NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

TRANSMISSION OF LOW-BIT-RATE MPEG-4 VIDEO SIGNALS OVER WIRELESS CHANNELS

by

Evangelos A.Adam

March 2002

Thesis Advisor:	Murali Tummala
Second Reader:	Robert Ives

Approved for public release; distribution is unlimited

Report Documentation Page

Report Date 29 Mar 2002	**Report Type** N/A	**Dates Covered (from... to)** -

Title and Subtitle Transmission of Low-Bit-Rate MPEG-4 Video Signals Over Wireless Channels	**Contract Number**
	Grant Number
	Program Element Number
Author(s) Adam, Evangelos	**Project Number**
	Task Number
	Work Unit Number
Performing Organization Name(s) and Address(es) Naval Postgraduate School Monterey, California	**Performing Organization Report Number**
Sponsoring/Monitoring Agency Name(s) and Address(es)	**Sponsor/Monitor’s Acronym(s)**
	Sponsor/Monitor’s Report Number(s)
Distribution/Availability Statement Approved for public release, distribution unlimited	
Supplementary Notes The original document contains color images.	
Abstract	
Subject Terms	
Report Classification unclassified	**Classification of this page** unclassified
Classification of Abstract unclassified	**Limitation of Abstract** UU
Number of Pages 122	

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: Title (Mix case letters)
Transmission of Low-Bit-Rate MPEG-4 Video Signals over Wireless Channels

5. FUNDING NUMBERS

6. AUTHOR(S) Evangelos A. Adam

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT ***(maximum 200 words)***

The objective of this thesis is to study the performance of the MPEG-4 video coding standard in the presence of highly erroneous media, such as a wireless channel. MPEG-4 treats video sequences as a collection of objects rather than a collection of frames. A Matlab encoder that conforms to this approach is built for compressing raw video signals at various compression rates. A two-state Markov channel was used to simulate a wireless channel that introduces errors in the video bitstream and a decoder that utilizes error concealment techniques to hide these errors from the user was used to reconstruct the video sequence. The error resilient tools that the MPEG-4 standard provides to enhance the robustness in the presence of errors were simulated and proven to be advantageous compared to methods used in previous standards (MPEG-2, H.263, etc.). At the decoder, the use of error concealment techniques significantly enhanced the quality of the reconstructed video in high bit error rate environments.

14. SUBJECT TERMS
MPEG-4, Video, Communications, Wireless Channels, Error Resilience, Error Concealment.

15. NUMBER OF PAGES
122

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

THIS PAGE INTENTIONALLY LEFT BLANK

TRANSMISSION OF LOW-BIT-RATE MPEG-4 VIDEO SIGNALS OVER WIRELESS CHANNELS

Evangelos A. Adam
Lieutenant, Hellenic Navy
B.S., Hellenic Naval Academy, 1993

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2002

Author:	Evangelos A. Adam
Approved by:	Murali Tummala Thesis Advisor
	Robert Ives Second Reader
	Jeffrey B. Knorr Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The objective of this thesis is to study the performance of the MPEG-4 video coding standard in the presence of highly erroneous media, such as a wireless channel. MPEG-4 treats video sequences as a collection of objects rather than a collection of frames. A Matlab encoder that conforms to this approach is built for compressing raw video signals at various compression rates. A two-state Markov channel was used to simulate a wireless channel that introduces errors in the video bitstream and a decoder that utilizes error concealment techniques to hide these errors from the user was used to reconstruct the video sequence. The error resilient tools that the MPEG-4 standard provides to enhance the robustness in the presence of errors were simulated and proven to be advantageous compared to methods used in previous standards (MPEG-2, H.263, etc.). At the decoder, the use of error concealment techniques significantly enhanced the quality of the reconstructed video in high bit error rate environments.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1
A. BACKGROUND 1
B. THESIS OBJECTIVES 2
C. RELATED WORK 3
D. THESIS ORGANIZATION 3

II. VIDEO COMPRESSION OVERVIEW 5
A. IMAGE FORMATS 5
B. STRUCTURE COMPONENTS 7
C. TRANSFORM CODING 8
D. ENTROPY CODING 11
E. MOTION ESTIMATION AND COMPENSATION 12
F. VIDEO QUALITY MEASURE 15
G. GENERIC VIDEO CODEC 16
H. SUMMARY 17

III. MPEG-4 VIDEO COMPRESSION STANDARD 19
A. ARCHITECTURE 19
1. Features and Functions 20
a. Content-Based Interactivity 20
b. Compression 20
c. Universal Access 20
2. Targets-Scope-Application Areas 21
3. Structure Syntax 21
4. Profiles in MPEG-4 23
B. OBJECT BASED APPROACH 25
1. Video Object Plane Generation 25
2. Bounding Box Generation 27
C. SHAPE CODING 28
1. Chain Coding Algorithm 30
D. TEXTURE CODING 31
1. Shape-Adaptive Discrete Cosine Transform (SA-DCT) 33
2. Quantization and Run Length Encoding 34
E. MOTION ESTIMATION 37
1. Fractional Pixel Accuracy 39
2. Coding of Motion Vectors 40
F. SCALABILITY 41
1. Spatial Scalability 42
2. Temporal Resolution 44
G. ISO/IEC 14496(MPEG4) VIDEO REFERENCE SOFTWARE 46
H. SUMMARY 49

IV. ROBUST TRANSMISSION OF MPEG-4 VIDEO 51

A. WIRELESS CHANNEL MODEL 52
1. Gilbert Channel Model 52
2. Error Correction Codes 53
B. ERROR RESILIENCE TOOLS IN MPEG-4 54
1. Resynchronization Markers 55
2. Reversible Variable Length Codes 58
3. Data Partitioning 62
4. Bit Stream Syntax 62
C. ERROR CONCEALMENT 63
1. Spatial Error Concealment 64
2. Temporal Error Concealment 67
D. SIMULATION RESULTS 69
E. SUMMARY 77

V. CONCLUSIONS 79
A. SIGNIFICANT RESULTS 80
B. FUTURE WORK 80

APPENDIX A. MPEG-4 VIDEO REFERENCE SOFTWARE 83
A. ENCODING A VIDEO SEQUENCE 84
B. DECODING A COMPRESSED FILE 84
C. RATE CONTROL 85
D. ERROR RESILIENCE 85
E. SUPPORTED TOOLS 85
F. RESULTS 86

APPENDIX B. SIMULATION RESULTS FOR SEQUENCES “SUZIE” AND “MISS AMERICA” 91

LIST OF REFERENCES 97

INITIAL DISTRIBUTION LIST 101

LIST OF FIGURES

Figure 2-1. SubSampling: (a) Sampling Pattern for 4:2:2 and (b) Sampling Pattern for 4:2:0, From Ref. [1].. 6
Figure 2-3. (a) Zigzag Scan Method, (b) Frequency Distribution, After Ref. [3]. 11
Figure 2-4. Illustration of Motion Compensation. 13
Figure 2-5. The Previous and Current Frames in the Search Window, From Ref. [1]. 14
Figure 2-6. Block Diagram of a Generic Video Encoder using Motion Estimation and Compensation, From Ref. [1]. 17
Figure 3-1. MPEG-4 Video Bitstream Logical Structure, From Ref. [5]. 22
Figure 3-2. Block Diagram of MPEG-4 Video Codec, After Ref. [5]. 23
Figure 3-3. Edge Detection Results for Three Sequences (Claire, Suzie, Mother and Daughter). The Left Column Shows the Original Frames and the Right Column Shows the Edge Detection Results. 26
Figure 3-4. Generating a VOP for One Frame of "Claire" Sequence. 27
Figure 3-5. Generation of Bounding Box Algorithm , From Ref. [1]. 28
Figure 3-6. Definition of Blocks in Texture Coding, After Ref. [3]. 29
Figure 3-7. Chain Code for Pixels with Eight Neighbors, After Ref. [1] and Ref. [2]. 30
Figure 3-8. Macroblock Based Texture Coding for Arbitrarily Shaped VOP, From Ref. [3]. 32
Figure 3-9. SA-DCT Transform on an 8×8 Foreground Object, After Ref. [2]. 34
Figure 3-10. Comparison between Using Flat and Variable Step Sizes Quantization Tables. 36
Figure 3-11. Reconstructed Frame Using Flat and MPEG-4 Quantization Table. 37
Figure 3-12. Motion Estimation for Arbitrary Shaped VOPs, After Ref. [3]. 38
Figure 3-13. Padding of Boundary Macroblocks, From Ref. [7]. 38
Figure 3-14. (a) Extended and Normal Padding for VOP (star), From Ref. [6]; (b) Priority of Boundary Macroblocks Surrounding an Exterior Macroblock, From Ref. [8]. 39
Figure 3-15. Sub-Pixel Search Positions, around Pixel Coordinate A, From Ref. [1]. 40
Figure 3-16. Motion Vector Prediction, From Ref. [1]. 41
Figure 3-17. Block Diagram of a Two-Layer Spatial/Temporal Encoder, From Ref. [1]. .. 42
Figure 3-18. Effect of UpSampling and DownSampling of One Frame of "Suzie". 44
Figure 3-19. Formation of VOPs in Base and Enhancement Layers, From Ref. [5]. 44
Figure 3-20. Types of Temporal Scalability in MPEG-4, After Ref. [8]. 46
Figure 3-21. Frame 5 from the Video Sequence "News" at Various Bit-Rates. 47
Figure 3-22. Rate-Distortion Curve for "News". 48
Figure 3-23. Compression Ratio versus Bit-Rate for "News". 48
Figure 3-24. PSNR versus Compression for "News". 49
Figure 4-1. Video Transmission Scheme. 51
Figure 4-2. A Two State Markov Model Representing the Basis for Gilbert-Elliot Channel. 53
Figure 4-3. Rate ½ Convolutional Encoder with Constraint Length 4, After Ref. [22]. ... 54

Figure 4-4. Actual and Detectable Error Location in a VLC Bitstream, After Ref. [12]. . 56
Figure 4-5. Slice Structure Mode Adopted in MPEG-4 versus GOB in Baseline H.263, After Ref. [2]. 57
Figure 4-6. Effect of Discarding Data between Two Resynchronization Markers when Errors Occur in the Bit Stream. 58
Figure 4-7. Illustration of Error Propagation Reduction and Difference between Odd and Even-Indexed Bit Errors in Reversible Exp-Golomb Codes of Table 4-1, After Ref. [26]. 61
Figure 4-8. Bit Stream Organization inside an MPEG-4 Video Packet, After Ref. [13]... 63
Figure 4-9. Spatial Averaging Method for Error Concealment. Gray Area is the Reference Area of the Neighboring Macroblocks, After Ref. [17]. 65
Figure 4-10. Spatial Error Concealment Results in a Frame from Sequence "Suzie" where 20% of the MBs are Corrupted. 66
Figure 4-11. Spatial Error Concealment in a Frame from Sequence "Suzie". 67
Figure 4-12. Temporal Concealment in One Frame of Sequence "Suzie". 68
Figure 4-13. Gain in BER for Channel with ε = 0.01 due to FEC. 70
Figure 4-14. Illustration of Hybrid Error Concealment Scheme on Frames 1 and 2 (I-, B- frames) of Sequence "Claire" when Transmitted in a Channel with $\varepsilon = 0.01$ with Resulting Channel BER = 9.3×10^{-4}. 72
Figure 4-15. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.1. No Error Concealment. 74
Figure 4-16. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.01. No Error Concealment. 74
Figure 4-17. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.001. No Error Concealment. 75
Figure 4-18. Different Error Concealment Schemes for a Video Sequence using RVLC through a Gilbert Channel with ε = 0.1 75
Figure 4-19. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.01 76
Figure 4-20. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.001 76
Figure A-1. Rate-Distortion Curve for "Carphone" and "Foreman" Sequences. 87
Figure A-2. Compression versus Bit-Rate for "Carphone" and "Foreman" Sequences..... 87
Figure A-3. Mean PSNR versus Compression Ratio for "Carphone" and "Foreman" Sequences. 88
Figure A-4. "Carphone" Sequence at Various Bit-Rates. 89
Figure A-5. "Foreman" Sequence at Various Bit-Rates. 90
Figure B-1. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.1 for Video Sequence "Miss America". 91
Figure B-2. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.1. No Error Concealment. Video Sequence "Miss America". 91
Figure B-3. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.01 for Video Sequence "Miss America". 92

Figure B-4. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.01. No Error Concealment. Video Sequence “Miss America”. 92
Figure B-5. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.001 for Video Sequence “Miss America”. 93
Figure B-6. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.001. No Error Concealment. Video Sequence “Miss America”. 93
Figure B-7. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.1 for Video Sequence “Suzie”. 94
Figure B-8. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.1. No Error Concealment. Video Sequence “Suzie”. 94
Figure B-9. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.01 for Video Sequence “Suzie”. 95
Figure B-10. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.01. No Error Concealment. Video Sequence “Suzie”. 95
Figure B-11. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.001 for Video Sequence “Suzie”. 96
Figure B-12. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.001. No Error Concealment. Video Sequence “Suzie”. 96

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 2-1. Image Resolution for Image Formats, From Ref. [2]....................................... 7
Table 3-1. Subset of MPEG-4 Video Profile and Level Definitions, From Ref. [9]........ 25
Table 3-2. Huffman Table for Differential Chain Code, From Ref. [1]. 31
Table 3-3. Quantization Tables.. 35
Table 4-1. Non Reversible and Reversible Exp-Golomb Codes, After Ref. [13]. 59
Table 4-2. BER With and Without FEC Achieved for the Same P_b and P_g for Three Channel Conditions. 73
Table 4-3. Probabilities of Errors for the Bad (P_b) and the Good Channel (P_g) Used in Order to Achieve a PSNR of 25 dB Under Three Channel Conditions. 73
Table A-1. Supported Tools in MPEG-4 Video Reference Software. 86

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This work is primarily dedicated to my wife Elizabeth for her courage and unlimited support that she provided through those two difficult years and to my son Christos-Aristidis for giving meaning to my life.

I also dedicate this work to my parents and especially my mother because she taught me that a man is only limited by his own will to achieve his goal.

Special thanks to my professor Dr Murali Tummala for his effort to bring this work into completion. His guidance and support were unique and the main reason for the completion of this thesis.

I would also like to earnestly thank my second reader Dr Robert Ives for the help he provided in fine-tuning this work. His understanding for the deadlines I had to meet was extraordinary.

Lastly, I would also like to thank my fellow student LT K. Kamaras H.N for the donation of the code of the Gilbert-Elliot channel.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The emergence of multimedia applications in the last decade and the network-centric philosophy in the organization of future military operations make reliable transmission of video signals over wireless channels an important topic. However, video signals inherently demand considerable bandwidth. Compression techniques are utilized to reduce the required bandwidth by sacrificing the quality of the reconstructed video. In this case, the properties of the human vision system (HVS) are exploited to allow the exclusion of information with a resultant savings in the required transmission bandwidth. The bandwidth limitation of wireless channels along with their error prone nature makes the transmission of video over such media a challenging task.

This thesis seeks to study the algorithms described in the visual part of the MPEG-4 video compression standard for efficient coding of video sequences and the error resilience tools that enhance robustness in the presence of errors. An additional objective of this work is to apply error concealment techniques in an effort to hide from the user the visual effects of the transmission errors introduced in the channel.

In order to meet the goals of this work, a Matlab encoder was built to compress video signals at various compression rates. The compressed bitstream enters a forward error correction encoder followed by a two-state Markov channel. From the channel, the corrupted bitstream enters the forward error correction decoder to correct some of the channel errors and finally the video decoder, where the video is reconstructed. In the video decoder, error concealment techniques are applied in order to hide from the user the visual effects of remaining errors.

The Matlab encoder uses a motion compensation scheme and the discrete cosine transform to remove redundancies in the temporal and spatial domains. In order to exploit the content-based philosophy of the standard, shape coding is utilized. This method enables the encoder to consider the video sequence as a collection of video objects rather than frames. Nevertheless, the encoder is far from being a fully compatible MPEG-4 encoder since it does not utilize entropy coding in the form of variable length codes to achieve the maximum compression, which decreased the compression efficiency. The

encoder achieved a compression ratio of 9:1 as opposed to a ratio of 190:1 achieved by the regular MPEG-4 encoder.

The simulation results demonstrated the advantage of using an error isolation technique, such as reversible variable length codes, when compared to the approaches used in previous standards (for example, MPEG-2 and H.261). The advantage of fixed packet size was also demonstrated. The simulations indicated that the utilization of this technique is more advantageous than the group of blocks approach used in MPEG-2 since fewer macroblocks are contained in the packet in high activity areas thereby making these areas less vulnerable to errors.

The use of error concealment at the decoder improved the quality of the reconstructed sequence. Even under high error conditions (for example, BER = 7×10^{-2}), the decoder managed to keep the quality measured as peak signal-to-noise ratio above 20 dB. The hybrid method using spatial averaging in the first I-frame and simple temporal concealment in all other frames provided an additional boost in performance in all cases when compared to the pure spatial concealment method.

During the course of this work several topics for possible future effort were considered. Suggestions for future work include the implementation of a robust image segmentation technique to make the encoder work with more complex sequences than a “talking head” with uniform background, the enhancement of the encoder with the ability to code color sequences and the use of unequal error protection for the object of interest (i.e., foreground) and all other objects in order to enhance performance without sacrificing much of the coding efficiency.

In summary, MPEG-4 is a powerful tool for a variety of multimedia applications, including those applicable to wireless networks. The standard provides excellent compression of video signals and addresses the issues of errors and losses encountered in wireless channels using several error resilience and concealment tools. That MPEG-4 has been adopted in the 3G cellular networks is a testimonial to its wide acceptance in the industry.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

The emergence of multimedia applications in the last decade and the network-centric philosophy in the organization of future military operations make reliable transmission of video signals over wireless channels an important topic. As the available time for decision-making is reduced in order to limit the window of dynamic responses to meet hostile threats, the need for reliable multi-node video conferencing tools that would provide accurate information flow between the field and the decision points is growing.

Inherently, video signals demand considerable bandwidth. On the other hand, bandwidth limitations for wireless channels along with their error prone nature makes the transmission of video over such media a challenging task. Typically, compression techniques are utilized to reduce the required bandwidth by sacrificing the quality of the reconstructed video. In this case, the properties of the human vision system (HVS) are exploited to allow the exclusion of information with a resultant savings in the required transmission bandwidth while maintaining reasonable quality.

Current standard compression techniques provide bit-rates from as low as a few kbps up to over 1 Mbps using a combination of intra-frame and inter-frame coding (MPEG 2, H.263, H.261, etc.), or intra-frame only (Motion-JPEG). The additional achievable compression with inter-frame coding, which is a form of temporal coding of consecutive frames, makes it a better choice for the transmission of video over low bandwidth media.

In order to achieve higher compression ratios, all existing compression standards use entropy-coding techniques, such as variable length codes. This, however, makes the bit stream highly sensitive to errors. Depending on the location of the errors, the decoder can lose synchronization with the encoder and the file can be declared undecodable, or in the best case scenario, can be decoded with significant degradation in quality. Thus, error resilience tools that attempt to limit the sensitivity of the bit stream in the presence of errors are essential for retaining quality in the reconstructed video.

Low delay is an important factor in real time video conferencing applications. This additional constraint significantly affects the quality of the reconstructed sequence as more robust error correction techniques, such as automatic-repeat request (ARQ) have limited use.

Even though the use of error correction mechanisms and error resilience tools limit the appearance of errors, these errors have proven to be inevitable in such hostile environments as wireless channels. A robust video codec must utilize effective error concealment techniques in order to hide the presence of errors from the end user. The remaining redundancies in the spatial or temporal domain are exploited in this case to produce the best possible result and the effect of errors on the quality of the reconstructed video is minimized.

B. THESIS OBJECTIVES

The objective of this thesis is to investigate the performance of the transmission of ISO/IEC 144496 (MPEG-4) coded signals (low-bit-rate video) over wireless channels. Specifically, this thesis seeks to study the algorithms described in the visual part of this standard for efficient coding of video sequences. In order to gain insight into these algorithms and tools described in the standard, a Matlab codec is built. The codec supports object-based encoding techniques with three types of frame (I, P and B) coding.

Since MPEG-4 targets transmission of low bit-rate video over wireless channels, it provides error resilient tools that are used to provide robustness in the presence of errors. The use of these tools will be simulated and the resultant improvement in the reconstructed video quality will be demonstrated.

Finally, error concealment techniques that the decoder can effectively utilize to hide the effect of errors in the reconstructed frame will be investigated. Specifically, two error concealment techniques, operating in the spatial and temporal domain, will be simulated. Additionally, the performance of a technique combining the spatial and temporal methods of error concealment will be studied.

C. RELATED WORK

In [20], Stuhmuller et al. developed an analytical model covering the complete transmission of video over wireless channel including rate-distortion performance, forward error correction and inter-frame error propagation using a two state Markov channel. In [28], Lu et al. modeled a correlated mobile fading channel using a finite-state Markov model. The authors simulated the transmission of video over this channel and compared simulation results concerning the performance of block codes as error correction mechanisms with the ones derived from the analytical model. In this thesis, a two state Markov model similar to the one used in [20] and [28] served as a model for the wireless channel over which compressed MPEG-4 video signals are transmitted.

The advantages of the error resilience tools have been demonstrated by the MPEG-4 committee through a core experiment as detailed in [21]. This thesis utilized these tools to enhance the quality of the reconstructed video. In [16], Wang et al. presented a review of error control strategies and error concealment techniques that can be used for reliable video transmission over error-prone media. In this thesis, a combination of temporal and spatial error concealment techniques is used, and the effectiveness of this method in enhancing the quality of reconstructed video is demonstrated.

D. THESIS ORGANIZATION

This thesis is organized into five chapters and two supporting appendices. Chapter II provides an overview of the tools used to limit the redundancies of a video signal in spatial and temporal domains. Chapter III describes the tools that are utilized in the newest video compression standard and illustrates the use of the content-based philosophy of ISO/IEC 14496 (MPEG-4) . Chapter IV deals with the robust transmission of MPEG-4 video signals over erroneous media and compares the performance of the tools proposed by the standard. Simulation results of the transmission of an MPEG-4 compressed video signal over a wireless channel, modeled by a two-stage Markov channel, are presented in this chapter. Additionally, this chapter demonstrates the advantages of using error concealment techniques in the decoder to produce visually desirable results. Chapter V summarizes the work done in this thesis and concludes with recommendations for further research. Finally, Appendix A includes a brief manual for using the MPEG-4 video reference software, publicly available from ISO, and shows results obtained from the use of this software by compressing two video sequences while Appendix B shows results obtained from encoding two video sequences with a Matlab encoder and their transmission over the channel.

II. VIDEO COMPRESSION OVERVIEW

Video communication plays an important role in the world of telecommunication and multimedia systems. Since 1960 when the first videophone appeared in a primitive format [1], much research has been conducted and many standards have appeared with the goal to provide tools that would allow the transmission and storage of moving pictures in an efficient manner. In this chapter, the basic tools of video compression are presented.

A. IMAGE FORMATS

Images are two-dimensional signals. Each image is characterized by the number of lines and pixels per line. This is called the *resolution* of the picture and the larger the number of pixels in the picture, the better the quality. Each pixel is represented by three values: a luminance value and two chrominance values. The human eye is more sensitive to luminance values than chrominance values. Therefore, the first step in compression is *subsampling* of the chrominance values. Thus, if a pixel is represented by a fraction rather than all of the components, it is possible to achieve a first stage compression by omitting some information without degrading the quality.

In most cases, the video signal is digitized directly from the camera using the CCIR-601 format. The resolution of the CCIR-601 differs between North America and Europe. In North America and the Far East, the resolution is 525 lines per frame with 30 frames per second while in Europe there are 625 lines per frame with 25 frames per second [1], [2]. The active lines, not counting the blanking lines used for synchronization are 576 and 480, respectively. The number of pixels per line is 720 for both cases.

Lower resolutions are used for efficient storage and coding. Source input format (SIF) is defined as having half the resolution of the CCIR-601 both in the spatial and temporal sense. Therefore, the resolution of SIF in North America and the Far East is 240 lines per frame and 360 pixels per line with 25 frames per second while in Europe, the SIF resolution is 288 lines/frame and 360 pixels/line with 30 frames per second.

In CCIR-601, the chrominance bandwidth is half that of the luminance. As the human visual system is less sensitive to chrominance values, CCIR-601 omits half the chrominance values thereby resulting in a 2/3 reduction in data rate with no visual artifacts. This subsampling pattern known as 4:2:2 is shown in Figure 2-1(a). For every four luminance values, there are only two pairs of chrominance instead of four pairs. In SIF, the subsampling pattern 4:2:0 is used to reduce the data rate by ½ as shown in Figure 2-1(b). For every four luminance values, one pair of chrominance values is used. Thus, in SIF, the horizontal and vertical resolutions of luminance will be half of the source resolution, but for the chrominance, while horizontal resolution is halved, the vertical resolution is reduced to ¼ of its original value [1].

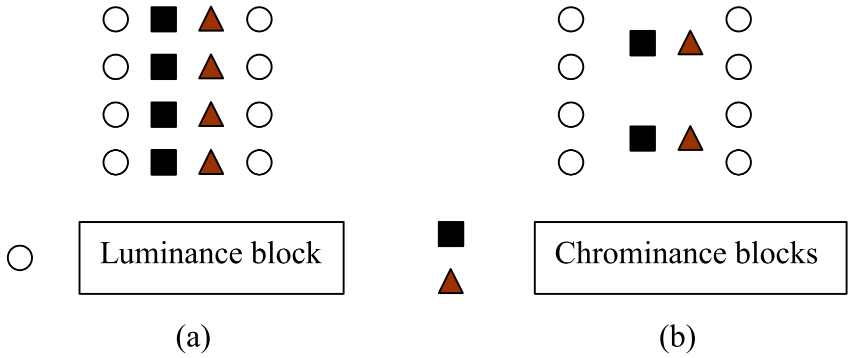


Figure 2-1. SubSampling: (a) Sampling Pattern for 4:2:2 and (b) Sampling Pattern for 4:2:0, From Ref. [1].

For videoconferencing applications, a common format called common intermediate format (CIF) was adopted. CIF retains the high spatial resolution of the North American standard of 288 lines per frame and the high temporal resolution of the European standard of 30 frames per second. The image sampling pattern used is 4:2:0, similar to SIF. Since 360 is not divisible exactly by 16, which is the number of pixels in one line of a macroblock as defined in Section B, the number of pixels per line is 352.

In applications, such as video conferencing and videophones, which are the main focus of this thesis, images with lower resolutions than the original are preferred. Furthermore, for certain applications such as video over mobile communications channels, it is possible to reduce the frame rate from 30 frames per second to 12.5 or 8.3

frames per second [1]. Reduced resolutions, such as Quarter-SIF (QSIF) and Quarter-CIF (QCIF), can be used for SIF and CIF pictures, respectively. In these formats, the spatial resolutions are halved in each direction. These formats are used in very low-bit-rate video applications and this thesis uses the QCIF format for video sequences. In applications, such as mobile video telephony, reduced resolutions, such as Sub-QSIF and Sub-QCIF, are also used. Table 2-1 summarizes the image resolutions of some of the image formats.

	Sub-QCIF	QCIF	CIF	4CIF	16CIF
Width (pixels)	128	176	352	704	1408
Height (pixels)	96	144	288	576	1152

Table 2-1. Image Resolution for Image Formats, From Ref. [2].

B. STRUCTURE COMPONENTS

The building block of an image is called a pixel, which is an abbreviation for *picture element*. In order to cope with images more efficiently, other structure elements are defined by grouping pixels together. Thus, a *block* is an 8 × 8 group of pixels while a group of 4 blocks or a group of 16 × 16 pixels, is called a macroblock (MB). A *slice* is a row of macroblocks. A slice can start from any macroblock in the picture and end at any macroblock as long as the first slice starts at the first macroblock and the last ends at the last macroblock. Slices serve as resynchronization points at the decoder. When an error occurs in one slice and the synchronization at the decoder is lost, the decoder can look for the beginning of the next slice and resynchronize. These methods and the manner in which slices can be used to eliminate errors are discussed in Chapter IV of this thesis.

Higher in the hierarchy of the structure elements of a video sequence are the *frames,* which are the actual pictures in the video sequence. Frames are divided into Intra-frames (I-frame), Inter-frames (P-frame) and Bidirectional frames (B-frames). An I-frame is a frame where each block is coded without any reference to its previous or following frame. A P-frame has macroblocks coded using information from the previous frame and a B-frame uses information from both the previous and the following frame in order to code its macroblocks. The manner in which macroblocks are coded in each frame will be described in the following sections of this chapter.

Finally, a group of pictures (GOP) is a number of frames together. Each GOP starts with an I-frame and ends with either a P-frame or a B-frame if the codec supports bidirectional frames. This hierarchy is very important in order to identify different points inside a video sequence and use them as a reference. For example, the GOP number characterizes a scene in a DVD movie using the MPEG-2 format. Furthermore, if the user wishes to fast forward to a part of the video file while watching it, the decoder decodes only the I-frames and skips the others. The coding of one frame using another as well as the coding efficiency achieved following this structure will be discussed in more detail later in this chapter. Figure 2-2 illustrates the hierarchy of the structure elements of a video sequence.

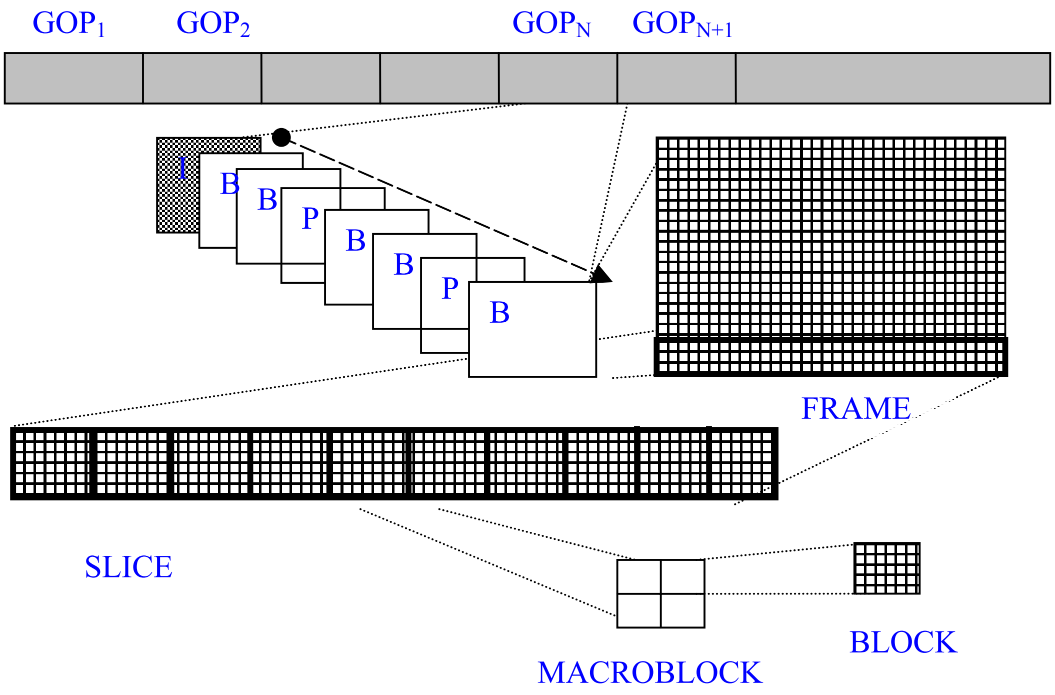


Figure 2-2. Hierarchy of the Structure Elements of a Video Sequence.

C. TRANSFORM CODING

A common characteristic of most images is that the neighboring pixels are correlated and therefore contain redundant information. The foremost task then is to find a less correlated representation of the image. Two fundamental components of compression are redundancy and irrelevancy reduction. The goal of *redundancy reduction* is to remove duplication from the signal source (image/video). *Irrelevancy reduction* omits parts of the signal that will not be noticed by the end user. In the preceding section, the irrelevancy reduction methods, or subsampling methods, were briefly discussed. Redundancy reduction methods based on the discrete cosine transform (DCT) will be presented in this section.

The goal of transform coding is to reduce the redundancy of information at the pixel level by transforming the values into another domain prior to data reduction. The strength of transform coding is that in most images, the energy is concentrated in the low frequency region, and thus, by retaining only a small portion of the coefficients, the decoder is able to reconstruct the image accurately or at least in such a manner that the human visual system is unable to distinguish the missing information.

The discrete Karhunen-Loeve transform (DKLT) decorrelates data using a kernel based on the eigenvalue decomposition of the covariance matrix [2]. The DKLT is optimal in terms of coding gain (energy compaction) and decorrelation of the image. However, it is not efficient for real life applications for two main reasons. First, the signal statistics , i.e., the covariance matrix, have to be known a priori. Second, the eigenvalue decomposition is computationally intensive, thus making the transform inefficient for real-time applications. On the other hand, the Discrete Cosine Transform (DCT) is signal independent and computationally efficient, hence widely used in real-time implementations.

The 1-D DCT for a vector with N elements is given by:

$$F(v) = C(v)\sum_{n=0}^{N-1} f(n)\cos\left(\frac{\pi(2n-1)v}{2N}\right), v = 0,...,N-1 \qquad (2.1)$$

where $f(n)$ is the value of the n^{th} element, N is the number of elements that are transformed, $F(v)$ the transformation coefficient and

$$C(v)=\begin{cases}\dfrac{1}{\sqrt{N}}, & \text{if } v=0\\ \dfrac{2}{\sqrt{N}}, & \text{if } 1\le v\le N-1\end{cases}$$

The inverse DCT is defined as follows:

$$f(n)=C(n)\sum_{v=0}^{N-1}F(v)\cos\left(\frac{\pi(2n-1)v}{2N}\right),\ n=0,\ldots,N-1 \qquad (2.2)$$

where

$$C(n)=\begin{cases}\dfrac{1}{\sqrt{N}}, & \text{if } n=0\\ \dfrac{2}{\sqrt{N}}, & \text{if } 1\le n\le N-1\end{cases}$$

A 2-D DCT transform is implemented by performing two consecutive 1-D transforms in the vertical and horizontal directions, respectively. The 2-D transform for an *M×N* image is defined as:

$$F(u,v)=C(u)C(v)\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m,n)\cos\left(\frac{\pi(2m-1)u}{2M}\right)\cos\left(\frac{\pi(2n-1)v}{2N}\right),\ \begin{matrix}0\le u\le M-1\\ 0\le v\le N-1\end{matrix} \qquad (2.3)$$

where C(v) is defined above and

$$C(u)=\begin{cases}\dfrac{1}{\sqrt{M}}, & \text{if } u=0\\ \dfrac{2}{\sqrt{M}}, & \text{if } 1\le u\le M-1\end{cases}$$

A block of *M×N* pixels has *MN* coefficients with the *F*(0,0) coefficient defined as the *DC coefficient* and all others as the *AC coefficients.* Most of the energy is typically concentrated in the low frequency region [1]. If a portion of the coefficients is omitted, the decoder is capable of reconstructing the image without a significant loss in quality. The coefficients are reordered into a zig-zag fashion in a one-dimensional array as illustrated in Figure 2-3 (a) and are quantized using either a fixed or variable quantization step size. Figure 2-3 (b) illustrates the distribution of energy as a function of frequencies.

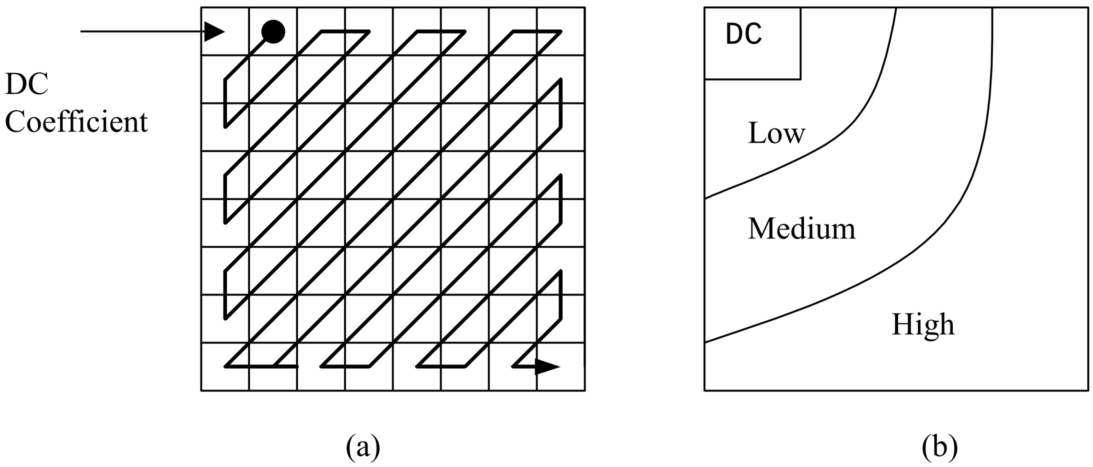


Figure 2-3. (a) Zigzag Scan Method, (b) Frequency Distribution, After Ref. [3].

D. ENTROPY CODING

The rearrangement of coefficients (see Figure 2-3) places the DC coefficient at the first location of the array and the AC coefficients are arranged from low to high frequency in both the horizontal and the vertical directions. The assumption is that the quantized DCT coefficients at higher frequencies would likely be zero, thereby separating the non-zero and zero parts.

The rearranged array is coded into a sequence of run-level pairs. A *run* is defined as the distance between two non-zero coefficients in the array. A *level* is the non-zero value immediately following a sequence of zeros. The run-length encoded values are further coded using Huffman Coding [2].

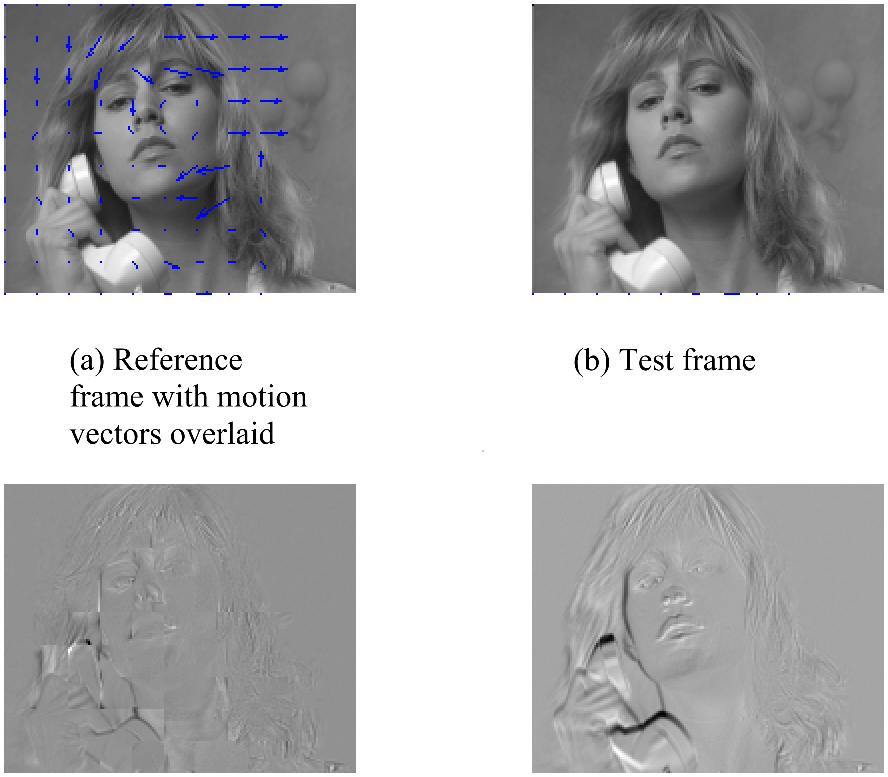
Huffman is a variable length code algorithm in a sense that the codewords do not have a fixed length. This can cause synchronization problems at the receiver. If an error occurs, the receiver cannot synchronize and decode the rest of the code. This can lead to unacceptable quality out of the receiver and is something that the designer of a codec must recognize. In fact, most recent video codecs using MPEG-4 and H-263 have taken this problem into account and introduced points in the bitstream that the decoder looks for if a loss of synchronization occurs. These tools are described in Chapter IV.

Even though Huffman is a lossless algorithm, the quantization of DCT coefficients leads to an overall lossy compression scheme. The coarser the quantization step size for the coefficients, the more distortion the codec is going to introduce in the original information. This leads to a trade-off in video compression codecs: the more the compression required, the more distorted the result will be and vice versa.

E. MOTION ESTIMATION AND COMPENSATION

In the preceding paragraphs, techniques for decorrelating data in the spatial sense were described. Discrete cosine transform and quantization are used to remove the spatial redundancies within a frame of a video sequence. This type of coding is called intraframe coding. When dealing with video sequences, however, another type of redundancy exists that a codec must be able to exploit.

Temporal redundancy refers to the redundancies that exist from one frame to the next. In most natural video sequences, especially those with little motion activity such as a speaker in front of a uniform background, consecutive frames possess a great deal of similarities and further reduction in the required bits can be achieved using inter-frame coding. A codec can compute the differences between successive frames and code only those differences. In general, these differences have small values and tend to be zero when sequences with low motion or luminance variations are being coded. Further compression can be achieved when instead of using the differences, the motion compensated differences are coded. This scheme requires motion estimation techniques to be adopted by the codec. Figure 2-4 illustrates an example of the use of motion compensation. In the figure, (a) is the anchor of the frame in question and (b) uses it to predict its motion vectors. The motion compensated error frame (c) and the frame produced from the subtraction of the two frames (d), i.e., without motion compensation, illustrate that the motion compensated error frame is much smoother and contains less high frequency coefficients after entropy coding, and thus, requires fewer bits to code it.



(a) Reference frame with motion vectors overlaid

(b) Test frame

(c) Motion compensated error frame.

(d) Difference frame without motion compensation.

Figure 2-4. Illustration of Motion Compensation.

Motion estimation is performed at the block or at the macroblock level. Most modern standards use the macroblock level approach since in most natural scenes there is not much difference between adjacent pixel values. The most commonly used motion estimation technique in all standard video coders is the block matching algorithm (BMA). In BMA, each frame is divided into macroblocks and the motion of each macroblock in reference to the position of the macroblock in the previous frame is estimated. Then for maximum motion displacement of w pixels, the macroblock in the current frame is matched with the corresponding macroblock in the previous frame by applying a minimization criterion. The shifted position that yields the minimum metric is defined as the motion vector for this macroblock. Figure 2-5 shows the block matching algorithm technique.

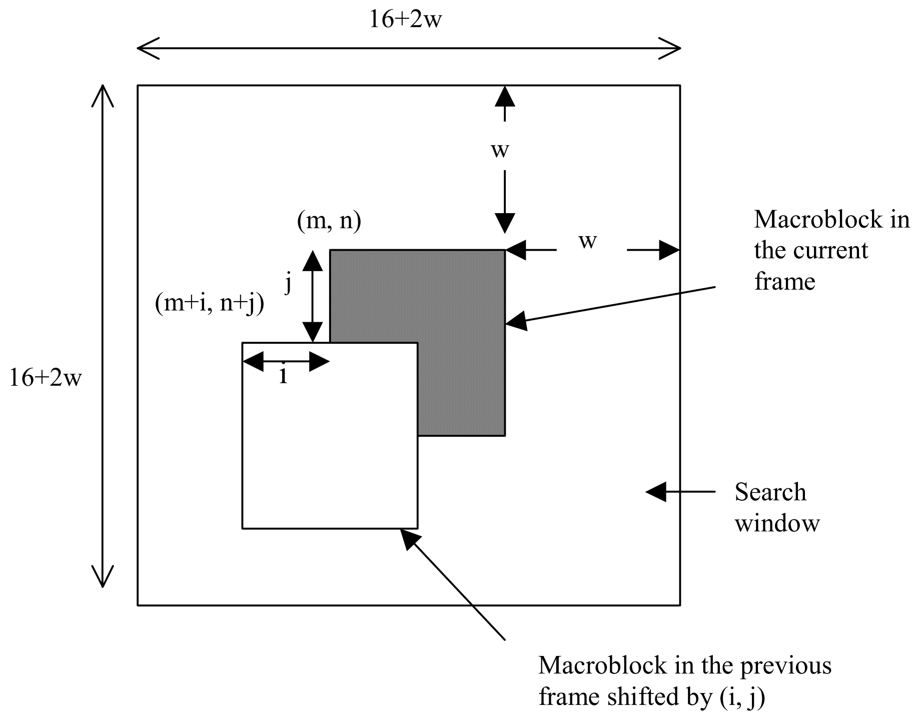

16+2w
16+2w
w
w
(m, n)
j
(m+i, n+j)
i
Macroblock in the current frame
Search window
Macroblock in the previous frame shifted by (i, j)

Figure 2-5. The Previous and Current Frames in the Search Window, From Ref. [1].

The minimization criterion used is either the mean squared error, defined as

$$D_{MSE}(i,j) = \frac{1}{16^2}\sum_{m=1}^{16}\sum_{n=1}^{16}(f_2(m,n) - f_1(m+i,n+j))^2,\ -w \le i,j \le w \qquad (2.4)$$

or the mean absolute difference,

$$D_{MAD}(i,j) = \frac{1}{16^2}\sum_{m=1}^{16}\sum_{n=1}^{16}\left|f_2(m.n) - f_1(m+i,n+j)\right|,\ -w \le i,j \le w \qquad (2.5)$$

where $f_2(m,n)$ is the luminance pixel value of the current frame in position (m,n), and $f_1(m+i,n+j)$ is the luminance pixel value of the same pixel shifted by (i,j) in the previous frame. Note that, based on the assumption that movement in a picture is only due to object motion, when there is no motion, the above metrics tend to be zero, i.e., the motion vectors are zero.

Nevertheless, this exhaustive search for the best match requires $(2w+1)^2$ evaluations of the matching criterion. For this reason, MAD is preferred and used in almost every standard codec today. Search methods aimed at reducing the number of calculations have been proposed in order to make systems suitable for real-time applications [1]. All faster methods rely on the fact that the number of search points can be reduced by checking specific points first and then move in the direction of the point that produces the minimum difference assuming that the distortion measures decrease monotonically as the search moves towards the best matched point. These algorithms can reduce the required number of computations needed to $5+\log_2 w$, where w is the displacement [1].

F. VIDEO QUALITY MEASURE

Since quantization introduces distortion into the reconstructed video sequence, it is necessary to define an objective criterion that can measure the difference between the original and the reconstructed sequence. Most video coders today are designed to minimize the mean square error between two sequences.

If f_o and f_r are the original and the reconstructed video sequence, respectively, containing K frames and MN pixels in each frame, the mean square error or distortion is defined as:

$$D_{MSE} = \frac{1}{KMN}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(f_o(m,n,k) - f_r(m,n,k)\right)^2 \qquad (2.6)$$

Usually, however, the peak-signal-to-noise-ratio (PSNR) in dB is used instead of the mean squared error as given by

$$PSNR = 10\cdot\log_{10}\left(\frac{f_{\max}^2}{D_{MSE}}\right) \qquad (2.7)$$

where $f_{\max}$ is the maximum intensity value of the video signal, which is 255 for an 8-bit video. In this thesis, the PSNR measure is used for comparison between the original and the reconstructed video signals. For the calculation of PSNR over an entire video sequence, an average value of distortion over the corresponding frames is used in Equation (2.7) to compute the mean PSNR.

The PSNR does not correlate very well with perceptual distortion between images. Thus, for a fixed $f_{\max} = 255$, the higher the resolution and the number of pixels, the smaller the distortion due to quantization (which can produce a large D_{MSE}), thus a small PSNR. Additionally, this method does not take into account the relative position of the distortion within a frame or sequence. Conceptually, distortion in an area of the image that contains high frequencies, for example, text and edges, is more annoying to the human eye than the same distortion in a uniform background. The simplicity and the lack of better alternatives made the PSNR a widely accepted quality measure for images.

G. GENERIC VIDEO CODEC

The basic tools for video compression described in previous sections are part of a generic video codec shown in Figure 2-6. As input video enters the coder, a decision is made on the type of coding to be performed, either I-, P-, or B-frames. If the frame is going to be coded as an I-frame, DCT and quantization (Q) are performed at the macroblock level. Each macroblock contains 4 blocks on which DCT is performed. Inverse quantization and inverse DCT are performed in order for the coder to use the restored frame as a reference for the inter-coding of other frames. The coded I-frame coefficients are entropy coded to produce variable length codewords as described in section D, and the bitstream is transmitted over the medium. In the case of inter-frame coding, motion estimation and compensation are performed using as a reference frame one of the previously stored I- or P-frames. The difference between the original frame and the motion compensated frame is coded in a manner similar to intra-frame coding and the resulting codewords are transmitted. Motion estimation (ME) is performed at the macroblock level.

The output bit-rate of the encoder can be made constant by using a buffer (traffic shaper). The bitstream before entering the channel fills the buffer, which sends out bits at the required channel rate. A feedback signal to the quantizer concerning the occupancy of the buffer and the bit-rate requirements can be included. When the buffer tends to overflow, the quantizer increases the step size to increase the compression by sacrificing quality and vice versa [1].

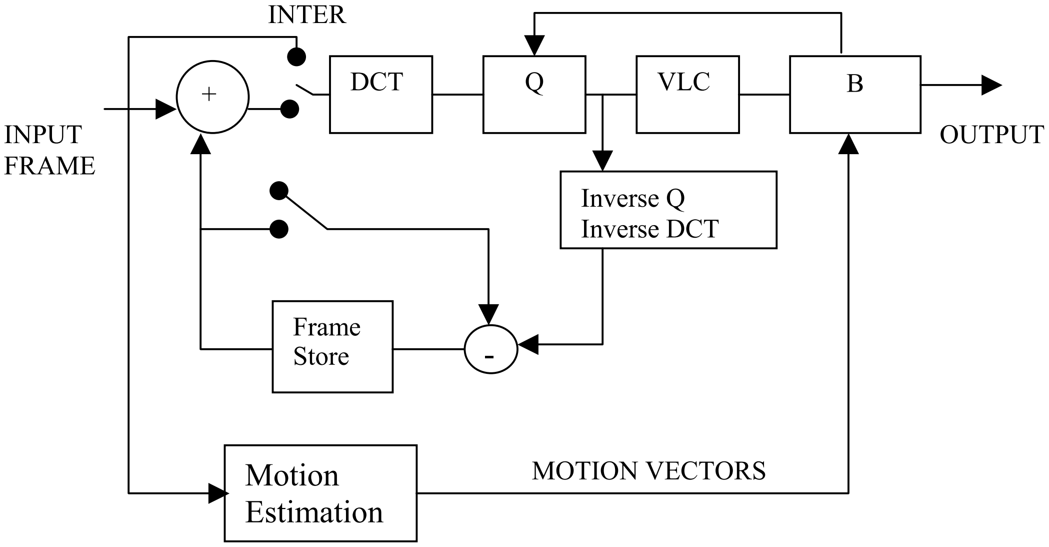


Figure 2-6. Block Diagram of a Generic Video Encoder using Motion Estimation and Compensation, From Ref. [1].

H. SUMMARY

In this chapter, basic functional blocks of video compression were discussed. Intra- and inter-frame coding techniques, utilized in order to remove the spatial and temporal redundancy between successive frames, were introduced. The hierarchical structure of a video sequence was described and a generic video coder was presented.

In the following chapter, the extensions of these techniques, with the necessary modifications to support the new content-based approach adopted by the MPEG-4 video compression standard, will be discussed.

THIS PAGE INTENTIONALLY LEFT BLANK

III. MPEG-4 VIDEO COMPRESSION STANDARD

In Chapter II, general video compression techniques were presented. These techniques are widely used in standard video codecs today. MPEG-4, however, adopts a new approach to video coding. The content or object-based approach treats video sequences more like a collection of objects. New algorithms had to be developed for this reason. This chapter describes the structure of the video part of the standard and the associated new algorithms. Furthermore, a Matlab code was written aimed at building an entire codec (encoder/decoder) that would use these tools.

Section A introduces the MPEG-4 standard by describing the new algorithms and the structure of the video sequence. Section B presents the adopted object-based approach and the need for new algorithms. Section C describes algorithms used in shape coding, especially the chain coding method. Section D deals with texture coding techniques while Section E describes motion estimation techniques in more detail. Finally, Section F discusses scalability and specifically presents the two scalability modes, spatial and temporal, while Section G presents results of encoded sequences using the MPEG-4 video reference software publicly available from ISO.

A. ARCHITECTURE

The Moving Picture Experts Group (MPEG) developed audiovisual information standards MPEG-1 and MPEG-2 for the efficient storage of digital video in CD-ROMs and the coding of DVD and HDTV, respectively. The growing need for multimedia applications, which can be seen as a platform for the exchange of information coming from different sources, such as synthetic and natural, necessitated a new universal standard that would provide tools to compress video, audio and images in a flexible and interactive manner while covering areas that were neglected in the previous standards. Thus, this standard would support a very broad spectrum of applications [3].

In 1993, the MPEG group started working on MPEG-4 and after six years, the standard was adopted by the ISO as “Generic Coding of Audio-Visual Objects: Part 2 - Visual,” ISO/IEC JTC1/SC29/WG11 N1902, FDIS of ISO/IEC 14496-2. The new standard adopts a video coding technique called the *object based approach* and suggests tools to implement such an approach while at the same time giving the designer the freedom to apply different kinds of tools to solve a variety of problems.

1. Features and Functions

In a broad sense, multimedia is assumed to be a general framework of interaction with information originating from different sources, including video. A multimedia standard is expected to provide support for a large number of applications [5]. The MPEG-4 visual standard consists of a set of tools that enable applications by supporting several classes of functions, such as security, low delay and synchronization. Apart from the above, the MPEG-4 committee was seeking solutions to support eight key functions that were not supported by existing standards. These new functions have been divided into three major classes based on the requirements they support.

a. Content-Based Interactivity

Four functions are included in this class providing interactivity between the user and the data: *content-based multimedia data access tools*, *content based manipulation and bitstream editing*, *hybrid natural and synthetic data coding*, and *improved temporal random access*. Data retrieval from on-line libraries, interactive home shopping, and movie production and editing are applications that could use these functions.

b. Compression

This class contains two functions: *improved coding efficiency* and *coding of multiple concurrent data streams*. They aim at applications requiring an efficient storage or transmission of audiovisual information and their efficient synchronization. Information browsing over the Internet and virtual reality applications can make use of the above functions.

c. Universal Access

Robustness in error-prone environments and *content-based scalability* fall into the last class. These functions allow MPEG-4 encoded data to be accessible over a wide range of media, and with various qualities in terms of temporal and spatial resolutions for specific objects, which could be decoded by a range of decoders with different complexities. Applications benefiting from these functions are wireless communications, database browsing and access at different content levels, scales, resolutions, and qualities [3].

2. Targets-Scope-Application Areas

In order to support the above mentioned features, the representation and coding methods of arbitrary shaped objects need to be investigated. The object based approach of MPEG-4 is the most important aspect in video coding. Algorithms for segmenting frames into objects of interest and the method to represent and code objects with arbitrary shape are introduced. This approach improves interactivity since the position of an object in a frame can be altered.

Compression efficiency and robustness to errors are important factors in the standard since video transmission over wireless channels is expected to be the next major application area. Since wireless video communication operates in error-prone environments, tools for providing robustness and fast resynchronization are required.

The MPEG-4 visual standard has been explicitly optimized for three bit-rate ranges: below 64 kbps, 64-384 kbps and 384 kbps-4 Mbps. For high quality applications, higher bit-rates are also supported while using the same set of tools and the same bit stream syntax for those available in the lower bit-rates [5].

3. Structure Syntax

In order to provide the maximum user interactivity as needed in multimedia applications, a special structure had to be adopted. An MPEG-4 video sequence consists of one or many video sessions (VS) of the same or different subjects. Each video session consists of one or many video objects (VO), which are the 2-D representation of an object inside a scene and the shape can be rectangular or arbitrary.

A video object can be encoded in scalable (multi-layer) or non-scalable (single layer) form, depending on the application, represented by the video object layer (VOL). The VOL provides support for scalable coding. Spatial or temporal scalability can be used going from coarse to fine resolution. Depending on parameters, such as available bandwidth, computational power and user preferences, the desired resolution can be made available to the decoder [5]. Video object planes (VOP) are the elementary structure components of an MPEG-4 video sequence and they are the instantaneous representations of a video object. The generation of VOP and VOL are discussed later in this chapter in Sections B and F, respectively.

Finally, in order to provide random access to the sequence, optionally video object planes can be grouped together to form a group of video object planes (GOV). Figure 3-1 shows the hierarchy of structure components in an MPEG-4 sequence.

Each video object plane is coded differently much like the frames are coded in all the preceding standards. Additionally, shape-coding tools are introduced in order to provide the ability to represent arbitrary shaped objects. Depending on the coding mode of each VOP (I-, P- or B-VOP), the information needed to represent each VOP includes shape, quantized DCT coefficients for intra-frame coded VOP's, quantized DCT coefficients of the motion compensated error VOPs, and motion vectors for the inter-frame coded VOPs.

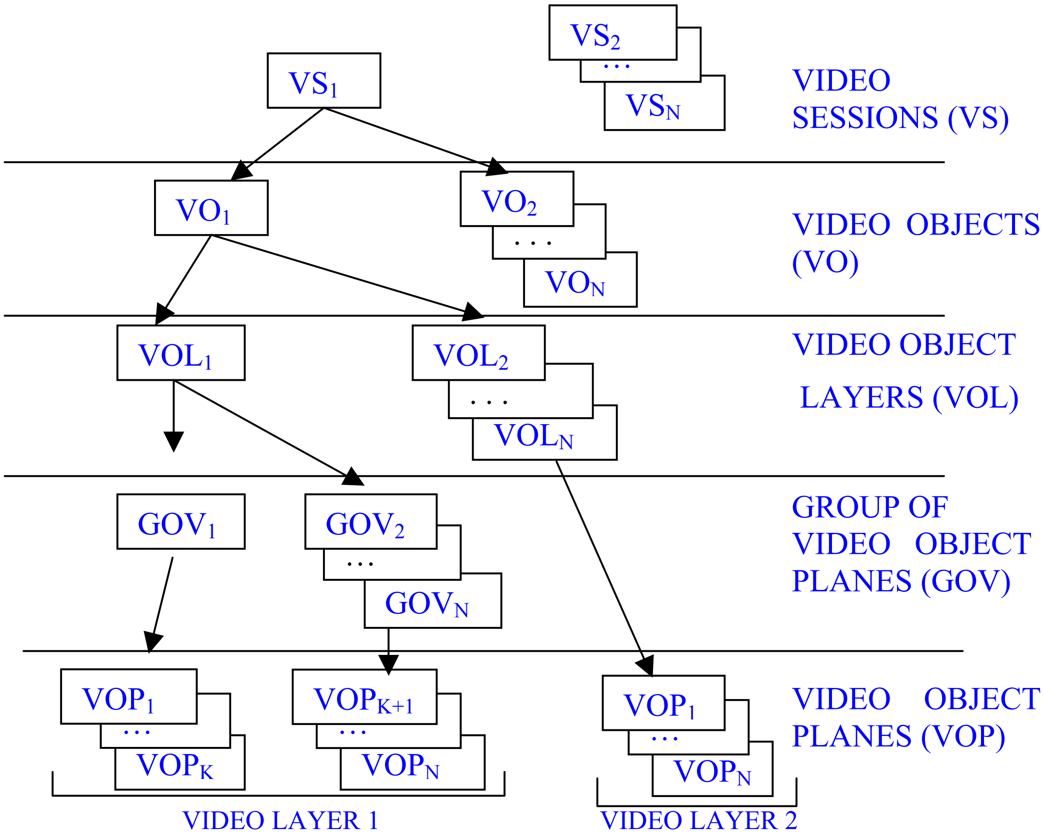


Figure 3-1. MPEG-4 Video Bitstream Logical Structure, From Ref. [5].

Figure 3-2 shows the block diagram of an MPEG-4 codec. The input frame is decomposed into multiple VOs. The multiplexer combines the coding information for each video object along with scene descriptions on where and when each object is going to be displayed in the sequence. The decoder decodes the information for each VO and uses the composition information to regenerate the sequence.

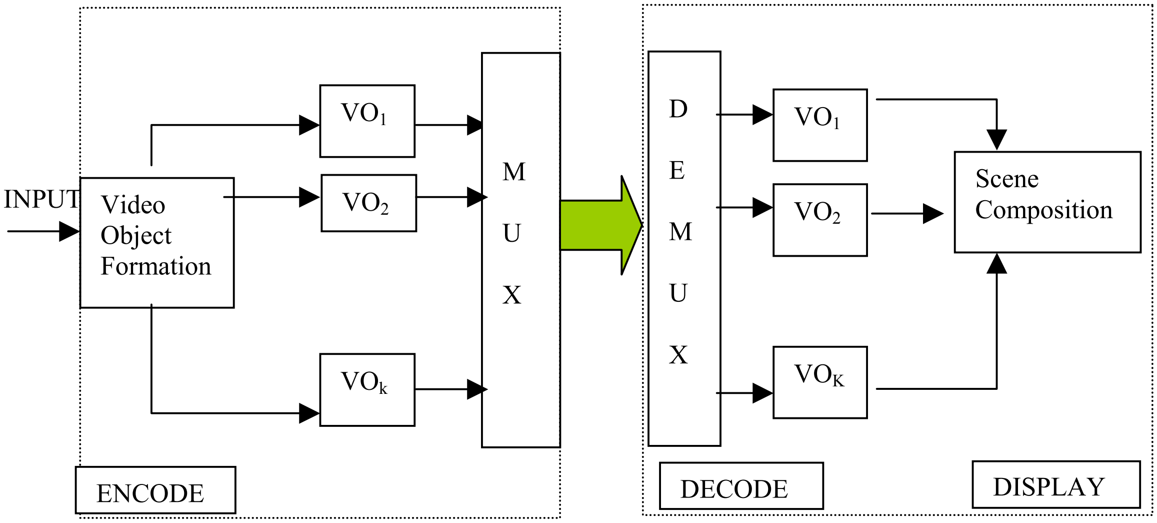


Figure 3-2. Block Diagram of MPEG-4 Video Codec, After Ref. [5].

4. Profiles in MPEG-4

MPEG-4 provides a large and rich set of tools for the coding of audio-visual objects. For effective implementation, subsets of the MPEG-4 systems have been identified for specific applications. These subsets, called ‘profiles’, limit the tool sets that a decoder needs to implement [9]. For each of these profiles, one or more levels have been defined to restrict the computational complexity. A profile-level combination allows a codec builder to implement only a subset of the standard while maintaining interworking with other MPEG-4 devices built to the same combination and to check whether MPEG-4 devices comply with the standard.

Profiles exist for various types of media content, such as audio, visual and graphics, and for scene descriptions. MPEG does not prescribe or advise combinations of these profiles but care has been taken to ensure that good matches exist between the different areas [9]. The basic profiles are termed simple, simple scalable, core and main.

The simple visual profile provides efficient, error resilient coding of rectangular video objects suitable for applications on wireless networks. Three levels are defined with bit-rates in the range 64-384 kbps. I-VOPs, P-VOPs, unrestricted motion vectors, slice resynchronization, data partitioning and reversible VLC are supported. A decoder using this profile can decode H.263 video streams [2].

The simple scalable visual profile adds support for B-VOPs, spatial and temporal scalability to the simple visual profile. It is useful for applications that provide services at more than one level of quality due to bit-rate or decoder resource limitations, such as Internet use [9].

The core visual profile adds support for coding of arbitrary-shaped and temporally scalable objects to the Simple Visual Profile. It is useful for applications that provide relatively simple content-interactivity, such as Internet multimedia applications [2], [9].

The main visual profile adds support for the coding of semi-transparent and sprite objects to the core visual profile. Progressive and interlaced material along with gray scale shape coding is also supported in this profile. It is useful for interactive and entertainment-quality broadcast and DVD applications [2], [9].

Additional profiles were added during the development of the standard to support more specific applications and to give more flexibility to the designer by combining different tools. Version 2 of the standard added the advanced real-time simple (ARTS) profile, the core scalable profile and the advanced coding efficiency (ACE) profile suitable for real-time coding applications, such as the videophone, tele-conferencing and remote observations where additional coding efficiency is required combined with supplementary robustness to errors. Finally, version 3 added the advanced simple profile, fine granular scalability profile, simple studio profile and core studio profile [8], [9].

A level within a profile defines the constraints on the parameters in the bitstream that relate to the tools of that profile [9]. For core, simple, and main profiles, a subset of level constraints is given in Table 3-1.

Profile and Level		**Typical scene size**	**Bit-rate (bit/sec)**	**Maximum number of objects**
Simple Profile	L1	QCIF	64 k	4
	L2	CIF	128 k	4
	L3	CIF	384 k	4
Core Profile	L1	QCIF	384 k	4
	L2	CIF	2 M	16
Main Profile	L2	CIF	2 M	16
	L3	ITU-R 601	15 M	32
	L4	1920x1088	38.4 M	32

Table 3-1. Subset of MPEG-4 Video Profile and Level Definitions, From Ref. [9].

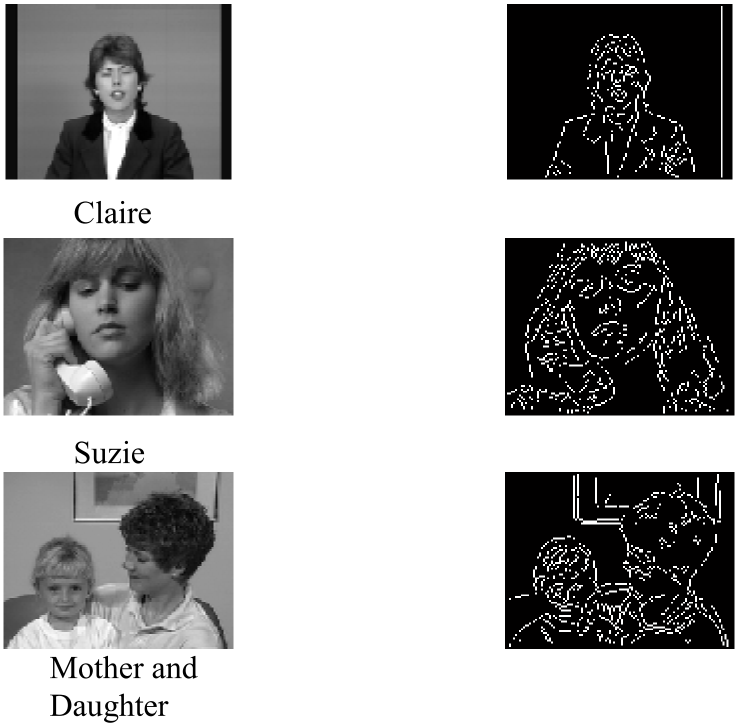
B. OBJECT BASED APPROACH

As mentioned previously, MPEG-4 treats video sequences as a composition of video objects with individual properties of shape, motion and texture. See Figure 3-1 for the hierarchical structure of the standard. For coding applications, the background that is considered as one video object can be coded only once if it is stationary over time while the foreground objects are coded through time. Since the foreground objects usually are only a fraction of the picture, this can lead to great compression efficiency [1].

1. Video Object Plane Generation

A video object at any given time is identified with a video object plane (VOP). Video object planes are the result of *image segmentation* algorithms applied on a frame. Each frame is segmented into two-dimensional semantic objects. There can be more than one VOP in each frame.

There are a number of techniques used for image segmentation [1]. In this work only sequences of a “talking head” with a uniform background are considered. Matlab provides a number of edge detection methods. These methods look for places in the image where the intensity changes rapidly by determining the first derivative of the intensity. In places where the first derivative is larger than a threshold, edges are highly probable. From these methods, the “Canny” method was used to obtain the edges in a frame. The Canny method applies a 2-D Gaussian filter at the beginning to smooth the image and alleviate any noise and then uses two different thresholds to detect strong and weak edges [24], [25]. Weak edges are included in the output only if they are connected to strong edges. Thus this method is more robust to noisy images as it is less likely to include outliers. Using the "Canny" operator and determining different threshold values in each sequence, the edge detection results for the video objects are shown in Figure 3-3. For more complicated frames, such as these with non-uniform background and multiple objects, the segmentation becomes a challenging task. Further investigation of segmentation was beyond the scope of this thesis. Since MPEG-4 does not specify a segmentation technique, a simple approach was adopted and only simple video sequences were chosen in this work.


Claire

Suzie

Mother and Daughter

Figure 3-3. Edge Detection Results for Three Sequences (Claire, Suzie, Mother and Daughter). The Left Column Shows the Original Frames and the Right Column Shows the Edge Detection Results.

After segmentation and the generation of contours of the foreground object, the results of a *binary alpha-map* were obtained. Construction of binary and gray-alpha maps will be discussed in Section C of this chapter. This map has the dimensions of the frame or a bounding box of a VOP; pixels with coordinates that correspond to the foreground object in the actual frame have value of 255 while pixels that correspond to the background have a value of 0.

Figure 3-4 shows the results from the segmentation of frame 10 in sequence “Claire” and the formation of the two VOPs (background and speaker). Since the background was not recorded separately from the foreground, when the information of the foreground object is subtracted from the frame, pixels in the missing territory had to be interpolated in order to create a uniform background that would be ready for further coding. This procedure was required in order to smooth the sharp edges generated from the missing pixel area in the background image.

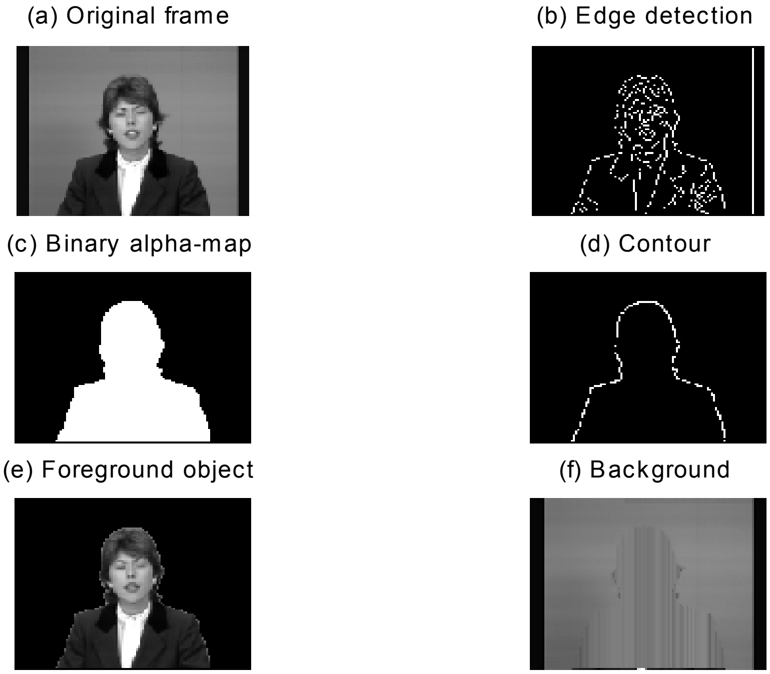


Figure 3-4. Generating a VOP for One Frame of “Claire” Sequence.

2. Bounding Box Generation

After segmentation of the frames and the extraction of the VOPs, the arbitrary shaped object must be encapsulated in a boundary rectangle such that the object contains the minimum number of macroblocks [1]. The *tightest rectangle* that surrounds the object is generated as shown in Figure 3-5. A *control macroblock* is the macroblock whose southeast point is the northwest point of the tightest rectangle.

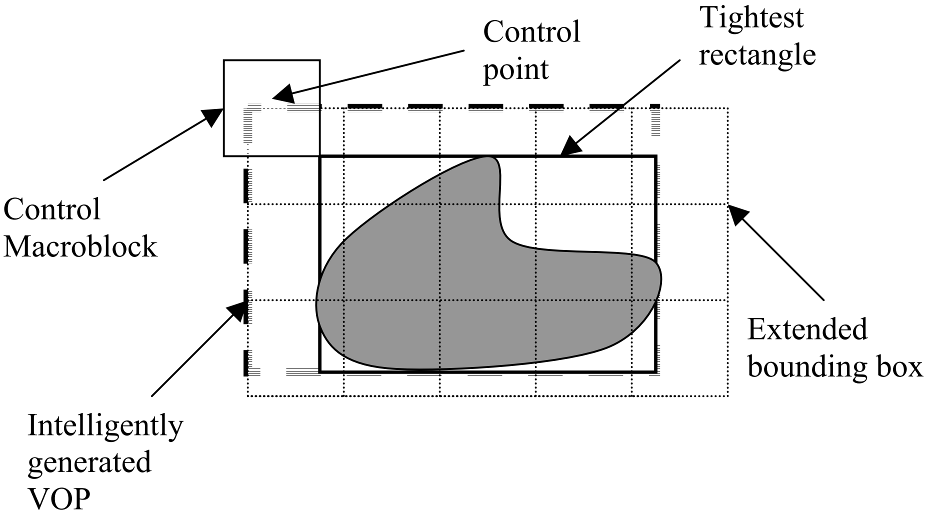


Figure 3-5. Generation of Bounding Box Algorithm , From Ref. [1].

Each point of the control macroblock is tested as the northwest point of a new rectangle that completely contains the object and consists of integer number of macroblocks. This is the *extended bounding box* in Figure 3-5. The point of the control macroblock that leads to a rectangle that contains the minimum number of macroblocks is chosen to be the *control point* of the bounding box. Lastly, the top left co-ordinate of the tightest rectangle is extended to the control point co-ordinate in order to form the *intelligently generated VOP* in the above figure.

C. SHAPE CODING

The need to represent arbitrary shaped objects led to the introduction of algorithms capable of coding the shape information of objects. Before describing these methods some terms used in MPEG-4 must be defined. An *alpha-map* is a template that contains the VOP. The shape of an object S in a $M\times N$ image is defined as:

$$S = \{s(m,n) \mid 0 \le m \le M, 0 \le n \le N\} \tag{3.1}$$

where $s(m,n)$ is defined as the alpha-map and the pixel values are in the range of 0 to 255. For each pixel when $s(m,n) > 0$, the pixel is inside the object while when $s(m,n) = 0$, the pixel is part of the background. When $s(m,n) \in \{0,255\}$, a *binary alpha-map* is created. When $s(m,n) \in [0,255]$, we have a *gray alpha-map,* which also contains information about the transparency of the object. Given that a background image is represented by f_b and a foreground object f_f along with an alpha-map s, the overlaying of the object on the background is represented by

$$f(m,n) = \left(1 - \frac{s(m,n)}{255}\right) f_b(m,n) + \frac{s(m,n)}{255} f_f(m,n) \tag{3.2}$$

When a gray alpha-map is constructed, it is possible to see how the information of the transparency of the object with respect to the background is used. The gray alpha-map provides an efficient tool to code objects with a different transparency in special effect scenes. In this work, however, only binary alpha-maps are used.

Macroblocks in the alpha-plane are called alpha-blocks or binary alpha blocks (BAB). BABs that lie completely outside the object have all elements zero and are called transparent BABs while BABs residing completely inside the object are called Opaque BABs. When a BAB contains pixels belonging both to the object and the background, it is called a boundary BAB. These are illustrated in Figure 3-6.

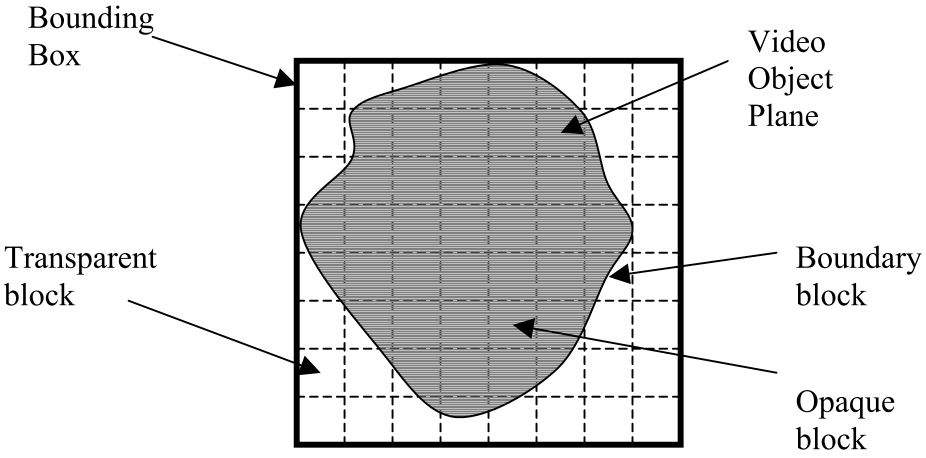


Figure 3-6. Definition of Blocks in Texture Coding, After Ref. [3].

The main effort of the shape coding algorithms is to efficiently code boundary blocks as opaque; the transparent blocks are not coded at all since the information needed is completely contained within the boundary block information. These blocks are only flagged as opaque or transparent in the coding process and the decoder decodes them accordingly.

The shape coding algorithms are categorized into bitmap coding and contour coding [2]. In the former, in its simplest form, the coder scans the alpha-map and transmits a zero or one depending on whether or not the pixel is part of the object. This is somewhat inefficient since the correlation between neighboring pixels is not exploited. However, algorithms, such as Modified Reed and Context Arithmetic Encoding (CAE) [2], that provide tools to increase the efficiency of these coders have been proposed. In the latter, the contour of the object is followed and the corresponding position of a pixel, which is part of the contour, is coded. In this thesis the contour-coding algorithm was chosen for shape-coding in the encoder.

1. Chain Coding Algorithm

Chain coding is a lossless coding technique used to code the contour of the foreground object. The contour of a foreground object can be derived after image segmentation and the construction of the binary alpha-map as shown in Figure 3-4 (c) and (d). The code contains the coordinates of the starting point and then the direction of the next pixel in the contour according to a predetermined pattern shown in Figure 3-7.

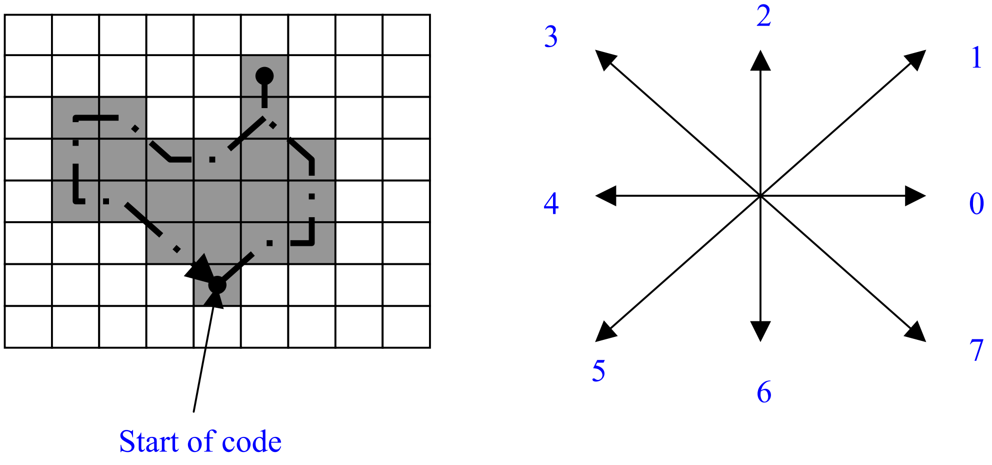


Figure 3-7. Chain Code for Pixels with Eight Neighbors, After Ref. [1] and Ref. [2].

In the example in the figure, the code transmitted is 1 0 2 2 3 2 6 5 4 3 4 6 0 7 7. To improve coding efficiency, a differential representation of the code is used. Exploiting the cyclic property of a chain code in eight directions, a differential chain code is constructed as follows [1]:

$$d = \begin{cases} c_n - c_{n-1} + 8, & \text{if } c_n - c_{n-1} < -3 \\ c_n - c_{n-1} - 8, & \text{if } c_n - c_{n-1} > 4 \\ c_n - c_{n-1}, & \text{otherwise} \end{cases} \tag{3.3}$$

where d is the difference, c_n the current chain code, and c_{n-1} the previous chain code. Next, the differential chain code is a variable length code according to Table 3-2 [1]. Consequently, the differential chain code for the example shown in Figure 3-6 is 1 –1 2 0 1 –1 4 –1 –1 –1 1 2 2 1 0.

At the decoder, after decoding the variable length code, the decoding of the differential chain code is performed using

$$c_n = (c_{n-1} + d + 8) \bmod 8 \tag{3.4}$$

Differential Code	Huffman Code
0	1
1	00
-1	011
2	0100
-2	01011
3	010100
-3	0101011
4	0101010

Table 3-2. Huffman Table for Differential Chain Code, From Ref. [1].

D. TEXTURE CODING

Texture information is the pixel values of the intra-VOPs or the pixel values of the error residual of the motion compensated VOPs [3]. Block based techniques, such as DCT, are applied and the coefficients are transmitted over a channel. The DCT operation is applied at the block level (8×8 pixels), but the information transmitted is grouped into macroblocks (16×16 pixels).

As shown in Figure 3-8, three kinds of macroblocks are encountered in a VOP's bounding box: macroblocks that lie completely inside the VOP's shape, macroblocks that lie inside the bounding box but outside the VOP, and macroblocks that partially lie inside the VOP. For the first case, a simple 2D-DCT operation is performed for each block inside the macroblock. In the second case, the macroblocks are not coded at all. In the last case, the blocks that reside outside the VOP are coded as zero blocks while for the blocks that are partially inside the VOP, two methods (texture extrapolation and Shape Adaptive DCT) can be used.

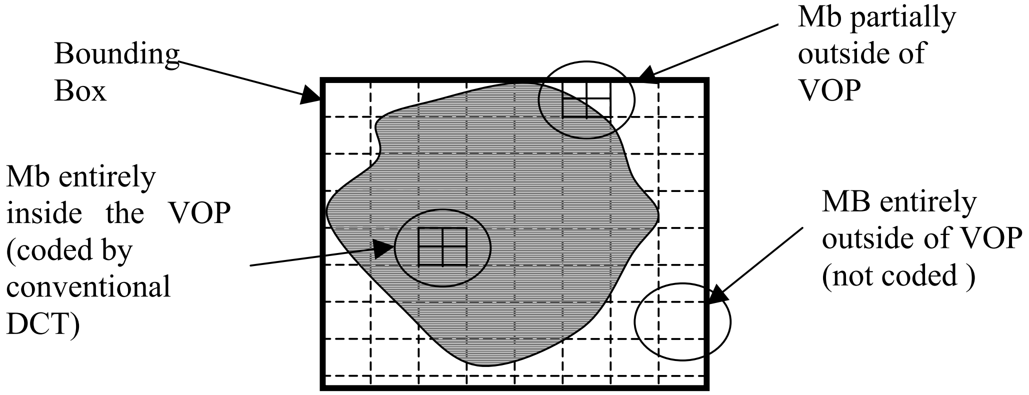

Bounding Box
Mb partially outside of VOP
Mb entirely inside the VOP (coded by conventional DCT)
MB entirely outside of VOP (not coded)

Figure 3-8. Macroblock Based Texture Coding for Arbitrarily Shaped VOP, From Ref. [3].

Edges in images represent high frequency components. In this manner, performing DCT in a block containing edges is going to produce a large amount of significant coefficients. This is eventually going to affect the bit-rate. On the other hand, pixels that are not part of the VOP, are not visible at the decoder, so their values can be manipulated in such a way that an unnecessary increase in bit-rate will not be introduced. A method called *texture extrapolation* stretches the values of the region of VOP to the boundaries of the block so that high frequencies do not exist [2]. This method of extrapolating the values of the boundaries of the object to the boundaries of the block is an essential pre-processing step for motion estimation in arbitrary shaped objects and will be presented in Section E. On the other hand, the decoder can correctly decode since shape information is available. This method produces as many coefficients as the pixels inside a block.

A more suitable method called shape adaptive DCT (SA-DCT), which increases the efficiency of texture coding, has been implemented for processing the boundary blocks in this thesis.

1. Shape-Adaptive Discrete Cosine Transform (SA-DCT)

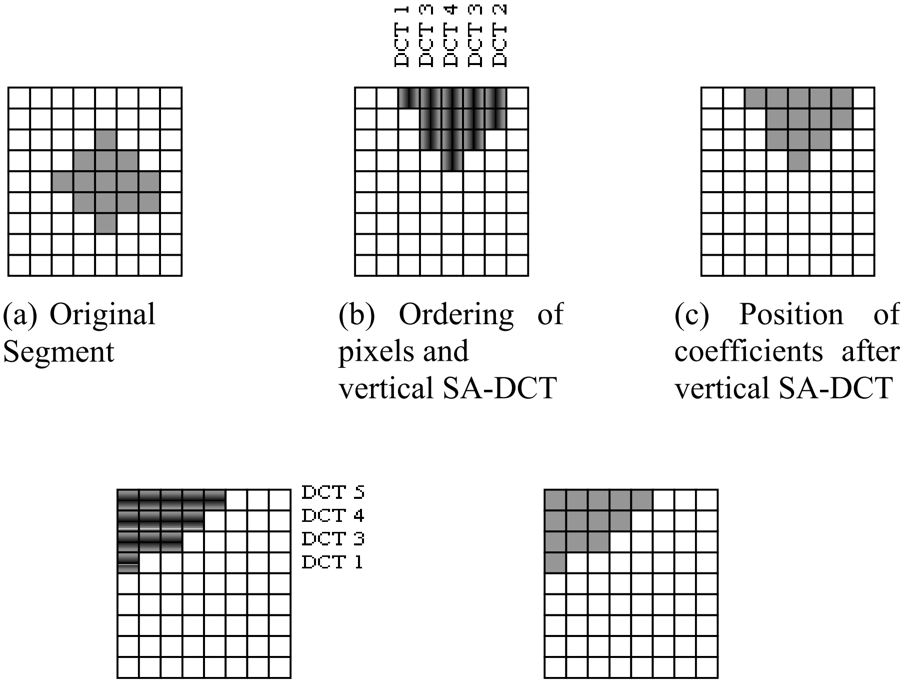
The SA-DCT is used for coding boundary blocks as it produces as many coefficients as the pixels inside a VOP. This improves coding efficiency when compared to a regular DCT in these kinds of blocks. No more computations other than the original DCT are required and the coefficients are in comparable positions as in the standard DCT.

The SA-DCT algorithm is based on representing M×M blocks in terms of predefined orthogonal sets of basis functions. Basic 1D-DCT operations used in Figure 3-9 illustrate the procedure. In Figure 3-9(a), the original 8×8 block is shown. First, the pixels are shifted vertically and aligned to the block boundary. The length of each column is computed and 1D-DCT transform is applied. For an N-point sequence the DCT is computed as stated in Equation 2.1

$$F(v) = C(v)\sum_{n=0}^{N-1} f(n)\cos\left[\frac{(2\pi+1)\cdot n\cdot\pi}{2N}\right], v = 0,...,N-1 \qquad (3.5)$$

where $C(v)=\sqrt{\frac{1}{N}}$ for $v=0$ and $C(v)=\sqrt{\frac{2}{N}}$ for $v\neq 0$. The resulting coefficients are then shifted horizontally and aligned with the left block boundary and the same transformation is now applied on the rows. The result is depicted in Figure 3-9(f), where the number of coefficients is equal to the number of pixels residing in the VOP area.

At the decoder, the procedure is applied in reverse order. First, the horizontal 1D-IDCT is performed and then the coefficients are shifted. The information on the shape is crucial in the decoding position in order for the decoder to correctly compute the amount of shifting of each row-column. Compared to texture extrapolation, this technique gains, on average, 1-3 dB PSNR measured over an image segment for the same bit-rate [2].



(a) Original Segment

(b) Ordering of pixels and vertical SA-DCT

(c) Position of coefficients after vertical SA-DCT

(d) Ordering of pixels and horizontal SA-DCT

(e) Final location of coefficients

Figure 3-9. SA-DCT Transform on an 8×8 Foreground Object, After Ref. [2].

2. Quantization and Run Length Encoding

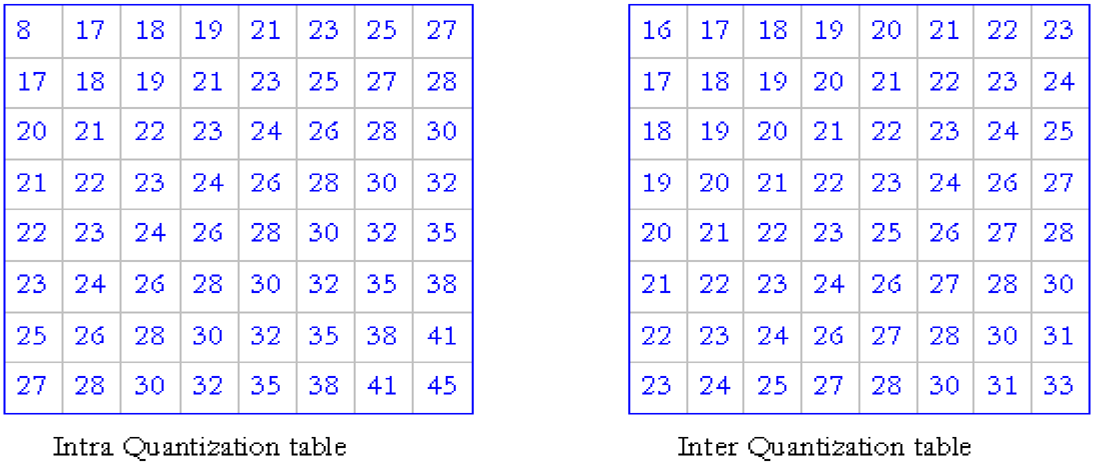
The human visual system is less sensitive to high frequencies than low frequencies. This property is exploited in order to decrease the required bandwidth. Thus, high frequency coefficients are quantized using a coarser step size than low frequency coefficients. This is done in practice by dividing the 8×8 block of DCT coefficients by an 8×8 matrix of step sizes and rounding the result to the nearest integer. The quantization tables used in MPEG4 are different for intra and inter macroblocks and are shown below in Table 3-3.

A quantization parameter (Q_P) is used in order to scale prior to quantization of the weighting matrix. The range of the quantization parameter is between 1 and 31. Fixed Q_P for all macroblocks results in a variable bit-rate output while for constant bit-rate needs, a rate control algorithm must be applied to constantly monitor the target bit-rate and change the Q_P by sacrificing quality in favor of compression or visa versa. A Q_P of 1 results in the higher bit-rate while a Q_P of 31 achieves the best compression but yields the worst quality.

It is obvious that the step sizes are increased when moving from lower to higher coefficients, thus eliminating perceptually unimportant information. The quantization is performed according to:

$$F_q(u,v) = \text{round}\left(\frac{F(u,v)}{Q_{uv}Q_P}\right) \tag{3.6}$$

where *F(u,v)* and *F_q(u,v)* are the DCT coefficients before and after quantization at position *(u,v)*, *Q* is the quantization matrix, and Q_p is the quantization parameter.



8	17	18	19	21	23	25	27
17	18	19	21	23	25	27	28
20	21	22	23	24	26	28	30
21	22	23	24	26	28	30	32
22	23	24	26	28	30	32	35
23	24	26	28	30	32	35	38
25	26	28	30	32	35	38	41
27	28	30	32	35	38	41	45

Intra Quantization table

16	17	18	19	20	21	22	23
17	18	19	20	21	22	23	24
18	19	20	21	22	23	24	25
19	20	21	22	23	24	26	27
20	21	22	23	25	26	27	28
21	22	23	24	26	27	28	30
22	23	24	26	27	28	30	31
23	24	25	27	28	30	31	33

Inter Quantization table

Table 3-3. Quantization Tables.

Figure 3-10 shows the results obtained from coding a video sequence ("Claire") using flat quantization tables and Table 3-3. Even though the performance as measured by PSNR is superior when using flat tables (fixed step size for all coefficients), the perceptual quality of MPEG-4 compressed frames, using Table 3-3 for quantization, is very similar.

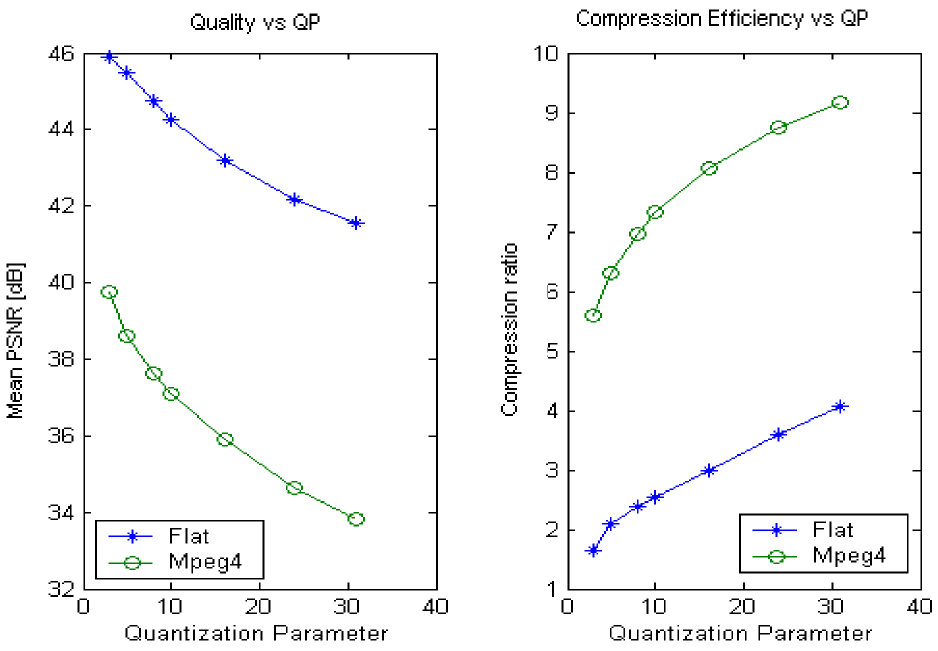


Figure 3-10. Comparison between Using Flat and Variable Step Sizes Quantization Tables.

After quantization, run length encoding of the quantized coefficients is performed. Coefficients are zigzag scanned (see Figure 2-3(a) in the previous chapter) which produces a trailing of zeros for each block. After run-length encoding, the values are Huffman coded to further reduce the bit-rate, and the coded stream is appended to the bit stream.

Figure 3-11 shows the original and the reconstructed frame of the sequence using flat and variable step size tables. Figure 3-11(a) shows the original frame while Figure 3-11(b) shows the reconstructed frame using a flat quantization table with step size one. Figure 3-11(c) shows the reconstructed frame using an MPEG-4 table. The figure shows the first frame in the sequence coded in intra mode using a quantization parameter of $Q_P = 24$.

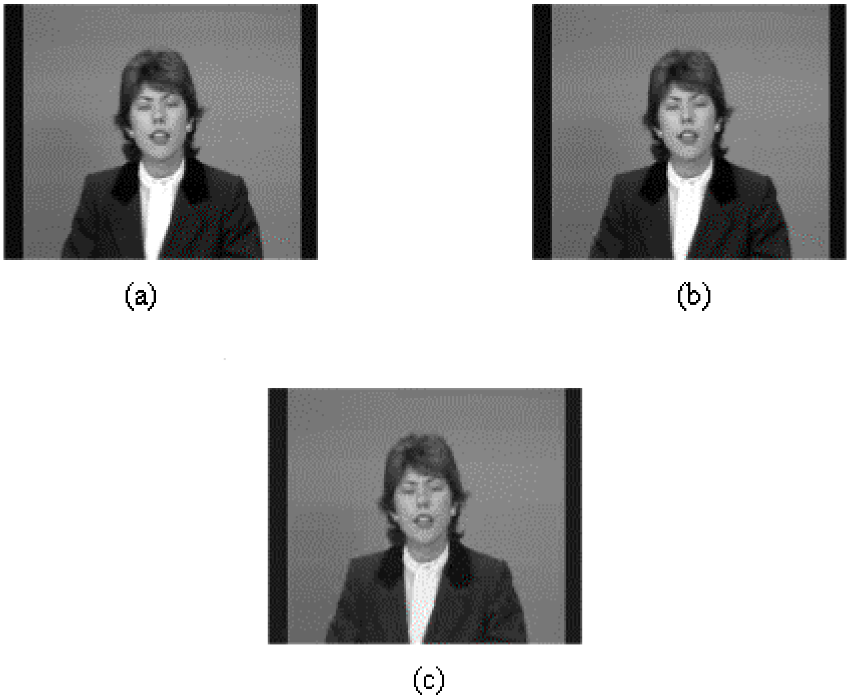


Figure 3-11. Reconstructed Frame Using Flat and MPEG-4 Quantization Table.

E. MOTION ESTIMATION

Motion estimation and compensation in MPEG-4 is performed at the macroblock level. This means that each macroblock is associated with one motion vector. The techniques used are similar to the techniques described for motion estimation in Chapter II with the exception that some modifications are needed to code arbitrary shaped objects and especially for boundary macroblocks. Additionally, fractional accuracy motion estimation is performed, thereby boosting the performance of motion estimation results.

As shown in Figure 3-12, macroblocks that reside completely inside the VOP are coded using standard techniques. Macroblocks residing completely outside the VOP but inside the bounding box are excluded from the motion estimation algorithm while for macroblocks containing the boundary of the VOP, a technique called *polygon based matching* is used. In this technique, the matching error is computed as the sum of the absolute value differences between those pixels of the current macroblock that are inside of the VOP shape and of the corresponding pixels in the reference VOP [3]. As some of the reference pixels used in the matching may be outside of the reference VOP, a block based repetitive padding is performed in order to extrapolate the values of these pixels from those inside of the reference VOP.

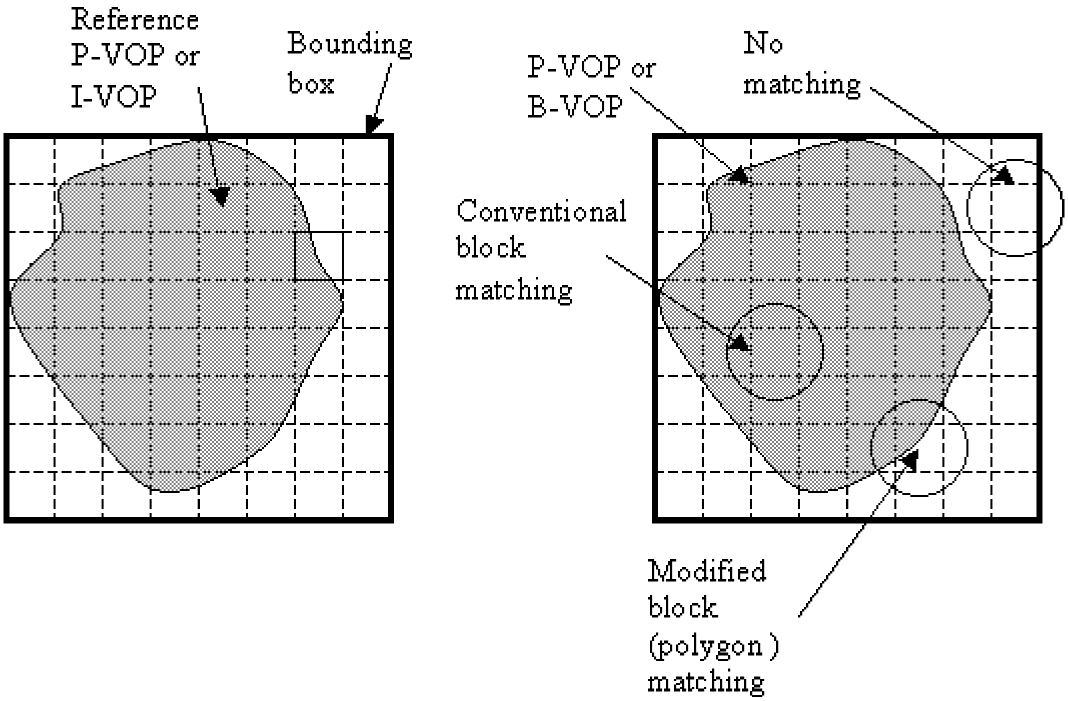


Figure 3-12. Motion Estimation for Arbitrary Shaped VOPs, After Ref. [3].

In horizontal padding, each pixel at the boundary of a VOP is extrapolated to the outside in order to fill the transparent region of the boundary macroblock [1]. Vertical padding is performed similarly. The procedure is shown in Figure 3-13.

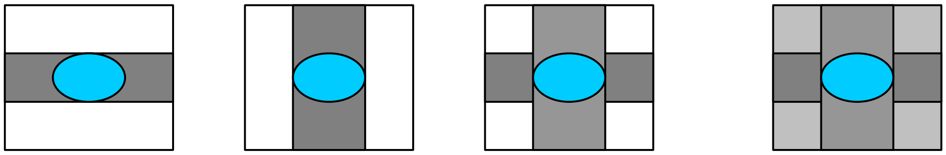


Figure 3-13. Padding of Boundary Macroblocks, From Ref. [7].

The macroblocks adjacent to the boundary macroblocks are called exterior macroblocks and are also padded with the values at the borders of the boundary macroblock. When an exterior macroblock has two or more neighboring boundary MBs it uses the values of the neighboring boundary macroblock with the highest priority number according to Figure 3-14(b). The remaining exterior macroblocks, which are not located next to any boundary macroblocks are filled with 2^{r-1},where r is the number of bits used to represent a sample. For an 8-bit luminance component and associated chrominance, this implies filling with a value of 128. Figure 3-14 (a) shows the boundary and external macroblocks for an arbitrary shape VOP (star). In the figure, gray denotes external macroblocks, shade denotes boundary macroblocks and white are macroblocks padded with 128.

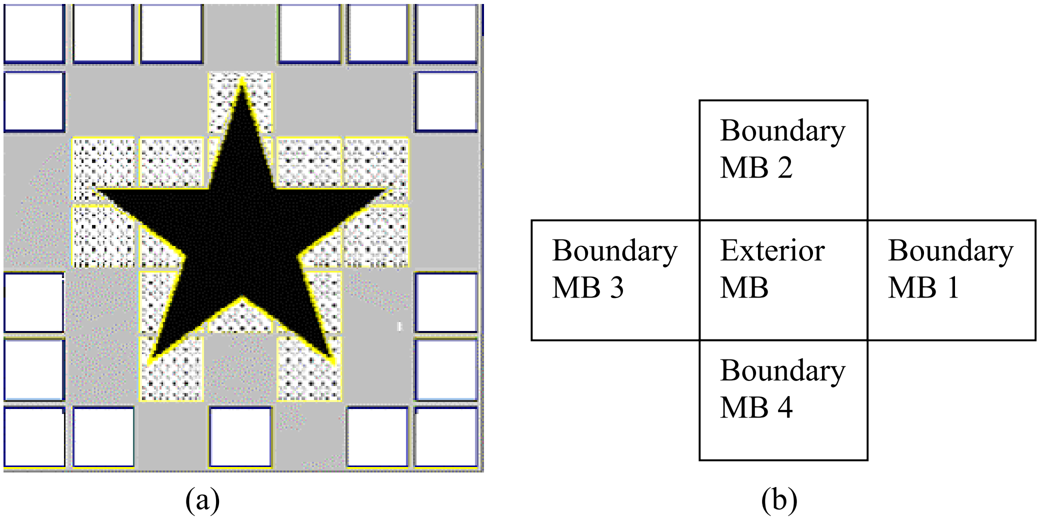


Figure 3-14. (a) Extended and Normal Padding for VOP (star), From Ref. [6]; (b) Priority of Boundary Macroblocks Surrounding an Exterior Macroblock, From Ref. [8].

1. Fractional Pixel Accuracy

In Chapter II, the block-matching algorithm (BMA) was presented using an integer value as step size. However, it is possible that a fractional number can be used for more accurate motion representation. In order to realize a step size of *1/K*, the reference frame has to be interpolated using bilinear interpolation by a factor of *K*. For *K* = 2, a half-pixel accuracy is defined. MPEG-4 uses half-pixel accuracy motion estimation since it can provide a significant improvement in estimation over integer accuracy, especially in low-resolution video [2].

The motion estimation is performed in two steps. First, integer estimation is performed using the conventional BMA technique. This is represented as point A in

Figure 3-15. Next, using the result from the first step, eight neighboring points, marked as X, Y and Z in Figure 3-15, are tested for better match. The fractional points are interpolated as follows

$$h = \frac{A+X}{2} \tag{3.7}$$

$$v = \frac{A+Y}{2} \tag{3.8}$$

$$c = \frac{A+X+Y+Z}{4} \tag{3.9}$$

Clearly, a fractional step size algorithm increases the computational complexity of the BMA by four. The overall complexity of the encoder is increased due to interpolation of the frames [2]. Because of the computational complexity, the half pixel accuracy algorithm was not implemented in Matlab in this work.

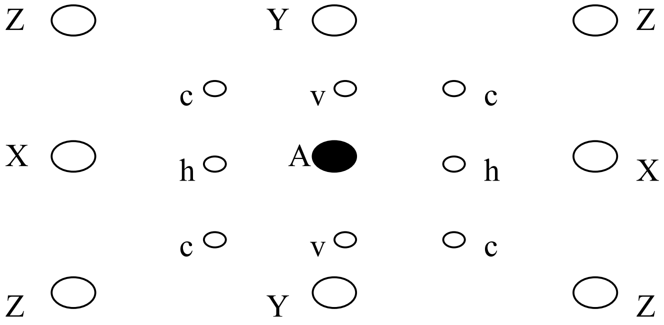
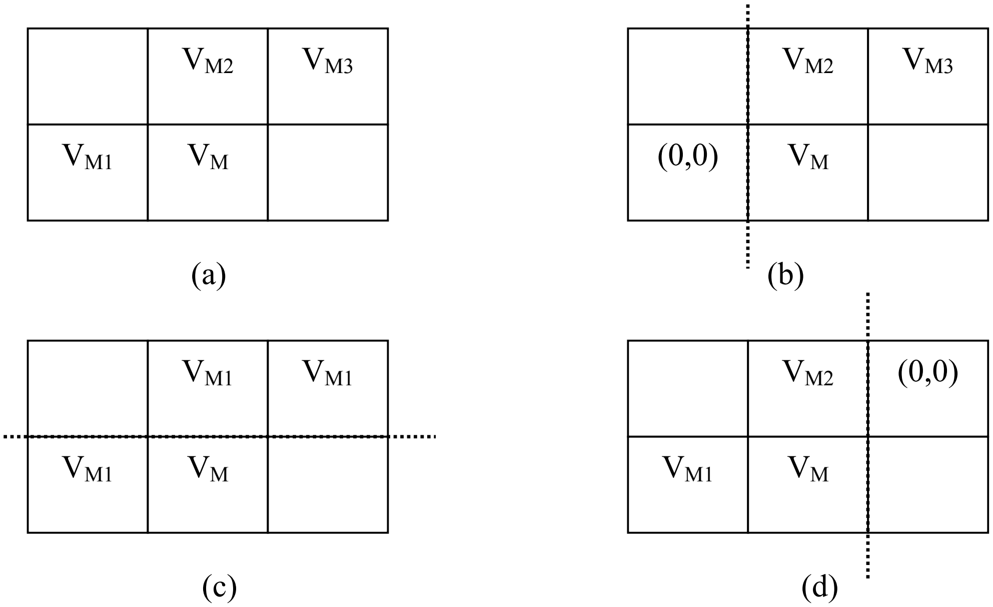


Figure 3-15. Sub-Pixel Search Positions, around Pixel Coordinate A, From Ref. [1].

2. Coding of Motion Vectors

Each motion vector is represented as coordinates of motion in the x and y direction. The individual components are differentially coded with predictions based on motion vectors from three surrounding macroblocks as shown in Figure 3-16(a); V_M is the current motion vector. For the special case of the surrounding macroblocks that are outside of the frame, the values of the surrounding motion vectors are taken as shown in Figure 3-16(b), (c) and (d). Figure 3-16(a) is applicable when the macroblock is surrounded by all the macroblocks needed for prediction. When the macroblock lies on the border of a frame (dotted line), Figure 3-16(b), (c) and (d) are applicable.



(a) (b) (c) (d)

Figure 3-16. Motion Vector Prediction, From Ref. [1].

For each component, the predictor is calculated as the median between the three surrounding motion vectors:

$$P = \text{Median}(V_{M1}, V_{M2}, V_{M3}) \tag{3.10}$$

This is done both in x and y directions and two predictors $(\hat{V}_{Mx}, \hat{V}_{My})$ for each direction are computed. The difference between the components of the motion vector (V_{Mx}, V_{My}) and the predictor are variable length coded:

$$D_{MV_x} = V_{Mx} - \hat{V}_{Mx} \tag{3.11}$$

$$D_{MV_y} = V_{My} - \hat{V}_{My} \tag{3.12}$$

F. SCALABILITY

Scalability refers to the capability of recovering physically meaningful video information by decoding only partial compressed bit streams [2]. *Scalable* or *layered* video coding was proposed during the design of the H.261 in the late 1980s [1]. The idea was to create two bit streams from the same video input. One bitstream, called the *base layer*, contains vital information for the decoding of the sequence and is transmitted over relatively reliable channels or with added protection and the other, called the *enhancement layer,* contains information that when decoded would increase the quality of the decoded sequence but is not vital for decoding. Figure 3-17 shows a spatial/temporal encoder with two layers, the base layer and an enhancement layer.

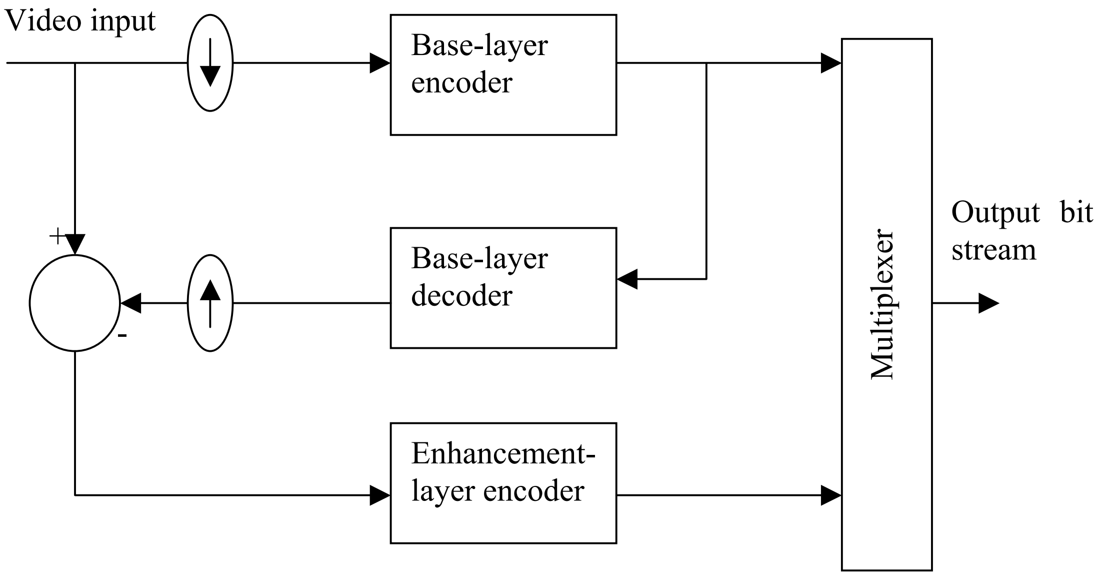


Figure 3-17. Block Diagram of a Two-Layer Spatial/Temporal Encoder, From Ref. [1].

Scalable or layered video can also offer adaptivity to channel error characteristics. For wireless applications, scalability allows the use of unequal error protection in response to channel characteristics. For Internet applications, layered coding provides a platform to deal with network congestion effectively; data in the enhancement layer can be discarded without resulting in undecodable files. Users with slow modems may decode only the base layer and receive reduced resolution video while users with fast modems may decode both base and enhancement layers.

1. Spatial Scalability

Spatial scalability is the representation of the same video at different spatial resolutions [2]. The base layer is coded by itself and the enhancement layer employs a spatially interpolated base layer and together they provide the full spatial resolution of the input video source [1].

An input frame is first spatially downsampled and coded using a conventional non-scalable coder. The result is the base layer contains coded information of the video sequence at a lower resolution. Before entering the channel, the encoder decodes the compressed bit stream and upsamples the output. This is then subtracted from the original frame and the resulting error frame is fed into another non-scalable coder to produce the enhancement layer.

The procedure is shown in Figure 3-18 for a given frame from video sequence "Suzie". Figure 3-18(a) is the input frame at the original resolution (144×176). Figure 3-18(b) is the downsampled frame by a factor of 2 to a resolution of 72×88. This frame is coded and forms the base layer. The decoder receives the base layer and upsamples if the original resolution is requested from the end user. The result is depicted in Figure 3-18(c). Figure 3-18(d) is the difference between 3-18(a) and 3-18(b); this difference is coded to realize the enhancement layer. Upon receiving and decoding the enhancement layer, the decoder adds the enhancement layer and the upsampled version of the base layer and displays the result at the original resolution as shown in Figure 3-18(e).

In MPEG-4, the spatial scalability definition VOPs in the base layer are encoded as I-VOPs or P-VOPs while the VOPs of the enhancement layer are encoded as P-VOPs or B-VOPs. If a VOP in the enhancement layer is temporally coincident with an I-VOP in the base layer, it could be treated as a P-VOP. VOPs in the enhancement layer that are coincident with P-VOPs in the base layer could be coded as B-VOPs. Since the base layer serves as the reference for the enhancement layer, VOPs in the base layer must be encoded before their corresponding VOPs in the enhancement layer. Figure 3-19 shows how a video sequence would be encoded using the above convention.

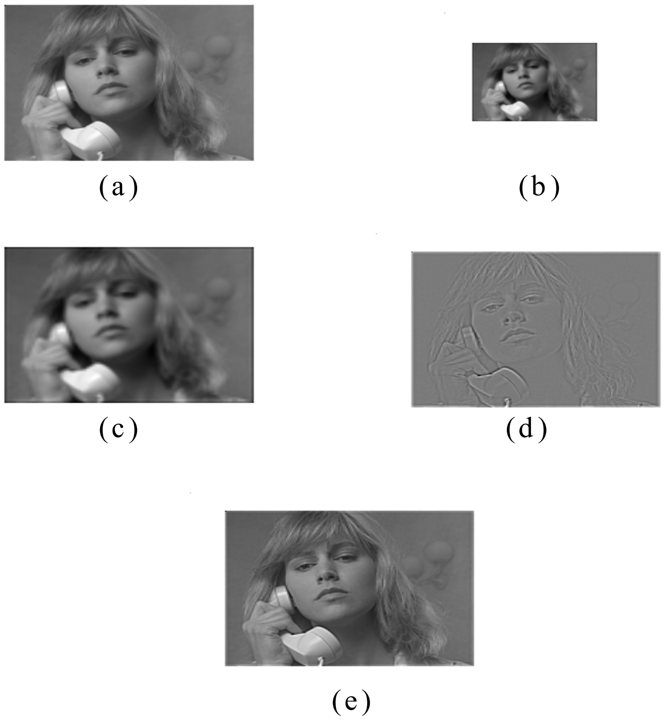


Figure 3-18. Effect of UpSampling and DownSampling of One Frame of “Suzie”.

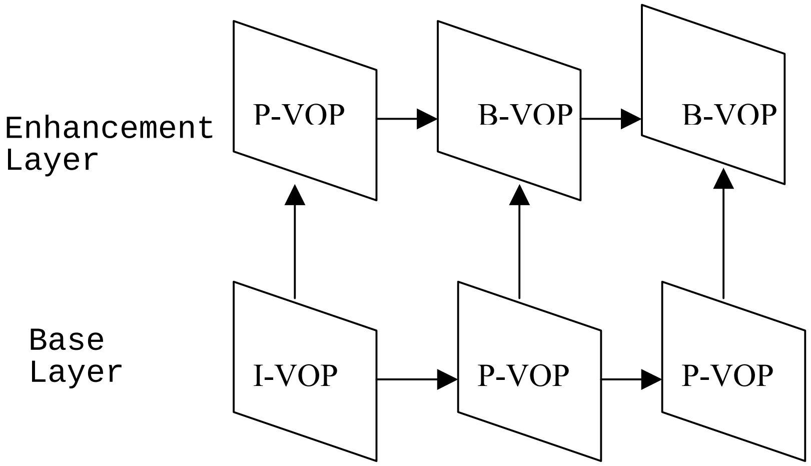


Figure 3-19. Formation of VOPs in Base and Enhancement Layers, From Ref. [5].

2. Temporal Resolution

Temporal scalability is defined as the representation of the same video sequence in varying temporal resolutions or frame rates. The spatial resolution of the layers is assumed to be the same as the input video. In temporally scalable coders, the upsampling and downsampling are done in the time domain. The simplest way to perform temporal scalability is by frame skipping. A temporal downsampling by 2:1 is done by skipping every other frame and vice versa.

For motion compensation, a frame in a base layer can only use frames in the base layer as a reference frame while a frame in the enhancement layer can use reference frames from both the base and the enhancement layers [1].

The simplest way to achieve temporal scalability is by coding I- and P-frames of a sequence as the base layer and B frames as the enhancement layer. Note that B-frames are not used for prediction so a loss in the enhancement layer would not degrade the visual quality of the decoded video but affect only the smoothness in the temporal domain.

ISO/IEC 14496 refers to two types of temporal scalability modes. In Type I, only the object is enhanced temporally. In Type II, the whole frame is enhanced [2], [8]. Figure 3-20 shows the two enhancement types in temporal concealment that MPEG-4 supports. In this figure, the gray area denotes the portion of the frame that is enhanced.

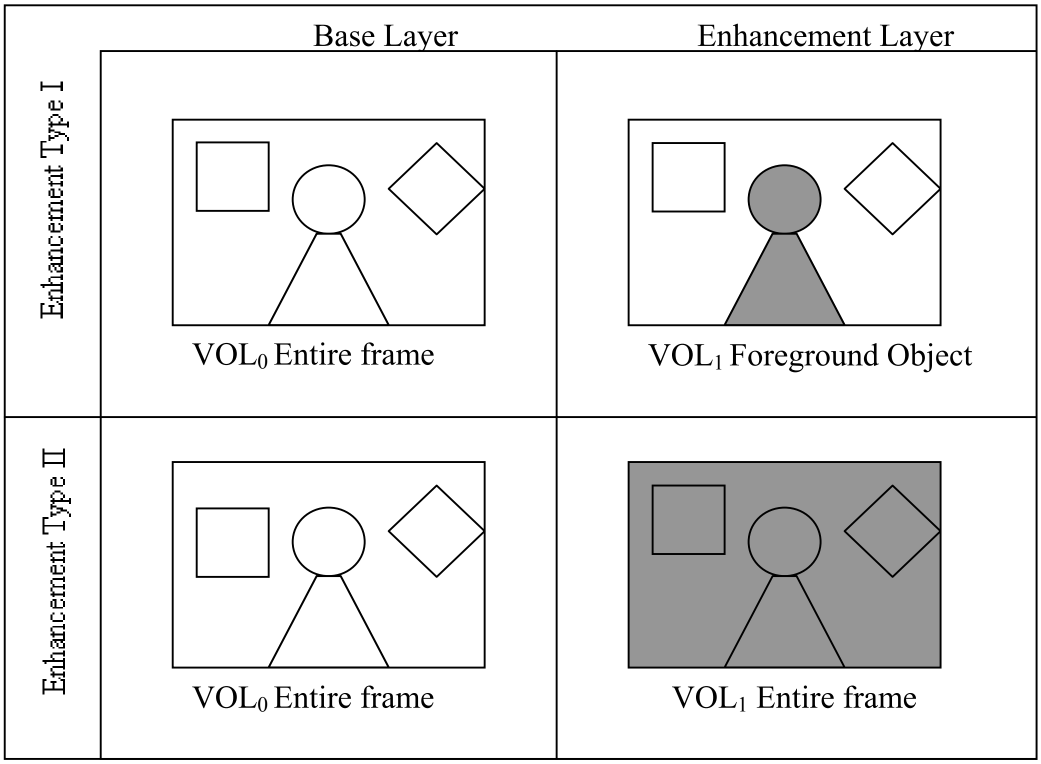


Figure 3-20. Types of Temporal Scalability in MPEG-4, After Ref. [8].

G. ISO/IEC 14496(MPEG4) VIDEO REFERENCE SOFTWARE

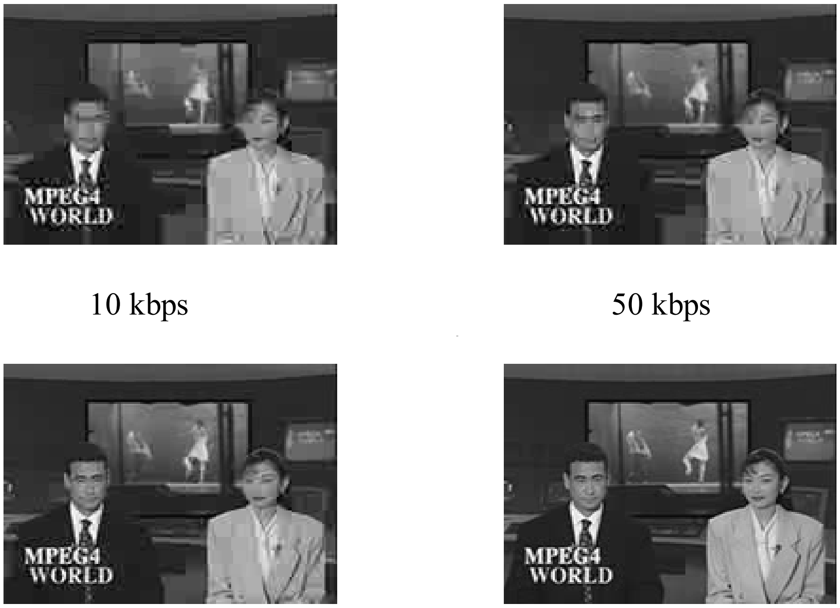
In the previous sections of this chapter, algorithms that are part of the standard were described. In this work, a Matlab codec was constructed in order to be able to encode a sequence of frames using most of these algorithms. However, due to the complexity of the standard, not all of the above algorithms were employed. For example, the Matlab codec, even though it supports I-, P-, B-frames and object based encoding, does not use Huffman coding to variable length code the result and produce a bit stream, but stays at the symbol level.

Additionally, the codec does not contain any form of rate control and produces variable bit-rate results. The compression is done according to a specified quantization parameter, which is fixed for the entire sequence. Furthermore, only gray scale sequences were used in order to keep the programming complexity and execution load within reasonable levels.

All of the above significantly affected the performance of the Matlab codec since it differed somewhat from the real MPEG-4 encoder. In order to provide a feel for the results that a real MPEG-4 encoder would produce, the reference software for the MPEG-4 verification model was used. The software is written in C++ and was built originally at Microsoft during the experimental phase of the standard to support the verification model. It runs under Windows and the commands are executed from the DOS command prompt. The user has to manually specify the encoding parameters in a parameter text file. In this section, some results from the coding of various sequences with this encoder are presented. In Appendix A, a manual for the software appears.

Specifically, three sequences were used in order to test the software. The first was the “Foreman” sequence showing a speaker in front of a non-uniform but stationary background. The second was the “Car Phone” sequence of a speaker inside a car (moving background) and the third sequence was “News” showing two speakers with a ballet moving in the background. From these three, only the results from sequence “News” are shown in this section while results from the other sequences are displayed in Appendix B.

The “News” sequence contains 50 QCIF frames (144×176 resolution) recorded at 30 frames per second. Figure 3-21 shows the results of coding the sequence at bit-rates of 10, 50, 100 and 300 kbps, and Figure 3-22 shows the measured performance as mean PSNR. Figures 3-23 and 3-24 show the compression efficiency achieved for this sequence.



10 kbps

50 kbps

100 kbps

300 kbps

Figure 3-21. Frame 5 from the Video Sequence “News” at Various Bit-Rates.

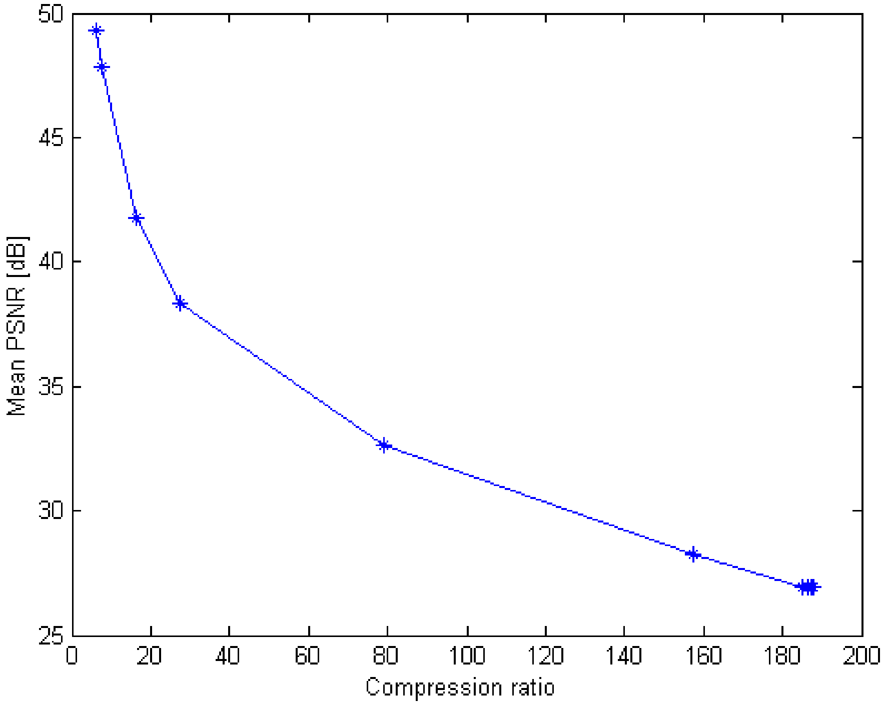


Figure 3-22. Rate-Distortion Curve for “News”.

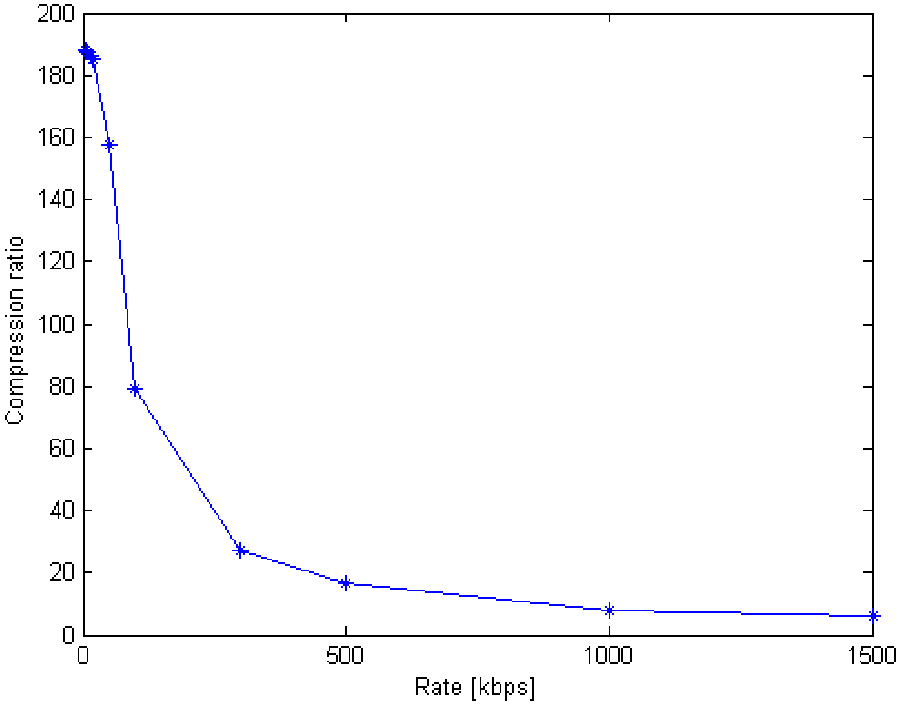


Figure 3-23. Compression Ratio versus Bit-Rate for “News”.

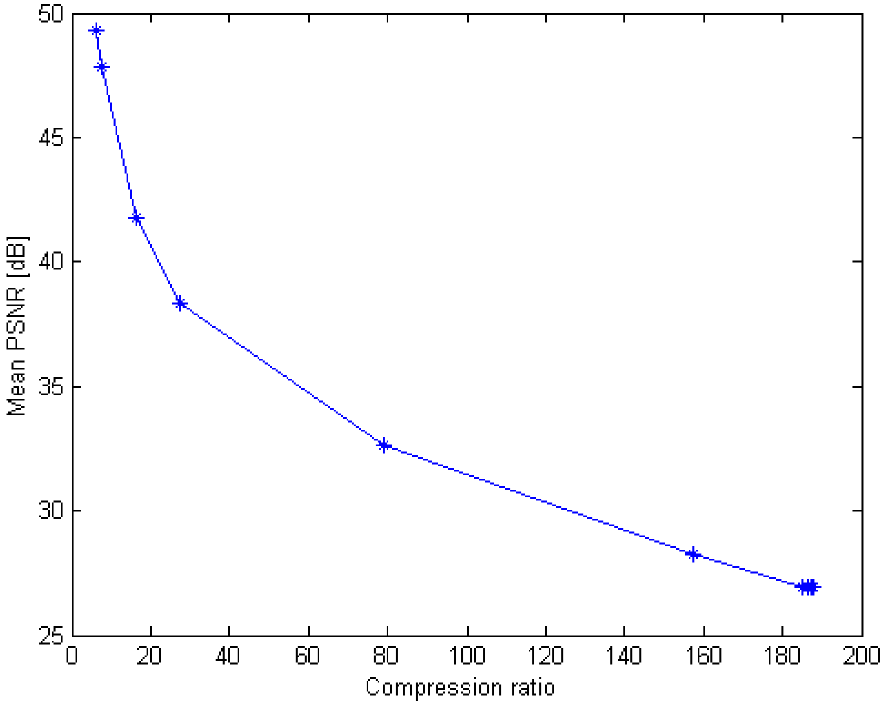


Figure 3-24. PSNR versus Compression for "News".

H. SUMMARY

In this chapter, the visual part of the ISO/IEC 14496 (MPEG-4) video coding standard was presented. Functions and the hierarchy along with several algorithms that are part of the visual portion of the standard were described. These tools were implemented in a codec in Matlab that is able to encode video sequences at a variety of compression rates. Results of coding some test video sequences using the reference software were also presented.

The next chapter will present tools of the standard that are designed to improve robustness in error prone environments. The effect of these tools and the performance in the presence of errors will be studied. Furthermore, error concealment techniques and their performance will be discussed.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ROBUST TRANSMISSION OF MPEG-4 VIDEO

The importance of networked video applications within the context of future naval information systems along with the growing commercial demand for mobile communications makes the issue of transmitting reliable video over error-prone environments, such as a wireless channels, a topic of considerable interest.

The impact of residual errors varies from application to application. In voice applications, residual errors result in speech quality degradation, which can be designed to be within the range of tolerance [11]. In contrast, applications such as image transfers need error protection since errors lead to lost blocks. Furthermore, due to the predictive nature of a video encoding algorithm, residual errors in the compressed video files may corrupt an entire frame or a sequence of frames leading to unacceptable quality. In Figure 4-1, a video transmission scheme is depicted. Raw video enters the video encoder and the resulting compressed bit stream is transmitted over the medium. The decoder receives the bit stream containing errors and decodes it to produce the reconstructed video. Quality degradation of the reconstructed video is due to the lossy compression in the encoder and the errors introduced by the channel.

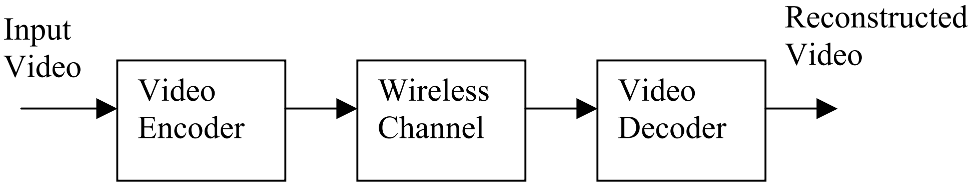


Figure 4-1. Video Transmission Scheme.

In this chapter, the effect of errors in video produced in an error-prone environment, such as a wireless channel, is studied. Section A describes the model used to simulate the wireless channel. Such channels can cause severe degradation because effects such as *shadowing, multipath loss, large and small scale fading* effects can introduce consecutive bit errors or *burst errors*. Error resilience and error concealment techniques help mitigate this effect of noisy channels. Error resilience tools described in the MPEG-4 standard are discussed in Section B and error concealment techniques used in the decoder are presented in Section C.

A. WIRELESS CHANNEL MODEL

The mobile radio channels are hostile media for video transmission. Beside absorption, the propagation of electromagnetic waves is influenced by the three basic mechanisms of reflection, diffraction and scattering. In conjunction with the mobility of the transmitter and the receiver, obstacles such as hills and buildings along with weather conditions can cause consecutive errors (bursts) leading to unacceptable quality of decoded video files. The bursty nature of these channels is captured by a two-stage Markov channel first introduced by Gilbert and generalized later by Elliot [18], [19].

1. Gilbert Channel Model

The physical channel is modeled as a two-state Markov chain as shown in Figure 4-2. In the "bad" state, errors occur with probability P_B while in the "good" state, errors occur with probability P_G. The channel is completely described by the channel transition matrix

$$P = \begin{bmatrix} P_{BB} & P_{BG} \\ P_{GB} & P_{GG} \end{bmatrix} \tag{4.1}$$

where P_{GG} and P_{BB} are the probability that the channel remains in the same state as before while P_{GB} and P_{BG} are the transition probabilities that the channel changes from a given state to the other state. The steady state probability that the channel is in the bad state is defined as:

$$\varepsilon = \frac{P_{GB}}{P_{BG} + P_{GB}} \tag{4.2}$$

which essentially characterizes the bursty nature of the channel. The average bit error rate is given by the following formula:

$$P_e = \varepsilon P_B + (1 - \varepsilon) P_G \tag{4.3}$$

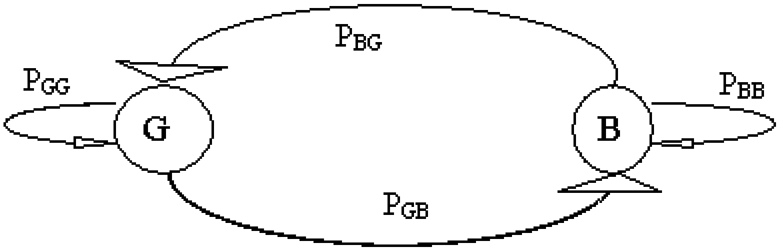


Figure 4-2. A Two State Markov Model Representing the Basis for Gilbert-Elliot Channel.

The average burst length, defined as the number of time units the channel stays in the bad state, is a geometric random variable with a mean of $1/P_{GB}$ while the times the channel is in the good state is a geometric random variable with a mean of $1/P_{BG}$. The values of ε used in the simulations used reported here are 10^{-1}, 10^{-2} and 10^{-3} .

2. Error Correction Codes

In erroneous media such as a fading channel, error correction techniques are used to improve signal quality by reducing the residual errors. Error correcting schemes use forward error correction codes (FEC) or automatic repeat-request (ARQ) techniques or a combination of both. The former is not as robust as the latter since the decoding capability of a FEC is constrained by the nature of the code while the latter two guarantee error free delivery. The drawback is that an ARQ scheme can introduce delays that may be unacceptable for real-time applications. Thus, only the FEC was considered in this thesis. Specifically, a convolutional error correction code was considered.

Convolutional error correction codes work by adding redundant bits in the bit stream. A convolutional encoder converts a *k*-bit data stream into an *n*-bit codeword, thus adds *n-k* redundant bits. The rate of the encoder is defined as *r* = *k/n*. Figure 4-3 shows a rate ½ convolutional encoder using a linear shift register mechanism [22].

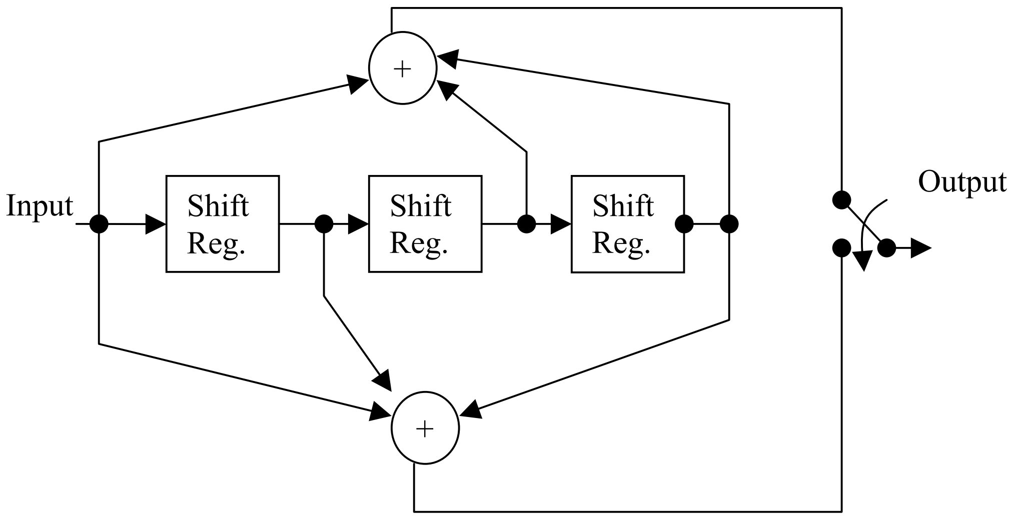


Figure 4-3. Rate ½ Convolutional Encoder with Constraint Length 4, After Ref. [22].

The constraint length is defined as the maximum number of shifts over which a single information bit can affect the output. If the encoder has more than one branch of shift registers, then the constraint length is the number of shift registers in the longer branch plus one. In Figure 4-3, the encoder has one branch with three shift registers, thus the constraint length is 4.

Free distance is another important figure of merit for convolutional codes. Free distance is the minimum Hamming distance between any two code sequences. During decoding using the Viterbi algorithm with hard decision decoding [12], the error correction capability of a convolutional code is directly related to free distance as:

$$t = \left\lfloor \frac{d_{free} - 1}{2} \right\rfloor \tag{4.4}$$

where d_{free} is the free distance and a convolutional code can correct up to t errors occurring within a time span corresponding to one constraint length. In the above equation $\lfloor \bullet \rfloor$ indicates rounding down.

B. ERROR RESILIENCE TOOLS IN MPEG-4

The previous section described the bursty nature of wireless channels along with one of the many different error correction techniques that can be used for minimizing the residual errors. FEC, however, may not be able to correct all errors in the bit stream especially when these occur in bursts. The design of the code with a minimum free distance limits the capability of the code to correct consecutive errors as Equation (4.4) states [22]. In this section, coding tools that can produce a bit stream that is robust to transmission errors, such that an error will not adversely affect the decoder operation and lead to unacceptable distortion in the reconstructed video quality, are presented.

1. Resynchronization Markers

One of the main reasons for the sensitivity of a compressed video stream to transmission errors is that the video coder uses VLC codes to represent various symbols. The nature of the VLC codes makes the code susceptible to severe errors even in the presence of single bit errors. Single bit errors can cause the decoder to lose synchronization with the encoder, thereby making it unable to decode the rest of the bit stream or causing a loss of the ability to correctly identify the precise location in the frame where the data belongs [12].

In the typical DCT block based coder with motion compensation, errors can result in syntax or semantic violations [10]:

- Not valid codewords (syntax violation)
- Motion vectors are out of range (semantic violation)
- DCT coefficients are out of range (semantic violation)
- The number of coefficients inside a block exceeds 64 (semantic violation).

The decoder checks for the above conditions and when one of them occurs it then flags an error. One approach that deals with this problem is the insertion in the bit stream of uniquely identifiable markers (resynchronization markers) at various locations. This marker is a codeword that cannot be produced from any combination of the video algorithm's table of VLC codewords. Upon detecting an error, the decoder hunts for the next resynchronization marker to gain synchronization with the encoder, thus localizing an error between two resynchronization markers.

In spite of this, erroneous codewords are occasionally processed by the video decoder before a syntactic or semantic violation occurs. Thus, the distance between error location and error detection can vary significantly as shown in Figure 4-4. Therefore, the data between these two markers must be discarded since it is assumed to be incorrect.

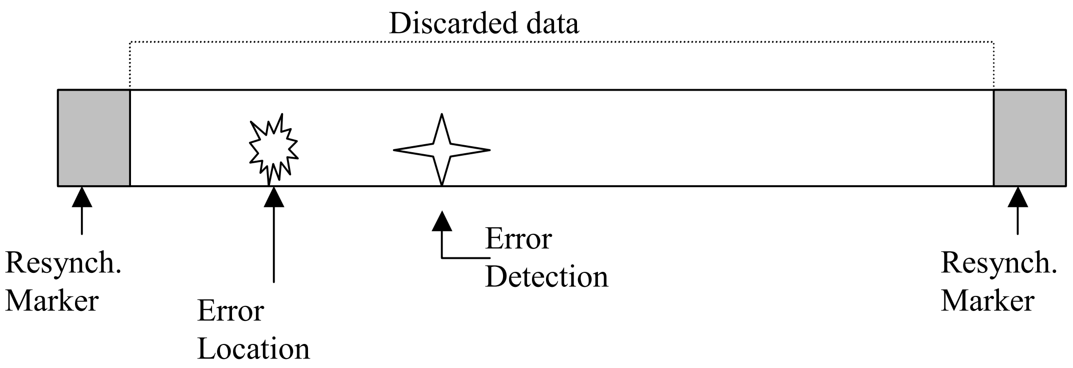
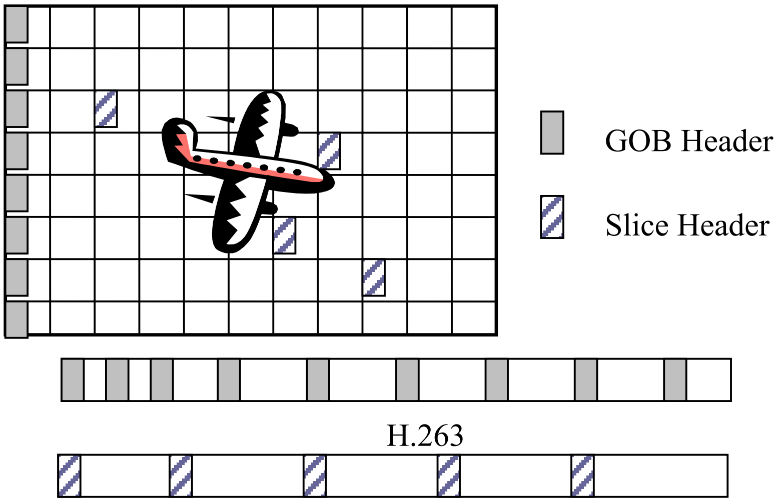


Figure 4-4. Actual and Detectable Error Location in a VLC Bitstream, After Ref. [12].

Previous approaches, such as those used in H.261, H.263 and MPEG-2, logically partition each of the images to be encoded into rows of macroblocks (MBs) called group of blocks (GOB). These GOBs correspond to a horizontal row of MBs for QCIF images, or in the case of CIF frames, two GOBs exist in one macroblock row. Resynchronization markers are inserted at the start of each MB row. Thus, a detected error in one row of pixels in the image would cause the effect of a missing MB row in the reconstructed frame. MPEG-4 adopts a different approach. Instead of inserting resynchronization markers at the start of each row, the encoder has the capability to divide the stream into video packets where each is made up of an integer number of consecutive MBs. These MBs can span several numbers of macroblock rows within an image and can even include partial rows. The idea is to insert markers after a certain number of bits. This method, called the *slice structure mode*, was first adopted for H.263 [23].

Figure 4-5 shows the position of the resynchronization markers in a MPEG-4 bit stream and in a stream generated by a MPEG-2 or baseline H.263 encoder. When significant activity is present, the packets will contain information corresponding to fewer macroblocks while when activity is low, the corresponding number of macroblocks in the slice can be very large. For example, in the case of an I-frame, where each macroblock is intra coded there can be more slices in a frame than for a P-frame. Thus, an error will only affect a small portion of the frame. This is desirable because I-frames are the anchor frames used for motion compensation for other frames.



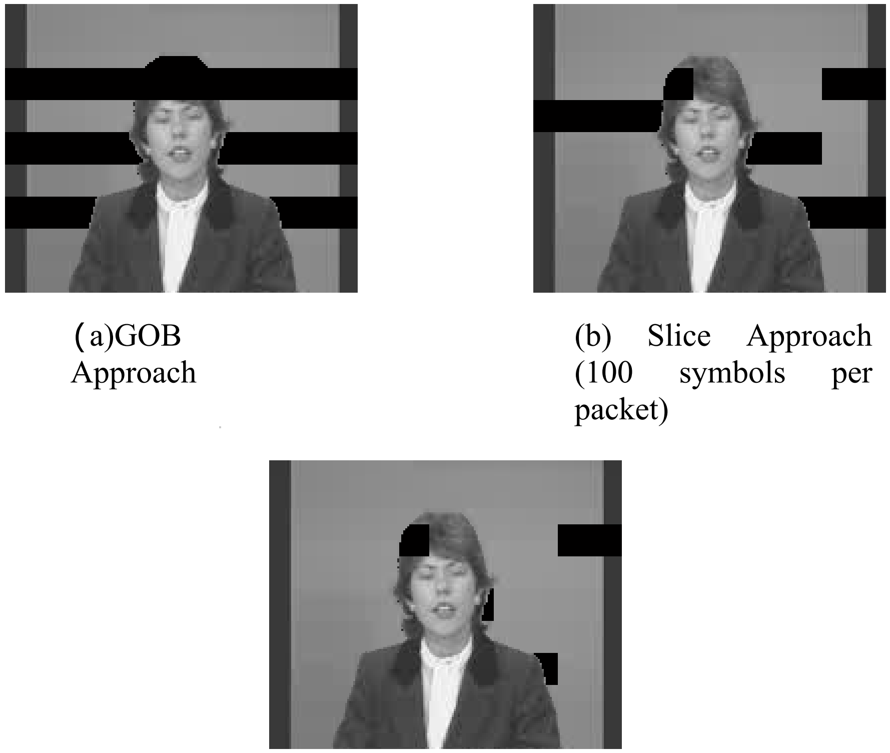
H.263

Slice Structure Mode used in MPEG-4

Figure 4-5. Slice Structure Mode Adopted in MPEG-4 versus GOB in Baseline H.263, After Ref. [2].

In the same manner, errors in P- and B-frames will have a considerable effect on the quality of the reconstructed video but error concealment techniques, described in the next section of this chapter, could be used to mitigate this loss. Considering the case of an I-frame, a macroblock lying in the background needs fewer symbols for coding than a macroblock in the foreground, where more of the high frequencies occur. Thus, the effect of an error will be smaller in the foreground object since the packets contain a small number of macroblocks.

Figure 4-6 shows the effect of errors in a reconstructed frame of sequence "Claire" when using the GOB and the slice structure in the insertion of the markers. The effect of the same errors is much smaller in the latter case. Packets of 50 and 100 symbols were used in the encoding of these sequences. An additional advantage of MPEG-4 content-based encoding approach is shown in this figure. Since the encoder constructs essentially two bit steams, one for the foreground and one for the background, missing blocks that are produced due to errors in one VOP do not propagate into the other. This is due to the construction method of the frame using Equation (3.2). This will not be the case if frame-based encoding was used.



(a)GOB Approach

(b) Slice Approach (100 symbols per packet)

(c) Slice Approach (50 symbols per packet)

Figure 4-6. Effect of Discarding Data between Two Resynchronization Markers when Errors Occur in the Bit Stream.

A resynchronization marker is placed at the start of the header of each video packet. The header contains information necessary to restart the decoding process, including the address of the first MB contained in the packet and the Q_P for the first MB. These are followed by a single bit called the header extension code (HEC). If this is set to 1, the information specified in the VOP header, such as timing information, temporal reference and VOP prediction type, is duplicated in this packet header. HEC enables the decoder to correctly utilize the data contained in the current packet even if the packet containing the VOP header is lost or corrupted.

2. Reversible Variable Length Codes

From the above discussion, the length of the video packet plays a critical role in the quality of the reconstructed video as all the data inside a packet must be discarded. Ideally, small packet size would limit the amount of wasted data in the event of an error causing loss of synchronization. This, however, would decrease coding efficiency since a large number of resynchronization markers would have to be added. An error localization mechanism that would efficiently localize the error would minimize the amount of discarded data when an error occurs. In this connection, the regular VLC codes do not have any features that would help in error localization as discussed in the previous section.

In the MPEG-4 standard, the use of reversible variable length codes (RVLC), which have the property of being uniquely identifiable in both forward and backward directions, is proposed. In [13], a set of RVLCs called Exp-Golomb Rice codes is described along with the algorithm used to generate them. An example is shown in Table 4-1. These codes match well the characteristics of a run-length code used to code DCT data and provide an efficient method to code and maintain good coding efficiency. [13]

	Non-Reversible Variable Length Exp-Golomb Code k = 1		Reversible Variable Length Exp-Golomb Code k = 1	
	Prefix	Suffix	Prefix	Suffix
0	0	0	0	0
1	0	1	0	1
2	100	0	101	0
3	100	1	101	1
4	101	0	111	0
5	101	1	111	1
6	11000	0	10001	0
7	11000	1	10001	1
8	11001	0	10011	0
9	11001	1	10011	1
10	11010	0	11001	0
11	11010	1	11001	1
etc.				

Table 4-1. Non Reversible and Reversible Exp-Golomb Codes, After Ref. [13].

In Exp-Golomb Rice codes, the number of codewords of a given length grows exponentially with length. One way to construct reversible codes from non-reversible Exp-Golomb Rice codes is by substituting the prefix portion of the code with a prefix starting and ending with one. All odd-indexed bits in the prefix with the exception of the first and the last one are required to be zero. For example the codeword that represents number 6, in Table 4-1, has a prefix of 11000 and a suffix 0 when a non-reversible Exp Golomb Rice code is used. In order to construct a codeword that is part of a reversible code the prefix is replaced by 10001 while the suffix remains the same. In order to distinguish between codewords with the same length the even-indexed bits of the prefixes can vary arbitrarily allowing $2^{(l-1)/2}$ possible prefixes of length l, where l is odd. The prefix is concatenated by 2^k possible suffixes of length k [13]. When $k = 1$, as Table 4-1 shows, the resulting number of possible codewords of length L is $2^{L/2}$. Basically, codewords with the same codeword length are grouped by having equivalent odd-indexed bits while the even-indexed bits distinguish each individual codeword within the same group [26].

Single bit errors on these codewords can be classified as propagating or non-propagating bit errors, depending on whether bit errors occur in odd-indexed or even–indexed bits, respectively. An error that occurs in an odd-index (i.e. 1st, 3rd, 5th, etc.) bit of a codeword will result in a non-valid codeword while errors that occur in even-indexed (i.e. 2nd, 4th, etc.) bits of the codeword will cause an error codeword, although a valid one. Even-index bit errors will not result in a loss of synchronization whereas odd-index bits errors will. Error propagation in these codes could be minimized by bi-directional decoding. Since the codes are uniquely identifiable both in forward and backward direction, the decoder can start decoding in the backward direction to retrieve as much information as possible.

Figure 4-7 illustrates an example of the ability of bi-directional decoding to minimize the propagation of errors and the difference when errors occur in odd and even indexed bits. In this example, codewords from Table 4-1 were used. X denotes decoded codewords not defined in the VLC table or illegal codewords, and therefore, further decoding cannot occur.

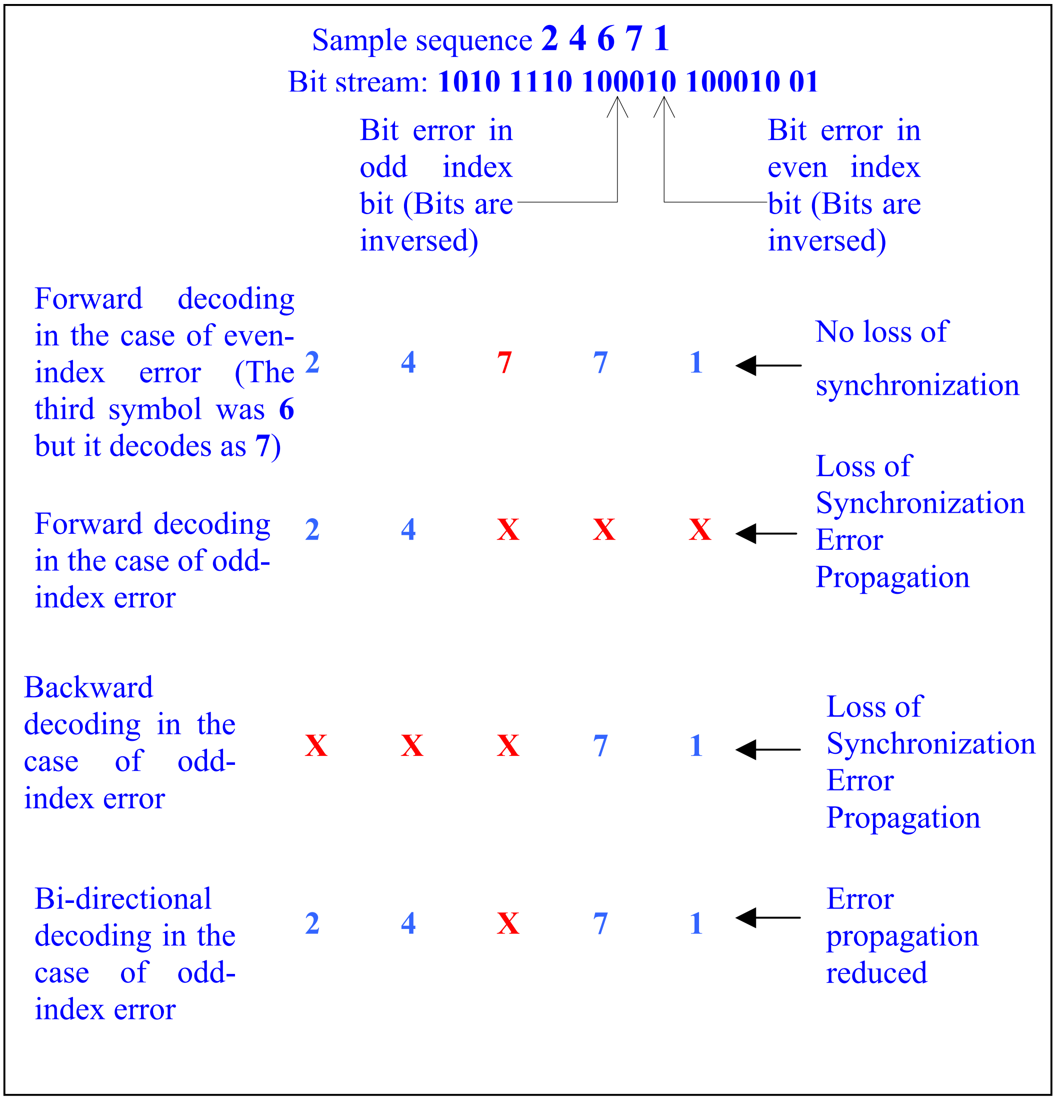

Sample sequence 2 4 6 7 1
Bit stream: 1010 1110 100010 100010 01
Bit error in odd index bit (Bits are inversed)
Bit error in even index bit (Bits are inversed)
Forward decoding in the case of even-index error (The third symbol was 6 but it decodes as 7)
2 4 7 7 1
No loss of synchronization
Forward decoding in the case of odd-index error
2 4 X X X
Loss of Synchronization Error Propagation
Backward decoding in the case of odd-index error
X X X 7 1
Loss of Synchronization Error Propagation
Bi-directional decoding in the case of odd-index error
2 4 X 7 1
Error propagation reduced

Figure 4-7. Illustration of Error Propagation Reduction and Difference between Odd and Even-Indexed Bit Errors in Reversible Exp-Golomb Codes of Table 4-1, After Ref. [26].

The Matlab encoder developed in this thesis does not implement the reversible variable length codes since it operates at the symbol level as previously mentioned. In order to simulate the effect of error localization and the minimization of error propagation that RVLCs introduce, symbol errors that occur in the vector containing the run-length coded DCT coefficients are separated into errors that will cause loss of synchronization and errors that will not. When a symbol error occurs in the run part of the sequence, the error causes loss of synchronization and the corresponding macroblock is declared missing, thus limiting the error propagation to within the macroblock boundaries.

3. Data Partitioning

Another tool enhancing the robustness of the video file inside an erroneous media is the *data-partitioning* mode. In this mode, motion and texture information are separated inside the bit stream using another kind of marker called motion boundary markers (MBM). The MBM is computed from the motion VLC tables using a search program such that this marker word is Hamming distance 1 from any possible valid combination of the VLC tables [13]. Thus, this word is uniquely decodable from the motion VLC tables.

From the VLC tables in MPEG-4, the MBM is a 17-bit word whose value is 1 1111 0000 0000 0001 [8], [13]. When an error is detected in the motion part and no error is detected in the texture part, the decoder can use the texture information to decode the corresponding macroblocks using, for example, zero motion vectors. Similarly, when an error is detected in the texture information and the motion information is available, the decoder can use the texture information of the corresponding macroblock in the previous frame and the decoded motion vectors of the macroblock in the current frame to reproduce an estimation of the macroblock without losing the information of the entire region. Some of these techniques are called error concealment techniques and are discussed in the following section. In order to simulate the data partitioning mode, motion and texture information of the macroblocks in the compressed stream are separated prior to sending over the channel, i.e., an error that can cause error propagation in the texture part of one MB does not necessarily affect the motion part at the same time.

It is obvious that data partitioning when combined with RVLCs can minimize the effect of errors in the bit stream. Additionally, the use of fixed bit video packets as described earlier in this section leads to an error resilient bit stream.

4. Bit Stream Syntax

Based on the above, the syntax for a video packet is shown in Figure 4-8 with data partitioning enabled. The numeric subscripts indicate a MB number. In Figure 4-8, C_{MB} is a one-bit field indicating whether the MB is coded or not. MB_{TYPE} is a variable length code indicating the type of the MB (i.e., intra or inter). D_Q is an optional 2-bit fixed length field to indicate the incremental modification of the quantization value with respect to the previous MB. B_{MB} is a VLC indicating which of the blocks of the MB is processed.

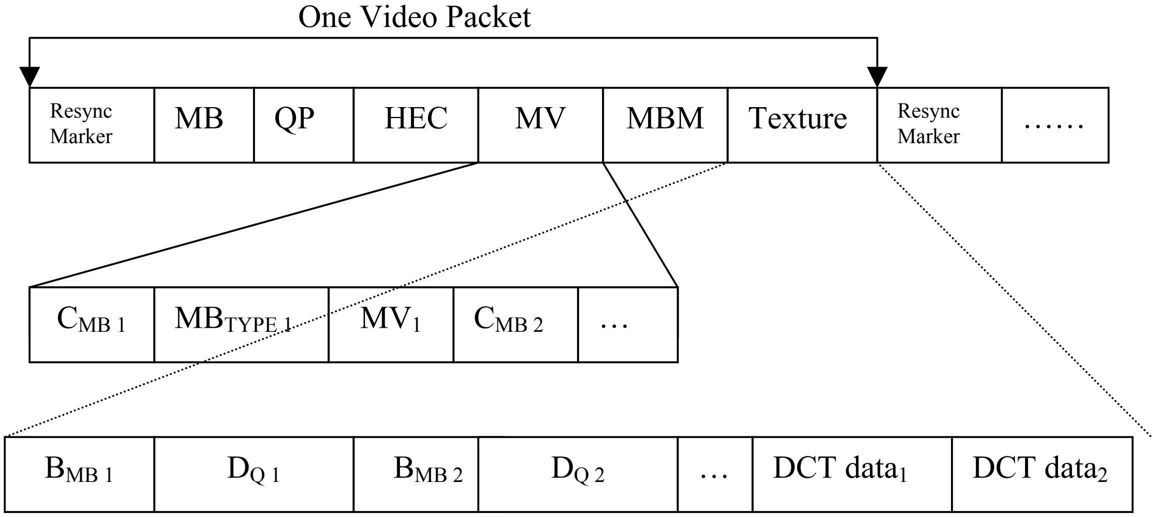


Figure 4-8. Bit Stream Organization inside an MPEG-4 Video Packet, After Ref. [13].

C. ERROR CONCEALMENT

As discussed in Section A, even the use of error control mechanisms, when transmitting video data in extremely erroneous media such as a wireless channel, does not guarantee reliable transmission. Utilizing error resilience tools, described in Section B, enhances but does not guarantee the quality of the reconstructed video, which can be poor since whole packets may be missing, or in the best scenario, whole macroblocks may not be decoded leading to annoying artifacts.

The goal of error concealment is to estimate and fill the missing macroblocks. The underlying idea is that there are still enough redundancies in the sequence to be exploited. In particular, in I-frames, it is possible to have a lost macroblock surrounded by intact macroblocks that are used to interpolate the missing data. In I-frames, the video packet contains fewer numbers of MBs. In P- and B-frames, it is possible to have entire rows of macroblocks missing. In this case, spatial interpolation, to be described later in this section, will not yield acceptable reconstructions. However, the motion vectors of the surrounding regions can be used to estimate the lost vectors, and the damaged region can be reconstructed via temporal interpolation. These techniques are called s*patial* and *temporal concealment,* respectively, and they fall into the category of post-processing techniques described in [15] and [16]. They are part of the decoder and interaction with the encoder is not necessary.

1. Spatial Error Concealment

Let f be an $M \times N$ frame of a decoded MPEG sequence as received in the output of the channel, possibly containing missing MBs. Each frame consists of Q macroblocks that have 16×16 pixels. Let MB_j be the lexicographic ordering of the j^{th} MB in f. Supposing that the p^{th} MB is missing, the goal is to estimate the MB_p from the surrounding macroblocks.

Let $\hat{f}_{m,n}$ denote the reconstructed value of the sample at the m^{th} row and n^{th} column of MB_p and let J_p denote the set of indices of the pixels belonging to MB_p:

$$J_p = \left\{ (m,n) \mid f_{m,n} \in MB_p \right\} \tag{4.5}$$

In the spatial concealment method, every pixel in MB_p is reconstructed by spatially averaging the values of its four closest neighbors as shown in Figure 4-9 and given by:

$$\hat{f}_{m,n} = \lambda \left[\mu_1 f_{m,-1} + (1-\mu_1) f_{m,16} \right] + (1-\lambda) \left[(1-\mu_2) f_{-1,n} + \mu_2 f_{16,n} \right] \tag{4.6}$$

where $f_{m,-1}$ is the closest element in the macroblock to the left of f_p, $f_{m,16}$ is the closest element of the macroblock to the right of f_p, $f_{1,n}$ is the closest element in the macroblock above of f_p and $f_{16,n}$ is the closest element in the macroblock below f_p, μ_1 is the distance in pixels between the lost pixel and the closest pixel lying to the macroblock to the left of f_p, μ_2 is the distance in pixels between the lost pixel and the closest pixel lying in the macroblock above to f_p, and λ is a coefficient weighing the contribution of macroblocks on either side, above or below.

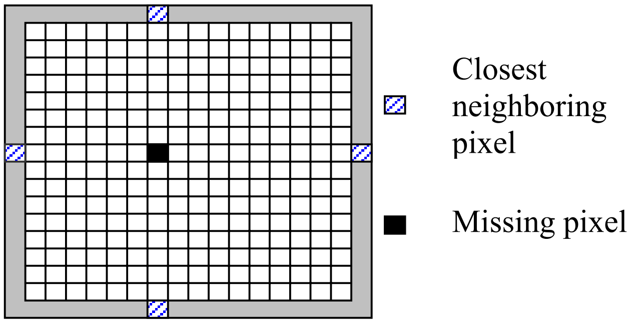
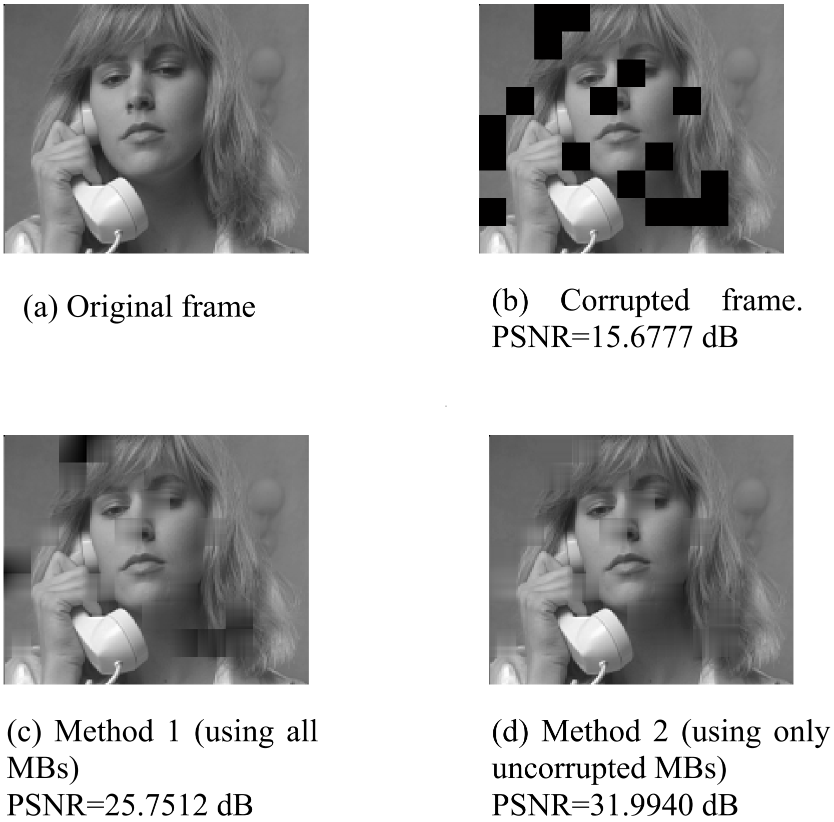


Figure 4-9. Spatial Averaging Method for Error Concealment. Gray Area is the Reference Area of the Neighboring Macroblocks, After Ref. [17].

Missing macroblocks are processed in a raster scan order. Equation (4.6) takes into account all the macroblocks that are neighbors to the macroblock in question. In a high bit error rate environment, however, adjacent macroblocks or even whole rows may be missing. In this case, only the uncorrupted neighboring macroblocks should be taken into account for reconstruction.

The method does not attempt to accurately estimate the missing macroblock: rather to hide it from the user. The reconstructed macroblock appears to be blurred due to the averaging nature of the algorithm, which is especially visible when areas with critical details are missing. Even though the gain measured in PSNR is significant, the reconstructed frame is not lacking artifacts as shown in Figure 4-10. Figure 4-10 shows the result of the spatial error concealment technique using all neighboring macroblocks (Method 1) and only uncorrupted neighboring macroblocks (Method 2) for a frame of sequence "Suzie" where 20 % of the MBs are missing. Note that when there are adjacent MBs missing, Method 1 does not perform as well as Method 2. Additionally, when the missing MB contains an area with critical details, e.g., nose or eye, both of these methods produce annoying artifacts. Nevertheless, the visual quality is much improved when error concealment is used.



(a) Original frame

(b) Corrupted frame. PSNR=15.6777 dB

(c) Method 1 (using all MBs)
PSNR=25.7512 dB

(d) Method 2 (using only uncorrupted MBs)
PSNR=31.9940 dB

Figure 4-10. Spatial Error Concealment Results in a Frame from Sequence “Suzie” where 20% of the MBs are Corrupted.

To measure the performance of this technique, a frame from sequence “Suzie” was used. MBs were declared missing randomly in the range of 1-30 % of the total number of MBs in the frame (a total of 99 MBs for this QCIF frame). Method 1 and Method 2 were simulated separately. When the percentage of missing macroblocks is high, the probability to have adjacent missing macroblocks is also high. So the additional gain in PSNR that the second method adds is proportional to the number of missing MBs. Figure 4-11 illustrates the advantage of using only the uncorrupted neighboring MBs for the reconstruction of the missing MB especially in cases where many macroblocks are missing. Each point in the graph is obtained by averaging 20 runs.

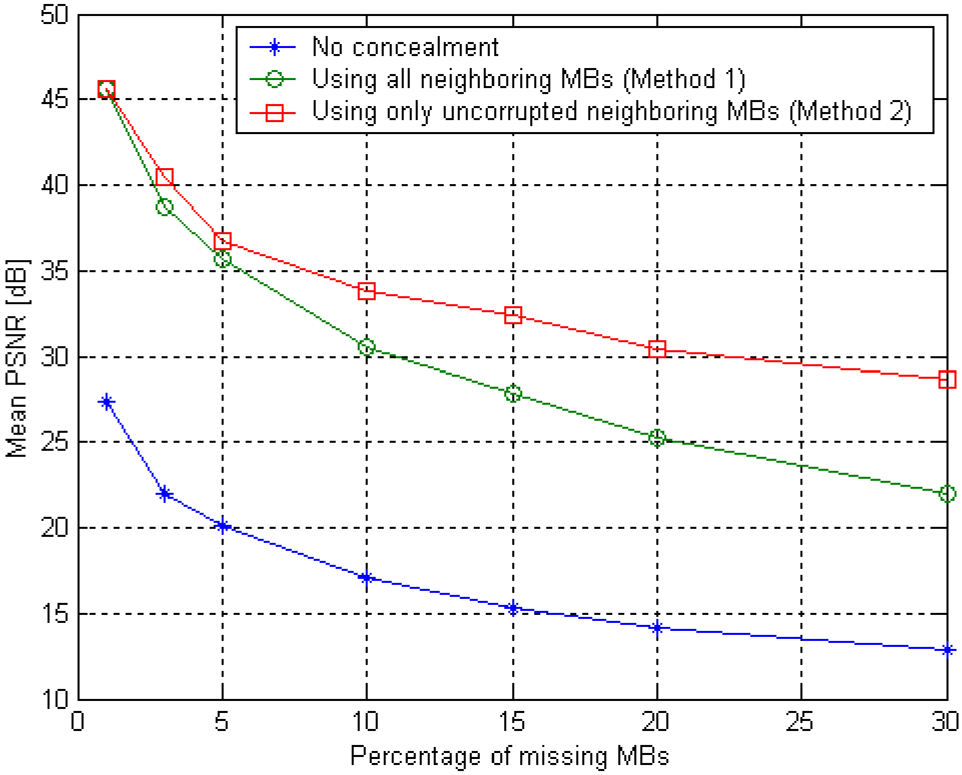


Figure 4-11. Spatial Error Concealment in a Frame from Sequence "Suzie".

From Figure 4-11, the difference in PSNR of the two methods is proportional to the number of missing MBs in the frame. On average, with respect to the case of no concealment, Method 1 yielded a PSNR gain of approximately 13.8434 dB while Method 2 enhanced the PSNR gain by approximately 17.0277 dB. When 30% of the MBs were missing, compared to Method 1, Method 2 produced a PSNR gain of 6.5748 dB.

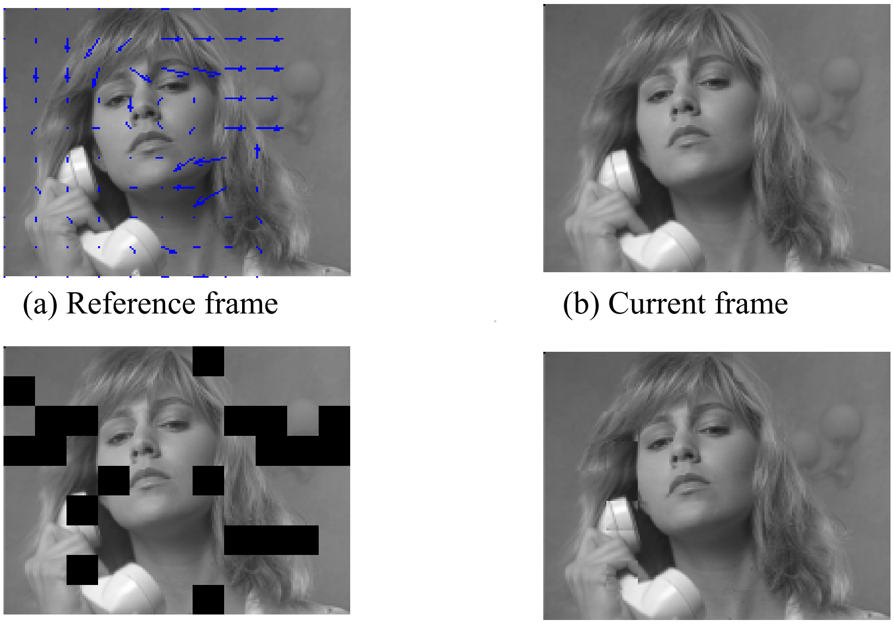
2. Temporal Error Concealment

Temporal error concealment technique exploits the temporal redundancy in the video sequences to conceal the missing MBs. The estimation of motion vectors can be performed by several operations [2].

When data partitioning mode is enabled during the encoding process, it is possible that the texture data of an MB is missing and motion vectors are available or vice versa. When only texture is missing, the decoder can perform motion compensation using the texture of the corresponding macroblock in the previous frame. When motion vectors are missing, they have to be estimated. By simply assuming zero motion vectors, a copy of the corresponding MBs from the previous frame is obtained. This can work well in sequences with low motion activity. However, this approach will not perform well with sequences with frequent scene changes or high motion activity.

Different methods to estimate missing motion vectors have been proposed [2]. The median or the average of motion vectors from spatially adjacent MBs can be used for this purpose. Using the motion vectors of the corresponding MBs in the previous frame can be beneficial also. Since, in this thesis, only low motion sequences were used, the first method of assuming zero motion vectors was utilized in the decoder.

Figure 4-12 shows the results of performing temporal concealment on a frame from sequence "Suzie". In Figure 4-12 (a), the reference frame with the motion vectors overlaid is shown while Figure 4-12 (b) shows the current frame. When 20% of the MBs are declared missing the PSNR is 13.7505 dB; when temporal error concealment is used, the PSNR improves to 32.8882 dB. Note that this method does not leave extensive visual artifacts like those of the spatial concealment method shown in Figure 4-10.



(a) Reference frame

(b) Current frame

(c) Current frame 20% of MBs are missing.

(d) Concealed current frame

Figure 4-12. Temporal Concealment in One Frame of Sequence "Suzie".

D. SIMULATION RESULTS

Three different gray scale video sequences were used for the simulations. In all cases, the activity is low and the background is uniform to cope with the restrictions of the edge detection technique discussed in Chapter III. From these sequences, only the first 13 frames were used to minimize the size of the input file. For the same reason, a quantization parameter of 24 was chosen. The frames used for the video sequences consisted of three P-frames every I-frame and two B-frames every P-frame, resulting in a total of two I-frames, three P-frames and eight B-frames:

I	**B**	**B**	**P**	**B**	**B**	**P**	**B**	**B**	**P**	**B**	**B**	**I**	Frame Type
1	2	3	4	5	6	7	8	9	10	11	12	13	Frame No

In this section, only the results for sequence “Claire” are included. In Appendix B, the same graphs are provided for sequences “Suzie” and “Miss America”.

Error concealment was implemented in the decoder. Both Methods 1 and 2 of spatial error concealment technique discussed in Section C were used. Additionally a hybrid scheme of spatial and temporal techniques was implemented in order to take advantage of the better performance of the temporal error concealment scheme illustrated in Figure 4-12. In this mechanism, the first frame of the sequence was concealed using Method 2 (only uncorrupted neighboring MBs were used in the reconstruction of the missing MB) of spatial error concealment while for the rest, temporal error concealment was used.

In order to simulate the use of RVLCs, the assumption of limited error propagation within the same MB where a detected error occurred was made. Thus, when an error was introduced in the odd indexed symbol of the run-length sequence of the code of the texture information, the corresponding macroblock was declared erroneous and was replaced with zero values. Errors in the motion vectors were assumed to be without error propagation. Note that the encoder used in this simulation study does not reach the bit level as a VLC code was not utilized.

Three channels with different steady state probability that the channel is in the bad state ε were used (0.1, 0.01, 0.001) and served as the erroneous media. The compressed file produced from the Matlab encoder was passed through the channel and the received file decoded. The results plotted are based on averaging results from five runs. A rate ½ convolutional encoder with constraint length 7 and free distance 10 was used as the error correction mechanism. This encoder corrects up to 4 errors occurring within a time span corresponding to one constraint length as given in Equation (4.4); the BER is significantly lower after error correction decoding. This improvement in BER when using a channel with ε = 0.01 is depicted in Figure 4-13. In the figure, the x axis denotes the channel BER before error correction decoding and the y-axis the residual BER.

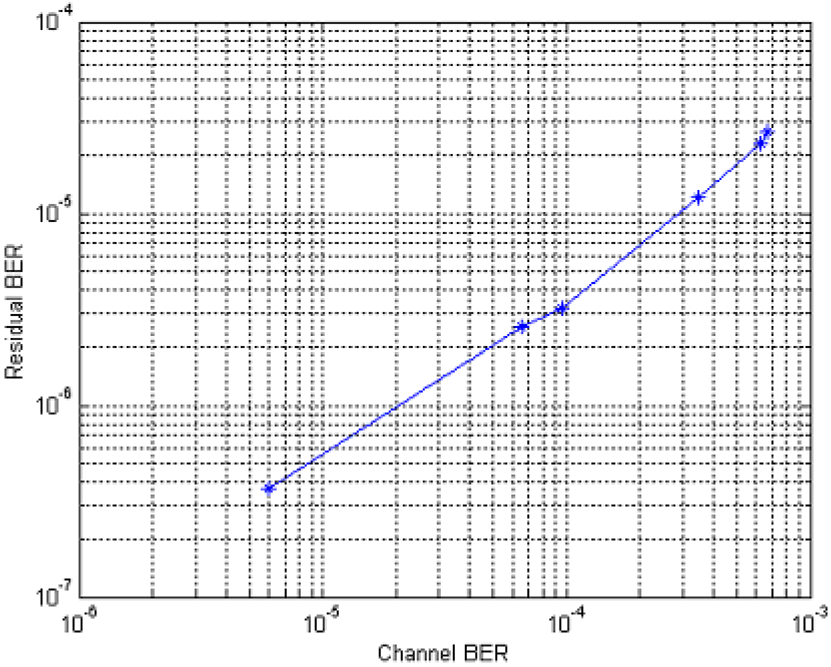
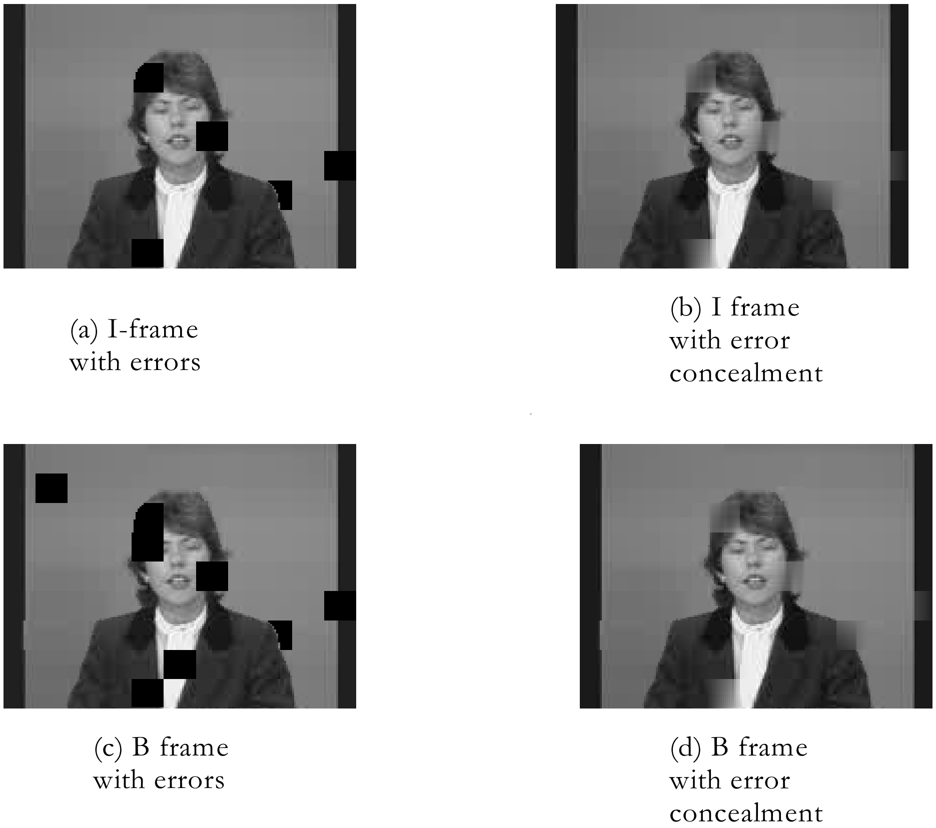


Figure 4-13. Gain in BER for Channel with ε = 0.01 due to FEC.

Figure 4-14 shows the results of the hybrid error concealment scheme in frame 1 and 2 of the “Claire” sequence (I-, B-frames) when the file was transmitted over a communication channel with BER 9.3×10^{-4}. Spatial error concealment was used to conceal the missing macroblocks in the first frame while temporal concealment was used for all the other frames. Missing MBs in the first frame are replaced by blurred MBs due to the nature of the spatial concealment algorithm. The second frame is an inter-frame using the first frame as a reference frame so the missing MBs of the first frame are propagated to the second. Additionally, the second frame has missing MBs due to errors that occurred in that frame (see the missing MB around the left eye of Claire). The temporal concealment method used for the second frame replaced the missing MBs with the corresponding MBs of the first frame after spatial concealment was performed. However, the macroblock covering the left eye of Claire was not missing in the first frame, so this method left no visual artifact for this MB since the motion is very low for this sequence. If spatial concealment was performed in all frames, this would not be the case since all missing MBs would have left visual artifacts after concealment.



(a) I-frame with errors

(b) I frame with error concealment

(c) B frame with errors

(d) B frame with error concealment

Figure 4-14. Illustration of Hybrid Error Concealment Scheme on Frames 1 and 2 (I-, B- frames) of Sequence "Claire" when Transmitted in a Channel with $\varepsilon = 0.01$ with Resulting Channel BER = 9.3×10^{-4}.

In Section B, the use of resynchronization markers was discussed and the slice structure mode adopted in the H.263+ and MPEG-4 video coding standards along with variable bit size packets used in previous standards was discussed. In order to show the improved performance that the slice structure mode gives in the presence of errors, three different approaches when dealing with the construction of the video packets were used. In the first approach, video packets were constructed containing one row of macroblocks (GOB) regardless of the symbol number used to code them (MPEG-2 and H.263 Baseline approach). In the second, the slice structure mode was simulated constructing packages containing 50 symbols and in the third approach packages with 100 symbols were simulated. When a detected error occurred, the whole packet was declared missing and replaced with zero values. The results depicted in Figures 4-15, 4-16 and 4-17 for the three channels clearly show that the when RVLC is used the same errors result in higher

PSNR. It is also obvious that the smaller the packet the better the performance since less information is discarded in the presence of errors that cause loss of synchronization.

Figures 4-18, 4-19, and 4-20 clearly show that the hybrid error concealment scheme that combines both spatial and temporal techniques performed better in all cases. In these tests, the effect of using RVLC was simulated.

In the simulations, different BER values were used for the bad and good states of the channel. The steady state probability that the channel is in the bad state affects the resulting BER of the channel given by Equation (4.3). For a BER of 5×10^{-1} for the bad channel and 5×10^{-4} for the good channel, the results are summarized in Table 4-2. It is clear that smaller ε resulted in lower overall bit error rates, which were further reduced with the use of the error correction mechanism.

Channel Condition ε	**BER before FEC**	**BER after FEC**	**Mean PSNR** **No Concealment**
0.1	3.14×10^{-3}	1.39×10^{-4}	17.846
0.01	3.47×10^{-4}	1.235×10^{-5}	25.419
0.001	6.75×10^{-5}	1.89×10^{-6}	34.181

Table 4-2. BER With and Without FEC Achieved for the Same P_b and P_g for Three Channel Conditions.

Lower bit error rates result in higher PSNR, so the steady state probability that the channel is in the bad state is directly related to the quality of the reconstructed video. Table 4-3 summarizes the probabilities of error for the bad (P_b) and good channel (P_g) that were used in order to achieve a PSNR of ~25 dB without concealment. The channel with a lower ε achieved the same performance with higher BER for the bad and good channels as a channel with a higher ε and lower BERs.

Channel Condition ε	P_g	P_b
0.1	5×10^{-3}	5×10^{-2}
0.01	5×10^{-4}	5×10^{-1}
0.001	10^{-2}	10^{-1}

Table 4-3. Probabilities of Errors for the Bad (P_b) and the Good Channel (P_g) Used in Order to Achieve a PSNR of 25 dB Under Three Channel Conditions.

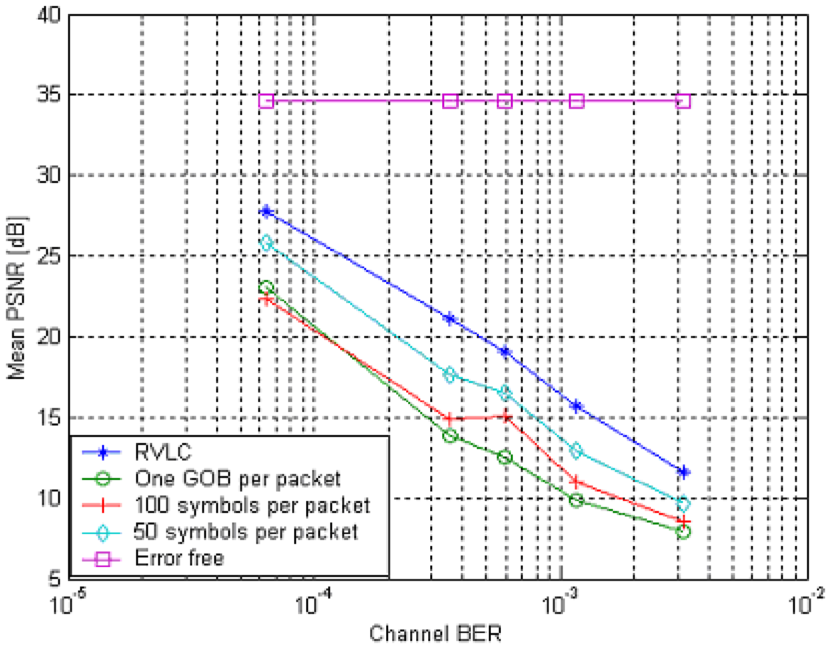


Figure 4-15. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.1. No Error Concealment.

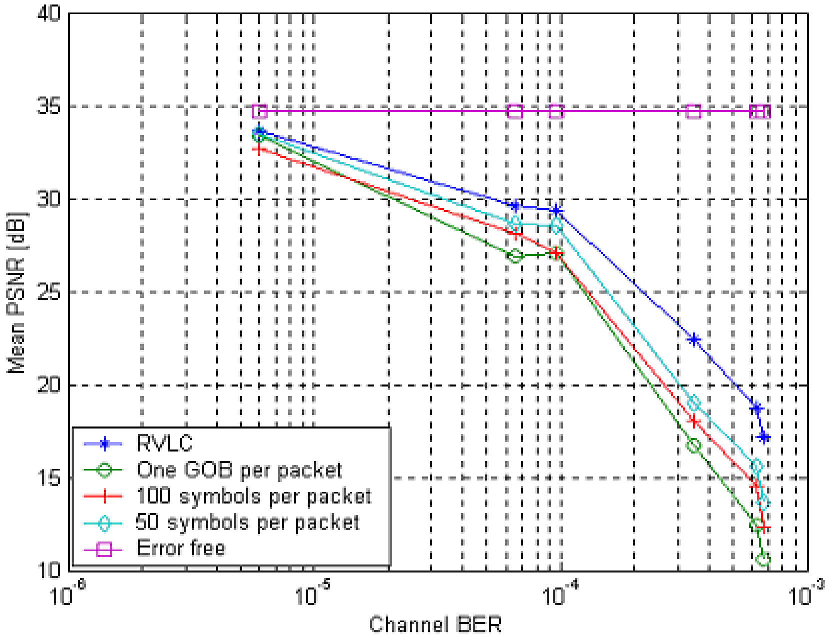


Figure 4-16. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.01. No Error Concealment.

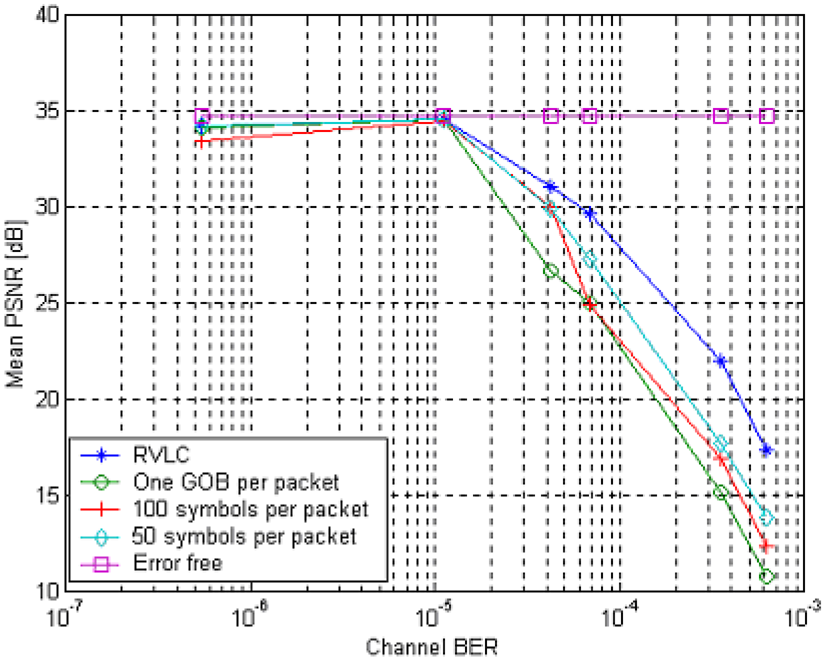


Figure 4-17. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.001. No Error Concealment.

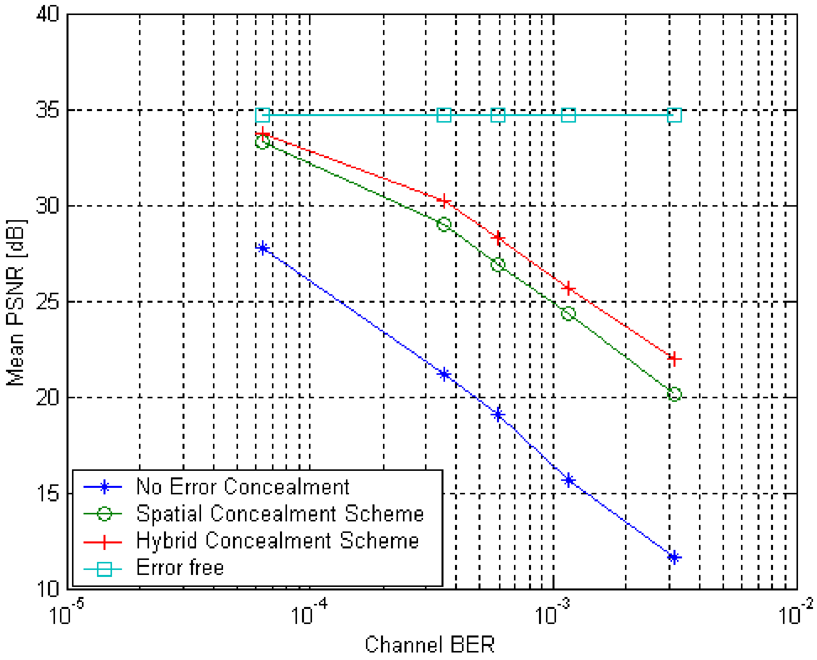


Figure 4-18. Different Error Concealment Schemes for a Video Sequence using RVLC through a Gilbert Channel with ε = 0.1

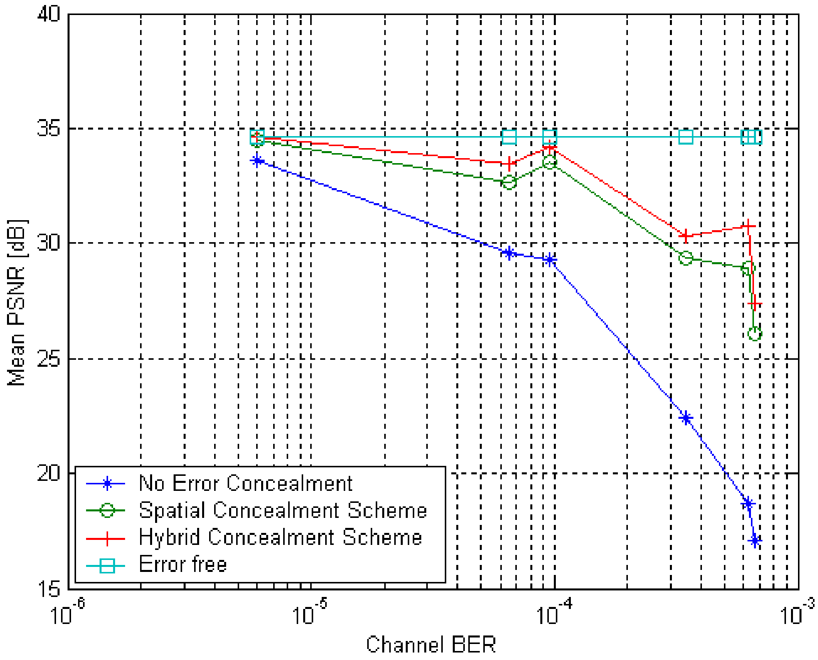


Figure 4-19. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.01.

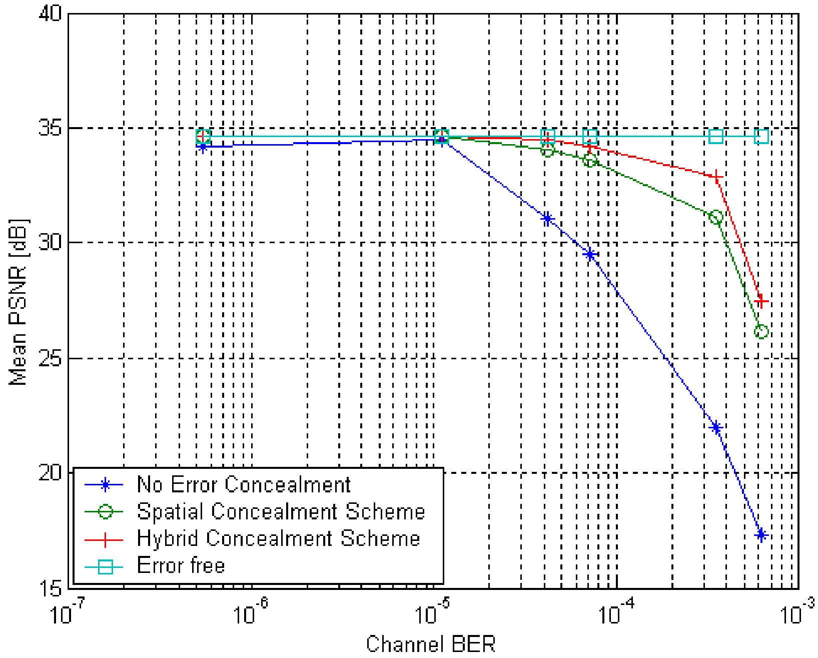


Figure 4-20. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.001.

E. SUMMARY

In this chapter, the issue of compressed video transmission over erroneous media was investigated. The MPEG-4 error resilience tools were described and the results of a series of simulations demonstrating the advantages of these tools were presented. These tools enhance the transmission performance of compressed video files over erroneous media. Two error concealment techniques that can be effectively utilized in the decoder in order to improve the visual quality of the reconstructed video in the presence of errors were discussed.

The simulations showed that the use of RVLC helped minimize the error propagation within the same frame. The size of the video packets plays an important role in the quality of the reconstructed video. When slice structure mode is enabled, resynchronization markers are placed in fixed bit intervals inside the bitstream. The combination of the two error concealment techniques performed better in all cases when compared to the spatial error concealment technique alone since the averaging nature of the algorithm leaves visible artifacts, especially when trying to conceal missing MBs that cover areas with critical details.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS

The objective of this thesis was to investigate the performance of MPEG-4 video transmission over erroneous media. In order to achieve this objective, a Matlab encoder was built to compress files at various compression rates. A compressed bitstream enters a forward error correction encoder followed by a two-state Markov channel. From the channel, the corrupted bitstream enters the forward error correction decoder to correct some of the channel errors and finally the video decoder, where the video is reconstructed. In the video decoder, error concealment techniques are applied in order to hide the effect of remaining errors from the user.

The initial intention was to use the MPEG-4 video reference software as an encoding tool of raw video signals that were to be transmitted over the channel. However, this software, even though it supports all the error resilience tools described in Chapter IV, does not have the ability to decode files with actual errors. For this reason, files compressed with the encoder built in Matlab were used for the simulations.

The Matlab encoder uses motion compensation and DCT to remove redundancies in the temporal and spatial domains. Additionally, in order to exploit the content-based philosophy of the standard, shape coding is utilized. Three types of frames are supported: I-, P- and B-frames. This encoder, however, is far from being an MPEG-4 encoder since it does not utilize entropy coding in the form of VLC codes to achieve the maximum compression. Simple rate control was used in order to produce constant quality at the lowest possible variable bit-rate by assigning a constant quantization parameter throughout the video sequence.

A two-state Markov channel with three different channel conditions was considered for the simulation of the wireless channel. Error resilience tools, such as the RVLC, slice structure mode and data partitioning, were simulated. The use of RVLC was simulated assuming that errors in the bitstream are not propagated outside the macroblock in which they occur. At the decoder, three error concealment techniques, spatial, temporal and a combination of spatial and temporal, were used and compared with each other.

A. SIGNIFICANT RESULTS

The absence of entropy coding in the encoder decreased the compression efficiency. The Matlab encoder could only achieve a compression ratio of 9:1 as shown in Figure 3-10 when $Q_P = 31$ was used while the C++ MPEG-4 encoder achieved a compression ratio of 190:1 as shown in Figure 3-23. Even though the sequences used in the two cases were not the same, the difference in compression ratios is significant and illustrates the performance limitations of the Matlab encoder.

The simulation results in Figures 4-15, 4-16 and 4-17 demonstrate the advantage of using an error isolation technique, such as reversible variable length codes, when compared to the approaches used in previous standards (for example, MPEG-2 and H.261). The advantage of fixed packet size was also demonstrated (see Figure 4-6). The simulations indicated that the utilization of this technique is more advantageous than the GOB approach used in MPEG-2 since fewer macroblocks are contained in the packet in high activity areas thereby making these areas less vulnerable to errors.

The use of error concealment at the decoder improved the quality of the reconstructed sequence. Even under high error conditions (for example, BER = 7×10^{-2} with $\varepsilon = 0.1$), the decoder managed to keep the quality measured as PSNR above 20 dB as shown in Figure 4-18. The hybrid method using spatial averaging in the first I-frame and simple temporal concealment in all other frames provided an additional boost in performance in all cases when compared to the pure spatial concealment method (see Figures 4-18, 4-19 and 4-20).

B. FUTURE WORK

The main limitation of the encoder is the lack of an algorithm that would effectively segment images into background and foreground objects. In this thesis a simple edge detection method was used that worked well for sequences containing a "talking head" with uniform background. More efficient image segmentation methods could be applied to improve the encoder performance for more complex sequences.

The absence of entropy coding in the form of variable length codes limited the coding performance of the encoder in this work. The use of RVLC was simulated by assuming that errors do not propagate outside the macroblock boundaries. The implementation of an exponential Golomb-Rice reversible code may enhance the coding efficiency and may lead to more accurate error isolation.

In this thesis, the same forward error correction scheme was used to limit the presence of errors for both the foreground and background objects. However, missing macroblocks in the foreground are more annoying than missing macroblocks in the background since the error concealment techniques used in this work performed better in areas containing low details (background). By using more powerful error protection schemes to protect the objects of interest (e.g. foreground) compared to a simple scheme for all other objects the performance would be enhanced without sacrificing coding efficiency.

In this thesis, only the video aspects of natural sequences of the standard were exploited. MPEG-4 also includes algorithms for compressing still images with the use of wavelet decomposition comparable to JPEG-2000. A study to compare the two standards for the compression of still images will be of interest.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. MPEG-4 VIDEO REFERENCE SOFTWARE

In this appendix, a summary of the MPEG-4 video reference manual, originally developed by the Microsoft Corporation during the experimental phase of the MPEG-4 standardization process, is provided. The original manual was provided by Simon A. J. Winder of Microsoft Corporation [27]. The software is freely available form ISO/IEC site.

The software runs under a Windows console and compiles into three executables: encoder.exe used for encoding and reconstructing raw YUV files, decoder.exe used to decode compressed bit streams, and convertpar.exe used to convert old version parameter files needed for the encoding process.

This software can only deal with 4:2:0 YUV. In this file format, the Y, U, and V planes are written one after the other in that order. If there are *K* frames of video, then the components are written consecutively: Y_0, U_0, V_0, Y_1, … ,Y_K, U_K, V_K. The planes are simply raw pixel bytes written row-by-row, top left to bottom right. If the image size is $M \times N$, then the size of the Y plane is *MN* bytes and the U/V planes are each $MN/4$ bytes. The total number of bytes in the file is $3KMN/2$. No header information is present in the file.

A video sequence can be encoded both in frame based encoding mode and object based encoding mode. In the latter case a segmentation mask file has to be provided. Files with segmentation information must have extension *.seg. These files typically contain binary segmentation maps for shape-based coding or for grayscale alpha plane coding. Each pixel is represented by a byte and these bytes are organized as one plane per frame and each plane is written row-by-row, top left to bottom right. There is no header information. If the image size is $M \times N$, then the size of the alpha plane is *MN* bytes and for an *K* frame sequence, the total number of bytes in the file is *KMN*. In the case of a binary alpha mask, the pixel value is either 0 or 255, indicating transparent or opaque pixels, respectively. In the case of a segmentation mask, the pixel value indicates to which one of multiple binary masks the pixel belongs. For example, three regions of the image can be assigned values 0, 1, and 2. In this way, different parts of the image can be coded; depending on which mask number is chosen during the encoding process. In this case, regions of the image that are not equal to the chosen mask number would be considered to be transparent by the encoder. Finally, pixel values can be used to indicate transparency in the range 0 (transparent) to 255 (opaque). In the case of a grey-scale alpha-map, a file with extension **.aux* has to be present.

A. ENCODING A VIDEO SEQUENCE

The operation has to be performed from the DOS command prompt. For example, encoding sequence *news.yuv*, a QCIF sequence (176x144), the following command has to be executed

```
Encoder news.par
```

news.par is essentially a text file containing all the parameters for the specific encoding operation. The result from this encoding operation is a compressed file with extension **.cmp* and a reconstructed file with extension **.yuv* stored under the specified directory in the parameter file. Extensive information of the parameters in a parameter file can be found in the documentation that comes with the software.

B. DECODING A COMPRESSED FILE

The basic format for decoding a compressed file is the following:

```
Decoder [compressed filename] [reconstructed filename][frame width][frame height]
```

In the compressed filename, the extension **.cmp* has to be provided while this is not necessary in the reconstructed filename. For example, the decoding command for the previous coded sequence news would be:

```
decoder news.cmp news 176 144
```

If the encoding was frame based, the decoder will provide the file *news.yuv* in the same directory where file *decoder.exe* exists. In case that object based encoding was chosen the decoder will provide the segmentation file *news.seg* in the same directory where the binary mask was chosen or provide the file *news.aux* if gray scale alpha-map with transparency information was provided during encoding process.

C. RATE CONTROL

There are two types of rate control: MPEG-4 rate control and TM5 rate control. In MPEG rate control, the quantizer is only changed on a frame basis whereas the TM5 rate control is macroblock-based. For both rate control types, *RateControl.BitsPerSecond* must be set to indicate the number of bits per second.

MPEG rate control is available but only works correctly under limited situations. The main limitation is that it does not support B-frames. When used with I- and P-frames, the first frame must be intra-frame coded for the rate control to function correctly. It is also necessary to adjust the values of the quantization parameter for I-VOPs and P-VOPs for the initital intra- and inter-frame coded VOPs. For the remaining VOPs the rate control takes over and adjusts the quantization parameter accordingly. When the encoding operation is over, it reports a mean quantization parameter. The user must run the encoder again defining as quantization parameter for P-VOPs the reported mean quantization parameter. MPEG-4 rate control will skip frames if the required bit-rate is too low.

TM5 rate control is the recommended form of rate control (it is free from bugs) and works correctly for I-VOPs, P-VOPs, and B-VOPs without user intervention. If the bit-rate is set too low, TM5 rate control will eventually set the quantizer to 31 and it will not skip frames. TM5 rate control cannot be used with non-rectangular VOPs.

D. ERROR RESILIENCE

The encoder and decoder support the MPEG-4 error resilient syntax. The encoder will create an error resilient bitstream when the correct elements are enabled. However, the decoder will not be able to decode a stream containing actual errors because error recovery is not implemented. This is the reason why compressed files with this software were not used for the channel simulations in this thesis.

E. SUPPORTED TOOLS

Table A-1 lists the tools of the MPEG-4 standard that this software supports.

Tool	Version	Comments
Basic(I-VOP, P-VOP, AC/DC Prediction, 4MV, Unrestricted MV)	1	Supported
B-VOP	1	Supported. No MPEG rate control.
P-VOP with OBMC	1	Supported
Method 1, Method 2 Quantisation	1	Supported
Error Resilience	1	Syntax only. No recovery from error supported.
Short Header (H.263 emulation)	1	Decode only.
Binary Shape (progressive)	1	Supported. No automatic VOP generation.
Grayscale Shape	1	Supported
Interlace	1	Supported
N-Bit	1	Supported
Sprite	1	Supported. No warping parameter estimation.
Still Texture	1	Supported
NEWPRED	2	Upstream signaling is simulated not implemented.
Global Motion Compensation	2	Supported
Quarter-pel Motion Compensation	2	Supported
SA-DCT	2	Supported
Error Resilience for Still Texture Coding	2	Supported
Wavelet Tiling	2	Supported
Object Based Spatial Scalability (Base)	2	Supported
Object Based Spatial Scalability (Enhancement)	2	Supported
Multiple Auxiliary Components	2	Supported
Complexity Estimation Support	2	Bitstream syntax supported only.

Table A-1. Supported Tools in MPEG-4 Video Reference Software.

F. RESULTS

In this section, results obtained from compressing two video sequences (“Foreman” and “Carphone”). For the simulations, the first 50 frames of each sequence were used.

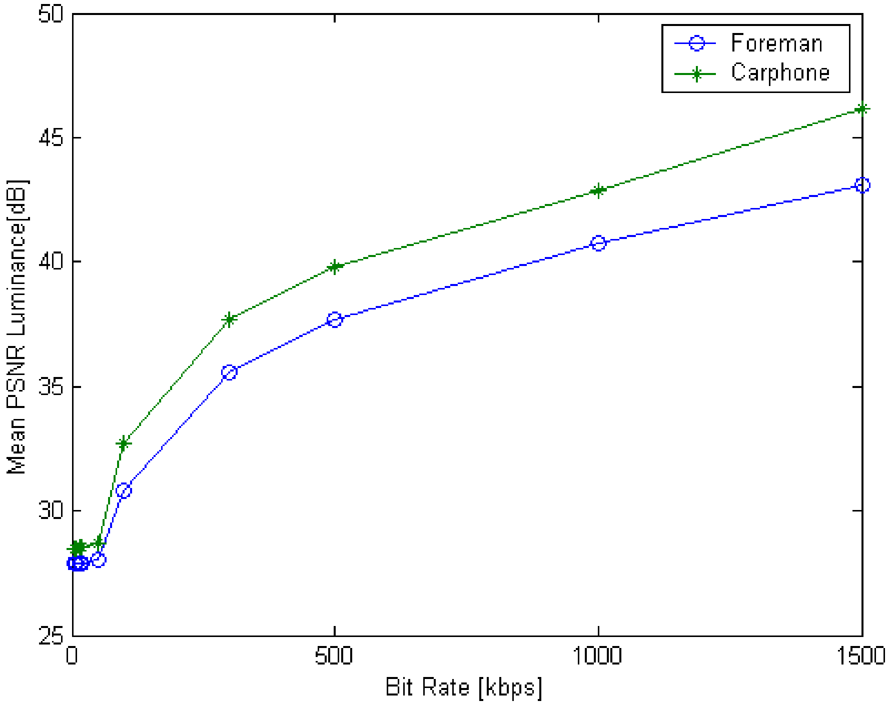


Figure A-1. Rate-Distortion Curve for "Carphone" and "Foreman" Sequences.

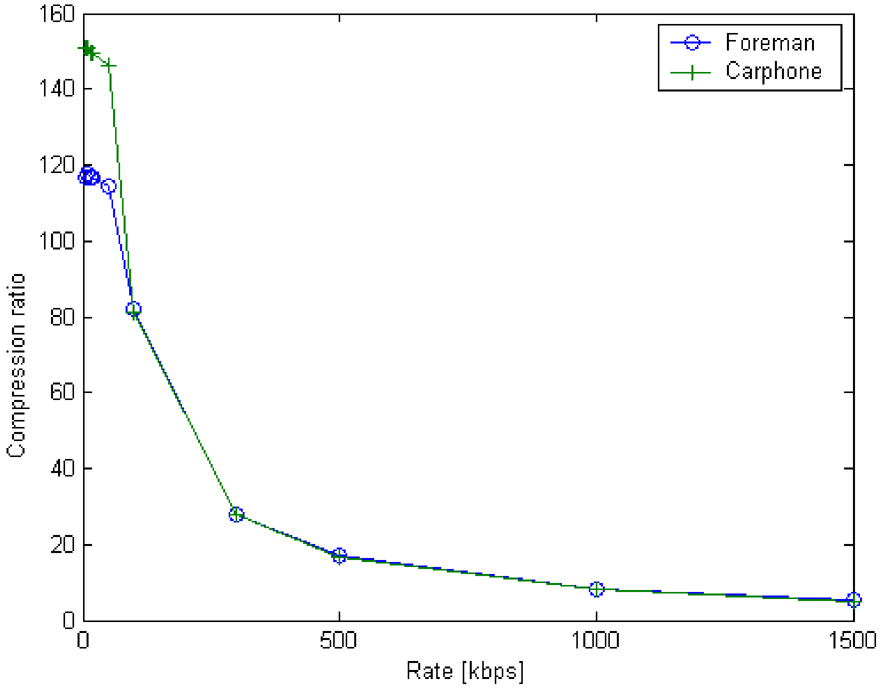


Figure A-2. Compression versus Bit-Rate for "Carphone" and "Foreman" Sequences.

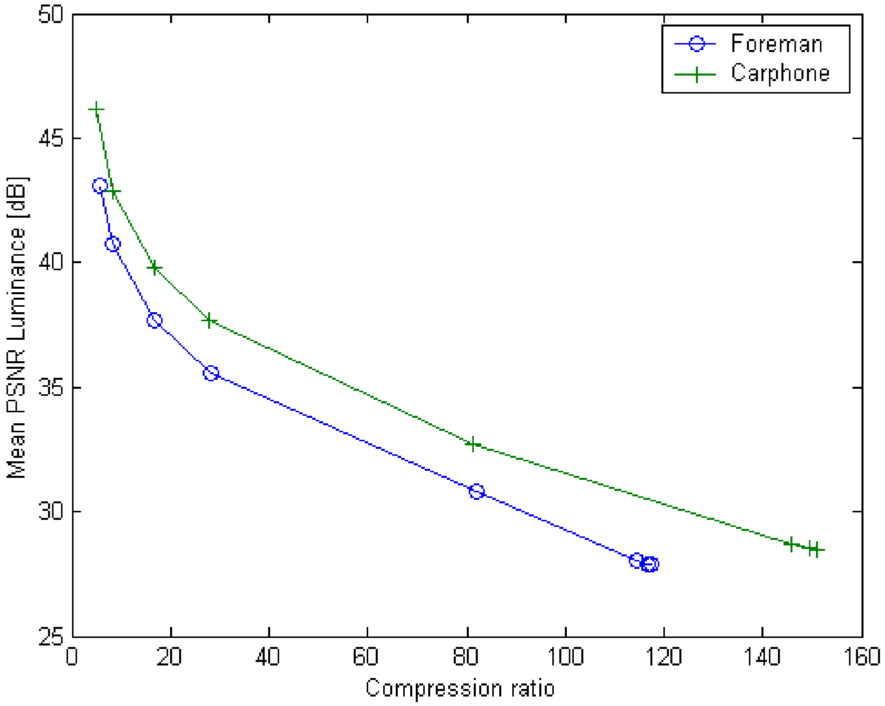
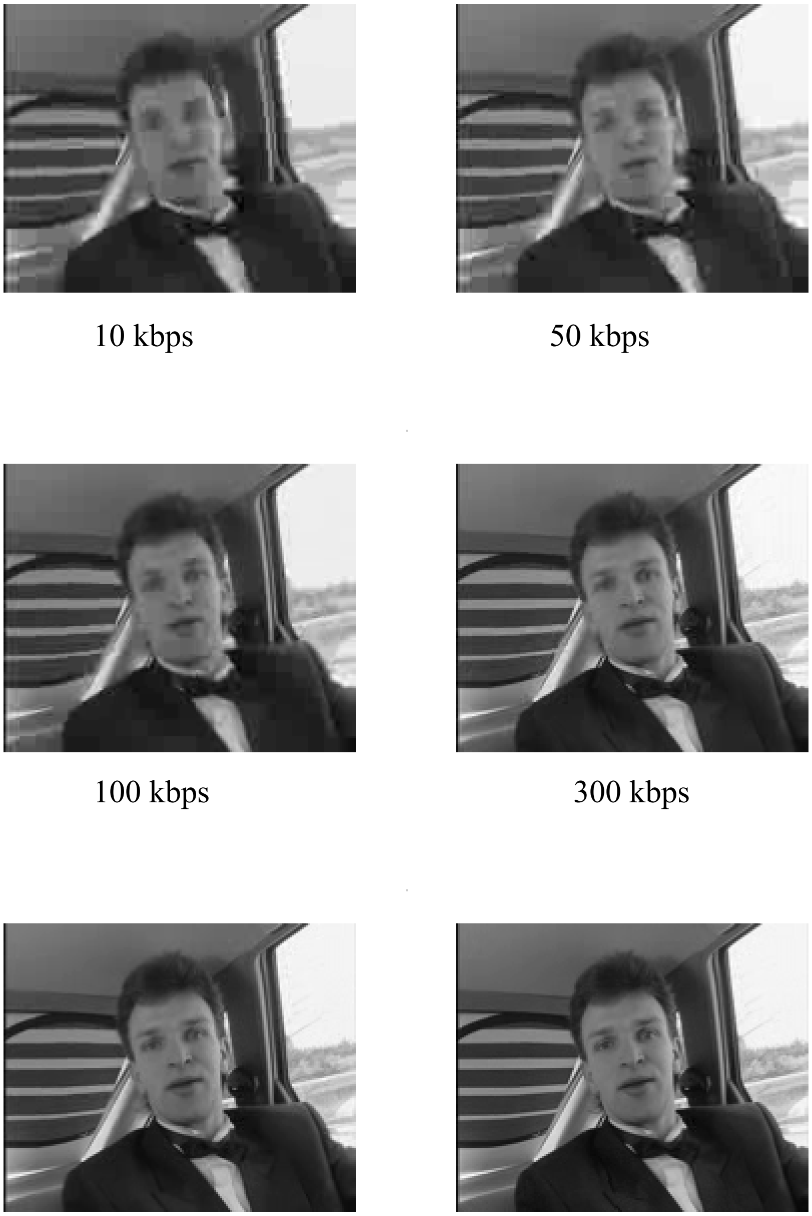


Figure A-3. Mean PSNR versus Compression Ratio for “Carphone” and “Foreman” Sequences.



10 kbps

50 kbps

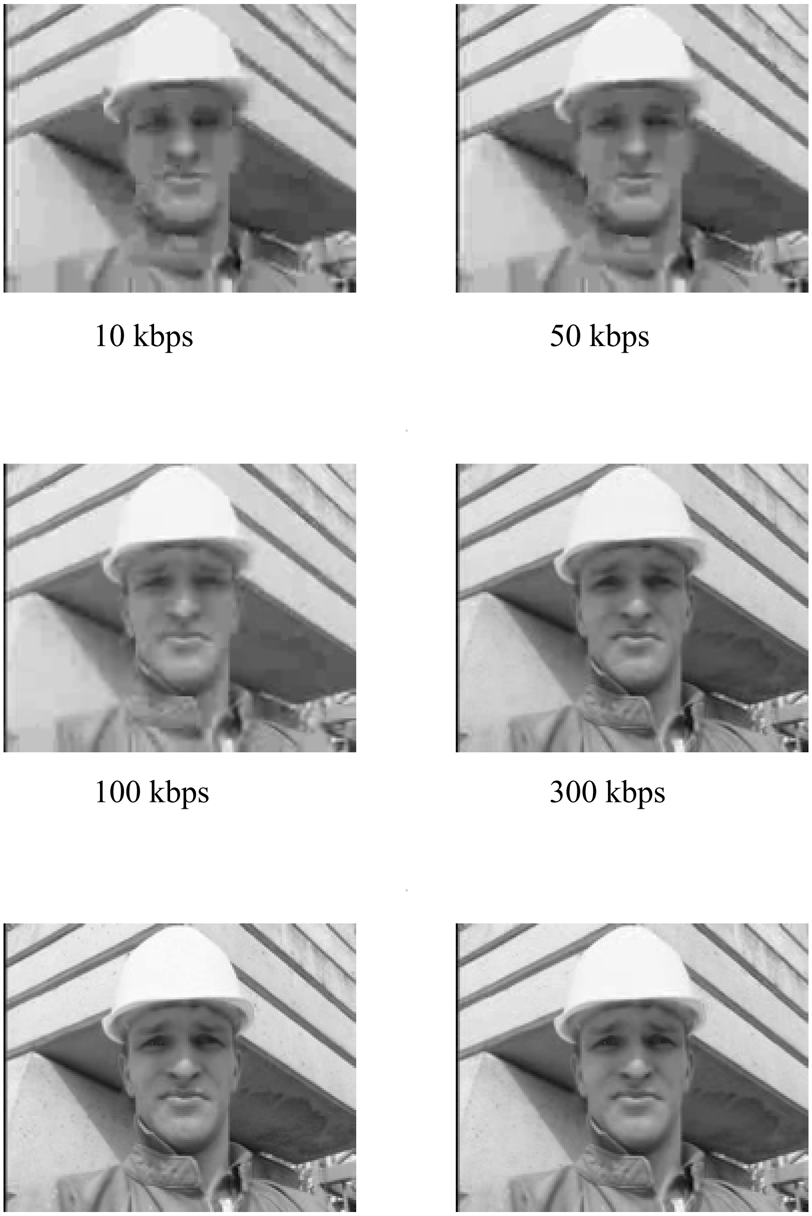
100 kbps

300 kbps

500 kbps

1 Mbps

Figure A-4. “Carphone” Sequence at Various Bit-Rates.



10 kbps

50 kbps

100 kbps

300 kbps

500 kbps

1 Mbps

Figure A-5. "Foreman" Sequence at Various Bit-Rates.

APPENDIX B. SIMULATION RESULTS FOR SEQUENCES "SUZIE" AND "MISS AMERICA"

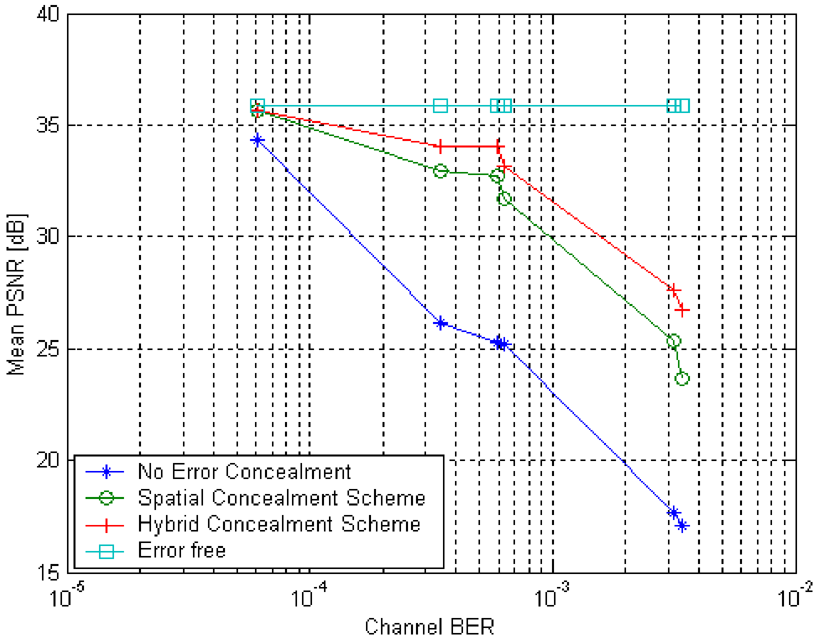


Figure B-1. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.1 for Video Sequence "Miss America".

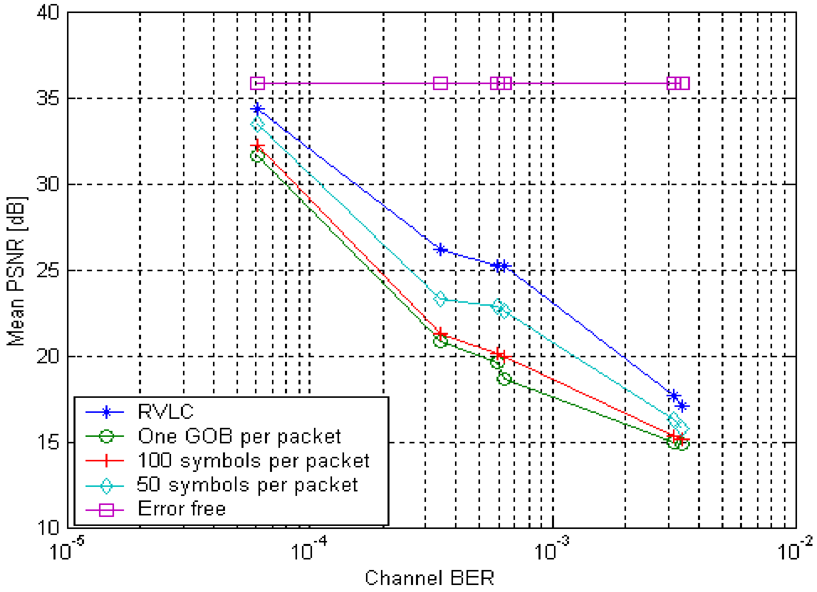


Figure B-2. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.1. No Error Concealment. Video Sequence "Miss America".

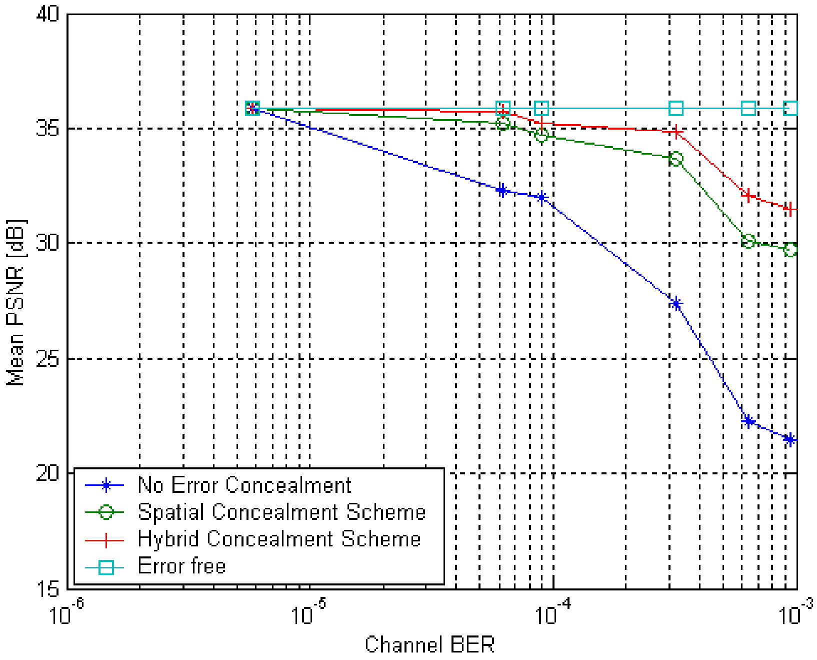


Figure B-3. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.01 for Video Sequence "Miss America".

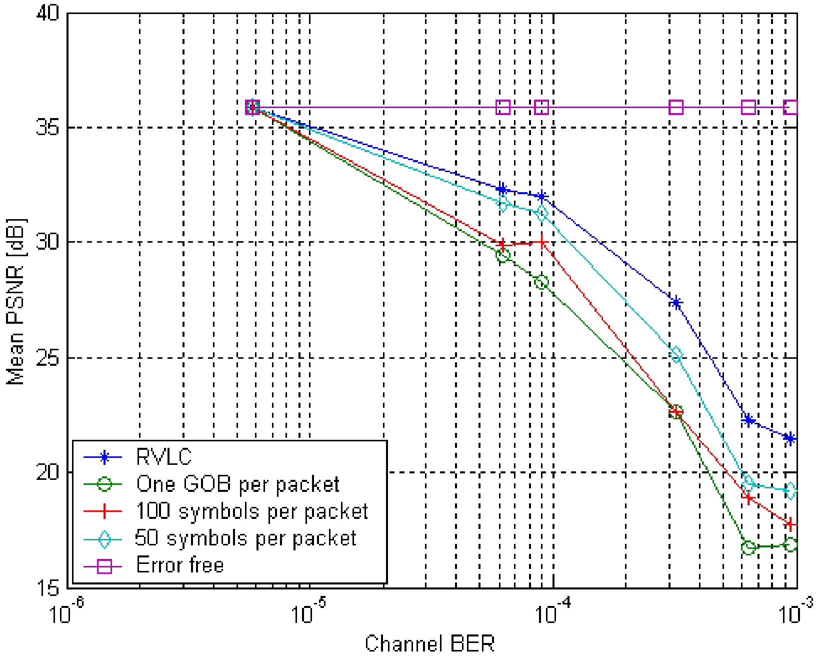


Figure B-4. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.01. No Error Concealment. Video Sequence "Miss America".

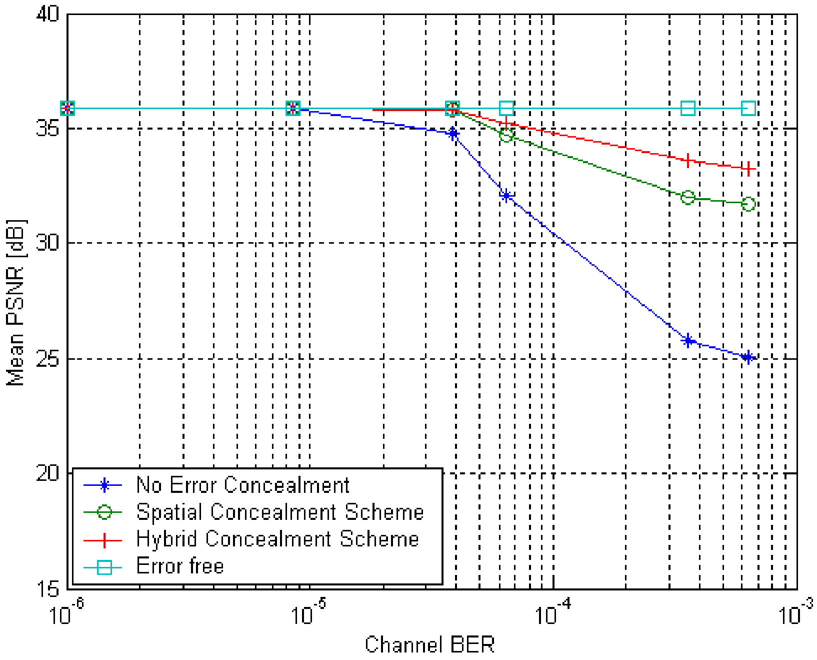


Figure B-5. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.001 for Video Sequence "Miss America".

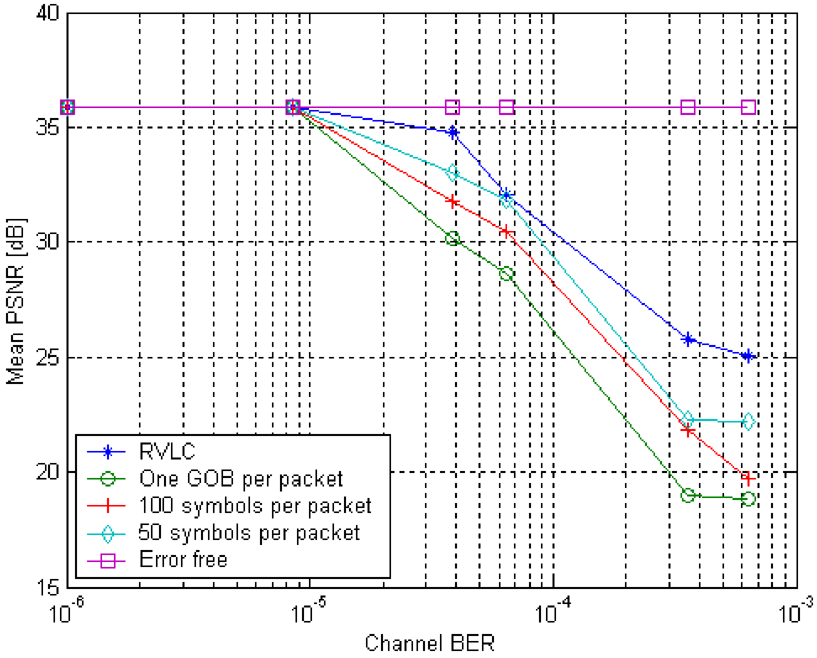


Figure B-6. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.001. No Error Concealment. Video Sequence "Miss America".

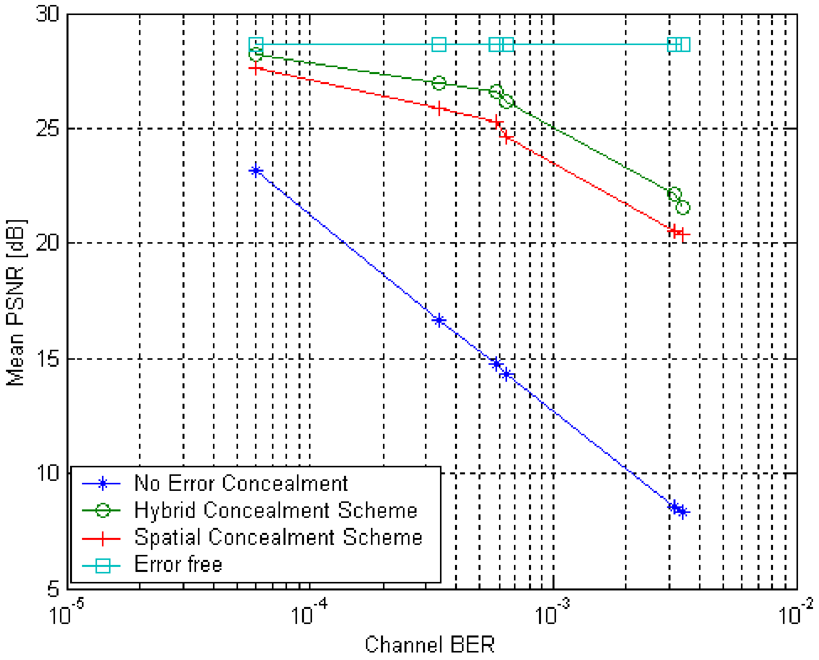


Figure B-7. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.1 for Video Sequence “Suzie”.

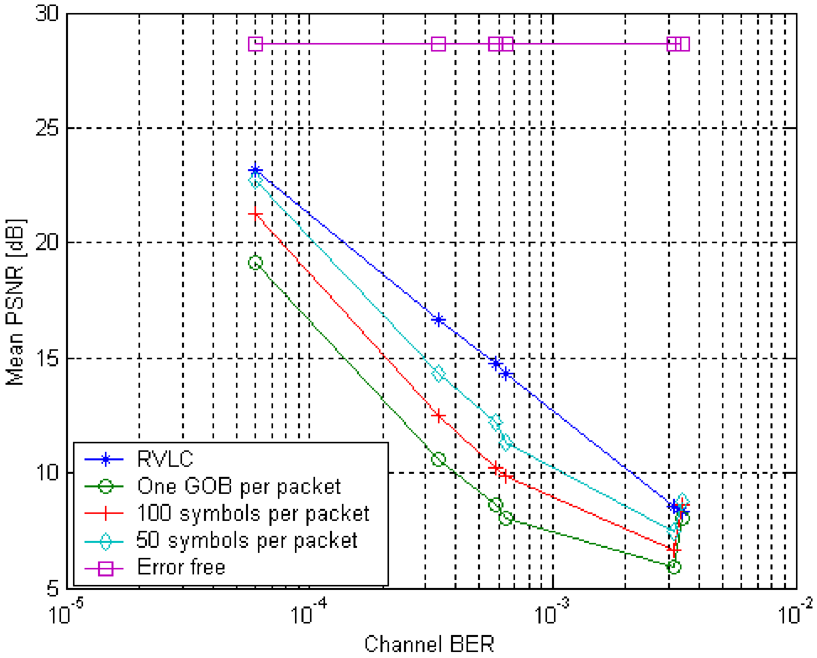


Figure B-8. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.1. No Error Concealment. Video Sequence “Suzie”.

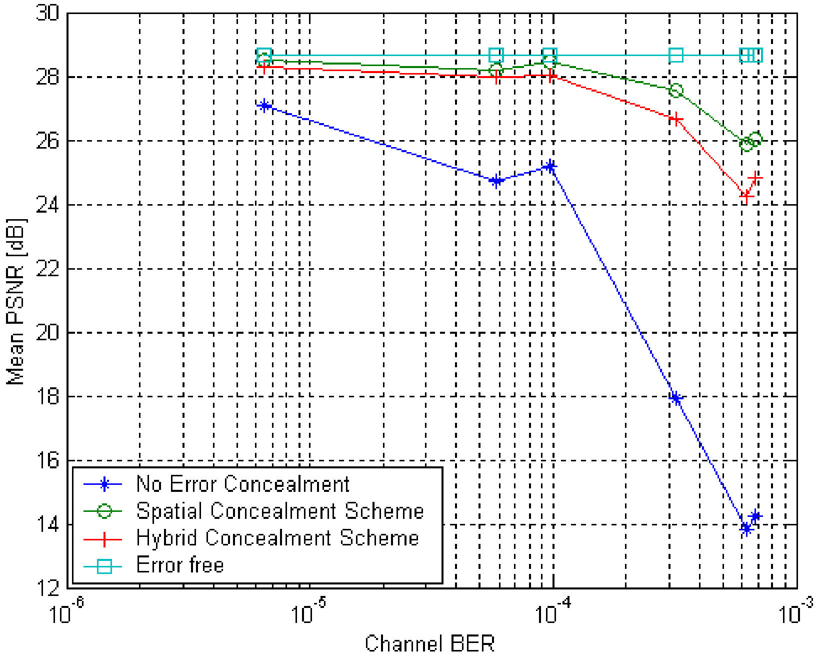


Figure B-9. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.01 for Video Sequence "Suzie".

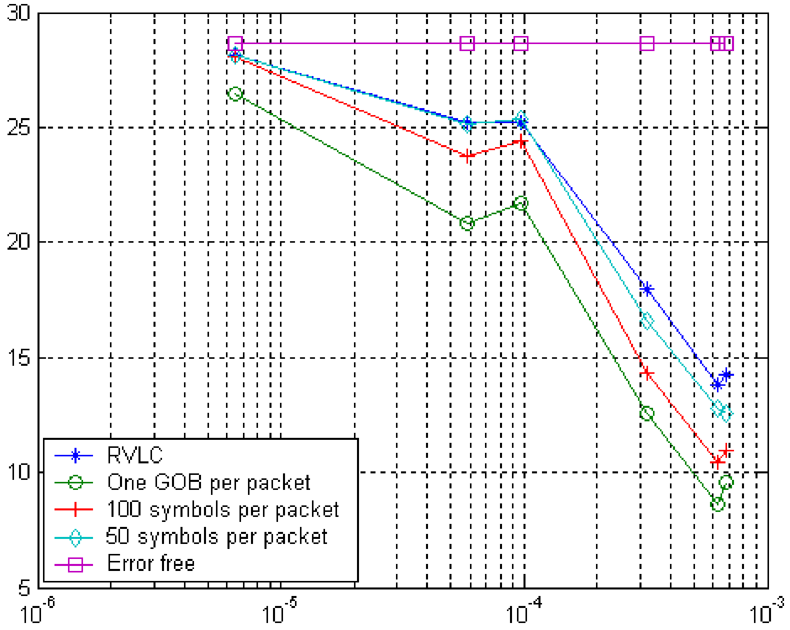


Figure B-10. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.01. No Error Concealment. Video Sequence "Suzie".

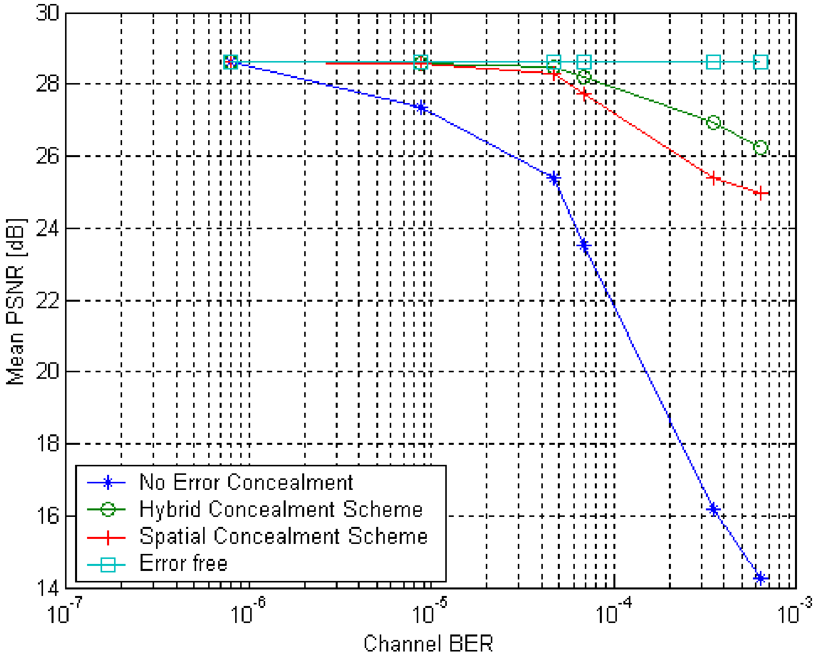


Figure B-11. Different Error Concealment Schemes for a Video Sequence using RVLC Through a Gilbert Channel with ε = 0.001 for Video Sequence “Suzie”.

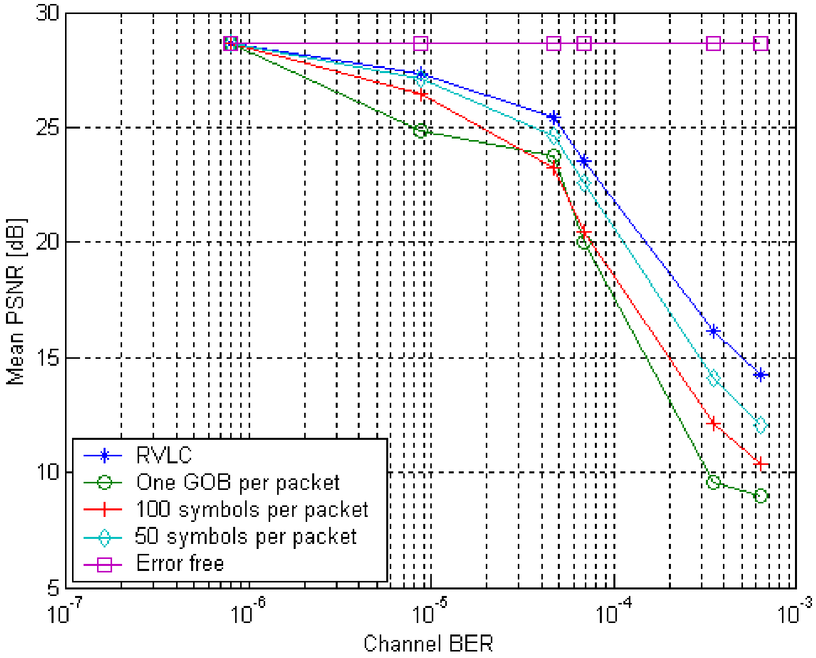


Figure B-12. Different Packet Construction Approaches for a Gilbert Channel with ε = 0.001. No Error Concealment. Video Sequence “Suzie”.

LIST OF REFERENCES

1. M. Ganhbari, *Video coding an introduction to standard codecs,* The Institution of Electrical Engineers, London, United Kingdom, 1999.

2. Y. Wang, J. Osterman, and Y. Zhang, *Video processing and communications*, Upper Saddle River, New Jersey, Prentice-Hall, Inc., 2002.

3. T. Ebrahimi, "MPEG-4 video verification model: a video encoding/decoding algorithm based on content representation", Signal processing Laboratory, Swiss Federal Institute of Technology [http://rtlab.kaist.ac.kr/~gunhan/MPEG4/paper5/paper5.html].

4. Technical Research Report CSHCN T.R. 97-5 (ISR T.R. 97-16), *An advanced image coding algorithm that utilizes shape adaptive DCT for providing access to content* by R. Haridashan.

5. T. Ebrahimi, C. Horne, "MPEG-4 natural video coding - an overview". [http://leonardo.telecomitalialab.com/icjfiles/mpeg-4_si/7-natural_video_paper/7-natural_video_paper.htm].

6. T. Ebrahimi, "MPEG-4 natural video tools presentation", Swiss Federal Institute of Technology, Lausanne [http://www.ist-metavision.com/docs/mpeg-naturalvideotools.pdf].

7. T. Chen, "MPEG-4 Presentation", Carnegie Mellon University [http://www.ece.cmu.edu/~ece796/mpeg4.pdf].

8. ISO/IEC 14496-2/1999Amd.1:2000(E): *Information technology-coding of audio-visual objects-part 2: visual, amendment 1: visual extensions*, International Organization for Standardizations.

9. R. Koenen, "ISO/IEC JTC1/SC29/WG11 N4030: executive overview of the MPEG-4 standard", March 2001.

10. B. Girod, N. Farber, "Compressed video over networks", Telecommunications Laboratory, University of Erlangen, Nuremberg, [http://citeseer.nj.nec.com/girod99wireless.html].

11. M. Zorzi, R.R. Rao, "Error control strategies for the wireless channel", ICUPC'96, Cambridge, MA, Sep.-Oct. 1996.

12. R.Talluri, "Error-resilient video coding in the ISO MPEG-4 standard", IEEE Communication Magazine, June 1998.

13. J. Wen, J, Villasenor, “Reversible variable length codes for efficient and robust image and video coding”, Proceeding of IEEE Data Compression Conference, Snowbird, Utah, April 1998.

14. S. Valente, C. Defour, F. Groliere, D.Shnook, “An efficient error concealment implementation for MPEG-4 video streams”, IEEE Transactions on Consumer Electronics, Vol. 47, No. 3, August 2001.

15. Y. Wang, Q. F. Zhu, “Error control and concealment for video communication: a review”, Proceedings of IEEE, Vol. 86, No. 5, May 1998.

16. Y. Wang, S Wenger, J. Wen, A. Katsaggelos, “Error resilient video coding techniques, real-time video communications over unreliable networks”, IEEE Signal Processing Magazine, July 2000.

17. P. Salama, N. B. Shro, E. J. Delp, “Error concealment in encoded video streams”, Video and Image Processing Laboratory, School of Electrical and Computer Engineering, Purdue University, West Lafayete, IN. [ftp://skynet.ecn.purdue.edu/pub/dist/delp/chapter-stream/chapter.pdf].

18. E. N. Gilbert, “Capacity of a burst-noise channel”, Bell System Tech. Journal, Vol. 39, pp.1253-1266, Sept. 1960.

19. E. O. Elliot, “Estimates of error rates for codes on burst-error channels”, Bell System Tech. Journal, Vol. 42, p. 1977, Sep. 1963.

20. K. Stuhlmuller, N. Farber, M. Link, and B. Girod, “Analysis of video transmission over lossy channels”, IEEE Journal on Selected Areas of Communications, Vol.18, No. 6, June 2000.

21. ISO-IEC/JTC1/SC29/WG11, “Report of the formal verification tests on MPEG-4 video error resilience, N2604”, December1998.

22. S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Upper Saddle River, New Jersey, Prentice-Hall, Inc., 1995.

23. ITU-T Rec. H.263, *Version 2, Video Coding for Low Bitrate Communication,* Jan. 1998.

24. Hypermedia Image Processing Reference, University of Edimburg, [http://www.cee.hw.ac.uk/hipr/html/hipr_top.html].

25. *Image Processing Toolbox for Use with Matlab*, The Mathworks Inc., 2001.

26. M. Novell, *Error Resilient Image Codec for Adaptive Wireless Terminals*, Master’s Thesis, University of California, 1999.

27. S. A. J. Winder, *ISO/IEC 14496 (MPEG-4) Video Reference Software Version: Microsoft-FDAM1-2.3-001213 User Manual*.
[http://www.iso.ch/iso/en/stdsdevelopment/techprog/workprog/TechnicalProgrammeSCDetailPage.TechnicalProgrammeSCDetail?COMMID=148].

28. J. Lu, K.B. Letaief and M.L. Liou, " Robust video transmission over correlated mobile fading channels", IEEE Transactions on circuits and systems for video technology, Vol. 9, No. 5, August 1999.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California, 93943-5121

3. Chairman, Department of Electrical and Computer Engineering
   Monterey, California, 93943-5121

4. Prof. Murali Tummala, Code EC/Tu
   Department of Electrical and Computer Engineering
   Monterey, California, 93943-5121

5. Prof. Robert Ives, Code EC/Ir
   Department of Electrical and Computer Engineering
   Monterey, California, 93943-5121

6. Dr. Richard C. North
   Code D855
   Spawar Systems Center
   53560 Hull Street
   San Diego, CA 92152

7. Thomas Darnell
   Titan Systems Corporation
   6207 Aviation Ave.
   Jacksonville, FL 32221

8. Hellenic Navy General Staff
   Department B/2, Stratopedo Papagou, Mesogeion 151,
   Holargos, 155-00, Athens
   Greece

9. Lt Evangelos A. Adam H.N
   92-94 Papanastasiou St
   K Patisia, Athens, 10445
   Greece