolved from simple communication tools, like email and ftp, to advanced communication services, such as interactive multimedia applications and media streaming. As the advances in connectivity, geographical reach and access networks continue, multimedia communications on the Internet are gaining more attention than ever before.

Interactive video applications, such as video phone and video conference, have become a feasible alternative to traditional telephony services. The block diagram of a typical teleconference system is shown in Fig. 1.1. Video conferencing is becoming increasingly popular for reporting news from remote locations, for business meetings with participants at multiple sites, and for virtual classrooms in distance education. At the same time, interactive video applications are very demanding; they require low latency, good visual and audio quality, and accessibility by a variety of devices, to provide better user experience compared to traditional telephony applications.

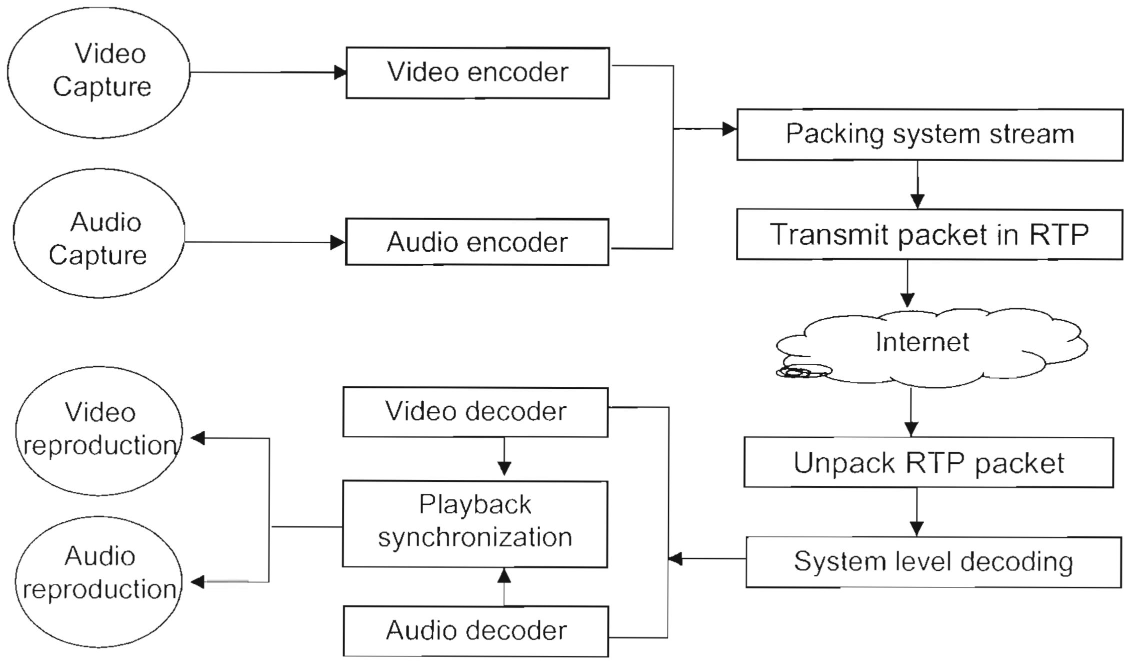


Figure 1.1: The teleconference system

Video characteristics and requirements. Video phone and video conferencing use the H.261, H.263, H.263+ and H264 standards that are designed specially to meet the delay constraints and very low bit rates. Low frame rate, e.g. 15 frames per second, and low resolution (CIF or QCIF) are typically used to enable real-time encoding. At the transport level, RTP (Real Time Protocol) is usually chosen to packetize the encoded video sequence, using an appropriate RTP profile [9]. To maintain good video quality, certain requirements in terms of loss, delay and delay variability need to be satisfied. First, low loss or no loss is desirable to maintain good video quality. Second, the temporal and spatial dependencies in the compressed video bring the risk of error propagation due to packet loss, thereby increasing loss sensitivity. Furthermore, in order to maintain good interactivity, especially for "lip-synchronization" between video and audio, the end-to-end video delay needs to be similar to the end-to-end audio delay. Finally, for a smooth video playback, little or no delay jitter is needed.

Audio characteristics and requirement. Packet voice has been studied intensively in the past two decades, and ITU-T has standardized a series of speech vocoder for voice over IP (VoIP) as well as video conferencing. These vocoders include G.711, G.726, G.729, G.723.1 [41][42][43][44] and produce a variety of bit rates from 5.3 kbps to 64 kbps. In VoIP, RTP is used to transport speech packets across an IP network at a fixed rate. The requirements for voice transmission are similar to those of video. First, low packet loss is needed to maintain good speech quality, and some error concealment algorithms are available that make a slight loss of up to 10% [10] tolerable. Second, the end-to-end delay for interactive voice conference should be lower than 150 ms to achieve an acceptable quality [17]. Echo-cancellation is a unique requirement in speech processing to get rid of unwanted acoustic echo caused by large end-to-end delay. Finally, if the audio is accompanied by the corresponding video signal, delay jitter should be kept under control to maintain lip-synchronization with video frames.

1.1.2 Impairments Introduced by the Network

A summary of impairments introduced by network transmission of audio and video is illustrated in Fig. 1.2. The importance of having low packet loss, low delay and low delay variability for good speech and video quality is well-known. However, today's Internet cannot guarantee any of these requirements. The experiments with interactive video applications show that any delay beyond 150 ms greatly worsens the user experience; delays above 400 ms make interactive communication virtually impossible [45].

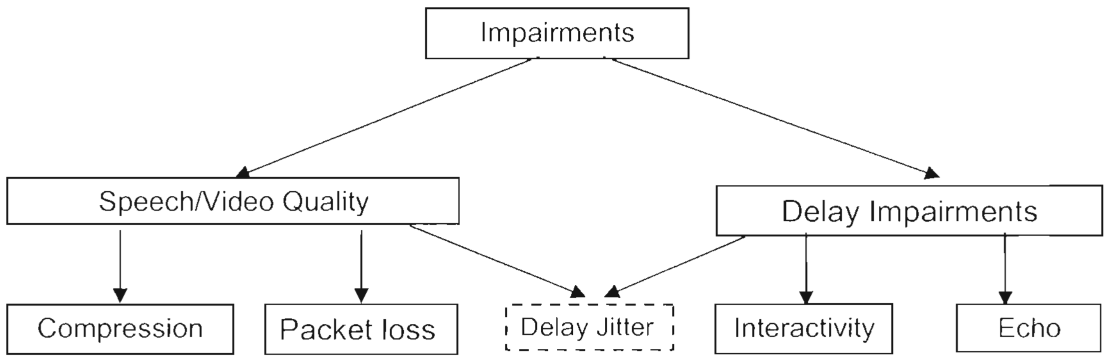


Figure 1.2: The network impairments to video teleconference

The overall end-to-end delay is composed of several components, including acquisition, processing at the transmitting and receiving end, processing in the intermediate nodes (routers), coding, decoding, and pure transmission delay due to the finite speed of the information carrier (electromagnetic waves). In the Internet, transmission delay alone may approach 150 ms, as indicated in Table 1.1.

Table 1.1: Several ping RTTs from Vancouver.

Host	RTT (ms)
ucla.edu	63
mit.edu	113
bbc.co.uk	187
epfl.ch	231
monash.edu.au	280
canterbury.ac.nz	304

The table shows several Round Trip Times (RTTs) measured on the afternoon of Feb. 24, 2007, by pinging various hosts from the Blenz Café in Yaletown (Vancouver, BC), using the FatPort broadband wireless access network. In this scenario, one can expect about 150 ms one-way transmission delay between Vancouver and New Zealand.

1.1.3 Impact of Delay on Video Performance

As interactive video applications become more popular, more attention is being paid to the impact of the delay impairments on the performance of video communication systems. Several research works have been conducted to characterize video traffic over data networks by using traffic measurement methods or simulation. The performance of H.261 and MPEG2 is studied over 10Base-T and 100Base-T Ethernet via simulation in [22]. The packet loss and delay jitter are studied by sending RTP/UDP-packetized MPEG video over the Internet between sites in Europe and the USA in [23]. A more recent study, [24], conducted a very large scale measurements by streaming MPEG-4 video to more than 600 sites in the US. Reference [25] assessed the quality of multimedia communications over the Internet backbone networks. All these research works assessed the network delay impairments from perspectives of different video compression standards, transmission scenarios as well as network environments.

In addition to the network delay in transmission, the interactive video system includes other delay components, such as i) acquisition, digitization ii) encoding by a real-time encoder, packetization iii) unpacking, decoding, playout at the receiver. Reference [17] gives an overview of the specific video delay components. However, video acquisition and encoding typically take longer than audio acquisition and encoding, so in practice, audio is often purposefully delayed in order to maintain lip synchronization [17].

1.1.4 Questions

Many interesting and challenging research questions arise when we try to provide high-quality audiovisual communication over the Internet. Packet loss, variable bandwidth and delay, as well as heterogeneity of end-user equipment all contribute to this problem. In this work, we particularly focus on the problem of delay in interactive video communications. As illustrated above, this delay can be significant.

1.2 Approach

In order to tackle the problem of end-to-end delay, we introduce the concept of predictive decoding [1]. In this scenario, video decoder predicts future video frames and displays them before they actually arrive. Since a frame that has not arrived at the decoder can be thought of as a frame subject to 100% loss, the problem of predictive decoding resembles error concealment [15]. Hence, as we will elaborate in this thesis, predictive decoding employs some error concealment techniques in its key processing unit. However, predictive decoding is more challenging than typical error concealment, since no information about the frame that is to be predicted is available at the decoder. In the following sections, we will first briefly review some popular error concealment algorithms, and then introduce predictive video decoding.

1.2.1 Error Concealment Techniques

Boundary Matching Algorithm (BMA) is a commonly used method in video packet loss recovery. Based on the boundary matching principle, the missing macroblock (MB) is recovered from previous or future video frames [11][12][13]. A variety of BMAs have been studied by researchers, and they in general achieve an excellent trade-off between complexity and visual quality.

Multiple reference temporal error concealment is another common method for better video packet recovery [14], where multiple reference frames are used in motion field interpolation (MFI) techniques.

Median motion vector concealment is also used in temporal error concealment algorithms to give better performance in both objective and subjective quality during video transmission [16].

These methods are designed to recover the packet loss for video transmission over packet network in error-prone environment. When part of a frame, which consists of multiple macroblocks, is missing due to packet loss, these methods can recover the missing area with a relatively good visual quality. However, we may ask what if the entire frame is missing? Or what if the frame arrives at the receiver too late for playout? Can we predict or recover the entire frame in these cases? This thesis gives a possible answer by proposing predictive decoding.

1.2.2 Predictive Decoding

Reducing the delay associated with video would be beneficial for interactive video applications. In this thesis, we present several frame prediction techniques that can help reduce the perceived transmission delay of video. Using the received video data, future frames are predicted and displayed before they arrive at the decoder, as illustrated in Fig. 1.3.

A future frame, which hasn't been received yet, may be thought of as a frame subject to 100% loss. Hence, some of the methods adopted in our frame prediction resemble popular error concealment techniques, taking advantage of spatial and temporal correlation. The predicted video frames are generated using one or more recently reconstructed frames, and their motion information.

This process inevitably reduces the quality of the displayed frames, especially when the motion is complex. But it also provides the user with the ability to trade-off quality for delay. Our results indicate that using these methods, it is possible to reduce the perceived end-to-end video delay by about 100 ms while maintaining reasonable video quality.

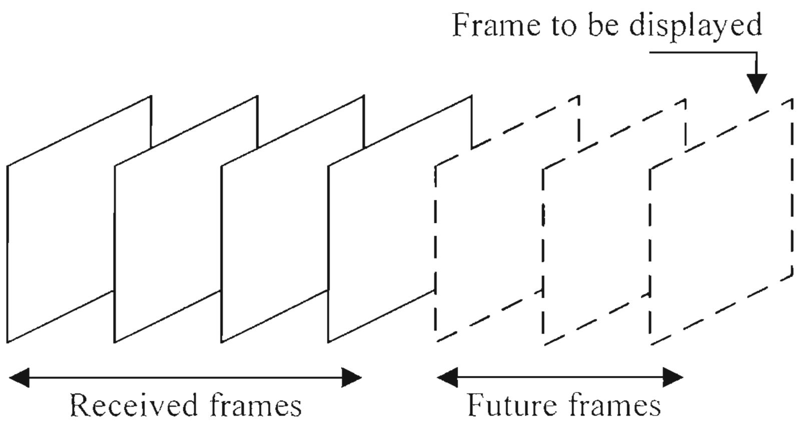


Figure 1.3: Perceived delay reduction by frame prediction.

1.3 Thesis Contribution and Outline

This thesis proposes predictive decoding for delay reduction in video communication, and examines its properties and performance on a variety of sequences with varying motion complexity. The thesis contributions and outline are summarized below.

Thesis contributions

The first contribution is the proposed framework for synthesizing the future frame based on motion information in previously reconstructed video frames. This framework essentially consists of motion vector prediction, temporal prediction and frame post-processing.

Our second contribution is the use of motion segmentation to improve motion vector prediction. Motion segmentation is implemented on a block-level, using variable block sizes. Its goal is to isolate all moving objects from the background area. Frame synthesis incorporating motion segmentation shows much better video quality with greatly reduced prediction noise level.

Finally, we introduced variable-block-size motion vector prediction to enhance the edges of moving objects. Frames synthesized in this way show better subjective quality on sequences with high motion levels.

Thesis Outline

The thesis is organized as follows. In Chapter 2, we present the system architecture of the proposed frame prediction module. In Section 2.1, we show how to interface the frame prediction module with standard video decoder in interactive video applications. In Section 2.2, we show the internal architecture of the prediction module. In Section 2.3, we present the frame prediction procedure.

In Chapter 3, we elaborate on the key processing steps in frame prediction. In Section 3.1, we briefly introduce the system and interface aspects in frame prediction module. In Section 3.2, we present the motion segmentation algorithm, including its control flow, seed pattern, region growing method, motion smoothing and its visual quality. In Section 3.3, we present variable-block-size motion prediction algorithm, and how it relates to segmented motion objects. In Section 3.4, we describe the temporal prediction method. In Sections 3.5 and 3.6, we present the frame post-processing unit for overlapped area as well as empty area, and we discuss how to combine linear interpolation and boundary matching algorithm to give a good trade-off between computation complexity and video quality.

In Chapter 4, we present the implementation of the frame prediction on XviD MPEG4 codec and some simulation results. The test sequences cover a variety of motion levels, and test cases are also categorized in terms of different prediction techniques and complexity. Finally, in Chapter 5, we conclude the thesis, summarize our findings, and discuss future directions.

CHAPTER 2
PREDICTIVE DECODER ARCHITECTURE AND DESIGN

Nowadays, video communications over the Internet are more prevalent than ever before. Video delivery methods vary in a variety of ways, from video clip downloading, through one-way video streaming (e.g., live newscasts), to interactive video applications (e.g., videoconferencing, video phone, etc.). Increased popularity of these applications has resulted in a significant increase in video traffic on the Internet in the past couple of years. This thesis addresses one of the crucial aspects of interactive video communications – that of the perceived end-to-end communication delay.

The perceived end-to-end delay and video quality are two main criteria by which interactive video applications are evaluated. First, low delay is crucial for effective interactive communication. For example, to maintain lip-synchronization, the delays associated with the audio stream and the video stream need to be approximately the same. In an interactive video application, the adaptive jitter buffers located in both receivers are carefully designed to deal with delay issues, and to smooth out any variations in delay. These buffers perform two important tasks: 1) they receive incoming audio and video packets and sort them in the proper order, and 2) they schedule the packets for decoding and playout. The size of jitter buffers is usually managed adaptively, but to maintain delay as low as possible they need to be relatively small, thus a late packet might not get the chance to be decoded and played out on time. Fig. 2.1 shows the buffer structure. The question raised here is how we can use the late received video packet for current playback instead of throwing it away.

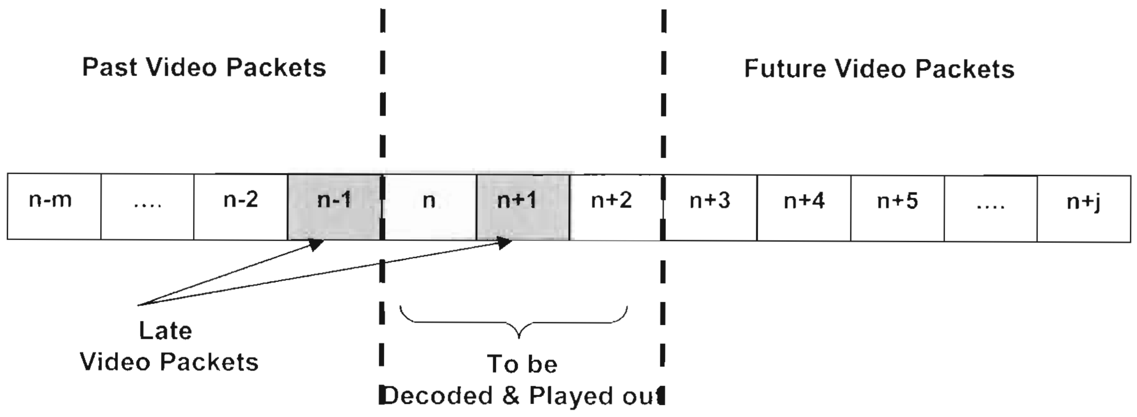


Figure 2.1: Jitter buffer for video packets.

The acceptable video quality is another requirement for interactive video communication, especially the temporal smoothness. According to [17], the performance degradation caused by video quality is less severe than that of large delay, and a moderate amount of compression and transmission artefacts is tolerable. Therefore, reducing the delay while maintaining acceptable video quality becomes the most important challenge for interactive video communication.

Most video compression methods are vulnerable to packet loss due to temporal dependency among different pieces of compressed video bitstream. A lost packet, or a late packet, can cause error propagation in the motion-compensated prediction loop at the decoder, and can even affect the other parts of the same frame due to intra prediction. In the worst case, the entire frame may fail to decode. Therefore, besides the traditional error recovery techniques [15], we need to consider delay control mechanisms to reduce the delay as well as improve the video quality.

In this Chapter, we will present an overview of our adaptive frame prediction at the decoder. In Section 2.1, we describe how the proposed frame prediction module fits into standard video decoders. In Section 2.2, we give an overview of the frame prediction module and its internal architecture. Finally, in Section 2.3 we explain the prediction procedure.

2.1 Frame Prediction Module

The block diagram shown in Fig. 2.2 represents a typical video receiver with adaptive frame prediction module inserted in its architecture. The system shown here is an extension of a standard video decoder, and the frame buffer is taken over by adaptive frame predictor for playout control.

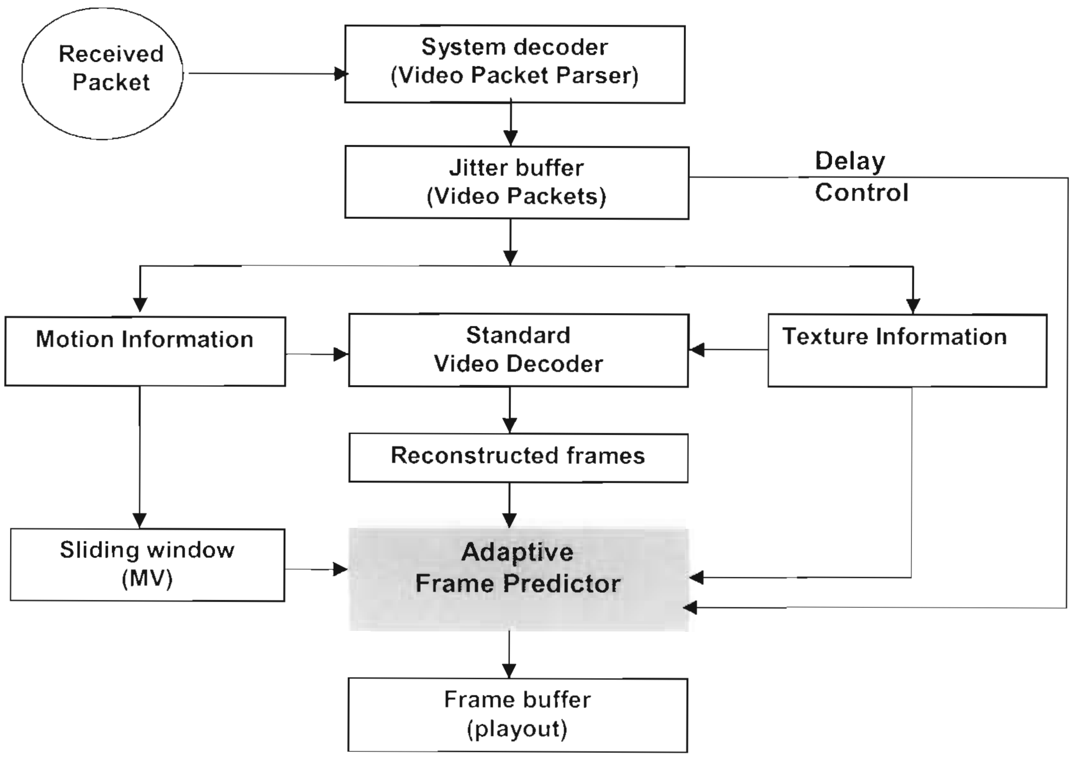


Figure 2.2: Video receiver system diagram

The interface design to the existing video receivers include the following aspects:

1. Motion vectors (MVs) extracted from past video packets are used to build statistics of historical motions, where a sliding window is used to control the motion tracing length.

2. Texture information, which includes the residual information of temporal prediction from past video frames, is combined with MVs to aid in motion prediction.

3. Reconstructed video frames form the basis for temporal predictor to synthesize future frames.

4. The final predicted frame is sent to the frame buffer for the actual playout.
5. The prediction depth is signalled by the delay monitor in the jitter buffer.

The whole module tries to make the best use of the available information so that a number of future frames can be predicted under different network delay condition. Jitter buffer sends its delay estimates to the prediction module, which then predicts a suitable number of future frames to maintain similar video and audio delay level for lip-synchronization. When the end-to-end delay is acceptable and no prediction needs to be done, the prediction module can be bypassed easily, so the last decoded video frame will be directly sent to the frame buffer for display.

A unique feature of this prediction module is that all the modifications are limited to the receiver itself, and the prediction procedure is essentially an open-loop system. By designing the frame prediction in this way, we can save all effort on the encoder side, thus the delay caused by the real-time encoder won't get any worse.

A variety of video codecs have been used in interactive video applications, such as H.261, H263, H263+ [18][19], MPEG4 [2], and H264 [3]. They all rely on block-based motion-compensated prediction coding, which makes it possible for us to unify the interface between the prediction module and a standard video decoder. Therefore, our frame prediction module is designed to be an "add-on" to standard video decoders without significant impact on video codec architecture.

In this thesis, MPEG4 video decoder is the sample platform we work on to develop the frame prediction algorithm.

2.1.1 Add-on Module to Standard Video Decoders

MPEG-4 [2], one of the popular video standards in video communications, has been chosen as the implementation platform in this work. Fig. 2.3 shows the block diagram of a typical MPEG-4 video decoder and indicates where the proposed prediction module fits.

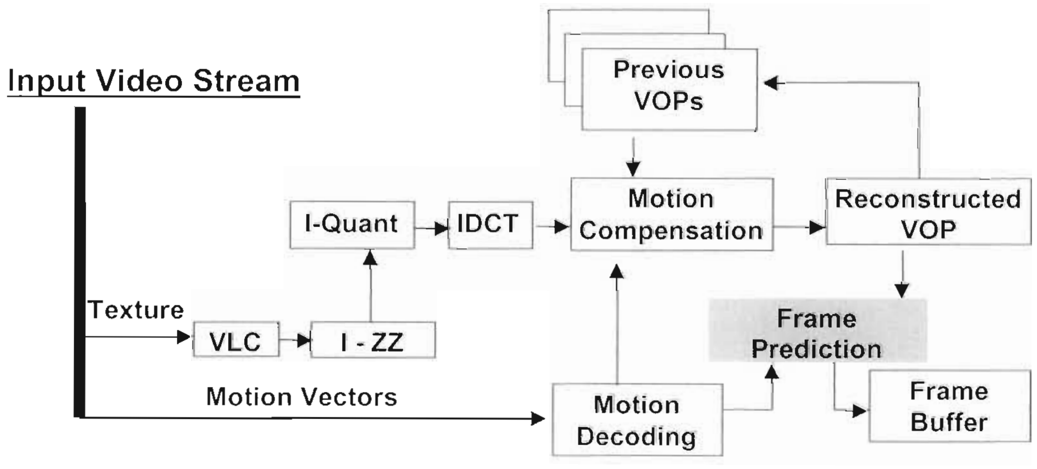


Figure 2.3: Prediction module in the standard MPEG-4 decoder.

The decoder feeds the last reconstructed frame and its corresponding motion vectors (MVs) to the prediction module, which maintains a buffer of several previous reconstructed frames and their MVs. These frames, along with their associated motion, are used to predict and synthesize future frames for display. For a video at 30 frames per second (fps), predicting one frame ahead corresponds to a delay reduction of 33.3 ms; predicting two frames ahead corresponds to a delay reduction of 66.7 ms, etc.

This architecture can be easily adapted to other video standards, like H.264 [3], as long as they use a similar block-based coding methodology. The following section describes the internal architecture of our prediction module.

2.2 Prediction Module Architecture

Fig. 2.4 shows the internal architecture of the prediction module with the following key blocks. Further operational details are provided in Chapter 3.

Unified Module Interface. This is the unified interface between the block-based video decoder and the adaptive frame prediction module. The interface includes all motion information from past video frames, the prediction error information, the delay signal for prediction depth control from the jitter buffer, and previous reference frames.

Motion Segmentation Unit. Combining the motion information and the prediction residual, this unit segments the motion objects from the background area on a block-by-block basis. This is a crucial step to predict the motion between the last decoded frame, and the future frame we would like to display.

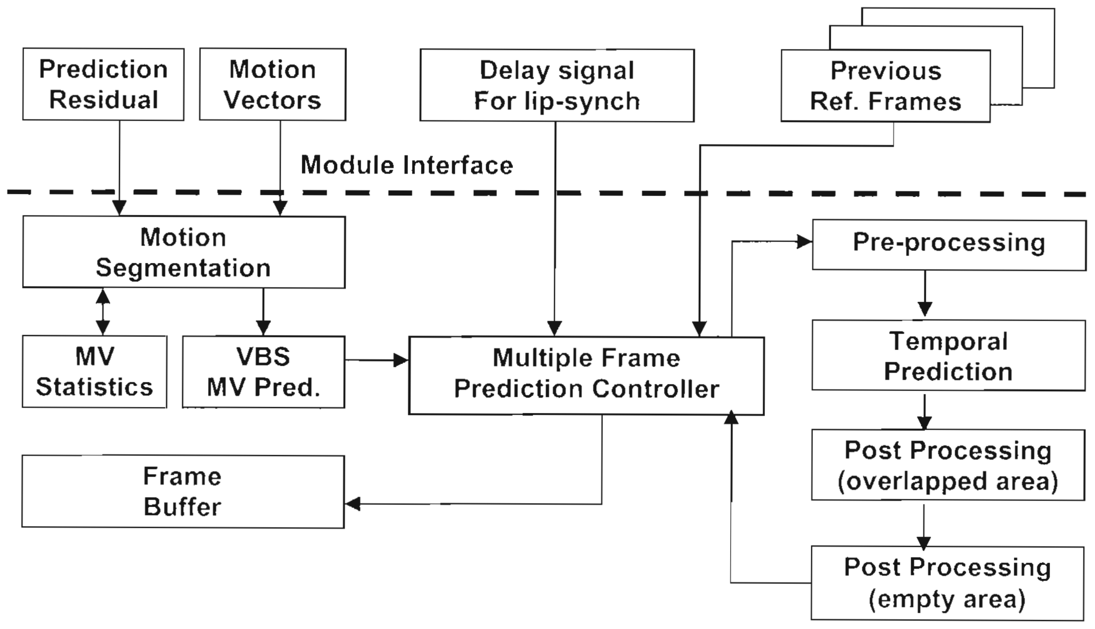


Figure 2.4: **Internal architecture of prediction module**

Variable-Block-Size Motion Prediction. Motion prediction is seen as the key step in frame prediction, and prediction is essentially done on the basis of each segmented object as well as the background area. A variable-block-size prediction algorithm is proposed to improve prediction near the edges of moving objects.

Temporal Prediction Unit. *Temporal Prediction* unit synthesizes the future frame using the blocks from previously decoded frames and predicted MVs.

Post Processing Units. As the motion objects which consist of blocks with consistent motion vectors get moved along the predicted MVs, they usually do not fill the entire future frame – some areas of the frame may remain empty (we call these “empty areas”), others may have multiple blocks landing on them (we call these “overlapped areas”). These issues are dealt with in the *post-processing* unit.

Multiple Frame Prediction Control unit. This unit uses the estimate of the current end-to-end delay to decide how many future frames are being synthesized, i.e., how many frames ahead are we predicting. The last predicted frame will be sent to the frame buffer for display.

2.3 Frame Prediction Procedure

Frame prediction module is inserted into the traditional video decoder, and utilized to reduce delay adaptively according to an estimate of the current end-to-end delay. The main issue in frame prediction is that of prediction error propagation as we synthesize more future frames. To clarify this difficulty, consider a simple prediction method where the motion of a block is predicted based on the motion vector of the last received block, and its magnitude is linearly increased depending on the number of frames that need to be predicted. A small prediction error is introduced when we predict the motion of the first future frame, and it is magnified as we predict more frames ahead.

This issue is depicted in Fig. 2.5 where, for the future frame (n), $MV_{real}(n)$ denotes the real MV, $MV_{pred}(n)$ denotes the predicted MV, and N is the number of future frames that need to be synthesized.

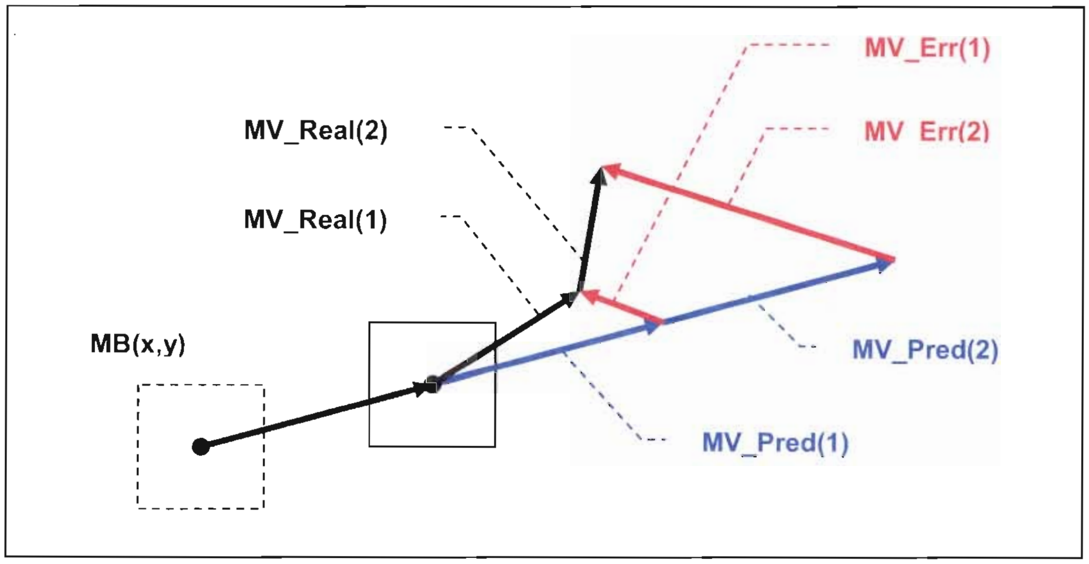


Figure 2.5: The error propagation of motion prediction.

The error of this simple motion predictor is:

$$MV_{err}(n) = \sum_{n=1}^{N} MV_{real}(n) - \sum_{n=1}^{N} MV_{pred}(n)$$

Eq. 2.1

Since the objects do not usually move along a straight-line trajectory, this simple motion prediction on the decoder side will wander off the real trajectory if not mitigated properly.

To mitigate the prediction error, we introduce sequential frame prediction, where we have moving-window to store motion information for several past frames, and we make use of motion segmentation in prediction so that we can refine MVs within a region with homogeneous motion. Furthermore, a sequential frame synthesis loop is used to predict motion vectors if more than one frame needs to be synthesized ahead of time. "Sequential frame prediction" means that if we need to predict several frames ahead, we first synthesize the first future frame, then the next future frame, and so on until the final frame. We do not attempt to synthesize the final future frame directly (unless we are predicting only one frame ahead of time). This sequential frame prediction mechanism is depicted in Fig. 2.6.

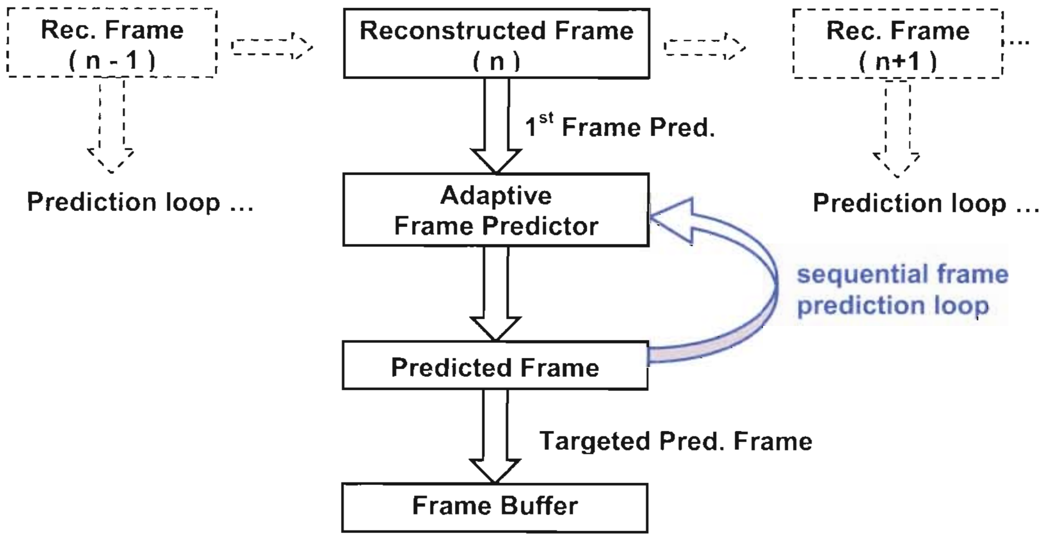


Figure 2.6: **Sequential frame prediction mechanism.**

We do not anticipate eliminating the error completely, but by adopting sequential frame prediction, the prediction error is greatly reduced as we synthesize all intermediate frames between the last reconstructed frame and the targeted future frame for display. The moving-window of motion information also helps with motion statistics and can smooth out the trajectory of some moving objects.

Chapter 3 elaborates on the key prediction unit, and emphasizes the way we minimize the prediction error.

CHAPTER 3
ADAPTIVE FRAME PREDICTION MODULE

In this Chapter, we are going to describe the key processing units of the frame prediction module, as well as its internal data flow. We start with the system and interface aspects of the adaptive frame prediction module, followed by the description of design motivations and algorithm details for each processing unit.

3.1 System and Interface Aspects

Fig. 3.1 shows the abstracted module interface and internal architecture of the proposed adaptive prediction module.

We have given a rough introduction about the system architecture of adaptive frame prediction in Section 2.2. A unified module interface is proposed for the prediction module so that all block-based video decoders can take advantage of it. Also, the core control loop is a sequential prediction loop which is responsible for synthesizing future frames. The frame prediction module essentially consists of motion segmentation, motion vector prediction, temporal prediction, and post-processing, and the details will be presented in the following sections.

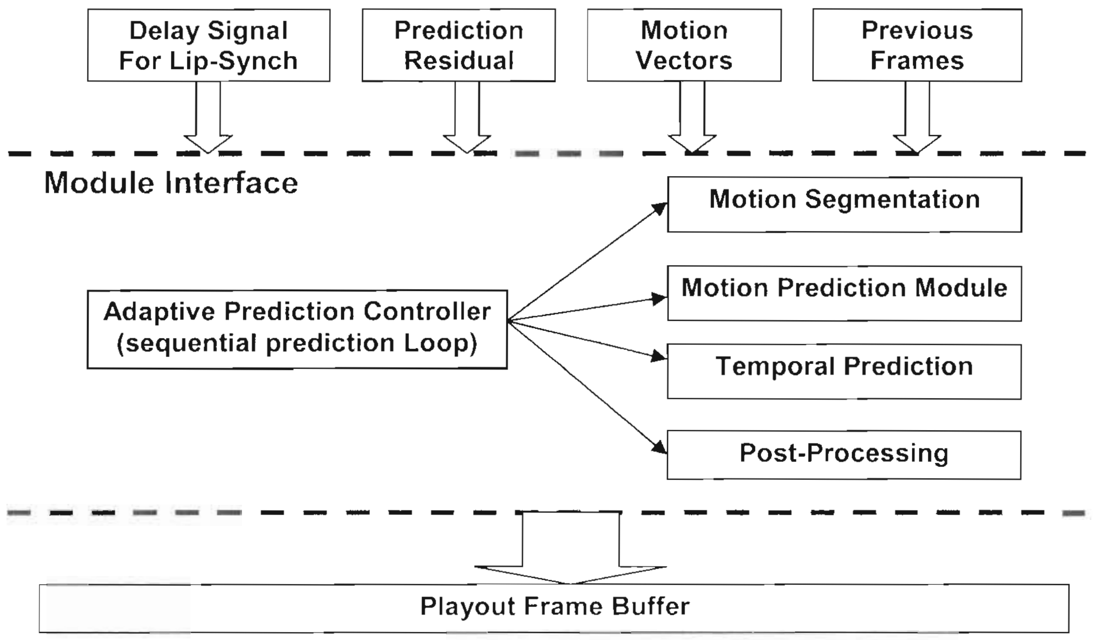


Figure 3.1: System and interface aspects

3.2 Motion Segmentation

3.2.1 Motivations

Predicting the moving direction precisely has proved to be crucial to synthesizing future frames in our experiments, and both motion vectors and prediction residuals from previous frames can contribute to MV prediction for future frames.

All recent block-based video encoders use variable-block-size motion estimation, and all these motion vectors represent moving directions of different parts of a particular MB. However, conventional motion estimator at the encoder side is built based on the criteria of minimum prediction error instead of motion homogeneity within certain region, and this might cause serious problems when we use all these motion vectors as the only information to predict future motion. The motion estimation results might not indicate real motion in the following cases [30]:

1) In flat areas, random vectors are likely generated due to the noise in the video sequence. The motion vectors associated with macro blocks in background region might end up differing from each other randomly.

2) In areas with repetitive patterns, false motion vectors may be generated as blocks get matched with other, further instances of the pattern.

3) When the motion is larger than the search area, no meaningful motion vectors can be obtained.

Motion estimation errors caused by these phenomena are sometimes referred to as "noise." Motion prediction that relies solely on these potentially false motion vectors becomes unreliable, especially for background areas.

Without eliminating the noise present in motion vectors, predicted frames may contain false and unnatural motion such as the "shaking background," etc. The problems gets worse as we predict further ahead, since prediction errors get amplified.

Another common issue concerns the relationship between blocks and object boundaries. It is commonly observed that the block-wise boundary doesn't match perfectly with the true object boundary [31], because the true object boundary may cut across the block. Therefore, frame prediction might mess up the object boundary since texture information might either disappear along the edge between different objects or appear along object boundaries in the future frames. To deal with all these issues, we develop a motion segmentation algorithm as the first step of variable-block-size motion vector prediction. Simulation results in Chapter 4 will illustrate the improvement of frame prediction based on motion segmentation, versus the prediction that does not use motion segmentation.

Motion segmentation has been a hot research topic in video processing. Reference [32] uses motion segmentation for very-low-data-rate object-based video coding. In [33], frames are adaptively partitioned into blocks of variable size with homogeneous motion, therefore a significant improvement of the compression ratio can be achieved. Reference [34] proposed a variational method for segmenting image sequences into spatio-temporal domains of homogeneous motion. Reference [39] addresses motion segmentation techniques in MPEG-4 video standardization, where hybrid block-based coding techniques are used. These works make use of motion segmentation in different scenarios.

Our motion segmentation aims to improve motion prediction at the decoder side. Here, the available motion information exists in the compressed video stream in the form of motion vectors, and it demands a different approach for the segmentation. References [35-36] have described methods for achieving multiple affine motion decomposition, with K-means clustering in the affine parameter space. Our proposal for motion segmentation algorithm is a combination of K-means clustering [29] and motion consistency model.

Fig. 3.2 shows the diagram of the main processing units for motion segmentation. The motion information and texture information associated with each macro block are extracted from the encoded video bit stream for low-precision motion segmentation, and the segmented motion objects are fed to the motion prediction module for object-based motion vector prediction. Detailed description of processing units is given in the following sections.

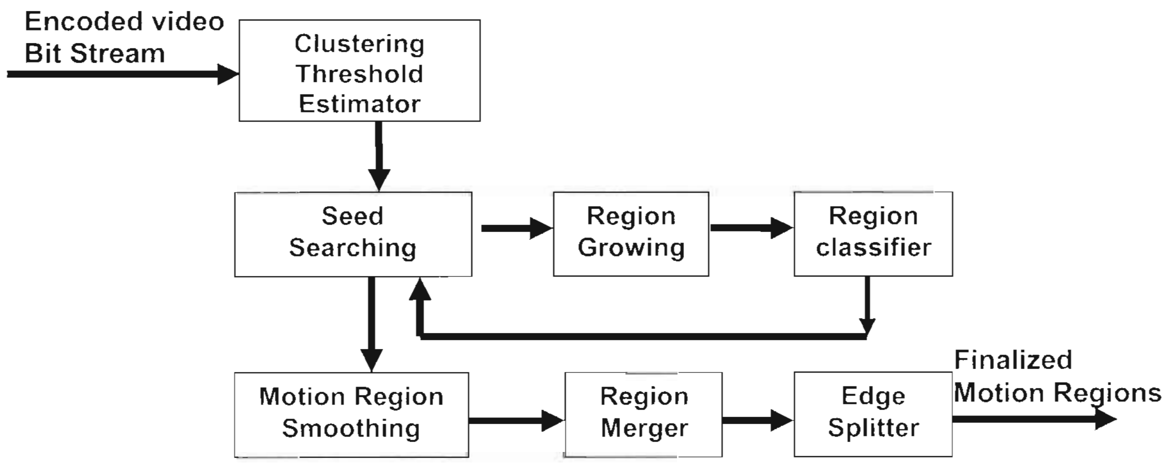


Figure 3.2: Diagram of motion segmentation

3.2.2 Clustering Threshold Estimator

Our experiments show that the motion level of each particular video frame greatly affects the motion segmentation performance. Without an appropriate knowledge of the motion level, after motion segmentation, one video frame might end up with a) too many moving objects, with some parts of an object segmented into different objects, or b) different moving objects clustered into the same object by mistake. Also, a good estimation of the motion level will speed up the segmentation. Therefore, we come up with a clustering threshold estimator to give the statistic of the motion level for motion segmentation.

Two important parameters are estimated in clustering threshold estimation: the region growing step size and the threshold of minimum moving object distance.

The region growing step size is used to control the speed of region growing. Large step size will make motion segmentation converge quickly, but it might increase the risk of grouping the blocks into a wrong motion region.

The threshold of minimum moving object distance is used in the region merge unit in Fig. 3.2. After motion objects are identified, we are going to calculate the distance between each pair of objects. If the distance between two particular objects is less than this threshold, these two objects are merged into one to get rid of falsely segmented moving objects.

To calculate these two parameters, a statistical model is used to estimate both the mean and variance of motion vectors in the frame that is to be segmented.

(1) The mean of the motion vectors

$$\mu_{mv} = E\left\{\overline{MV}_{i,j}\right\} = \frac{1}{MN}\left(\sum_{i=1}^{N}\sum_{j=1}^{M}\overline{MV}_{i,j}\right)$$

Eq. 3.1

(2) The variance of the motion vector magnitude

$$\sigma_{mv}^2 = \frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}\left\|\overrightarrow{MV}_{i,j} - \mu_{mv}\right\|^2$$

Eq. 3.2

Here, *N* and *M* denote the width and height of the frame in terms of 8x8 blocks, and $\overrightarrow{MV}_{i,j}$ denotes the motion vector of block $MB(i,j)$.

The region growing step size is set to be related to the variance, so that high variance leads to a large region growing step size. Currently in our software, we take the standard deviation as the region growing step size, and limit its maximum value to 8.

The threshold of minimum moving object distance is also decided by the motion vector variance. Currently in our software, we set the standard deviation as the threshold of minimum moving object distance, and limit the maximum value of the threshold to 4.

3.2.3 Seed Block Pattern

The seed-block pattern is defined as a group of blocks with minimum mean MV distortion, which works as a starting point for subsequent clustering. If we denote $\overrightarrow{MV}_i, i = 1, \cdots, N$ as the motion vectors of each block, where N represents the number of blocks inside a seed, and D_{seed} as the overall MV distortion, equation (3.3) is used to find a seed through the entire video frame.

$$D_{seed} = \arg\min_{i}\left(\sum_{j\neq k}\left\|\overrightarrow{MV}_{i,k} - \overrightarrow{MV}_{i,j}\right\|\right).$$

Eq. 3.3

It is also crucial to define a good pattern for seed-blocks. The main criterion in choosing a good pattern is to avoid false moving object seed, which can be caused, for example, by four identical motion vectors (8x8 blocks) split from one 16x16 MB. Fig. 3.3 shows two pattern candidates. Pattern on the left was found to give better performance in our motion segmentation experiments.

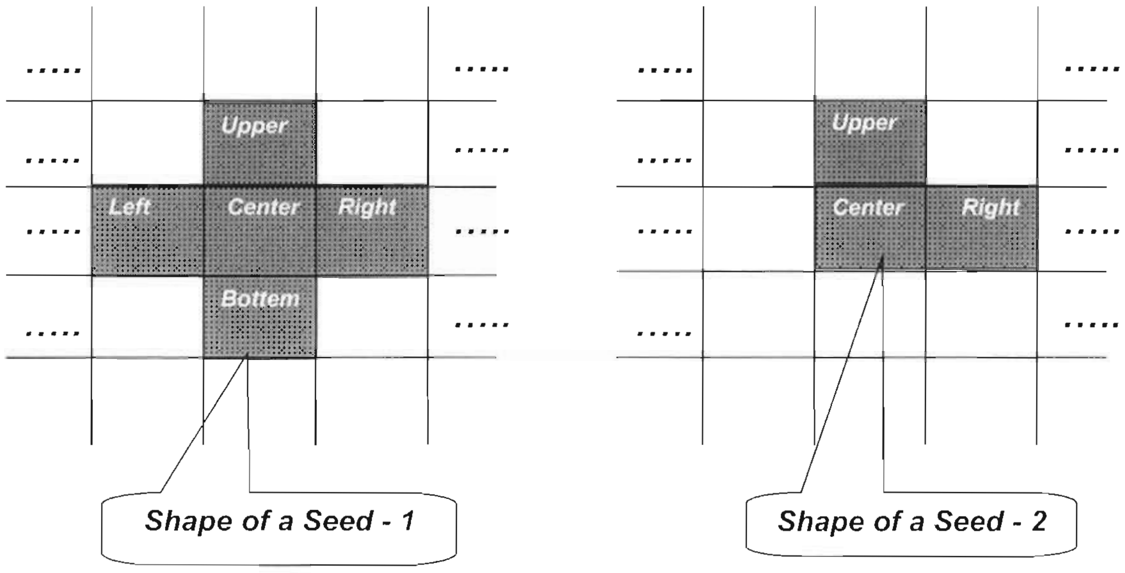


Figure 3.3: The seed pattern

After we identify a seed block, the neighbouring blocks in horizontal and vertical directions, which are shown in Fig. 3.4, are first clustered together to form a new motion region [37]. K-Means clustering is used to group surrounding blocks based on motion vector consistency, as explained below.

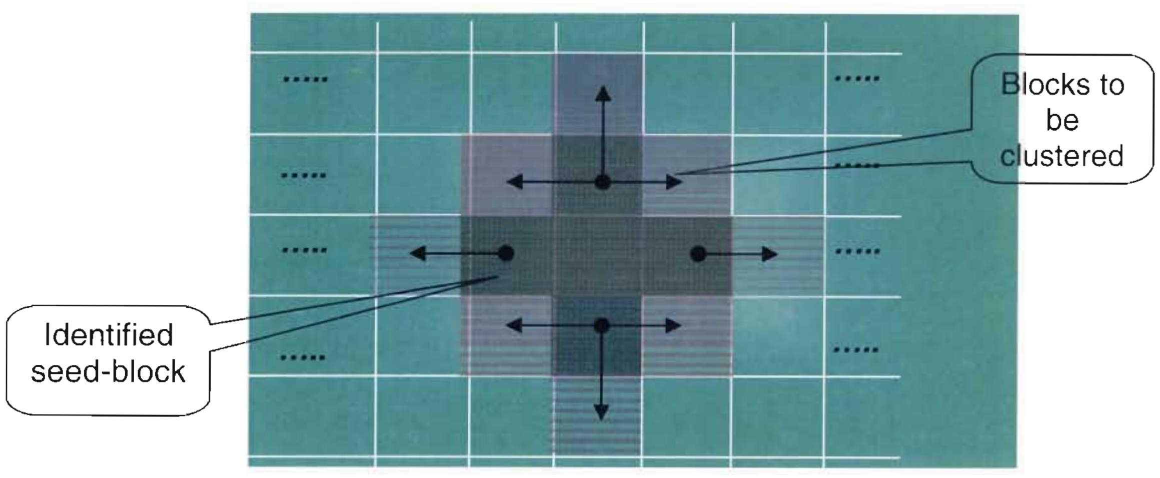


Figure 3.4: The region growing directions

3.2.4 Region Growing Criterion

Based on a Motion Consistency Model (MCM), the moving object can be identified by clustering ungrouped surrounding blocks that border the existing motion region into that region, if the motion vectors of these blocks are sufficiently similar to the motion inside the region, i.e., if they satisfy the consistency criterion in MCM. This procedure is called "region growing."

The MCM is established based on the minimum distortion criterion, and updated adaptively as the region is growing. Let's denote $\overrightarrow{MV}_{ext}$ as the MV for the considered block, $\overrightarrow{MV}_{centroid}$ as the centroid MV of the region, $\overrightarrow{MV}_{\text{int}}$ as MVs of all internal blocks within the region. Once a new seed-block is found, the centroid MV, $\overrightarrow{MV}_{centroid}$, is initialized, and the region growing threshold D_{TH} is computed as shown in equation (3.4). Essentially, D_{TH} is the average distance between the MVs in the region and the centroid MV of the region, increased by the offset D_{off} , where D_{off} indicates the region growing step size which is estimated in the Clustering Threshold Estimator, the first processing unit in Fig. 3.2.

$$D_{TH} = E\{\left\|\overrightarrow{MV}_{\text{int}} - \overrightarrow{MV}_{centroid}\right\|\} + D_{off}$$

Eq. 3.4

The region growing process checks whether an ungrouped block which borders the existing region has sufficiently similar motion to the prevalent motion inside the region, by checking if equation 3.5 is satisfied. Here, $\overrightarrow{MV}_{centroid,n}$ denotes the motion vector centroid of region n, $\overrightarrow{MV}_{ext}$ denotes the motion vector of the block that is being tested, $D_{TH,n}$ denotes the growing threshold of region n, and $D_{i,n}$ is the distance between motion vector centroid of the region and the MV of the block.

For a particular ungrouped block, if there is more than one bordering region for which equation 3.5 is satisfied, then we pick the minimum $D_{i,n}$ and group the block into that region accordingly.

$$D_{i,n} = \left\|\overrightarrow{MV}_{centroid,n} - \overrightarrow{MV}_{ext}\right\| \leq D_{TH,n}$$

Eq. 3.5

After the block is assigned to one of its neighboring regions, the motion vector centroid of that region gets updated as shown in equation 3.6. This process is repeated for other neighboring blocks until the region cannot grow anymore. Once the region stops growing, our estimate of the location of a moving object is formed.

$$\overrightarrow{MV}_{centroid} = E\left\{\overrightarrow{MV}_{\text{int},i}\right\} = \frac{1}{M}\sum_{i=1}^{M}\overrightarrow{MV}_{\text{int},i}$$

Eq. 3.6

3.2.5 Motion Region Smoothing

Fig. 3.5 illustrates some typical manifestations of motion vector "noise" addressed in section 3.1, such as false MV, background MV noise, etc. These problems are basically caused by MV estimator at the encoder side which utilizes the standard Mean Absolute Difference (MAD) error criterion [30]. To mitigate these problems, we add a special processing step, Motion Region Smoothing. In this step, MV filtering is applied to reduce the amount of MV noise.

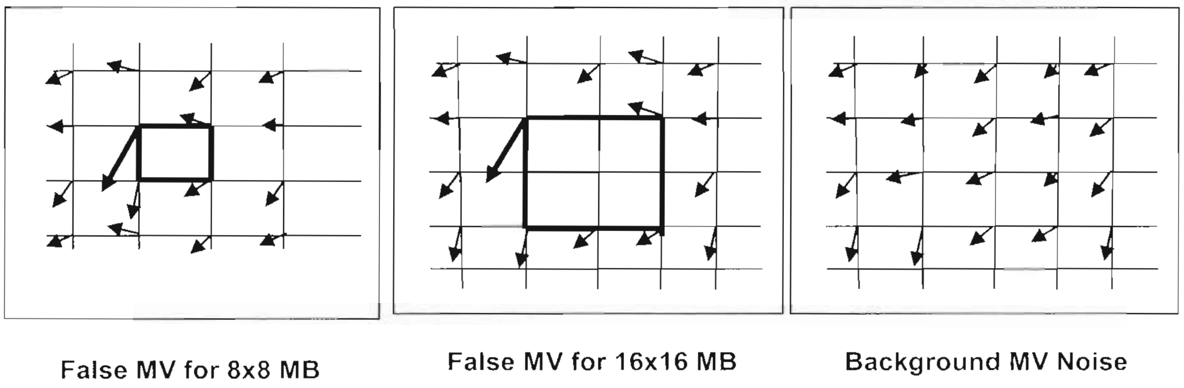


Figure 3.5: Typical motion vector noise

With segmented moving objects and background area in the previous section, it is possible to do separate motion vector filtering in each region. Two filtering methods are proposed here.

a) Replace MV of all internal blocks with the centroid MV of the entire region.

b) Use Vector Median Filtering (VMF) for motion vector smoothing [38].

For a set S of N motion vectors, $S=\{\overrightarrow{MV}_1,\overrightarrow{MV}_2,...,\overrightarrow{MV}_N\}$, vector median $\overrightarrow{MV}_{vm}$ is the MV whose sum of distances to all other MVs is the smallest:

$$\overrightarrow{MV}_{vm} = \arg\min_{k} \sum_{j \neq k} \left\| \overrightarrow{MV}_k - \overrightarrow{MV}_j \right\|.$$

Eq. 3.7

We use VMF in the following way. All original MVs of blocks in a certain region will be replaced by the vector median of all MVs in its 3×3 neighborhood that belong to the same region, as illustrated in Fig. 3.6. Near the edges of the frame or the boundaries of the region, we simply collect as many MVs from the 3×3 neighborhood which belong to same region to perform vector median filtering.

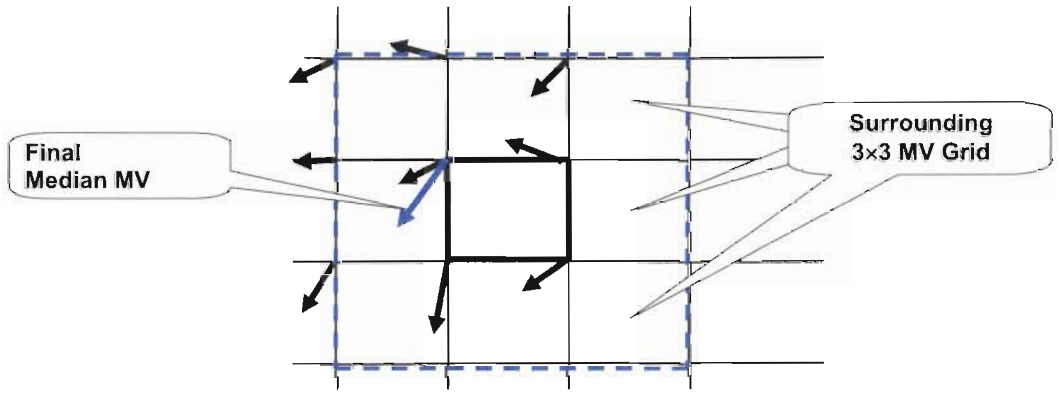


Figure 3.6: **VMF-based MV filtering**

3.2.6 The Processing Steps

In summary, the motion segmentation algorithm consists of the following steps:

step 1) The algorithm starts with extracting all block-based motion information from the received video stream, and a statistical approach is used to estimate the overall motion level in the clustering threshold estimator. Based on the variance of motion vectors of the frame to be segmented, the corresponding region growing step size is determined as well as the threshold of minimum moving object distance. These two parameters are used in the region growing unit and the object merge unit, respectively.

step 2) With the pre-defined seed pattern, a group of blocks with minimum MV distortion is found as the starting point for region growing, to identify moving objects.

step 3) With the threshold of MV distortion range from step 1) and seed-blocks from step 2), a region will be grown gradually by clustering bordering blocks into the region, if their motion is sufficiently similar to the prevalent motion inside the region. This step will be executed repetitively until no blocks are left which satisfy the minimum distortion criterion in the motion consistency model.

step 4) Classify all existing motion regions, and update all parameters for motion consistency model, such as the motion vector centroid, the region growing threshold, and so on.

step 5) Repeat Step 2) to Step 4) until no further seed can be found.

step 6) Motion region smoothing consists of two main parts: clustering all remaining ungrouped blocks to weed out potential false motion vectors, and using median filtering to reduce motion vector noise inside each region.

step 7) Calculate the distance between centroid MVs of adjacent regions, and merge two adjacent regions if the distance between their centroid MVs is less than the threshold of minimum moving object distance, which is determined in the clustering threshold estimator.

3.3 Variable-block-size Motion Prediction

In each coded frame, the MV associated with a block points to the most similar block in the reference frame, and can be interpreted as a motion path for that block. We assume the object will keep moving in a similar direction, thus a crucial step towards synthesizing a future frame is to predict the motion for each block between the last decoded frame and the future frame. With segmented motion objects as well as the background area, a variable-size block motion prediction algorithm is proposed in this thesis, and its system diagram is illustrated in Fig. 3.7.

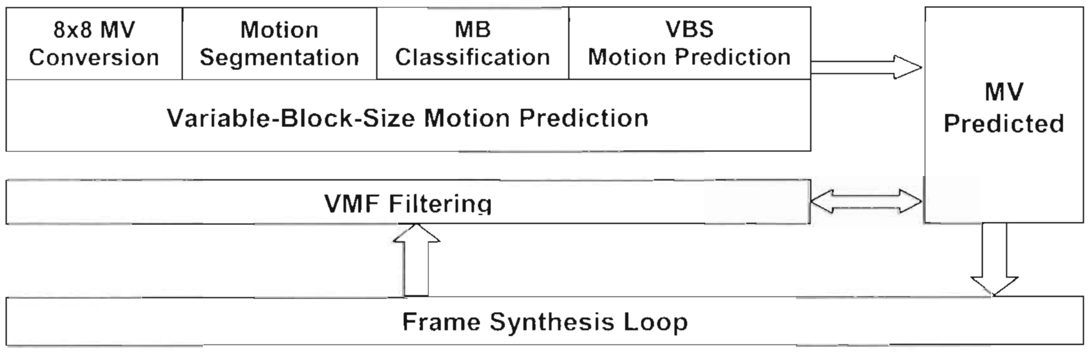


Figure 3.7: The system diagram of variable-block-size motion prediction

The whole MV prediction algorithm can be further divided into two processing phases in terms of frame prediction depth.

Phase a) When we synthesize the first future frame (i.e., when we predict one frame ahead of time), both motion information and prediction residual are extracted from the compressed video stream. Motion segmentation is used to isolate all motion objects from the background, and vector median filtering is applied to smooth each motion field out. To further improve the quality of motion estimates near object boundaries, all blocks are classified based on their position and the energy of the prediction residual. Finally, a Variable-block-size Motion Prediction (VBS-MP) is employed to predict motion path for each individual block [31], [33].

Phase b) To predict the motion path for other future frames (i.e., when predicting more than one frame ahead of time), the processing is slightly different due to the lack of prediction residual data. Therefore, instead of the VBS-MP method, vector median filtering is adopted to predict motion vectors near object boundaries.

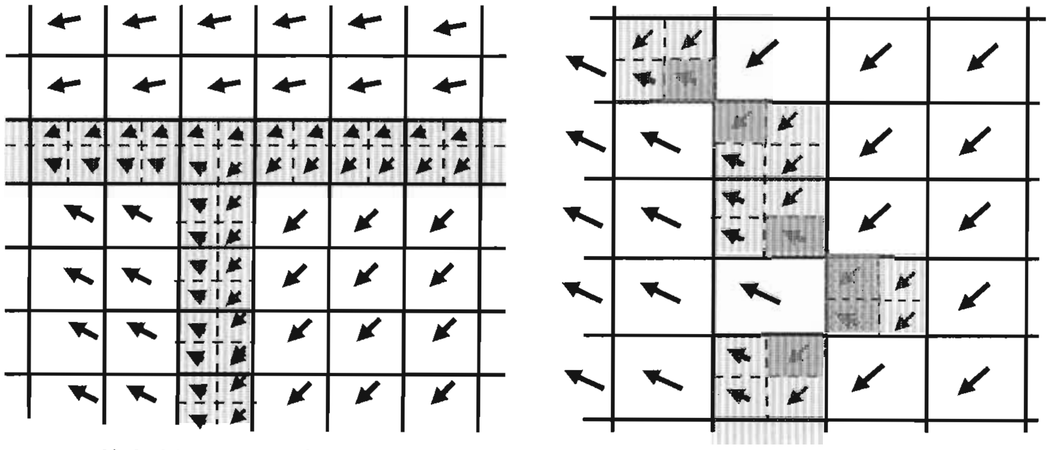
The predicted motion path is fed to the temporal prediction module to aid the synthesis of the future frame. The elaboration of all major processing units in Fig. 3.6 follows.

Conversion to 8x8-based MVs. Recall that in MPEG-4, MVs can refer to 8×8 blocks or 16×16 blocks. To simplify further processing, all MVs are converted to 8×8-based MVs by assigning the same MV to four 8×8 sub-blocks of a 16×16 block where necessary. For intra-coded blocks (I-blocks) which do not have a MV associated with, a Zero MV will be assigned.

Motion Segmentation was described in Section 3.2.

Block Classification. 8x8 blocks are classified prior to variable-block-size motion prediction. Within each region, each block is classified as either an internal block or a boundary block in terms of its position and the energy of the prediction residual. a) A block is classified as internal if it is surrounded by blocks which belong to the same motion region, or if its residual energy is lower than a pre-defined threshold whose value is set to 256 based on our experiments (which indicates an average prediction error of 4 per pixel). b) A block is classified as a boundary block if it is near the boundary between different regions and its residual energy is higher than the threshold.

Variable-Block-Size Motion Prediction. For the block classified as a boundary block, assigning a single MV may be inappropriate since a more complex motion structure is likely involved. Splitting the 8x8 block into smaller blocks, like 4x4 sub-blocks, and predicting MV for each of them can mitigate the risk of predicting a wrong single MV [17][50]. Fig. 3.8 shows two scenarios where blocks near the boundaries of moving regions are split into 4×4 sub-blocks, and an MV is assigned to each sub-block based on which region it is closest to.


A) 4x4 blocks, each of which is at the edge of one region

B) 4x4 blocks surrounded by multiple motion regions

Figure 3.8: The motion vector prediction for 4x4 sub-blocks

The way to assign MVs for sub-blocks is as follows:

a) If a 4x4 sub-block is surrounded by blocks from the same region, then the MV of that region is assigned to the 4x4 sub-block.

b) If a 4x4 sub-block is surrounded by multiple regions, then the MV distance is first calculated between these regions and the parent 8x8 block (before splitting). The MV of the region with minimum distance to the parent MV is assigned to the 4x4 sub-block.

Let $\overrightarrow{MV}_{orig}$ be the MV of the 8x8 parent block near the boundary, and let S be a set of *N* motion vectors, $S = \{\overrightarrow{MV}_{cen,1}, \overrightarrow{MV}_{cen,2}, \ldots, \overrightarrow{MV}_{cen,N}\}$, whose elements are the centroid MVs of the surrounding regions. Equation (3.8) is used to pick the most likely MV, $\overrightarrow{MV}_{4x4,k}$, for the *k*-th 4x4 sub-block.

$$\overrightarrow{MV}_{4x4,k} = \arg\min_i \left\| \overrightarrow{MV}_{cen,i} - \overrightarrow{MV}_{ori} \right\|$$

Eq. 3.8

VMF Filtering. As an alternative to variable-block-size motion prediction (VBS-MP), vector median filtering can be used to smooth the motion field in the boundary areas between moving objects if no residual information is available. This happens when predicting MVs to synthesize future frames that are more than one frame ahead of the last received frame. In these cases, we often find that multiple blocks land on the same area (overlapped area), as illustrated in Fig. 3.9. The question is how to predict the motion of that area into future frames, given that it may have come from different blocks in the previous frame, and so may take different motion trajectories into the future. Again, we found it useful to use the vector median of all candidate MVs as a predictor of motion for the overlapped area.

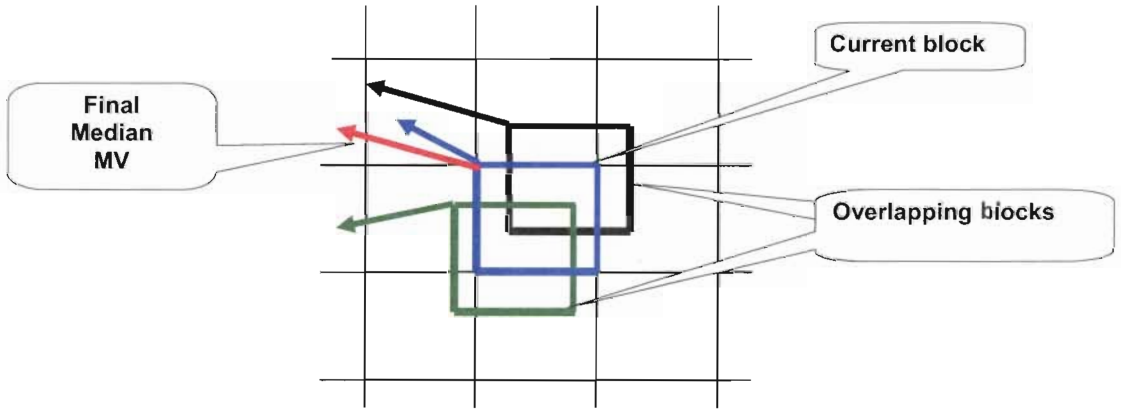


Figure 3.9: MV prediction for overlapped areas

3.4 Temporal Prediction

The term “temporal prediction” refers to the process of synthesizing the future frame after the motion between this future frame and the previous frame has been predicted as described in the previous few sections. Once MV estimates have been obtained, we move all the blocks including 4x4 sub-blocks of the previous frame along the predicted MVs. In this way, we synthesize a preliminary version of the future frame. Fig. 3.10 illustrates what a preliminary version of the future frame might look like.

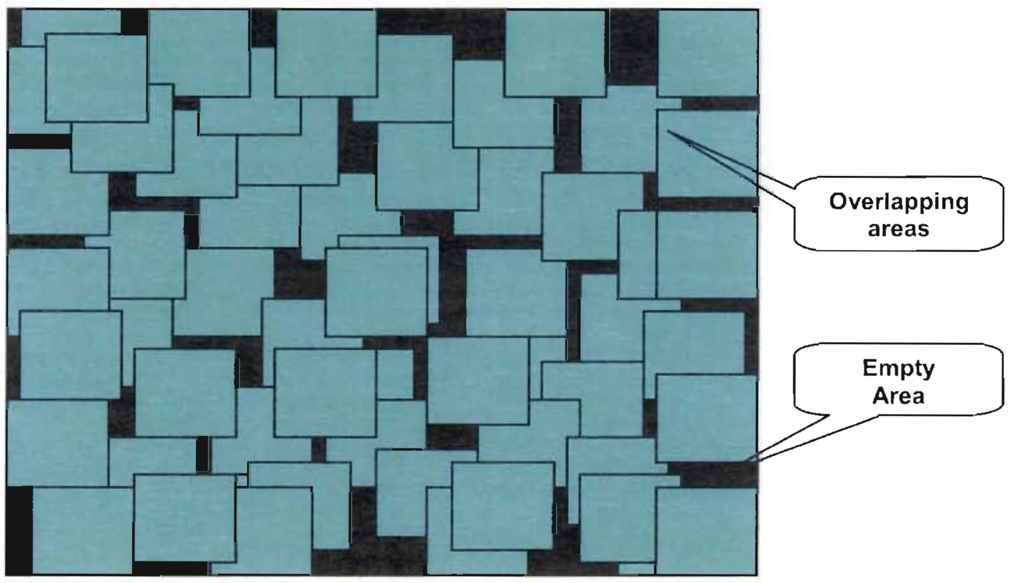


Figure 3.10: A preliminary version of the future frame

At this point, some areas of the synthesized frame may have multiple blocks landing on them – we call these areas “overlapped areas.” Other areas may remain empty, if no block lands on them. We need to decide which pixel values will be written into the overlapped and empty areas. These decisions are made in the two post-processing blocks whose operation is described below

3.5 Post-processing for Overlapped Areas

We distinguish two types of overlapped areas: “thin” areas are those whose width or height does not exceed 3 pixels, while “thick” areas are those whose both with and height exceed 3 pixels. Different post-processing is applied to each type of the overlapped area.

For thin areas, we apply a simple averaging of all candidate blocks. Let there be *N* blocks overlapping a certain area and let OV_k denote the *k*-th block. The pixel value at location (*x*, *y*) in the overlapped area is set to be the average of corresponding pixel values in each of the overlapping blocks:

$$P(x, y) = \frac{1}{N} \sum_{k=1}^{N} OV_k(x, y)$$

Eq. 3.9

Once all thin overlapped areas are processed, we are left with thick overlapped whose height and width exceed 3 pixels. An illustration is shown in Fig. 3.11.

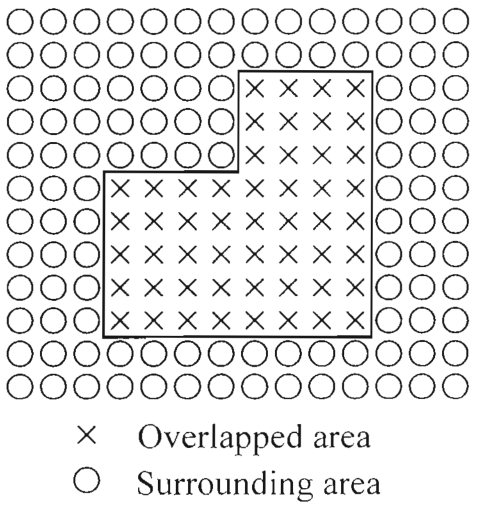


Figure 3.11: Boundary matching for overlapped areas.

These areas will be filled by pixel values from the block that fits the best into the surrounding area. To decide which block fits the best, we employ boundary matching by computing the mean square difference between the boundary pixels of candidate blocks, and the boundary pixels of the surrounding area.

Let $OV_k(x, y)$ be the pixel at location (x, y) in the k-th overlapping block. Let B be the set of boundary pixels of the overlapped area, and for each $(x, y) \in B$, let $n(x, y)$ be the value of the neighboring pixel across the boundary, in the surrounding area. The best matching block OV_{best} is the one whose square difference from the surrounding area along the boundary is the smallest, as in equation (3.10). Pixels from this block are used to fill the thick overlapped area.

$$OV_{best} = \arg\min_{k} \sum_{(x,y)\in B} \left\| OV_k(x,y) - n(x,y) \right\|^2$$

Eq. 3.10

3.6 Post-processing for Empty Areas

In addition to overlapped areas, we also find empty areas in the synthesized frame. These are the areas where no block has landed. A similar situation arises in error concealment, where a block of size 8×8 or 16×16 may be missing due to packet loss. However, in our case, empty areas may have different shapes and sizes. Again, we distinguish "thin" empty areas (those whose width or height does not exceed 3 pixels), from "thick" empty areas (those whose both height and width exceed 3 pixels). Different post-processing is applied to each type of the empty area.

Thin empty areas are filled using linear spatial interpolation [27]. Each pixel is estimated as a weighted sum of the boundary pixels in the surrounding areas. The weight of each boundary pixel is inversely proportional to the distance from the empty pixel whose value is being computed. An illustration of a thin empty area whose height is 3 pixels is shown in Fig. 3.12. Let $P(x, y)$ be the pixel value we wish to determine in an empty area, and let $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ be two of its nearest neighbors in the surrounding area.

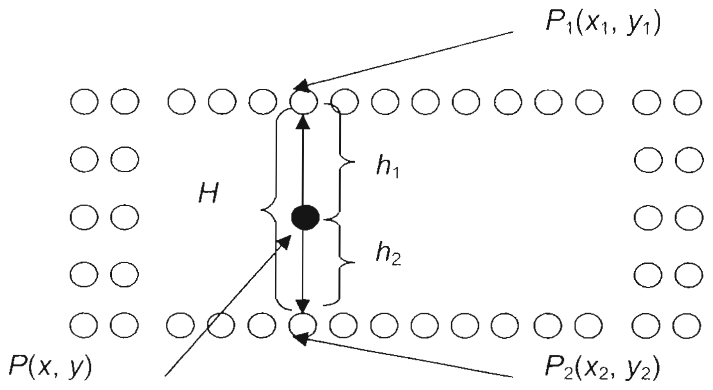


Figure 3.12: **Filling thin empty areas.**

In the situation depicted in Fig. 3.12, P_1 and P_2 are above and below P, so in this case $x_1 = x_2 = x$. The pixel in the empty area is linearly interpolated as

$$P(x,y) = \left(1 - \frac{h_1}{H}\right) P_1(x_1, y_1) + \left(1 - \frac{h_2}{H}\right) P_2(x_2, y_2),$$

Eq. 3.11

where h_1 and h_2 are the distances from P to P_1 and P_2, respectively, and $h_1 + h_2 = H$.

Simple linear interpolation works reasonably well for thin empty areas, but tends to produce excessive blurring when applied to thick empty areas. Therefore, we adopt a more sophisticated method for filling thick empty areas based on boundary matching [6-7].

An example of a thick empty area is shown in Fig. 3.13. First, we divide each thick empty area into rectangular regions, which we call "empty rectangles" (ERs), and label them ER_1, ER_2, ..., ER_N. We fill ERs in sequence, starting with

ER_1 and ending with ER_N. For each ER we extract the boundary pixels from the surrounding area and use them for boundary matching in previous frames.

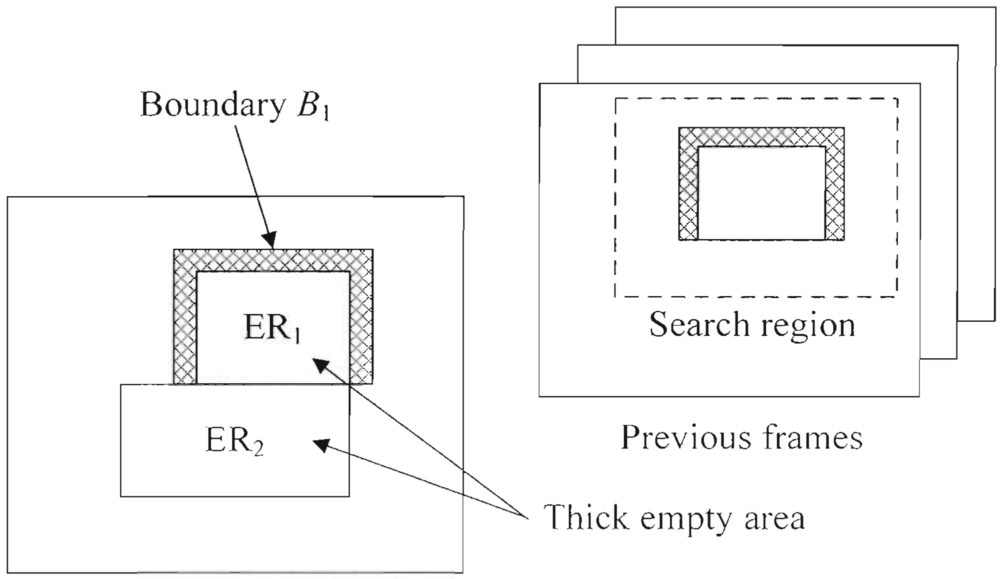


Figure 3.13: Filling thick empty areas.

Let B_n be the set of boundary pixel coordinates for ER_n. Denote the current frame as P, and previous K frames as P_1, P_2, ..., P_K. We will search in each of the previous K frames over an area of size $X \times Y$ pixels for the best matching boundary.

This boundary is found in frame P_k, offset by (dx, dy) from its position in the current frame, where

$$(k, dx, dy) = \arg \min_{k=1,2,\ldots,K} \min_{|dx| \le X/2, |dy| \le Y/2} \sum_{(x,y) \in B_n} \left\| P(x,y) - P_k(x+dx, y+dy) \right\|^2 .$$

Eq. 3.12

Once (3.12) is solved and the best matching boundary is found, we copy the corresponding rectangle from P_k to fill ER_n. At this point, ER_n is removed from the list of empty rectangles, and we continue with ER_{n+1}. The pixels of ER_n may now become boundary pixels for the remaining empty rectangles. The procedure is illustrated in Fig. 3.13.

An example of how empty areas are filled is shown in Fig. 3.14.

The figure shows a frame as it passes through the empty area post-processing block. The top left image shows the frame produced by temporal prediction and overlapped area processing. Thin vertical empty areas are filled first (top right), followed by thin horizontal empty areas (bottom left). The final predicted frame, obtained after filling thick empty areas, is shown at the bottom right of Figure 3.14.

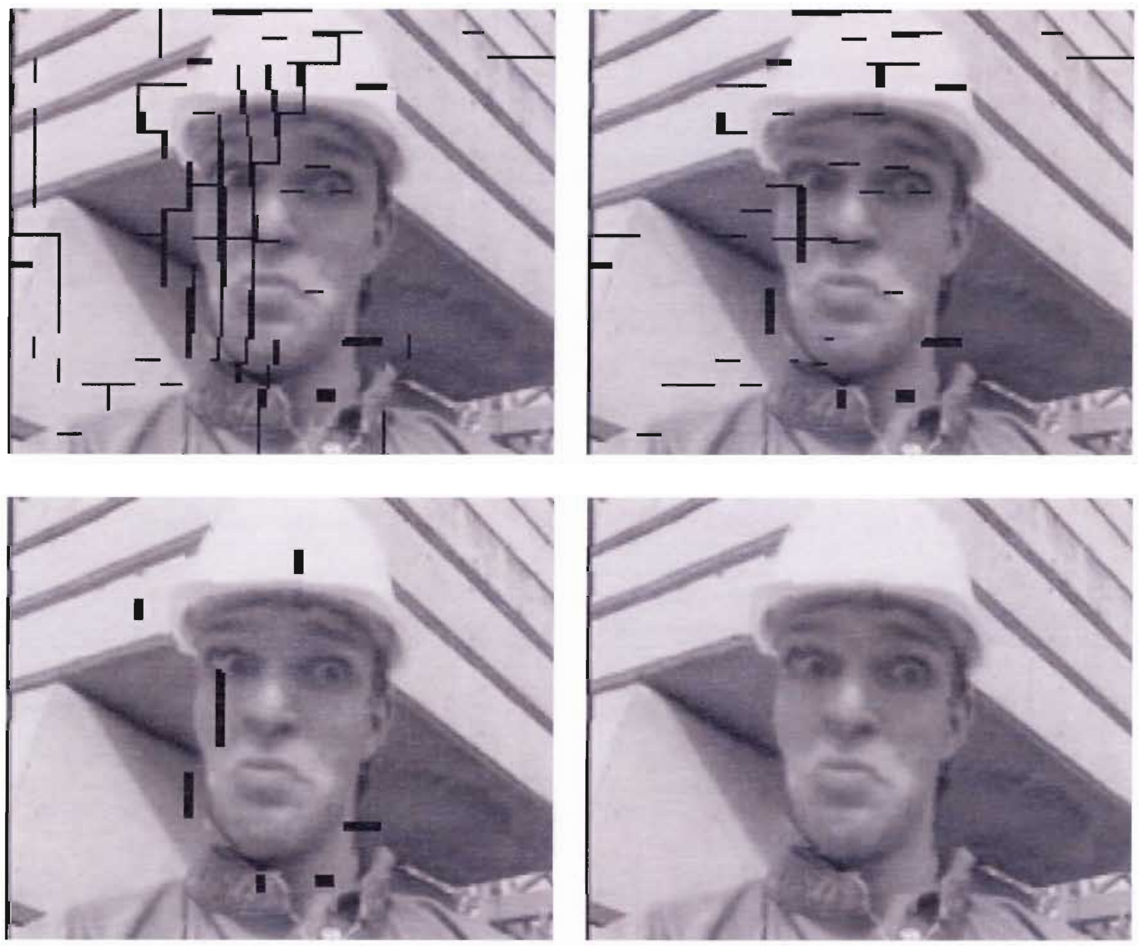


Figure 3.14: **Illustration of empty area post-processing where [top left] is the intermediate frame after post-processing of overlapped areas; [top right] is the frame after filling thin vertical empty areas; [bottom left] is the frame after filling thin horizontal empty areas; [bottom right] is the final frame after filling thick empty areas.**

CHAPTER 4
IMPLEMENTATION AND SMULATION RSULTS

In this Chapter, we describe the implementation of frame prediction in the XviD MPEG-4 decoder [26]. We also develop the methodology for assessing the motion segmentation as well as the overall predictive decoding process, and present the simulation results.

4.1 Test Sequences

We test the performance of the proposed frame prediction on several sequences with varying motion content. We used six sequences in our experiments, each at three different frame rates: 30, 15, and 7.5 frames per second (fps). These sequences are listed in Table 4.1. Frame prediction module was incorporated into the XviD implementation of the MPEG-4 video codec [26]. Up to 400 frames of each sequence were encoded using the IPPP... GOP structure. QCIF sequences were encoded at 128 kbps, and CIF/SIF sequences at 512 kbps.

On the decoder side, we tested prediction of up to 3 frames ahead. Depending on the frame rate (30 fps, 15 fps, or 7.5 fps) of the sequence, the perceived delay reduction is up to 100 ms, 200 ms and 400 ms when predicting three frames ahead of time. Using different combinations of frame prediction building blocks from the previous section, we constructed four prediction methods with different complexities, and compared their performance.

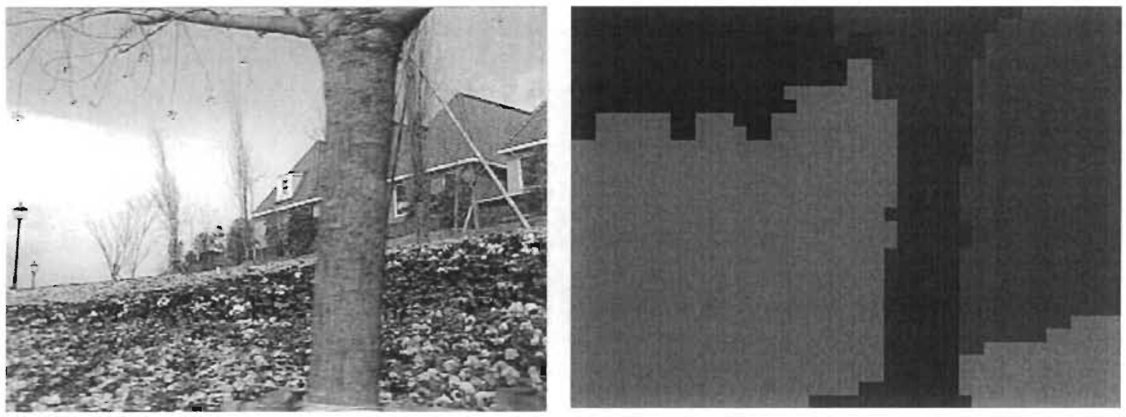
Table 4.1: Test sequences.

Sequence	Resolution	Motion
Carphone	QCIF	High
Flower Garden	CIF	High
Foreman	QCIF	Medium
Singer	SIF	Medium
Mother & Daughter	QCIF	Low
Miss America	QCIF	Low

4.2 The Assessment of Motion Segmentation

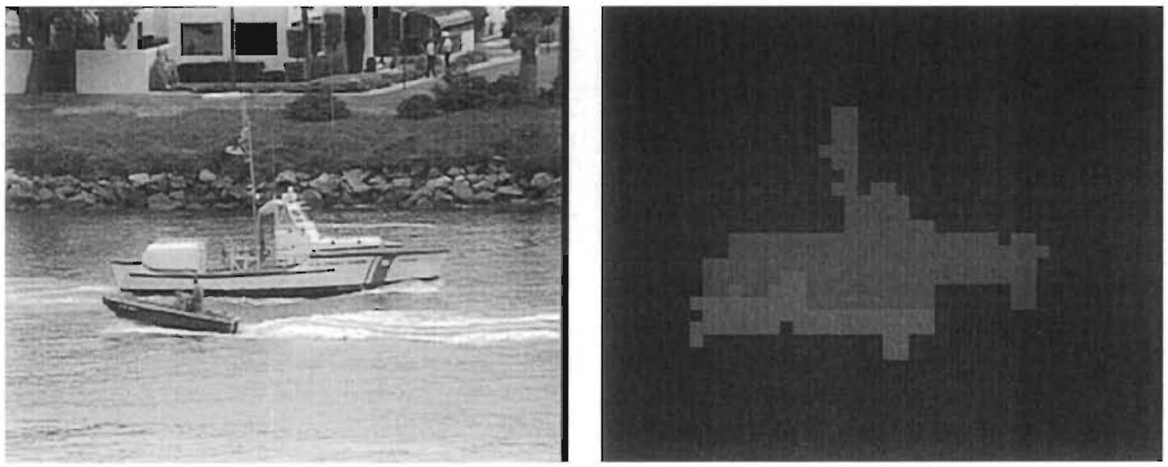
In Section 3.1, we proposed a motion segmentation algorithm whose aim is to improve the performance of motion prediction. Before we assess the overall performance of predictive decoding, we conduct several experiments to test the performance of motion segmentation itself.

Four video sequences, *Garden*, *Coastguard*, *Tennis* and *Football,* are used in this assessment. The input to the motion segmentation unit are the motion vectors extracted from the encoded video stream. Segmentation procedure is elaborated in section 3.1. Fig. 4.1 – Fig. 4.4 show the segmentation results where the original video frame is also displayed to give a reference on what the exact moving objects look like. To display the result of motion segmentation, moving regions are filled with different luminance values to distinguish them from each other.



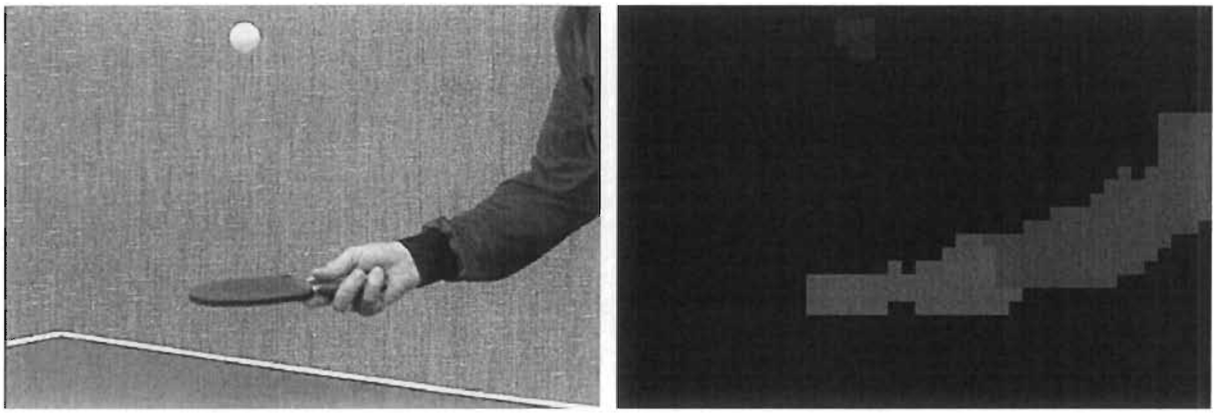
Original - *Garden* Motion Segmentation result

Figure 4.1: The motion segmentation result: *Garden*



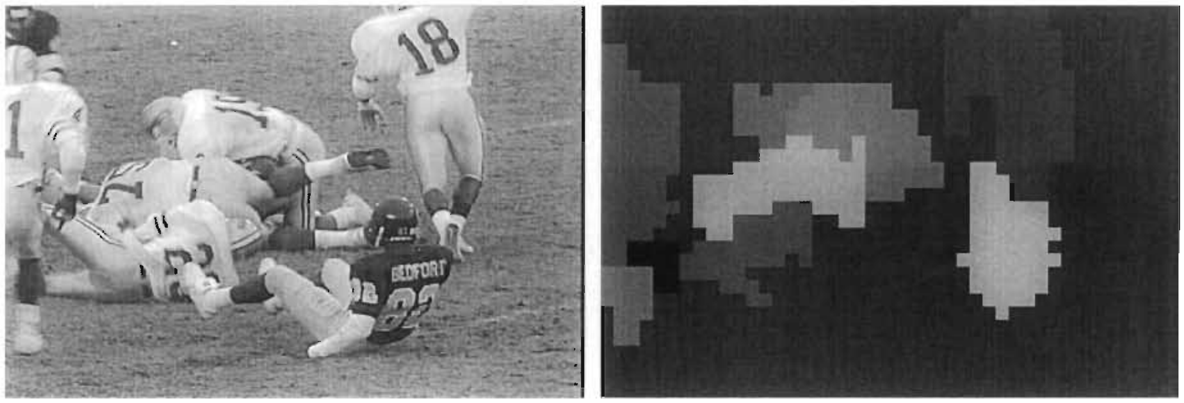
Original - *Coastguard* Motion Segmentation result

Figure 4.2: The motion segmentation result: *Coastguard*



Original - *Tennis* Motion Segmentation result

Figure 4.3: **The motion segmentation result:**
Tennis



Original- *Football* Motion Segmentation result

Figure 4.4: **The motion segmentation result:**
Football

From these results, one can see that the proposed motion segmentation algorithm nicely segments the moving objects at a block precision level. However, it is not hard to observe that the edges of each moving object cannot exactly match its original boundary. The jagged object boundaries are caused by the fact that motion estimation at the encoder is block-based, and uses the criterion of minimizing prediction error, which may lead to inaccurate estimates. Because of this, it is necessary to come up with a method to enhance segmentation performance and subsequent MV prediction near the edges. Our solution is the variable-block-size motion prediction, which was described in section 3.2.

4.3 Assessment Methodology

To assess the techniques described in Chapters 2 and 3, we construct four different prediction methods by combining different techniques. We then compare these methods to asses what benefit do the different techniques bring to frame prediction. We tested the four methods using both objective and subjective criteria.

4.3.1 Method 0: Zero Motion Vector

This method applies the simplest motion prediction model for frame prediction. The ZERO MV is assigned to all blocks so that there is no temporal movement at all regardless of the number of frames that need to be synthesized. In other words, the latest reconstructed frame is taken directly as the predicted frame for playout. This method is the basis against which we measure the performance of the proposed prediction techniques.

4.3.2 Method 1: Motion Extension

In this method, MV of each block is inverted and extended up to the frame we wish to predict. For intra-coded block, where there is no motion vector provided in compressed bit stream, the median neighbouring MV is chosen using the Vector Median Filter. Fig. 4.5 illustrates this simple motion prediction procedure.

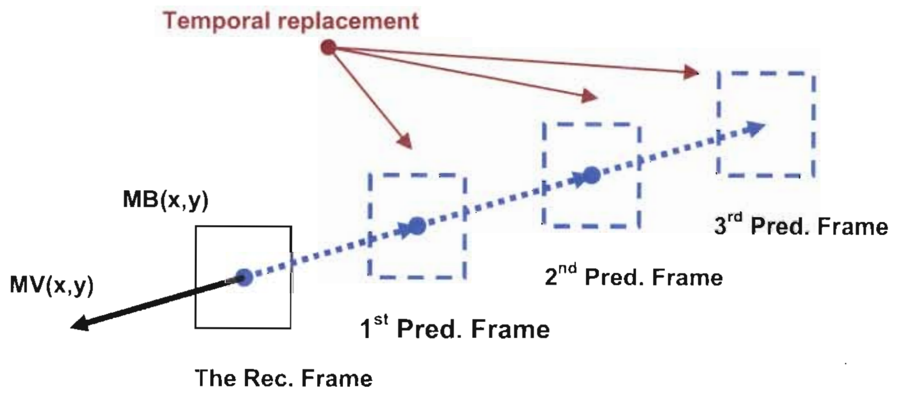


Figure 4.5: **MV inversion followed by motion extension.**

Motion vectors shown in dashed lines represent the predicted MVs. As the figure shows, the block will keep moving along the same direction that it came from. The moving distance is simply proportional to the number of frames to be synthesized. The post-processing for empty areas and overlapped areas is as described in Chapter 4.

The computation cost is slightly increased compared to method-0 due to temporal prediction and post-processing. However, since only the final predicted frame is synthesized (and no intermediate frames), the overall processing time remains almost the same no matter how far the prediction goes.

4.3.3 Method 2: Motion Segmentation, VMF, Sequential Frame Synthesis

Method-2 has more complicated motion prediction than method-1. First, this method includes motion segmentation and vector median filtering to reduce prediction errors. Second, the sequential frame synthesis loop is introduced, whereby each intermediate frame between the last received frame and the final future frame is synthesized one by one, using previously synthesized frames as references.

As illustrated in Section 4.2, the proposed motion segmentation algorithm nicely isolates moving objects from the background. Hence, motion prediction is able to identify areas of homogeneous motion and use some filtering techniques to predict consistent motion within each particular region. The vector median filter, which has an excellent trade-off between complexity and smoothing performance [5], is chosen for this purpose.

Since all intermediate frames are synthesized, the complexity of this method is obviously higher than the complexity of the previous two methods, and is proportional to the distance between the last received frame and the final frame we wish to synthesize.

4.3.4 Method 3: VBS Motion Prediction and Region Smoothing

This is the final full-featured prediction method. Compared to the previous three methods, here the motion smoothing is applied to get rid of false MVs and MV noise in large homogeneously-moving areas, and variable-block-size motion prediction is adopted near the edges of moving objects to further improve visual quality. This method has the highest complexity among all methods, but it also shows the highest visual quality and Peak Signal-to-Noise Ratio (PSNR), as will be illustrated below.

4.4 The Overall Assessment of Frame Prediction

In the assessment of the four frame prediction methods we described in the previous section, we measure both the objective and subjective quality of the predicted frames.

4.4.1 The Objective Quality

The predicted video frames can be qualified in an objective way, and such a measure is the mean square error (MSE) between the original raw video sequence and the predicted video sequence. The most widely used video quality measure is the Peak Signal-to-Noise Ratio (PSNR), defined in equation 4.1 in decibels (dB). The difference between luminance (Y) components of the original and predicted frame is calculated pixel by pixel, and then an average over the entire frame is taken.

$$PSNR(dB) = 10 \cdot \log_{10} \frac{255^2}{E\{(Y_{original} - Y_{decoded})^2\}}$$

Eq. 4.1

In this section, PSNR in dB is used to assess frame prediction in two different ways. First, we measure the displayed video quality for all four methods we discussed in section 4.3. We run the prediction for up to three frames ahead of time for each method, and PSNR is used for the comparison of their performance.

Second, we examine the effects of block size used for motion estimation at the encoder on the performance of frame prediction at the decoder. In particular, we compare the prediction performance in the case where encoder forces all MVs to be 8×8 against the case where the encoder can choose to assign a MV to either an 8×8 or a 16×16 block.

4.4.1.1 The Quality vs. Frame Prediction Depth

Fig. 4.6 – 4.8 shows how video quality measured by PSNR in dB depends on how far ahead we predict. The results are sorted from high motion level sequences to low motion level sequences. All these simulation were done with 30 fps sequences.

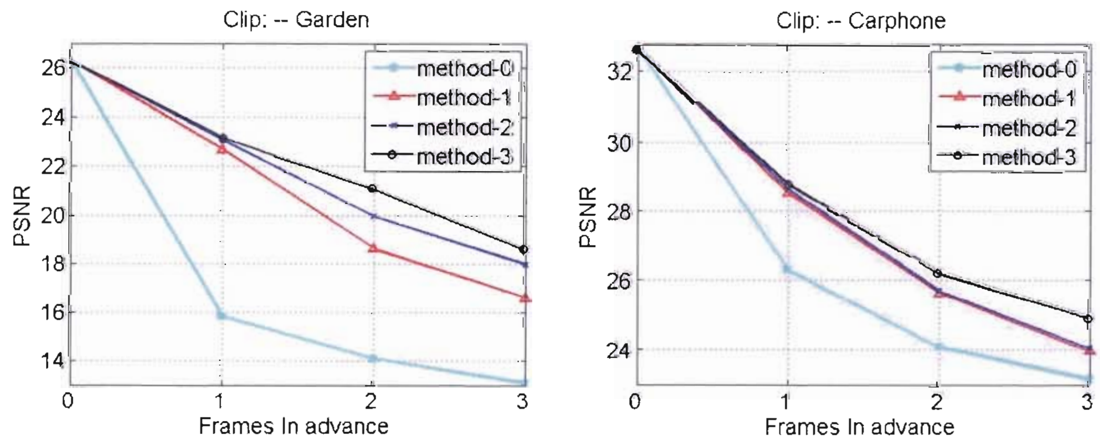


Figure 4.6: Prediction performance in PSNR (dB) – High Motion Level

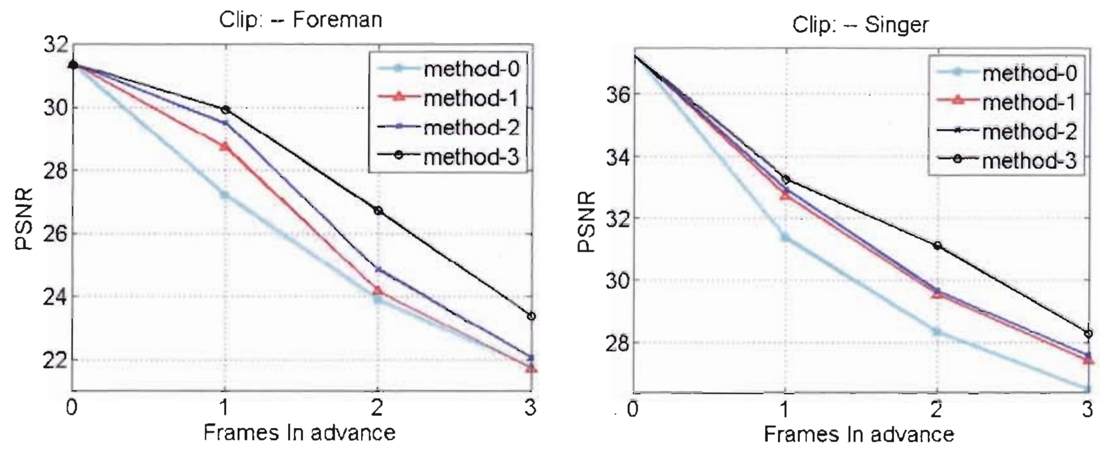


Figure 4.7: Prediction performance in PSNR (dB) – Medium Motion Level

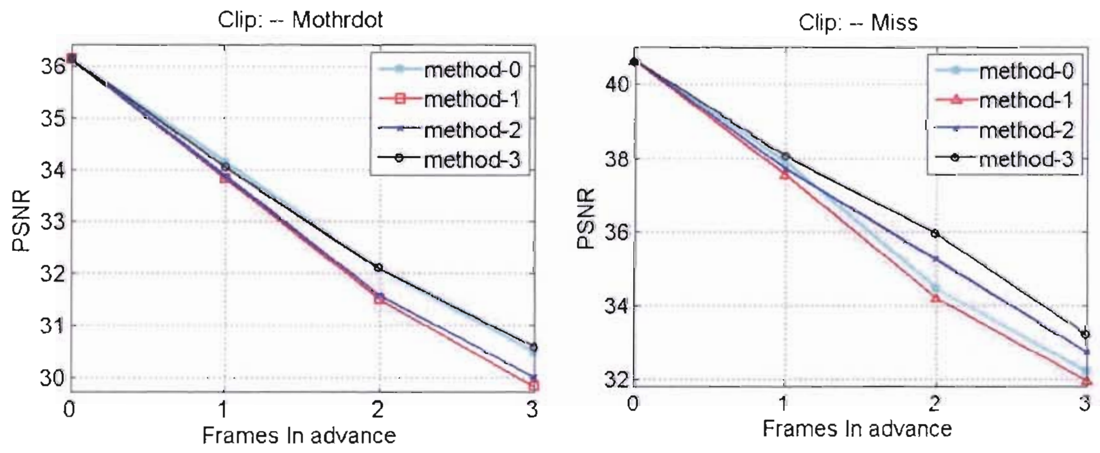


Figure 4.8: Prediction performance in PSNR (dB) – Low Motion Level

From the PSNR plots, we can observe the following:

- The PSNR is decreasing as the prediction goes further. The quality decay varies according to the motion activity level and texture pattern in the sequence. *Garden* has the most complicated texture pattern and its PSNR (method-3) drops up to 3 dB when predicting one frame ahead of time, while the PSNR of method-0 drops 10 dB after predicting the first frame.
- On sequences with relatively high motion levels, like *Garden*, *Singer*, *Carphone*, and *Foreman*, the full-featured method-3 outperforms other methods.
- For sequences with relatively low motion, like *Miss America* and *Mother & Daughter*, the sophisticated frame prediction method doesn't provide as much improvement as for high-motion sequences. On these sequences, even simple methods do reasonably well.

4.4.1.2 The Frame Prediction Performance vs. Frame Rate

The frame rate is another important factor which links prediction depth and delay reduction. Figure 4.9, 4.10 and 4.11 show how the frame prediction performs at different frame rates.

During the experiments, each test sequence is subsampletd from its native frame rate of 30 fps down to 15 fps and 7.5 fps. Prediction depth goes up to 3 frames ahead for each frame rate. Figure 4.9 illustrates the relationship between the frame prediction performance and frame rate for high motion sequences, while Figure 4.10 and Figure 4.11 are for medium motion and low motion sequences, respectively.

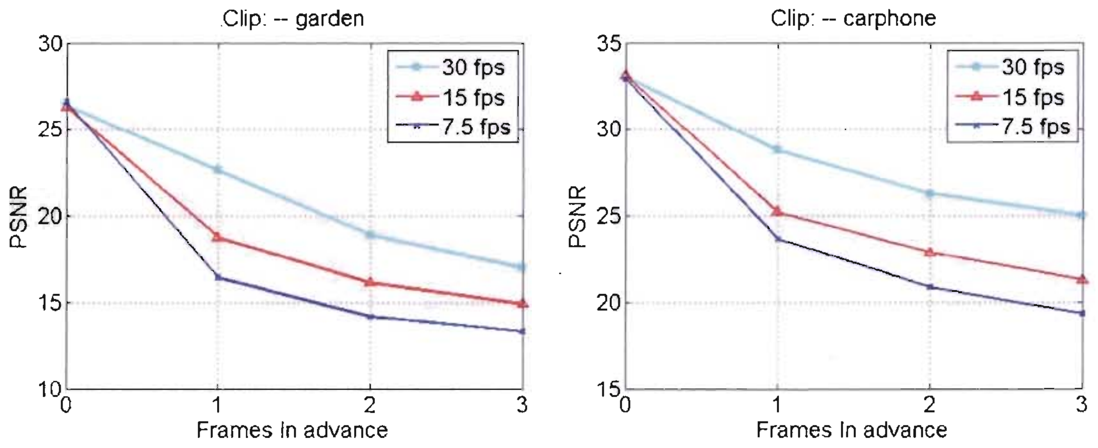


Figure 4.9: Prediction performance VS. Frame Rate – High Motion Level

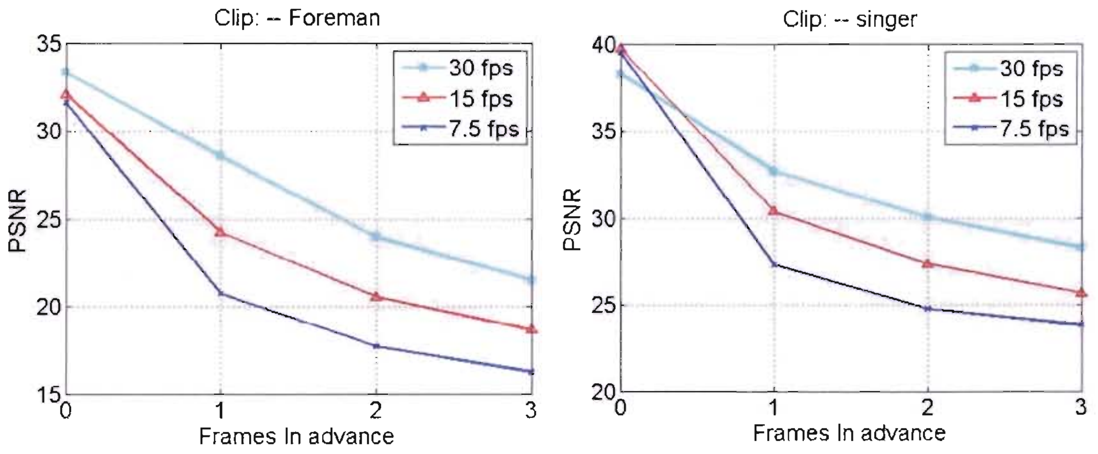


Figure 4.10: Prediction performance VS. Frame Rate – Medium Motion Level

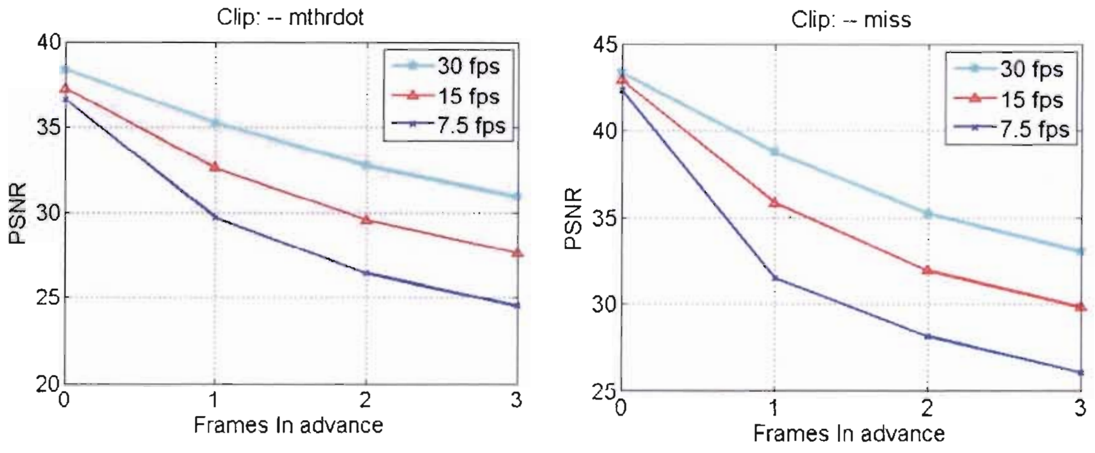


Figure 4.11: Prediction performance VS. Frame Rate – Low Motion Level

The simulation results are consistent with our expectations. Prediction is better at high frame rates. We can also observe that the quality (PSNR) of predicted frames is approximately determined by perceived delay reduction. For example, predicting one frame ahead with 15 fps *Foreman* gives us similar quality to predicting two frames ahead with 30 fps *Foreman*. In other words, the cost of quality loss to reduce perceived end-to-end delay for high frame rate source is similar to the quality loss incurred with its low frame rate counterpart.

This simulation shows that the frame prediction performance is mainly related to the amount of delay we want to reduce. However, video source with high frame rate offers us more flexibility in frame prediction than the low frame rate source. For example, if the user can accept the video quality at 150 ms delay reduction, then with 30 fps video communication, we will have options for delay reduction at 33ms, 67ms, 100ms, and 133ms, while for 15 fps video source we only have options 67ms and 133 ms, and only one option (133 ms) for 7.5 fps source.

4.4.1.3 The Frame Prediction Performance vs. Motion Level

The results in the previous figure indicate the average PSNR over the entire sequence. In this section, we further investigate the frame-by-frame PSNR fluctuation through the sequence, and look into the relationship between quality and motion level. For this experiment, choose four test sequences with different motion levels: *Carphone* (High), *Foreman* (Medium), *Miss America* (Low), and *Mother & Daughter* (Low). Method-3 is chosen as the prediction method, since it gave the best results overall in the previous section.

In Fig. 4.12 to Fig. 4.15, we plot the frame-by-frame PSNR for different prediction depths.

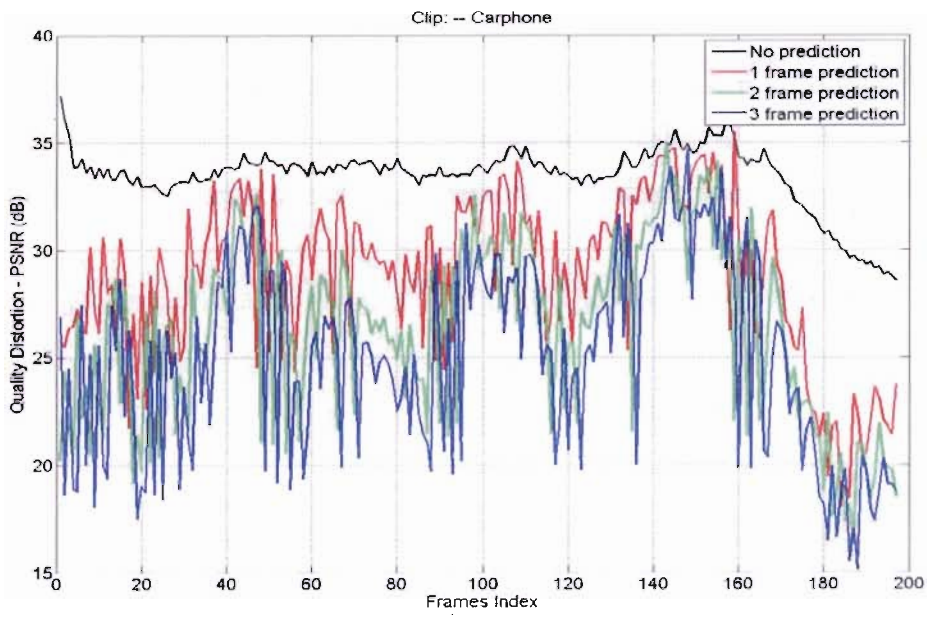


Figure 4.12:The PSNR of frame prediction vs. The motion level – *Carphone*

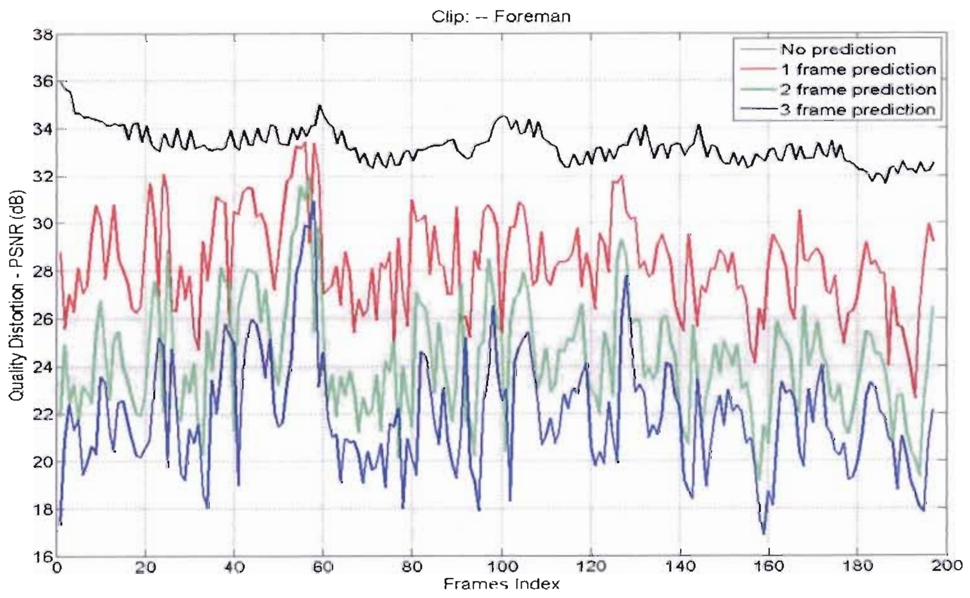


Figure 4.13:The PSNR vs. The motion level – *Foreman*

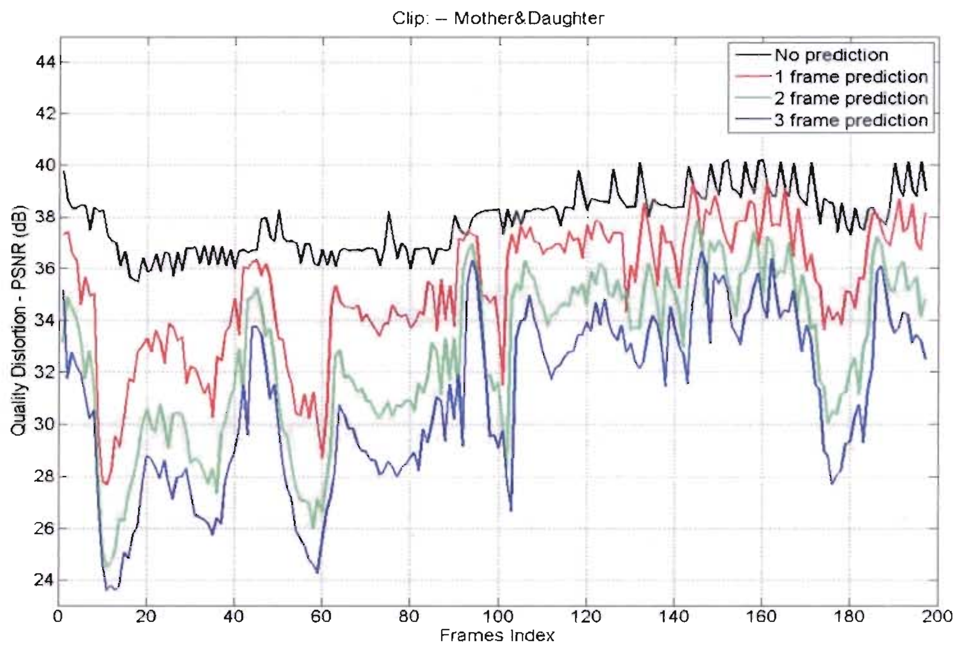


Figure 4.14:The PSNR vs. The motion level – *Mother & Daughter*

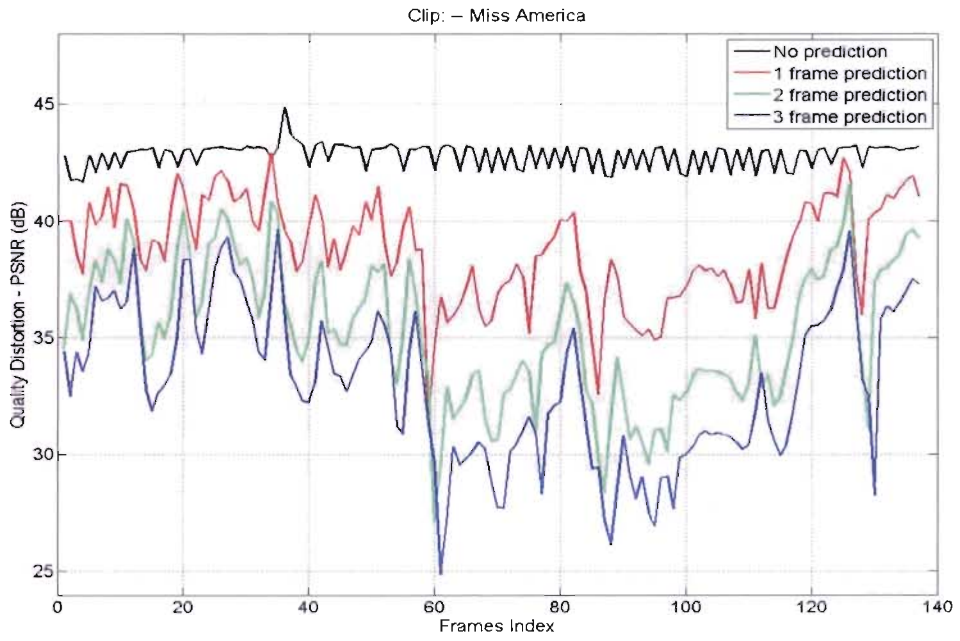


Figure 4.15:The PSNR vs. The motion level – *Miss America*

These plots indicate that prediction performance depends on the motion intensity in a particular segment of the sequence. Prediction is much better in low-motion sequences such as *Miss America* and *Mother & Daughter*, then it is on the higher-motion sequences like *Foreman* and *Carphone*. Within the *Carphone* sequence, prediction deteriorates towards the end of the sequence as the motion intensity increases.

4.4.1.4 The Frame Prediction Performance vs. Motion Field Density

As mentioned in Chapter 3, motion vector accuracy is a very important factor affecting the overall frame prediction performance. What we have discussed so far only concerns the decoder; the encoder operates independently without any knowledge that prediction will be carried out at the decoder. In this section, we investigate the impact of motion estimation at the encoder side, and how different coding techniques affect the prediction system. In particular, we investigate the effects of motion field density on prediction accuracy. MPEG-4 supports motion vectors based on 16×16 or 8×8 blocks. In variable-block-size motion estimation, the encoder may choose to split a 16×16 block into four 8×8 sub-blocks, if this reduces the energy of the prediction residual. One can also force the encoder to use fixed-size 8×8 blocks only.

Decisions regarding the block size for motion vectors (which in turn determine the "density" of the motion field) are made in the video encoder based on the Sum of Absolute Differences (SAD) information from motion estimator. When the function that makes these decisions is enabled, four 8x8 blocks are used for inter-coding if the SAD sum of four 8x8 blocks is less than the SAD of one 16x16 block; otherwise, 16x16 block is chosen.

An alternative is to force four 8x8 blocks for inter-coding for the entire sequence by disabling block size decision function. In this way, we might end up with a denser motion vector field, which may help prediction. On the other hand, the total bit rate will also increase due to the cost of encoding extra motion vectors.

The purpose of fixing the 8x8 coding mode is to increase the density of the motion vector field, which may benefit motion segmentation as well as subsequent motion prediction. However, in addition to increasing the bit rate, there is another risk associated with this. Smaller blocks may lead to more noise in the motion vectors. To identify how much improvement we can get in frame prediction in terms of motion field density and how much penalty we might suffer from video quality in terms of bit rate, we carried out the experiments on four sequences with different motion levels: *Foreman*, *Carphone*, *Miss America*, and *Mother & Daughter*.

In our experiment, we generate two compressed video files for each video sequence: one for adaptive 16x16 block size coding mode, and the other one for fixed 8x8 block size coding mode. These two video streams are fed to the frame prediction at the decoder separately, and their PSNR is measured for different frame prediction depth. Since the two coding modes might lead to different bit rates, to make a fair comparison we made the PSNR plots in two different ways:

- Fixed encoded video quality (PSNR) with different bit rates,

- Different encoded video qualities (PSNR) at the same bit rate.

Table 4.2 shows the bit rate difference when we use the two different coding modes to achieve the same quality (PSNR).

Table 4.2: Bit rates needed to maintain the same video quality.

Sequence	PSNR (dB)	Bit-Rate (Adaptive 16x16 mode)	Bit-Rate (fixed 8x8 mode)
Foreman	31.5	70 kbps	144 kbps
Carphone	33.7	65 kbps	144 kbps
Mother & Daughter	32.4	20 kbps	149 kbps
Miss America	39.6	55 kbps	134 kbps

As expected, the table shows that the adaptive 16×16 mode is more efficient that the fixed 8×8 mode. Efficiency difference between these two modes is highest on low-motion sequences, where lack of motion can be easily exploited by the large block size. The difference in efficiency reduces as the motion intensity and complexity increase.

Fig. 4.16 – Fig. 4.19 illustrate the frame prediction performance in terms of different motion field densities for four different video sequences. The plots on the right side correspond to the case where the encoded PSNR is the same for the two modes, and the plots on the left side correspond to the case where the bit rates of the two modes are the same.

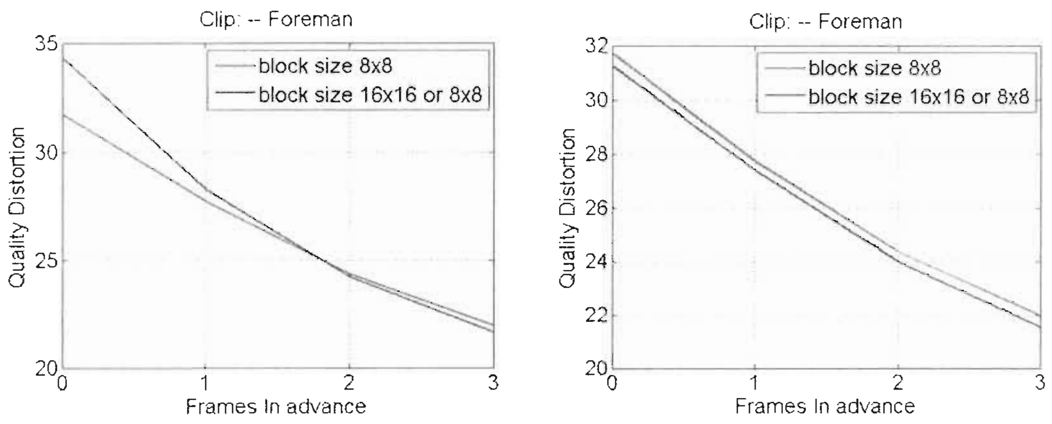


Figure 4.16: Prediction performance vs. Motion density: *Foreman*, Left: Same bit-rate, Right: Same encoded PSNR.

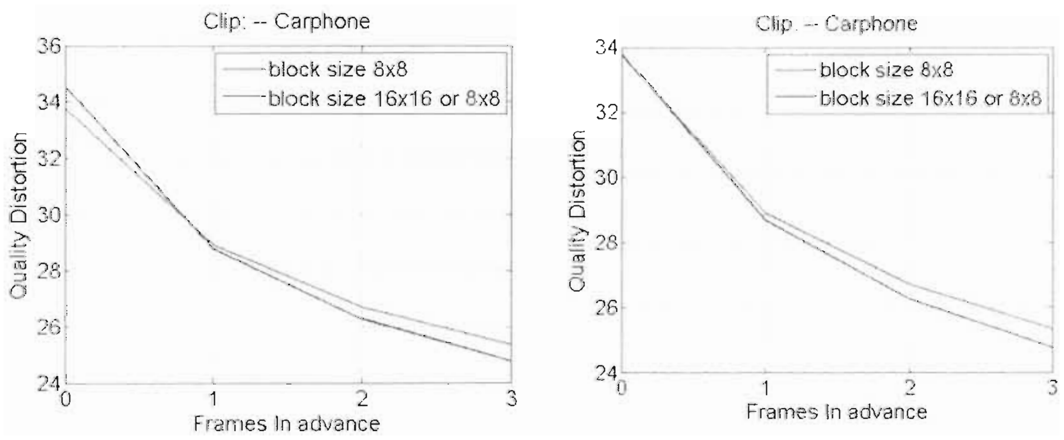


Figure 4.17: Prediction performance vs. Motion density: *Carphone*, Left: Same bit-rate, Right: Same encoded PSNR.

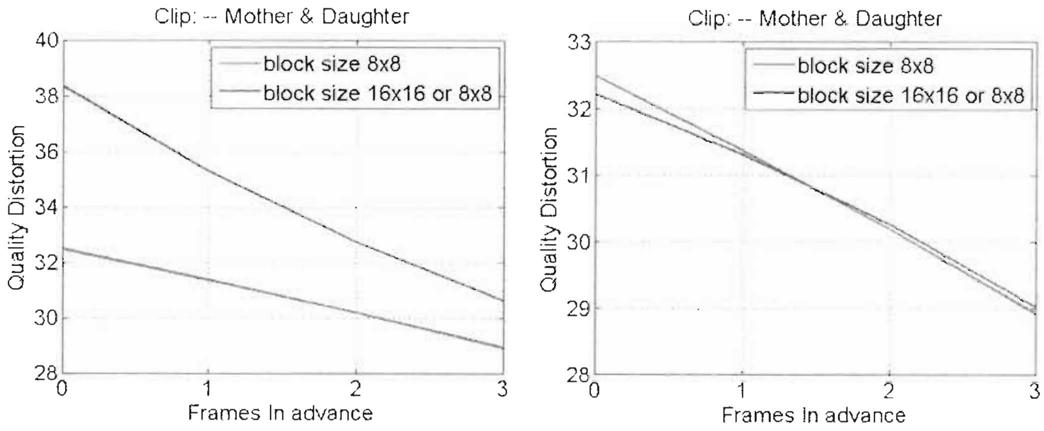


Figure 4.18: Prediction performance vs. Motion density: *Mother & Daughter*, Left: Same bit-rate, Right: Same encoded PSNR.

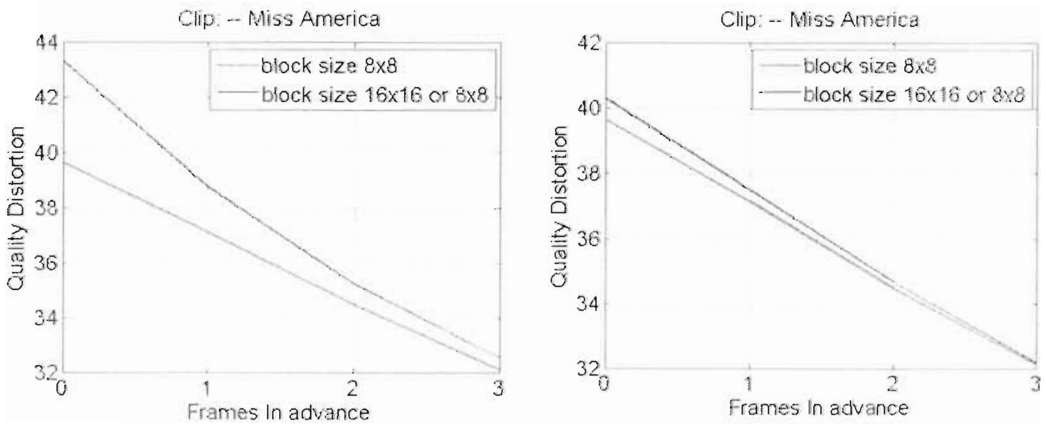


Figure 4.19: Prediction performance vs. Motion density: *Miss America*, Left: Same bit-rate, Right: Same encoded PSNR.

We can observe that increased motion vector density doesn't give too much improvement on the performance of frame prediction, especially when encoded bit streams have the same PSNR. The main reason is that increased motion density brings higher risk of false motion vectors in homogeneous areas, and motion accuracy turns out to be more important than motion field density in frame prediction.

By studying the impact of motion field density on frame prediction, we have come to the conclusion that increasing the motion field density by forcing small block size in motion estimation is not beneficial for predictive decoding. Using the default adaptive 16×16 block-based motion estimation produces less dense motion field, but has the benefit of improved coding efficiency and higher motion vector accuracy, which is crucial for prediction.

4.4.2 The Subjective Quality

In terms of subjective performance, the full-featured method is visibly better than all other methods on all sequences. An illustration is given in Fig. 4.20, which shows a sample *Foreman* frame produced by three different frame prediction methods (method-1 through method-3) when predicting three frames ahead. In the upper left corner, we also show the original frame for comparison.

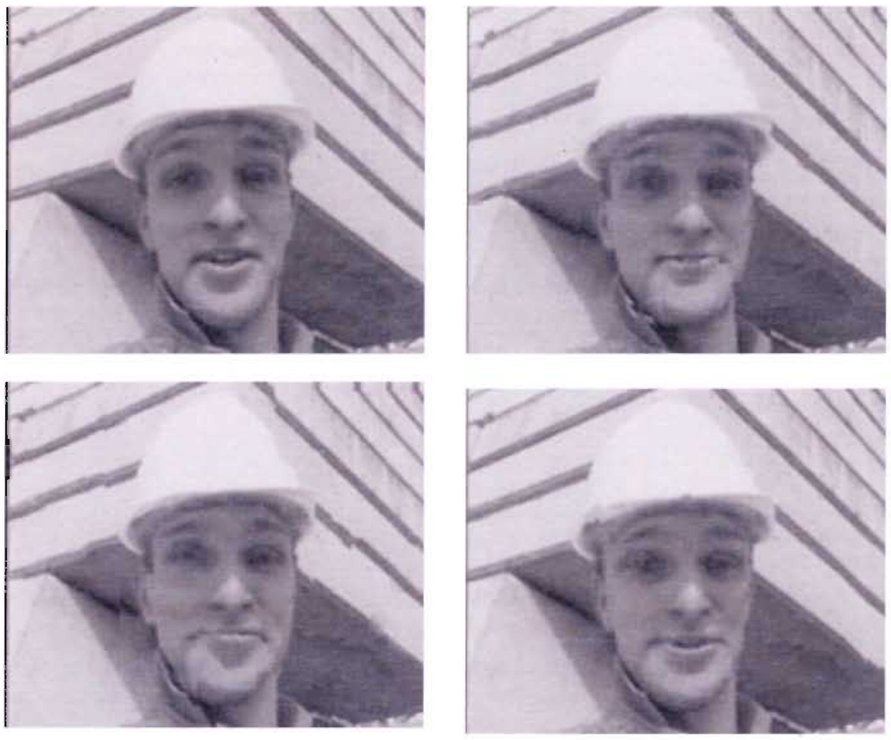


Figure 4.20: Visual comparison for frame prediction assessment. [Top left] is the frame without prediction, and other figures show the same frame produced when predicting three frames ahead by three prediction methods. [Top right]: Method 1, [Bottom left]: Method-2, [Bottom right]: Method-3

Finally, in Fig. 4.21 – Fig. 4.23, we show how the predicted frame quality deteriorates as the prediction depth increases from zero to three. The PSNR curve is also included in the figures. As expected, the further ahead we predict, the lower the quality of the predicted frames.

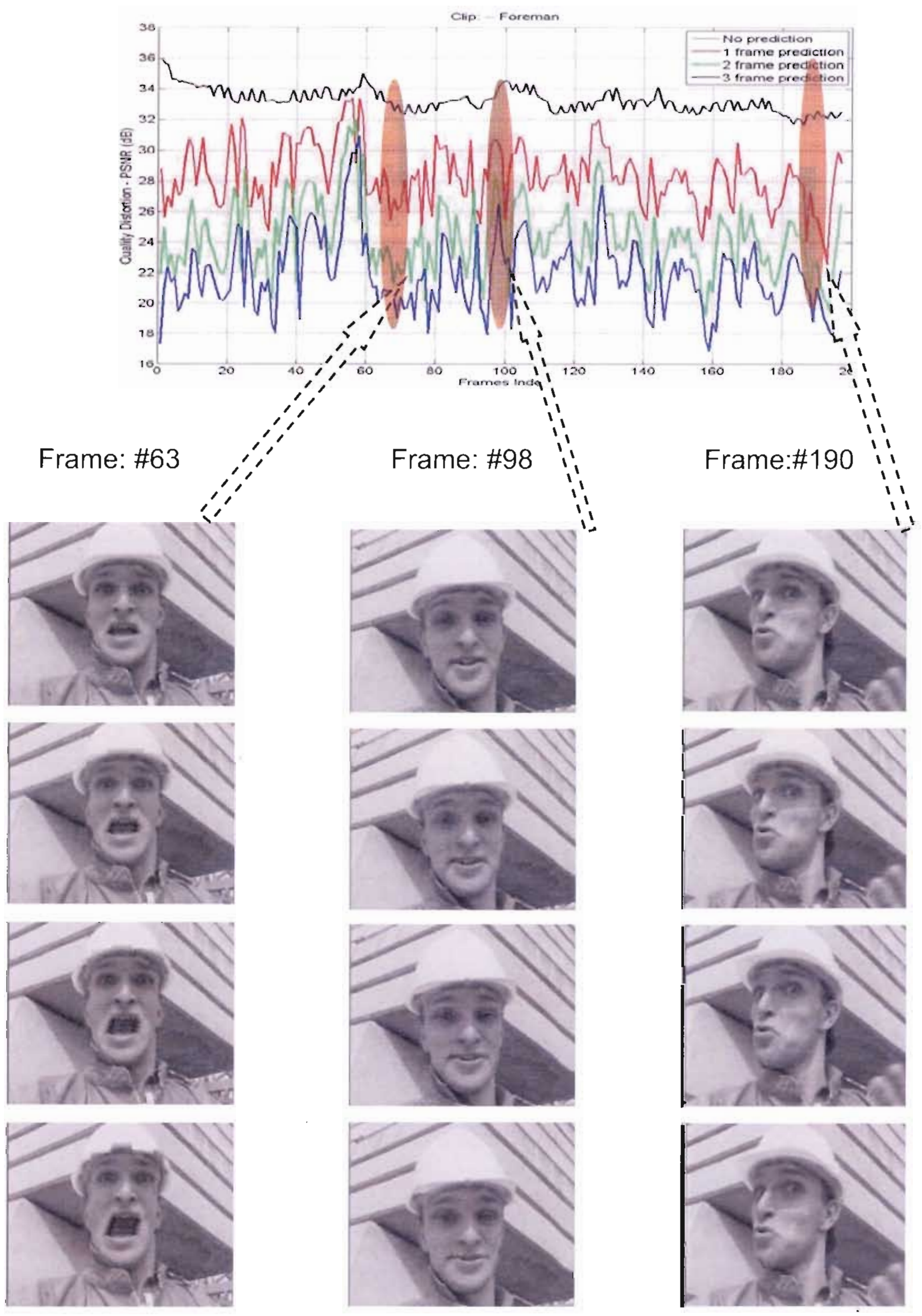


Figure 4.21: **Quality of frame prediction (*Foreman*).**

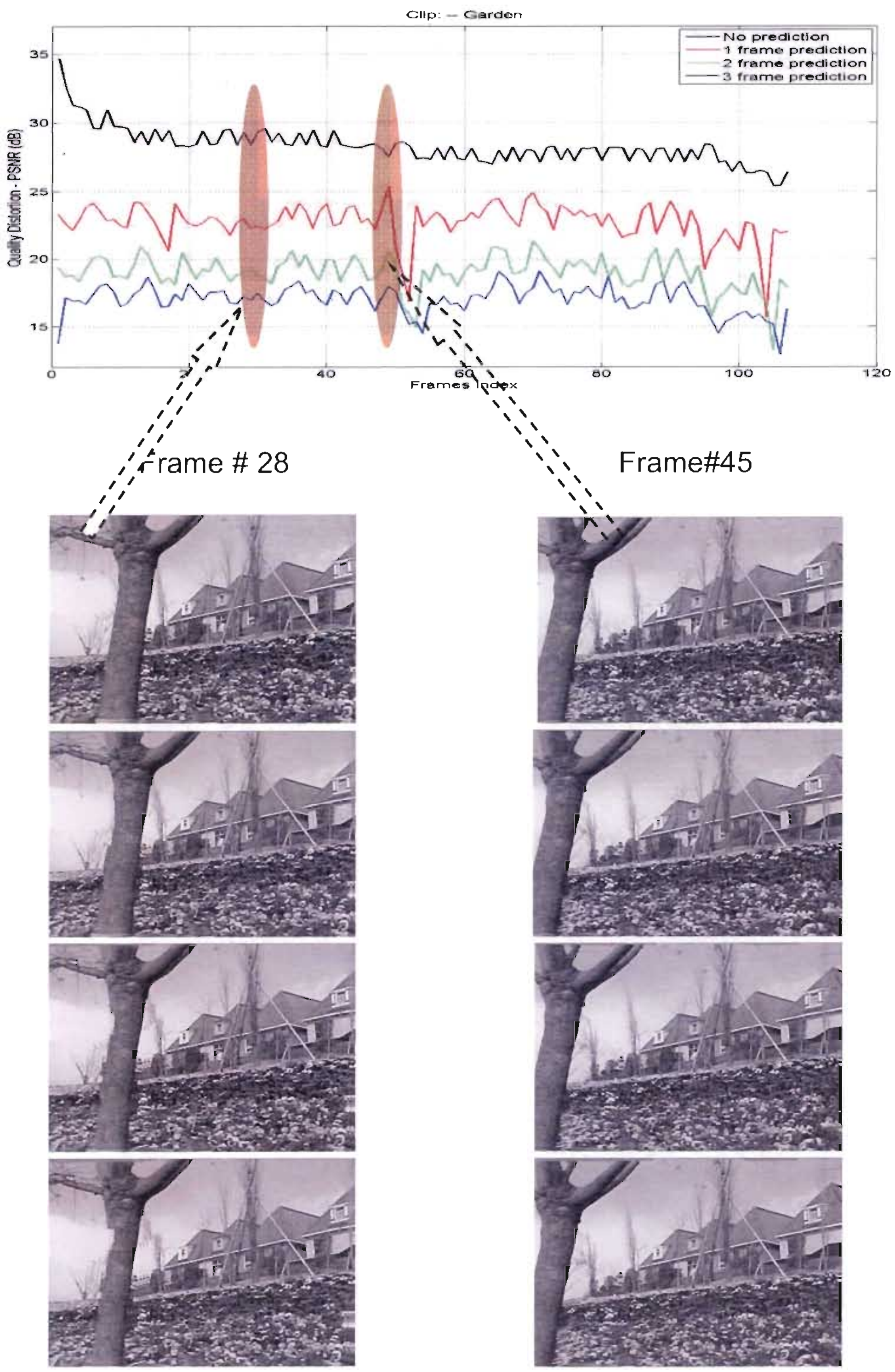


Figure 4.22: Quality of frame prediction (*Flower Garden*).

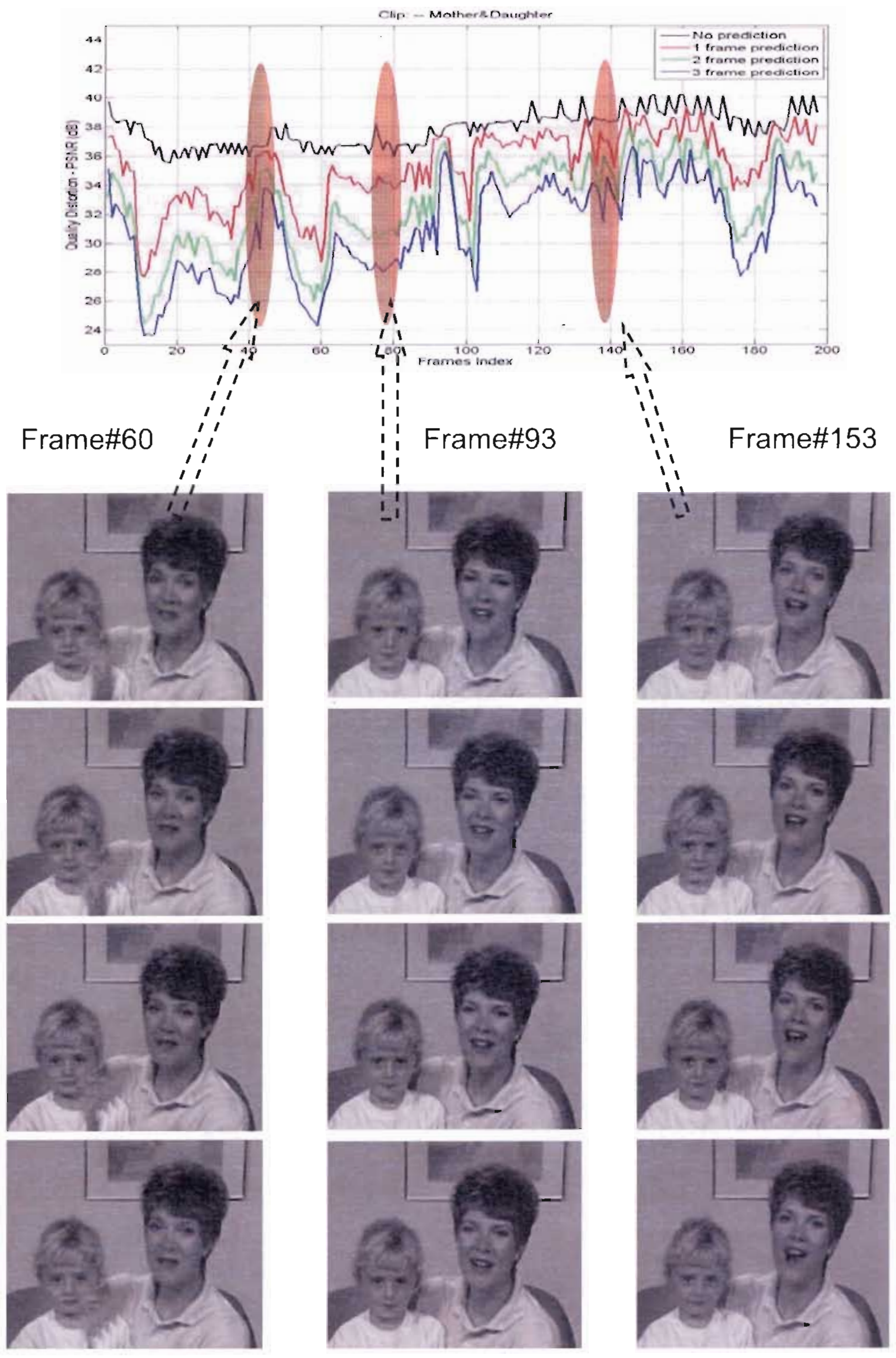


Figure 4.23: **Quality of frame prediction (*Mother & Daughter*).**

Fig. 4.21 – Fig. 4.23 illustrate how the quality of predicted frames deteriorates as frame prediction depth increases. In each of the figures, the PSNR curves and the sample frames are arranged from top to bottom in the following way: No prediction, prediction 1 frame ahead of time, prediction 2 frames ahead of time, prediction 3 frames ahead of time. The locations of the sample frames within the sequence are also indicated..

We observe that the degradation in PSNR is related to the motion level associated with the particular segment within the sequence. For those frames from very high-motion segments, the PSNR of the predicted frames drops quickly, and increased prediction depth leads to lower PSNR, such as Frame #60 in *Mother & Daughter*, and Frame #28 in *Foreman*. For frames associated with medium or very slow motion, the quality degradation is relatively low as prediction depth increases, which can be observed on Frame #93 in *Mother & Daughter*, and Frame #98 in *Foreman*.

From the visual quality comparison, we can conclude that having accurate motion prediction is the key to improve the quality of the entire frame prediction. This point will be addressed as a possible future work in the next chapter, where we suggest a couple of methods to improve motion vector prediction.

4.5 The Complexity Analysis

In this section, we analyze the complexity of frame prediction module in full-featured configuration (method-3 in section 4.3.4). The complexity analysis consists of two aspects:

(1) The overall time consumption of the frame prediction module compared to the decoding time consumed by standard MPEG4 video decoder;

(2) The Million Instructions Per Second (MIPS) dissipation on all processing units in the frame prediction module.

The entire frame prediction module is embedded into the XviD MPEG4 video decoder, and the simulations were conducted on a desktop PC with Intel Pentium CPU 3.0 GHz and 1.99 GB of RAM. Table 4.3 shows the simulation results in terms of time consumption for standard video decoder and multiple frames prediction for six 30 fps test sequences.

Table 4.3: Decoding and prediction time.

Video Clip	Format	Decoding Time - ms			
		STD Dec	1 ahead	2 ahead	3 ahead
Foreman	QCIF (176x144)	0.26	1.93	4.74	6.72
Carphone	QCIF (176x144)	0.16	1.92	3.69	6.41
Singer	CIF (352x288)	1.38	6.6	13.9	21.5
Mother & Daughter	QCIF (176x144)	0.42	1.35	3.28	4.69
Flower Garden	SIF (352x240)	2.87	8.84	17.95	27.3
Miss America	QCIF (176x144)	0.61	1.87	3.33	5.42

The goal of overall decoding and prediction time testing was to gather information about the resource demand of frame prediction embedded into MPEG4 decoder running on a PC platform in real time. The table 4.3 gives us a preliminary knowledge of algorithm complexity with an un-optimized version of the frame prediction module.

First, in order to achieve real-time video decoder with frame rate at 30 fps, current frame prediction implementation can support the video resolution up to CIF or SIF, while predicting three frames ahead of time, and reducing delay up to 100 ms. For lower resolution such as QCIF, the overall time consumption, including frame reconstruction and three-frame prediction, is less than 10 ms, thus it meets the requirement of running in real time. However, if a real-time encoder needs to run on the same system (as in typical PC-based videoconferencing), the current frame prediction module would need heavy optimization to reserve enough MIPS, especially for higher resolutions like CIF and SIF.

Second, compared to the standard video decoder, predicting one future frame costs significant extra computation - about 3 times more in low motion sequences such as *Miss America* and *Mother & Daughter*, and even higher in medium/high motion sequences like *Foreman* and *Carphone*. The motion level results affect the time consumption of frame prediction because a large portion of prediction effort is spent on MV prediction and motion segmentation. Therefore, it is necessary to further block down the MIPS dissipation over internal processing units of the prediction module, and come up with the corresponding optimization plan.

Figure 4.24 to Figure 4.26 show the MIPS distribution of internal processing units in the frame prediction module. We analyze the average MIPS dissipation on sequences with different motion levels. The processing units are categorized into Pre-Processing, MV prediction (including Motion Segmentation), Temporal prediction, Post-processing for overlapped areas, and Post-processing for empty areas.

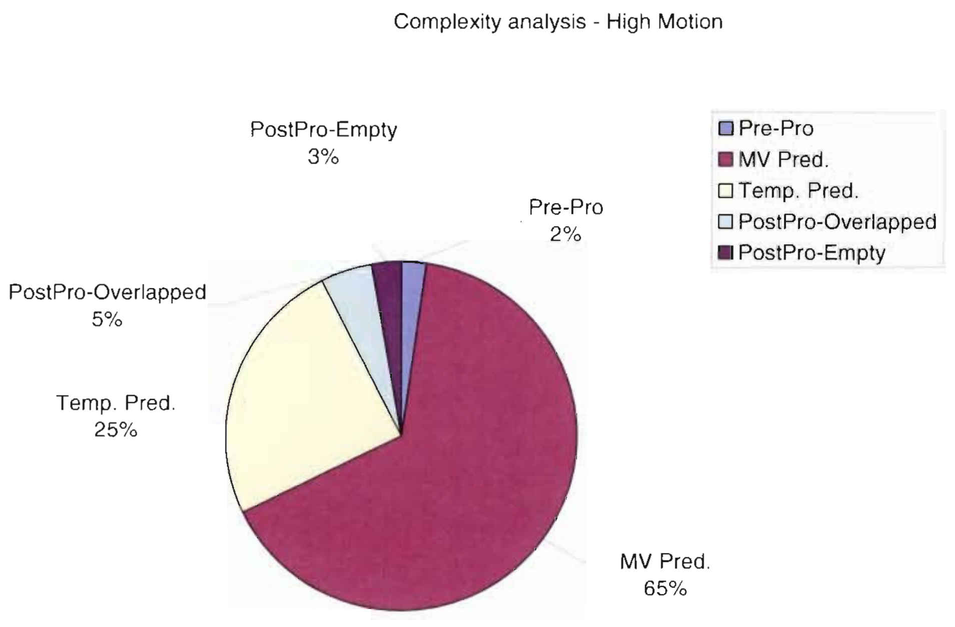


Figure 4.24: MIPS dissipation of frame prediction (*High Motion*).

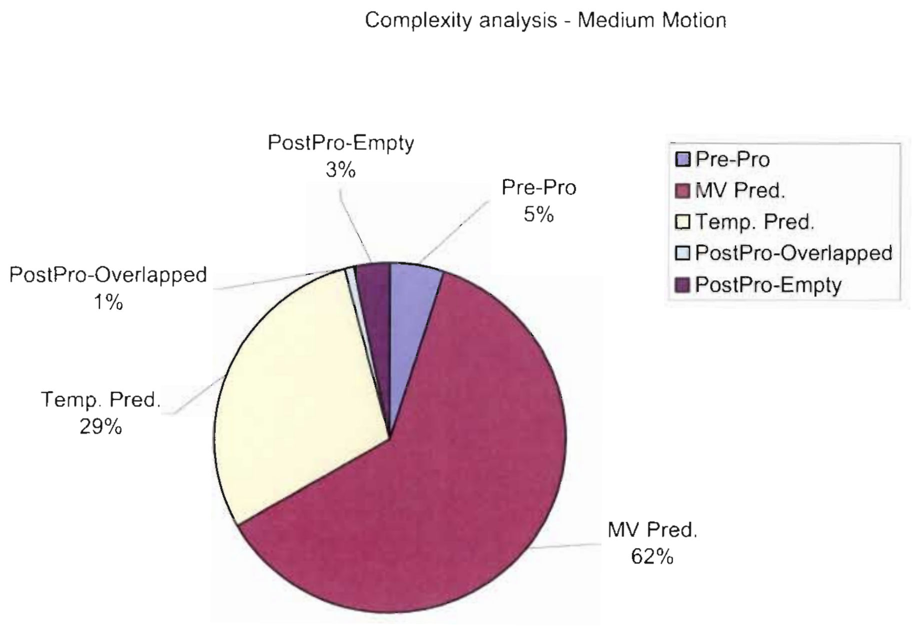


Figure 4.25: MIPS dissipation of frame prediction (*Medium Motion*).

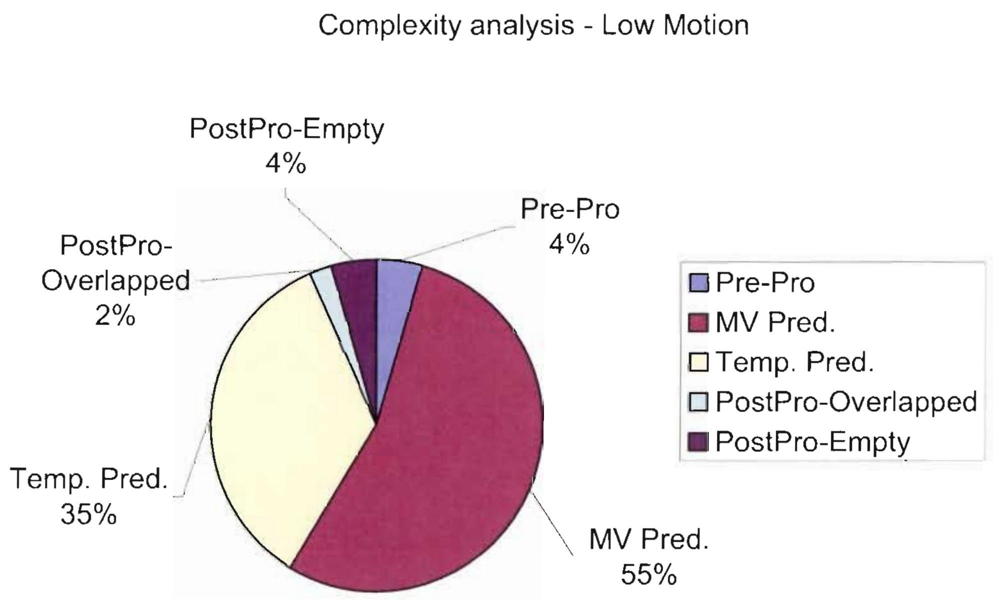


Figure 4.26: MIPS dissipation of frame prediction (*Low Motion*).

These pie charts show a clear picture of computational demand of different frame prediction processing units. First, more than 50% MIPS consumption comes from MV prediction part, which includes motion compensation and variable-block-size MV prediction, and takes a somewhat larger percentage of overall computation for higher motion sequences than for lower motion sequences. Second, temporal prediction has the second highest computational complexity, because this unit is responsible not only for synthesizing the preliminary future frame, but also for generating all position information for later post-processing, such as pixel mapping table and overlapped pixel table. Those processing steps are all time-consuming. These two parts, Temporal prediction and MV prediction make up to about 90% of overall MIPS consumption, thus our future optimization effort will mainly concentrate on these components.

By conducting the complexity analysis on MPEG4 video decoder, we have obtained a rough knowledge of computational resource demand of frame prediction integrated into a video decoder. However, as we mentioned in Chapter 2, our frame prediction module can also be ported to other block-based video codecs, such as H.261, H.263, and H.264. All these video codecs have different performance characteristic in terms of motion estimation, transformation techniques and so on, thus the complexity will vary accordingly as frame prediction module is tailored to fit each individual codec. In Chapter 5, we will discuss how the frame prediction module can be customized to H.264 decoder, and how this would affect the complexity.

CHAPTER 5
CONCLUSION AND FUTURE DIRECTIONS

In this work, we addressed the issue of delay in video communications, and proposed predictive decoding to reduce the delay. The main idea behind our delay reduction method is to incorporate a frame prediction module into a standard video decoder, predict the upcoming video frames from the available video data, and display them before they arrive at the receiver.

By using this "predictive decoding," it is possible to reduce the perceived end-to-end delay at the expense of displayed video quality. The key steps in this decoding scheme include: a) Motion segmentation-based motion vector prediction, which considers both prediction residuals from the last received frame and motion region homogeneity to improve motion prediction. b) Temporal prediction is used to synthesize future frames using the predicted motion vectors. c) A combination of bi-linear interpolation and boundary matching is used as a post-processing step to deal with overlapped areas and empty areas.

The benefits we may get from frame prediction depend on the complexity of the video. For scenes involving complex motion, the number of frames which can be predicted with acceptable quality is very limited. However, for scenes with less complex and relatively smooth motion, as we might expect in a video conference, predictive decoding shows promising performance.

5.1 Possibilities for Improving Prediction Performance

By studying the reasons behind predicted video quality degradation, we identified some possibilities for improving prediction performance.

First, our current motion prediction strategy only considers the last received frame, and all predicted motion vectors are based on this one frame and its motion vectors. Although motion objects are segmented carefully, significant amount of motion vector noise is still likely to remain in the motion field, which reduces prediction performance.

Second, the same prediction strategy is used for all frames. However, it is likely that motion activity changes from time to time in a video, and different prediction strategies may be appropriate for different segments. Hence, recognizing the type of motion in a particular segment and changing the prediction strategy accordingly may be another way to improve prediction performance.

Finally, without an efficient motion trajectory model, it is hard to keep a consistent motion region, and we commonly observed different shapes of motion regions in successive frames. Good motion trajectory models for objects that are commonly observed in particular applications (e.g. head or mouth in video conferencing) would go a long way towards improving prediction performance. It is also likely that using these models would increase the complexity of predictive decoding.

In summary, further improvements might be obtained by a) doing better motion prediction at the decoder side based on motion trajectory models, b) doing joint motion estimation and prediction, which optimizes the motion estimator at the encoder side according to a suitably chosen motion model, and transmitting precise motion information to aid the MV predictor at the decoder.

Motion Trajectory Model

By establishing a motion trajectory model, the historical motion information from past video frames are collected and used to extrapolate the motion trajectory. This way, it may be possible to predict more accurately the future position and orientation of a moving object in a time-variant environment.

There have been some studies concerning filtering techniques and autoregressive models to characterize motion in video processing. For example, reference [21] utilizes the predicted motion and measured motion to obtain the optimal estimate of motion vectors, where the predicted motion information is obtained by autoregressive models, and Kalman filters are used for prediction.

Joint Motion Prediction Method

The idea here is to optimize the motion estimator at the encoder side, so that the motion vectors generated by motion estimation can better approximate the actual moving directions of objects, instead of purely minimizing the prediction error. With more accurate motion vectors, the decoder may be able to do better motion prediction by setting up the same motion model, and having the entire predictive decoding improved.

5.2 The Frame Prediction in H.264/AVC

H.264/AVC is the latest video coding standard and it is designed for a variety of technical solutions including interactive applications [13]. H.264/AVC enhancees coding efficiency significantly compared to MPEG4 and H.263 through various optimizations such as rate-distortion optimal motion estimation [51][52], which would directly affect the performance of frame prediction. In this section, we will briefly discuss the R-D optimal motion estimation in H.264/AVC, and how it might affect frame prediction in a H.264/AVC decoder.

The rate-distortion optimal motion estimation algorithm in H.264/AVC leads to an efficient bit allocation between inter- and intra- frame information. The motion information is determined by searching the optimal solution of R-D pairs to minimize the total Lagrangian cost function

$$J = D + \lambda R$$

Eq. 5.1

where D denotes the prediction error, R represents the bit rate spent on motion vectors and λ is the Lagrange multiplier related to the quantization parameter QP [52]. With a reasonable bit rate, the motion vectors packed in the bit stream nicely reflect the real motion, therefore, the noisy motion vectors [30] might be suppressed in H.264/AVC. Frame prediction might benefit from R-D optimal motion estimation in following aspects as well.

1. Reduce the computational demand on motion segmentation and vector median filtering. The purpose of motion segmentation is to identify the

areas with homogeneous motion, while vector median filter smoothes out noisy motion vectors. Since the motion vectors have been locally refined by R-D optimal motion estimation in H.264/AVC, the need for motion segmentation may be reduced. This could save up to 40 percent of total computation.

2. H.264/AVC offers variable-block-size motion estimation [50], with block sizes down to 4x4. Also, there is an in-loop deblocking filter in the decoder to reduce the block boundary artifacts. These techniques may simplify the variable-block-size MV prediction, and improve both objective and subjective video quality.

To realize frame prediction algorithm in H.264/AVC, it is important to achieve a good balance between complexity reduction and quality improvement. It is also a part of our future work to give a comparison of prediction performance using different video codecs.

5.3 Comprehensive Video Quality Assessments

Besides seeking the possibility for improving prediction performance, accurate assessment of the perceptual quality of predicted frames is also important. PSNR is the main quality measure used in this thesis because of its simplicity. However, it is well-known that PSNR doesn't accurately model the perceptual quality.

There are quite a few of new video quality measurement approaches proposed in recent years, such as VQM [28], MPQM [46], SSIM [47], and NQM [49]. Each of them has a unique value in measuring perceptual video characteristics. How to combine these methods to give a comprehensive video quality assessment is also part of our future work.

The Video Quality Measure (VQM) [28]

The Video Quality Measure (VQM), developed by the Institute for Telecommunication Sciences (ITS), is based on feature extraction [28]. Compared to the PSNR, the VQM is more likely to identify the nature of quality loss. It measures the perceptual effects of video impairments including blurring, jerky/unnatural motion, global noise, block distortion, and color distortion, and combines them into a single metric.

Moving Pictures Quality Metric (MPQM) [46]

MPQM is an objective quality metric for moving pictures, which integrates two human vision characteristics into the quality assessment: contrast sensitivity and masking. Compared to PSNR, it considers the visual masking phenomenon in assessing video quality.

Structure Similarity Index Metric (SSIM) [47][48]

Reference [47] presents another video quality metric called Structure Similarity Index Metric, which uses the structural distortion measurement instead of mean square error. The reason considering the structural distortion is that the human vision system is highly specialized in extracting structural information from the viewing field, while it is not specialized in extracting errors. Thus, a measurement on structural distortion should give a better correlation to the subjective impression.

Noise Quality Measure (NQM) [49]

The video quality measurement metric addressed in reference [49] is called Noise Quality Measure (NQM), where the image quality is assessed based on a degradation model. Two sources of degradations are considered in this model: linear frequency distortion and additive noise injection, and it leads to two quality measures: a frequency distortion measure (DM), and a noise quality measure (NQM). Compared to SNR, the NQM weights the quality assessment on:

- Variation in contrast sensitivity with distance, image dimensions
- Variation in the local luminance mean
- Contrast interaction between spatial frequencies
- Contrast masking effects

REFERENCE LIST

[1] Y. M. Chen and I. V. Bajic, "Predictive Decoding For Delay Reduction in Video Communications," accepted for presentation at *IEEE Globecom'07*, Washington, DC, Nov. 2007.

[2] *Information Technology - Coding of Audio-Visual Objects, Part 2: Visual,* ISO/IEC 14496-2, 1999.

[3] *Advanced Video Coding for Generic Audiovisual Services*, ITU-T Recommendation H.264, Mar. 2005.

[4] J. Astola, P. Haavisto, and Y. Neuvo, "Vector median filters," *Proc. IEEE*, vol. 78, no. 4, pp. 678-689, Apr. 1990.

[5] N. C. Gallagher, Jr. and G.L. Wise, "A theoretical analysis of the properties of median filter," IEEE *Trans. Acoust., Speech, Signal Processing,* vol. ASSP-29, pp. 1136-1141, Dec. 1981.

[6] M. J. Chen, L. G. Chen, and R. M. Weng, "Error concealment of lost motion vectors with overlapped motion compensation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 3, pp. 560-563, Jun. 1997.

[7] Y. Chen, O. C. Au, C.-W. Ho, and J. Zhou, "Spatio-temporal boundary matching algorithm for temporal error concealment," *Proc. IEEE ISCAS*, pp. 686-689, May 2006.

[8] M. T. Orchard, "Predictive motion-field segmentation for image sequence coding," *IEEE Trans. Image Processing*, vol. 3, no. 1, pp. 54-70, Feb. 1993

[9] H. Schulzrinne, S. Casner, R. Federick, V. Jacobson, "RTP: A Transport Protocol for Real Time Applications," *IETF Requests for comments, RFC 1889,* January 1996.

[10] *ITU-T Recommendation G.113,* "Transmission impairments due to speech processing," February 2001

[11] L. Atzori, Francesco G. B. De Natale, C. Perra, "A Spatio-Temporal Concealment Technique Using Boundary Matching Algorithm and Mesh-Based Warping (BMA-MBW)," *IEEE Trans. Multimedia*, vol. 3, no. 3, Sep. 2001.

[12] T. H. Tsai, Y. X. Lee, and Y. F. Lin, "Video Error Concealment Techniques using Progressive Interpolation and Boundary Matching Algorithm," *Proc. IEEE ISCAS*, pp. 433-436, May 2004.

[13] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC Video coding Standard,” *IEEE Trans. Circuits Syst. Video Technol.,* , vol. 13, no. 7, pp. 560-576, July 2003.

[14] M. E. Al-Mualla, C. Nishan, D. R. Bull, “Multiple-Reference Temporal Error Concealment,” *Proc. IEEE ISCAS*, vol. 5, pp. 149-152, May 2001.

[15] Y. Wang and Q. F. Zhu, “Error control and concealment for video communication: A review,” *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974-997, May 1998.

[16] T. S. Chong, O. C. Au, W. S. Chau, T. W. Chan, “Temporal error concealment for video transmission,” *Proc. IEEE ICME*, vol. 2, pp. 1363-1366, June 2004.

[17] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video Processing and Communications*. Upper Saddle River, NJ: Prentice-Hall, 2002.

[18] ITU-T, H263+ public implementation, TMN-8, 1997.

[19] ITU-T, *Recommendation H.263 v.2,* “Video Coding for Low Bit Rate Communication,” January 1998.

[20] O. Komogortsev and J. Khan, “Predictive Perceptual Compression for Real Time Video Communication,” *Proceeding of the 12th annual ACM International Conference on Multimedia*, Tech. session 5, pp. 220-227, 2004.

[21] C. M. Kuo, C. H. Hsieh, Y. D. Jou, H. C. Lin, and P. C. Lu, “Motion Estimation for Video Compression Using Kalman Filtering,” *IEEE Transaction on Broadcasting.* vol. 42, pp. 110-116, Jun. 1996.

[22] F. Tobagi and I. Dalgic, “Performance evaluation of 10Base-T and 100Base-T Ethernet Carrying Multimedia Traffic”, *IEEE JSAC*, vol. 14, no. 7, pp. 1436-1454, September 1996.

[23] J. Boyce and R. Gaglianello, “Packet Loss Effects on MPEG Video Sent over the Public Internet”, *Proc. ACM Multimedia 1998*, pp. 181 - 190, Bristol, UK, September, 1998

[24] D. Loguinov and H. Radha, “Measurement Study of Low-Bitrate Internet Video Streaming,” *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement.* pp. 281 - 293, 2001

[25] A. P. Markopolou, “Assessing The Quality of Multimedia Communications Over Internet Backbone Networks,” *PhD Dissertation, Stanford University*, October, 2002

[26] XviD MPEG-4 codec: http://www.xvid.org/

[27] A. Katsaggelos and N. Galatsanos, *Signal Recovery Technique for Image and Video Compression and Transmission*, Kluwer Academic Publishers, 1998

[28] K. Kim and L. Davis, "A fine-structure image/video quality measure using local statictis", *Proc. IEEE ICIP 04*, vol. 5, pp. 3535 – 3538, Oct. 2004.

[29] T. Kanungo, D. M. Mount, et al., "An Efficient k-Means Clustering Algorithm: Analysis and Implementation," *IEEE Trans. On Pattern Analysis and Machine Intelligence* vol. 24, no. 7, pp. 881-892, July 2002.

[30] S. Desmet, B. Deknuydt, L. Van Eycken, and A. Oosterlinck, "Classified Motion Estimation for Video Coding," *Proceedings of SPIE*, vol. 2182, Image and Video Processing II, pp. 111 – 119, March 1994.

[31] B. D. Choi, J. W. Han, C. S. Kim, and S. J. Ko, "Motion-Compensated Frame Interpolation Using Bilateral Motion Estimation and Adaptive Overlapped Block Motion Compensation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 4, Apr. 2007.

[32] A. Shamim and J. A. Robinson, "Object-Based Video Coding by Global-to-Local Motion Segmentation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1106-1116, Dec. 2002.

[33] M. Silveira and M. Piedade, "Variable Block Sized Motion Segmentation For Video Coding," *Proc. IEEE ISCAS,* pp. 1294-1296, June 1997.

[34] D. Cremers and S. Soatto, "Variational Space-Time Motion Segmentation," *Proc. IEEE ICCV,* vol 2, pp. 886-892, Oct. 2003.

[35] J. Y. A. Wang and E. H. Adelson, "Representing Moving Images with layers," *IEEE Trans. Image Proc.*, vol. 3, no. 5, p.625-638, Sep. 1994.

[36] G. D. Borshukov, G. Bozdagi, Y. Altunbasak, and A. M. Tekalp, "Motion Segmentation by Multistage Affine Classification," *IEEE Trans. Image Processing*, vol. 6, no. 11, pp. 1591-1594, Nov. 1997.

[37] H. Y. Chung, Y. L. Chin, K. Wong, K. P. Chow, T. Luo, and S. K. Fung, "Efficient Block Based Motion Segmentation Method using Motion Vector Consistency," *Proc. IAPR Conference on Machine Vision Application*, pp. 550 – 553, May 2005.

[38] J. Zan, M. O. Ahmad and M. N. S. Swamy, "Median Filtering-based Pyramidal Motion vector Estimation," *Proc. IEEE ICASSP*, vol. 3, pp. 1605-1608, May 2001.

[39] A. Kaup, "Object-Based Texuture Coding of Moving Video in MPEG-4," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 1, pp. 5-15, Feb. 1997.

[40] S. Desmet, B. Deknuydt, L. Van Eycken and A. Oosterlinck, "A segmentation-based video codec with a block-based fall-back mode," *Proc. SPIE* Vol. 2925, pp. 276-285, Sep. 1996.

[41] ITU-T Recommendation G.723.1, "Dual Rate Speech Coder For Multimedia Communications Transmitting at 5.3 and 6.3 kbit/s," Mar. 1996.

[42] ITU-T Recommendation G.729, "Coding of Speech at 8 kbit/s Using Conjugate-Structure Algebraic-code-excited Linear Prediction (CS-ACELP)," March 1999.

[43] ITU-T Recommendation G.726, "40, 32, 24, 16 kbit/s Adpative Differential Pulse Code Modulation (SB-ADPCM)," 1996

[44] ITU-T Recommendation G.711, "Pulse Code Modulation (PCM) of Voice Frequencies," Nov. 1988.

[45] J. A. Zebarth, "Let Me Be Me [Video Telephony and Teleconferencing Tests]," *Proc. IEEE Globecom'93*, Vol. 1, pp. 389-393, Nov. 1993.

[46] C. J. Branden Lambrecht and O. Verscheure, "Perceptual Quality Measure using a spatio-Temporal Model of the Human Visual System," *Proc. SPIE* Vol. 2668, pp. 450-461, March. 1996.

[47] Z. Wang, L. Lu, and A. C. Bovik, "Video Quality Assessment Using Structural Distortion Measurement", *Signal Processing: Image Communication*, special issue on "Objective video quality metrics," Vol. 19, No. 2, pp. 121-132, Feb. 2004.

[48] Z. Wang and A. C. Bovik, "A Universal Image Quality Index", *IEEE Signal Processing Letters*, Vol. 9, pp. 81-84, Mar. 2002.

[49] N. Damera-Venkata, T. D. Kite, W. S. Geisler, B. L. Evans, and A. C. Bovik, "Image Quality Assessment Based on a Degradation Model," *IEEE Trans. Image Processing*, Vol. 9, No. 4, pp. 636-650, Apr. 2000.

[50] Z. Yang, J. J. Bu, C. Chen, and X. Li, "Fast Predictive Variable-block-Size Motion Estimation For H.264/AVC," *Proc. IEEE ICME'05*, 4 pp.-, Jul. 2005.

[51] J. Stottrup-Andersen, S. Forchhammer, and S. M. Aghito, "Rate-distortion-complexity optimization of fast motion estimation in H.264/MPEG-4 AVC," *Proc. IEEE ICIP'04*, Vol. 1, pp. 111-114, Oct. 2004.

[52] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-Constrained Coder Control and Comparison of Video Coding Standards," *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 13, No. 17, pp. 688-703 July 2003.