Efficient error concealment and error control schemes for H.264 video

Zheng, Jinghong

2006

Zheng, J. (2006). Efficient error concealment and error control schemes for H.264 video.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/4039>

<https://doi.org/10.32657/10356/4039>

Nanyang Technological University

Downloaded on 26 Apr 2023 16:41:19 SGT

Efficient Error Concealment and Error Control Schemes for H.264 Video

Zheng Jinghong

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2006

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research done by me and has not been submitted for a higher degree to any other University or Institute.

.....

Date

.....

Zheng Jinghong

Acknowledgements

First and foremost, I would like to express my deepest appreciation and sincere gratitude to my supervisor, Associate Professor Chau Lap Pui, for his precious time, invaluable instruction and great support. It is him who gave me strong support and sincere encouragement throughout the tough time of the research.

I would like to express my sincere thanks to Mr. Jing Xuan, Ms. Shu Haiyan, Mr. Gao Yunlong, Mr. Fang Tao and Ms. Wang Yu. Their useful advisement and patient discussion on my project gave me much help. Additional thanks are also due to other members in Media Technology Lab for their assistance and cooperation.

I would like to thank my family for their love and support that give me greatest inspiration.

Contents

Acknowledgements	i
Summary	vii
List of Figures.....	ix
List of Tables	xiv
List of Acronyms.....	xvi

Part I

1 Introduction.....	1
1.1 Concept of Video Compression	1
1.1.1 Video Coding Schemes.....	2
1.1.2 Motion Compensation	4
1.1.3 2-D Discrete Cosine Transform.....	8
1.1.4 Quantization	8
1.1.5 Reordering and Entropy Coding.....	10
1.1.6 Decoding	11
1.2 H.264.....	12
1.2.1 Motion Compensated Prediction	14
1.2.1.1 Variable Block-Size Motion Compensation	14

1.2.1.2 Quarter-Accurate Motion Compensation.....	16
1.2.1.3 Multiple Reference Pictures	18
1.2.2 Intra Prediction	19
1.2.3 Flexible Slice Size and Ordering.....	21
1.3 Objective and Contribution.....	22
1.4 Outline of the Thesis	23
2 Review	25
2.1 Forward Error Correction.....	26
2.2 Resynchronization.....	30
2.3 Error Concealment	32
2.3.1 Interpolation	33
2.3.2 Recovery of DCT coefficients.....	36
2.3.3 Boundary Matching.....	39
2.3.4 Block Matching	44
2.3.5 Other Error Concealment Methods	49
2.3.5.1 Spatial Error Concealment Methods.....	49
2.3.5.2 Temporal Error Concealment Methods.....	56
2.4 Multiple Description Coding	62
2.5 Intra Refresh.....	65
2.6 Conclusion	68

Part II Error Concealment Techniques

3 An Efficient Maximally Smooth Error Concealment Method	69
3.1 Introduction.....	69
3.2 Relative Spatial Error Concealment Methods.....	70
3.3 The Recovery of Lost Block	75

3.4 Interpolation of Inner Pixels	80
3.5 Performance Evaluation of the Maximally Smooth Error Concealment	82
3.6 Summary	85
4 First-Order Plane Estimation for Motion Vector Recovery	87
4.1 Introduction.....	87
4.2 Motion Vector Recovery	89
4.3 Algorithm of Linear Interpolation.....	90
4.3.1 Estimate Lost Motion Vectors by Corner Motion Vectors.....	91
4.3.2 Interpolation of Motion Vector for Each Pixel	94
4.4 Simulation Results of the Motion Vector Recovery Method	95
4.5 Conclusion	101
5 A Motion Vector Recovery Algorithm Using Lagrange Interpolation	102
5.1 Introduction.....	102
5.2 Approximation	103
5.2.1 Goals and Strategy.....	103
5.2.2 Interpolation Methods	105
5.2.3 Lagrange Polynomials.....	107
5.3 Described Function of Lost Motion Vectors	110
5.4 Recover the Motion Vectors of Lost Macroblock	112
5.5 Recovery Performance of Lagrange Interpolation	114
5.6 Conclusion	122
6 Efficient Temporal Error Concealment Based on a Polynomial Model	124
6.1 Introduction.....	124
6.2 Least-Squares Approximation.....	125
6.2.1 Fitting a Straight Line.....	126
6.2.2 Assess Fitting.....	128

6.3 Motion Vector Recovery Algorithm Using Polynomial Model.....	131
6.3.1 The Lost Macroblock and Its Neighbors	131
6.3.2 Polynomial of Neighboring Motion Vectors	133
6.4 Simulation Results of Polynomial Error Concealment	137
6.5 Conclusion	146
 Part III Error Resilience Techniques	
7 Periodic Macroblock for Error Resilient Video Transmission	147
7.1 Introduction.....	147
7.2 Multiple Reference Frames	151
7.3 Content-Based Periodic Macroblock	153
7.4 Simulation Results	160
7.5 Conclusion	164
 8 A Drift-Reduction Multiple Description Coder Using Multiple Reference Frames.....	165
8.1 Introduction.....	165
8.2 Side Information	167
8.3 A Multiple Description Coding Method Using Multiple Reference Frames.....	170
8.3.1 Side Information Based on Multiple Reference Frames	170
8.3.2 Selection of Macroblocks Contained in Side Information	174
8.4 Simulation Results	177
8.5 Conclusion	180
 9 Conclusion and Future Work.....	182
9.1 Conclusion	182
9.2 Future Research Suggestion.....	186

Author’s Publication.....187

Bibliography189

Summary

In the last decade, as the demand of video transmission over communication network grows rapidly, the algorithms and technologies in video processing have significant improvement.

In order to achieve good coding efficiency, the temporal and spatial redundancies in video are removed by compression schemes, such as motion compensated prediction and discrete cosine transform. On the other hand, these coding schemes make the encoded video very sensitive to the transmission error. Even single bit error in transmission may cause obvious degradation of video quality. To alleviate the effect of the transmission error, many error resilient algorithms have been developed. In this thesis, many popular error resilient algorithms have been studied, and several new error control methods are proposed for the new coding standard H.264, which supports many new video applications. There are three parts in the thesis.

In Part I that includes Chapter 1 and 2, an introduction about the fundamentals of video coding is presented. The codec of H.263 is used as an example to illustrate how to compress the digital video into a bitstream. Then the new coding standard H.264 is introduced, since some new coding techniques are adopted in this coding standard. In Chapter 2, many efficient error resilient methods are reviewed.

In Part II, several efficient error concealment methods developed for H.264 are presented. Error concealment is implemented in the decoder to alleviate the effect of

transmission error. Spatial and temporal redundancies are often exploited to reconstruct the corrupted area of video. In Chapter 3, a maximally smooth error concealment method is formulated for the reconstruction of 4x4 block. The information of the blocks that are neighboring to the lost 4x4 block is used to approximate the pixel values of the lost 4x4 block. To improve the coding efficiency, H.264 adopts more flexible motion compensation schemes than previous coding standards. A 16×16 macroblock can be divided into different shapes for motion estimation. By making use of this property, three different modes are established to simulate the motion change tendency within a small area. And the lost motion vectors can be predicted by these modes.

In Part III, two different error resilient methods that are implemented in encoder are presented. One method is named as periodic macroblock. Some macroblocks in every n^{th} frame are encoded as periodic macroblocks. The error propagation effect between two frames that contain periodic macroblocks can be efficiently alleviated. Another error resilient approach is multiple description coding with reduced drift. By using multiple description coding, the original video can be encoded into several independent bitstreams. The decoder can use either bitstream to reconstruct the video. The main problem of the multiple description coding is the mismatch of reference frames used in the encoder and the decoder. To solve this problem, an additional motion-prediction-loop is introduced into the multiple description encoder, and the concept of multiple reference frames is also adopted in this encoder.

List of Figures

Figure 1. 1 Generic video compression framework.....	2
Figure 1. 2 The temporal and spatial correlation in video sequence.	3
Figure 1. 3 The intra-frame and inter-frame. The P frame is dependent on the previous I or P frame. The B frame is predicted from the previous and future P or I frames... 3	
Figure 1. 4 The encoder structure of H.263.	4
Figure 1. 5 The decoder structure of H.263.	4
Figure 1. 6 Motion estimation and compensation block diagram.....	5
Figure 1. 7 Motion estimation.....	6
Figure 1. 8 Three-step search algorithm.....	7
Figure 1. 9 The distribution of DCT coefficients in a block.	9
Figure 1. 10 Zig-zag scan order.	11
Figure 1. 11 Basic structures of H.264. (a) H.264 encoder. (b) H.264 decoder.....	12
Figure 1. 12 Different macroblock division modes supported in H.26L.....	14
Figure 1. 13 Segmentation of a macroblock in H.264 for motion compensation. Top line: segmentation of macroblock, bottom line: segmentation of 8x8 partitions.	15
Figure 1. 14 Interpolation for fractional-sample accurate motion compensation. Capital letter indicates full-sample grid, and the lower case presents fractional sample. ...	17
Figure 1. 15 Multi-frame motion compensated prediction. The reference frame flag f will be transmitted with motion vector.	18

Figure 1. 16 (a) Intra_4x4 prediction that is conducted for pixel A-P using neighboring samples a-q. (b) Corresponding prediction directions for Intra_4x4 prediction....	19
Figure 1. 17 Five examples of Intra_4x4 block prediction modes.	20
Figure 1. 18 General slice structure when not using FMO.	21
Figure 1. 19 Using FMO to divide a QCIF frame into slices.....	21
Figure 2. 1 Reed-Solomon code.....	26
Figure 2. 2 Multicast system over wired and wireless network.....	27
Figure 2. 3 The data dependence of scalable video coding. (a) no enhancement layer in prediction. (b) using enhancement layer in prediction.	27
Figure 2. 4 Layered FEC for multicast framework.....	29
Figure 2. 5 Relation of synchronization markers and coded macroblocks in H.263 frame.	31
Figure 2. 6 Linear interpolation.	34
Figure 2. 7 Edge-oriented spatial interpolation algorithm proposed by Hsia.	34
Figure 2. 8 Multiple edge oriented interpolation.	35
Figure 2. 9 The location of lost block and its boundary vectors.....	36
Figure 2. 10 Smoothing directions for an 8x8 block.....	37
Figure 2. 11 The lost block M_5 and its neighboring blocks.	40
Figure 2. 12 Four coding modes of B frame. (a) Forward mode. (b) Backward mode. (c) Interpolation mode. (d) Direct mode.	43
Figure 2. 13 Schematic presentation of the split-matching algorithm.....	46
Figure 2. 14 Temporal matching of top and/or bottom available macroblock.....	48
Figure 2. 15 Content-based spatial error concealment.	49
Figure 2. 16 Conceal the edge block by directional interpolation	49
Figure 2. 17 Error resilient image coding bitstream structure.	52
Figure 2. 18 Post-processing error-concealment decoding flow.....	53

Figure 2. 19 Demonstration of the general approach.	54
Figure 2. 20 An example of the recovery of shape information in [83]. (a) α -plane of frame 0. (b) Received α -plane with one slice missing. (c) One possible reconstructed boundary for figure.....	54
Figure 2. 21 Illustration of impact of scene change on conventional temporal error concealment.....	56
Figure 2. 22 Modified average algorithm. (a) TB. (b) STB. (c) ASTB.....	62
Figure 2. 23 Example of a multiple description coding communication system.....	63
Figure 2. 24 Multiple description image coding using pixel interleaving.....	63
Figure 2. 25 Multiple description transform coding using a cascade of KL transform and a correlating transform.....	64
Figure 2. 26 Linear forward error tracking.....	66
Figure 3. 1 The lost block B_i and its neighboring blocks.	72
Figure 3. 2 The locations of maximally obtainable DCT coefficients.....	73
Figure 3. 3 The lost block and the four boundary pixel vectors.	76
Figure 3. 4 4×4 block and its neighboring pixels.	81
Figure 3. 5 PSNR of different bitrate. (a) Stefan: 5% blocks lost. (b) Foreman: 5% blocks lost.....	85
Figure 4. 1 The estimation of the vertices motion vectors.	92
Figure 4. 2 The plane built by neighboring motion vectors.	93
Figure 4. 3 Interpolation of motion vector for each pixel.	94
Figure 4. 4 Comparison. (a) Coastguard: 5% macroblocks lost, (b) Coastguard: 10% macroblocks lost.	100
Figure 5. 1 The lost macroblock and its neighboring macroblocks.....	111

Figure 5. 2 Recovered by motion vectors in both directions. (a) Tempete: 5% macroblocks lost. (b) Tempete: 10% macroblocks lost.	116
Figure 5. 3 Recovered by motion vectors in one direction. (a) Tempete: 5% packets lost. (b) Tempete: 10% packets lost.	119
Figure 5. 4 The subjective performance of the three algorithms. (a) Corrupted Video, PSNR = 12.2321. (b) Temporal Replacement, PSNR = 30.3096. (c) Boundary matching, PSNR = 30.9625. (d) Proposed Method, PSNR = 34.0804.	122
Figure 6. 1 Two examples for macroblock assignment to a slice in H.264.	132
Figure 6. 2 The location of a lost macroblock and its neighboring macroblocks. (a) A slice lost. (b) A macroblock lost.	132
Figure 6. 3 The motion vectors in the lost macroblock and neighboring macroblock.	134
Figure 6. 4 PSNR of different macroblock loss rates. (a) Tempete: 5% macroblocks lost. (b) Tempete: 10% macroblocks lost.	138
Figure 6. 5 PSNR of different slice loss rates. (a) Tempete: 5% slices lost. (b) Tempete: 10% slices lost.	142
Figure 7. 1 Multiple reference frames used in H.264.	149
Figure 7. 2 Multiple reference frame motion compensated predictor.	151
Figure 7. 3 Single reference frame Markov chain.	152
Figure 7. 4 Two reference frames Markov chain.	152
Figure 7. 5 Periodic Macroblock.	154
Figure 7. 6 The receipt probability of macroblock for single reference frame.	155
Figure 7. 7 The receipt probability of pixel in different scenarios.	156
Figure 7. 8 Comparison of PSNR. (a) Foreman: 5% macroblocks lost. (b) Foreman: 10% macroblocks lost. (c) Foreman: 15% macroblocks lost.	162
Figure 8. 1 The structure of the three-prediction-loop multiple description coder.	168

Figure 8. 2 The enhanced two-stage multiple description codec that is based on three-prediction-loop structure.....	169
Figure 8. 3 A typical multiple description coder.	171
Figure 8. 4 The proposed coder structure for multiple description coding.	172
Figure 8. 5 The decoder of multiple description coding.....	172
Figure 8. 6 Multiple description coding method used in simulation	178
Figure 8. 7 The PSNR of different situations. (a) Mobile: 5% macroblocks lost. (b) Mobile: 10% macroblocks lost. (c) Mobile: 15% macroblocks lost.....	180

List of Tables

Table 2. 1 Compare the performance of MMRF with other concealment methods	51
Table 2. 2 Average PSNR values for different error concealment schemes for the Flower Garden, Football, and Hockey sequences.....	52
Table 2. 3 Average PSNR comparisons for various schemes.....	59
Table 2. 4 PSNR comparison of different temporal error concealment methods for each P frame in the test sequence: table tennis.	60
Table 3. 1 Comparison of the PSNR in different situations	83
Table 4. 1 The simulation results of our algorithm. (a) Foreman. (b) Stefan. (c) Coastguard. (d) Mobile. (e) Tempete.....	97
Table 4. 2 Comparison between first-order plane and boundary matching algorithm. .	99
Table 5. 1 The corresponding coordinates of each motion vector.	111
Table 5. 2 The simulation results of proposed algorithm with interleaving. (a) Foreman with interleaving. (b) Stefan with interleaving. (c) Mobile with interleaving. (d) Tempete with interleaving.....	117
Table 5. 3 The simulation result of proposed algorithm without interleaving. (a) Foreman without interleaving. (b) Stefan without interleaving. (c) Mobile without interleaving. (d) Tempete without interleaving.	120

Table 6. 1 The corresponding coordinates of motion vectors.	134
Table 6. 2 Simulation results of different macroblock loss rates. (a) Tempete. (b) Foreman. (c) Mobile. (d) Stefan.....	139
Table 6. 3 Simulation results of different slices loss rates. (a) Tempete. (b) Foreman. (c) Mobile. (d) Stefan.	143
Table 7. 1 The PSNR of different macroblock loss rates.	163
Table 7. 2 Comparison between periodic macroblock and intra macroblock	164

List of Acronyms

ARQ:	Automatic Repeat Request.
BMA:	Boundary Matching Algorithm.
DCT:	Discrete Cosine Transform.
FEC:	Forward Error Correction.
FMO:	Flexible Macroblock Ordering.
LS:	Least Squares.
MAE:	Mean Absolute Error.
MAP:	Maximum a Posteriori.
MDSQ:	Multiple Description Scalar Quantizer.
MRF:	Markov Random Field.
MSE:	Mean Square Error.
NAL:	Network Abstraction Layer.
QP:	Quantization Parameter.
RS Code:	Reed-Solomon Code.
SAD:	Sum of Absolute Difference.
VLC:	Variable-Length Coding.

Chapter 1

Introduction

The explosion of the Internet and the acceptance of its interactive operation mode have clearly shown that the traditional television paradigm will no longer be sufficient for the services, people want to access to the video as they now have access to text and graphics. To satisfy this need, the original video sequences must be encoded and transmitted at low bit rates with acceptable quality. Compression is the process of compacting data into a smaller number of bits. In this chapter, the fundamentals of video compression are presented first. Then the new coding standard H.264 is introduced, and the outline of this thesis is presented.

1.1 Concept of Video Compression

Video compression (video coding) is the process of compacting or condensing a digital video sequence into a smaller number of bits. Compression involves a complementary pair of systems: an encoder and a decoder [1-3]. The encoder converts the source data into a compressed form prior to transmission or storage. And the decoder converts the compressed form back into a representation of original video, as shown in Figure 1.1.

The encoder and the decoder are often described as a CODEC. In this section, the techniques used in video coding are reviewed, and the structure of codec is examined.

Figure 1. 1 Generic video compression framework.

1.1.1 Video Coding Schemes

Most video coding methods exploit both temporal and spatial redundancies to achieve good compression efficiency. In the temporal domain, there is usually a high correlation (similarity) between frames of video that were sampled around the same time. The frames adjacent temporally (successive frames in time order) are often highly correlated, especially when the temporal sampling rate is high. In the spatial domain, there is usually a high correlation between pixels within a small area. In other words, the neighboring samples are often very similar. In Figure 1.2, the existing temporal and spatial relations in video are illustrated.

There are two kinds of frame adopted in most video coding standards: Intra-frame (I frame) and Inter-frame (P/B frame). For the intra-frame, the encoding procedure is similar to image coding. Only the spatial redundancy is removed. For the inter-frame, the encoding frame is predicted from the previously encoded frame, which is denoted as reference frame for the encoding frame, as shown in Figure 1.3. The difference between the reference frame and the encoding frame is calculated and encoded. Therefore, an inter-frame depends on its reference frame.

Figure 1. 2 The temporal and spatial correlation in video sequence.

Figure 1. 3 The intra-frame and inter-frame. The P frame is dependent on the previous I or P frame. The B frame is predicted from the previous and future P or I frames.

Until now, many different video coding standards have been established, such as MPEG-2 [4], H.261 [5], H.263 [6] and H.264 [7].

H.263 is a widely used video coding standard, and the flexibility of this standard makes it suitable for various applications. In this subsection, we use the codec of H.263 as example to illustrate how a video codec operates. The introduction is focused on the encoding schemes. The decoding methods can be derived from the encoding schemes, since each step of decoding has a corresponding inverse scheme in the decoder. The general structures of the encoder and the decoder are presented in Figure 1.4 and 1.5. Then we introduce each part of the diagram. Most video coding standards adopt similar

coding schemes. Although the introduction is based on H.263, it also can be conform to other coding standards with slight modification.

Figure 1. 4 The encoder structure of H.263.

Figure 1. 5 The decoder structure of H.263.

1.1.2 Motion Compensation

The goal of motion compensation is to remove the temporal redundancy between successive frames by forming a predicted frame and subtracting this frame from the currently encoding frame. The output of this process is a difference frame, which is referred to as residual frame, and the more accurate the prediction process is, the less energy is remained in the residual frame.

Block matching algorithm (BMA) is a simple and effective motion compensation method that is adopted by most video coding standards. The procedure of BMA is explained in Figure 1.6. Prior to the implementation of BMA, each frame is divided into rectangular block that is $M \times N$ sample region. Then the following procedure is carried out:

1. **Motion Estimation**--Search an area in the reference frame (past or future frame) to find a best matching $M \times N$ sample region. This is carried out by comparing the $M \times N$ block in the currently encoding frame with some possible $M \times N$ regions in the search area, which is usually a region centered on the currently encoding block, and finding the region that is the best matching with the encoding region, as shown in Figure 1.7.
2. **Motion Compensation**--The chosen candidate region becomes a predictor for the currently encoding $M \times N$ block and is subtracted from the current block to form a residual $M \times N$ block.
3. The residual block is encoded and transmitted. The offset, which is referred to as *motion vector*, between the currently encoding block and the position of the candidate region is also transmitted.

Figure 1. 6 Motion estimation and compensation block diagram.

This procedure can also be referred to as *motion compensated prediction*. In most

coding standards, the *macroblock*, corresponding to a 16×16 pixel region in a frame, is the basic unit for motion compensated prediction. As we mentioned before, the frame has two formats, one is intra-frame, and the other is inter-frame. The macroblock also has two different formats, intra-macroblock and inter-macroblock. All macroblocks in intra-frames must be intra-macroblocks. But there may be some intra-macroblocks in the inter-frames. The intra-macroblock is not predicted from reference frame. The pixels within intra-macroblock are directly transformed. It means that the motion compensated prediction is only utilized in the inter-macroblock encoding.

Figure 1. 7 Motion estimation.

The generally used search area is a 32×32 region in reference frame. And the most accurate search method is full-search algorithm. It searches all possible locations inside the search window in the reference frame to provide an optimal solution. However, its high computational complexity makes it often not suitable for real-time implementation. To solve this problem, many fast motion search algorithms have been proposed [8-13]. For these fast motion search algorithms, they assume that the block difference increases monotonically as the checking point moves away from the global minimum difference point. Thus, in each searching step, these fast motion search algorithms would choose the direction such that the block difference of the checking point in that direction is the minimum. A most widely used fast search algorithm is three-step search. It searches 9 points in the first step, and the step size is half of the search region. The point that results in minimum block difference will be the center

point in the next step, and the step size is halved in the second step. The third step is iteration of the second step, and the step size is a single pixel.

Figure 1. 8 Three-step search algorithm.

The criterion used to evaluate the block matching can be *mean square error*, *mean absolute error*, or *sum of absolute difference*. The mean square error (MSE) provides a measure of energy remaining in the difference block. MSE for $N \times N$ sample block can be calculated as Equation (1.1).

$$MSE = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (C_{ij} - R_{ij})^2 \quad (1.1)$$

where C_{ij} is a sample of currently encoding block, and R_{ij} is a sample in searching area.

Mean Absolute Error (MAE) provides a reasonably good approximation of residual energy and is easier to be calculated than MSE, since it requires an absolute calculation instead of a square calculation for each pair of samples.

$$MAE = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |C_{ij} - R_{ij}| \quad (1.2)$$

The comparison may be further simplified by neglecting the term $1/N^2$. And only the sum of absolute differences (SAD) is calculated.

$$SAD = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |C_{ij} - R_{ij}| \quad (1.3)$$

1.1.3 2-D Discrete Cosine Transform

The purpose of the transform stage in video encoder is to convert pixel value or motion-compensated residual data into another domain (the frequency domain). The choice of the transform depends on a number of criterions:

1. Data in the frequency domain should be de-correlated and compact. That means the data after transform should be minimal inter-dependent, and most of the energy in the transformed data should be focused on a small number of data.
2. The transform should be reversible.
3. The transform should be computationally tractable.

In H.263, the 2-D discrete cosine transform operating on a block of 8×8 samples is adopted. Block transform has low memory requirement and is very suitable to the compression of block-based motion-compensated residuals.

1.1.4 Quantization

The principle of compression in DCT-based coder is the quantization. One property of DCT is that most energy concentrates on the low frequency coefficients, which are located at the left-top area of transformed block as shown in Figure 1.9. For human

visual system, the low frequency coefficients are more important than the high frequency coefficients, thus the higher frequency coefficients can be more coarsely quantized or discarded. The implementation of quantization is to divide the DCT coefficients by corresponding elements in a quantization matrix, as described in Equation (1.4). Y is a matrix of DCT coefficients, and Z is the quantization matrix defined by Equation (1.5).

Each element of the quantization matrix can be any integer that is between 1 to 255, and defines the quantization step for the corresponding DCT coefficient. A quantization matrix with all ones will result in nearly lossless compression. In general, the larger the elements are, the bigger the loss is, and the better compression efficiency can be achieved.

Large quantization step will improve the compression efficiency. However, it may also cause obvious visual artifacts, such as blocking, and cause video distortion in the decompressed video. A good quantization matrix has to balance the need for low bit rate with the need for good video quality.

Figure 1. 9 The distribution of DCT coefficients in a block.

$$\text{Quantization: } Y'_{ij} = \text{round}[Y_{ij} / Z_{ij}] \quad (1.4)$$

$$Z = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 110 & 103 & 99 \end{bmatrix} \quad (1.5)$$

1.1.5 Reordering and Entropy Coding

After quantization, the transform coefficients are required to be encoded as compactly as possible, prior to storage and transmission. In a transform-based video encoder, the output of the quantizer is an array containing a few nonzero coefficients and a large number of zero coefficients. Reordering and efficient representation of zero coefficients are applied prior to entropy encoding.

The significant DCT coefficients of a block are typically the low frequency coefficients and the DC coefficient. After quantization, the DCT coefficients of a block are reordered to group the nonzero coefficients together. This can enable the efficient representation of the remaining zero coefficients. The optimum reordering route of a block depends on the distribution of nonzero DCT coefficients. For a typical block with a distribution similar to Figure 1.9, a suitable scan order is a zig-zag. Starting from the DC coefficient, each quantized coefficient is arranged into a one-dimensional array, as shown in Figure 1.10. Nonzero coefficients tend to be grouped together at the beginning of the reordered array, followed by long sequences of zero coefficients.

There will be a large number of zero coefficients in the end of reordered array. And these zero coefficients can be represented more compactly by the *run-length* coding. The *run* indicates the number of zeros preceding a nonzero coefficient, and the *length* indicates the magnitude of the nonzero coefficients.

Figure 1. 10 Zig-zag scan order.

The entropy encoder converts a series of symbols that represent the elements of the video sequence into a compressed bitstream suitable for transmission or storage [14-16]. Input symbols may include quantized transform coefficients in run-length format, motion vectors, markers that indicate resynchronization point, headers and supplementary information. Most entropy coding methods are variable-length coding (VLC), which maps the input symbols to a series of codewords, and different codewords may have various lengths of bits.

A widely used entropy coding technique is Huffman variable length coder [17-19]. Huffman coding assigns a VLC to each symbol based on the occurrence probability of the symbol.

1.1.6 Decoding

Each step of decoding procedure has a corresponding step in encoding. There is an entropy coding table in the decoder. By utilizing this table, the decoder can convert the entropy codewords into original symbols. Then the decoded symbols are reordered to obtain the original sequence, and are de-quantized by the corresponding quantization

table. In the encoder side, since only the quotient without residual is encoded in bitstream, the de-quantization cannot reconstruct the symbols exactly. And an inverse DCT is operated on the symbols. For the intra-frames or intra-macroblocks, they do not have reference frame and can be decoded directly. For the inter-frames, the previously decoded frames are stored in the decoder buffer and used as reference frames.

1.2 H.264

The aim of this thesis is to improve the error resilient performance of video compressed by H.264. In order to make the later parts of this thesis easy to understand, a detailed introduction of H.264 is presented in this section.

(a)

(b)

Figure 1. 11 Basic structures of H.264. (a) H.264 encoder. (b) H.264 decoder.

The encoder includes two dataflow routes, one is **forward** path and the other is **backward** path [2]. The dataflow path in the decoder is similar to the encoder. Before illustrating the details of H.264, we will describe the main steps in encoding and decoding a frame of video.

In the forward path, an input frame F_n is processed in units of a macroblock denoted as M_u , which is a region of 16×16 samples. Each macroblock M_u is encoded in intra or inter mode. For each block in the macroblock M_u , a prediction is implemented independently based on the previously encoded frame. In intra mode, prediction is based on samples in the current slice that have previously encoded and decoded.

In inter mode, prediction is formed by motion-compensated prediction from reference frame. In Figure 1.11(a), the reference picture is shown as the previous frame F'_{n-1} , but the reference frame for each macroblock partition may be chosen from a series of past or future pictures that have been encoded already. The prediction is to subtract the current block from a region of samples in the reference frame to produce a residual block D_n . Then D_n is transformed and quantized to give a set of quantized transform coefficients X , which are reordered and entropy encoded. The entropy-encoded coefficients, together with the head information required to decode each block within the macroblock M_u , form the compressed bitstream. Then the bitstream is passed to a Network Abstraction Layer (NAL) for transmission or storage.

In the backward route, the encoder decodes the macroblock M_u to provide a reference for future prediction. The coefficients X are de-quantized and inverse transformed to produce a difference block D'_n . The prediction block is added to D'_n to create a decoded block. A filter is applied to reduce the effects of blocking distortion, and the reconstructed reference frame is created.

The decoder receives a compressed bitstream from the NAL, and entropy decodes the data elements to produce a set of quantized coefficients X . There are de-quantization and inverse transform to give D'_n (identical to the D'_n shown in the encoder). Using the header information decoded from bitstream, the decoder creates a prediction block, identical to the original prediction block formed in the encoder. Then the prediction block is added to D'_n to produce M_u .

1.2.1 Motion Compensated Prediction

In the previous section, we have introduced the fundamental motion compensated prediction procedure. To achieve better compression efficiency, H.264 adopts more complexity motion compensated prediction scheme.

1.2.1.1 Variable Block-Size Motion Compensation

H.264 supports more flexibility in the selection of motion compensation block sizes and shapes than any previous standard, with minimum luminance motion compensation block size as small as 4×4 .

Figure 1. 12 Different macroblock division modes supported in H.264.

In the previous version of H.264, which is referred to as H.26L, there are 7 different macroblock division modes supported in motion compensation as shown in Figure 1.12. And in the later version of H.264, the macroblock division scheme becomes even more flexible.

Figure 1. 13 Segmentation of a macroblock in H.264 for motion compensation. Top line: segmentation of macroblock, bottom line: segmentation of 8x8 partitions.

In Figure 1.13, various block shapes used in the new version of H.264 for motion compensated prediction are specified. Partitions with luminance block size of 16×16 , 16×8 , 8×16 , and 8×8 samples are supported by the syntax. In case partition with 8×8 samples is chosen, one additional syntax element for each 8×8 partition is transmitted. This syntax element specifies whether the corresponding 8×8 partition is further partitioned into the partitions of 8×4 , 4×8 , or 4×4 luminance samples.

The prediction block for each predictive-coded $M \times N$ luminance block is obtained by displacing an area of corresponding reference picture, which is specified by a motion vector and a picture reference index. Thus, if the macroblock is coded using four 8×8 partitions and each 8×8 partition is further divided into four 4×4 blocks, a maximum of 16 motion vectors may be transmitted for a single 16×16 macroblock.

In order to conform to the different block size in motion compensation, a 4×4 integer transform that has the similar efficiency as discrete cosine transform is used in H.264 [22] [23]. The transform and inverse transform matrices used in H.26L, which is

an old version of H.264, are presented in Equation (1.6). And the transform and inverse transform matrices used in the new version of H.264 are presented in Equation (1.7).

$$C_{transform} = \begin{bmatrix} 13 & 13 & 13 & 13 \\ 17 & 7 & -7 & -17 \\ 13 & -13 & -13 & 13 \\ 7 & -17 & 17 & -7 \end{bmatrix} \quad C_{inverse-transform} = \begin{bmatrix} 13 & 17 & 13 & 7 \\ 13 & 7 & -13 & -17 \\ 13 & -7 & -13 & 17 \\ 13 & -17 & 13 & -7 \end{bmatrix} \quad (1.6)$$

$$C_{transform} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \quad C_{inverse-transform} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix} \quad (1.7)$$

1.2.1.2 Quarter-Accurate Motion Compensation

The motion compensation accuracy is in unit of one-quarter of the distance between luminance samples. In case the motion vector points to an integer-sample position, the prediction signal consists of the corresponding samples in the reference picture. Otherwise the corresponding sample is obtained by using interpolation to generate non-integer positions. The prediction values at half-sample positions are obtained by applying a one-dimensional 6-tap FIR filter horizontally and vertically.

Prediction values at quarter-sample positions are generated by averaging samples at integer and half sample positions. Figure 1.14 illustrates the fractional sample interpolation for samples a-k and m-w. The samples at half sample position labeled b and i are derived by first calculating intermediate values b_1 and i_1 respectively as following:

$$\begin{aligned} b_1 &= (E - 5F + 20G + 20H - 5I + J) \\ i_1 &= (A - 5C + 20G + 20M - 5R + T) \end{aligned} \quad (1.8)$$

Figure 1. 14 Interpolation for fractional-sample accurate motion compensation. Capital letter indicates full-sample grid, and the lower case presents fractional sample.

The final prediction values for location b and i are obtained as follows and clipped to the range of 0 to 255.

$$\begin{aligned} b &= (b_1 + 16) \gg 5 \\ i &= (i_1 + 16) \gg 5 \end{aligned} \quad (1.9)$$

The samples at quarter sample positions labeled as j are obtained by Equation (1.9), where intermediate values denoted as cc , dd , n_1 and ff are obtained in a manner similar to i_1 .

$$j_1 = cc - 5dd + 20i_1 + 20n_1 - 5ee + ff \quad (1.10)$$

The final prediction value k is then computed by Equation (1.11) and clipped to the range of 0 to 255.

$$k = (k_1 + 512) \gg 10 \quad (1.11)$$

1.2.1.3 Multiple Reference Pictures

The use of multiple frames for motion compensation in most cases provides significantly improved prediction gain [24]. Previous coding standard used only one previously encoded picture to predict the encoding frame. The new design of H.264 extends the selection region of reference frame to improve the coding efficiency. The implementation of the multiple reference pictures is presented in Figure 1.15. The same extension of referencing capability is also applied to motion-compensated bi-prediction.

Multi-frame motion compensated prediction requires both the encoder and the decoder to store the reference pictures used for inter prediction in a multiple frame buffer. The decoder duplicates the multiple-frame buffer in the encoder according to memory management control operations specified in the bitstream. And the indexes of the reference frames that are in the multi-frame buffer have to be signaled. This reference frame index is transmitted for each motion-compensated luminance block.

Figure 1. 15 Multi-frame motion compensated prediction. The reference frame flag f will be transmitted with motion vector.

1.2.2 Intra Prediction

Each macroblock can be encoded in one of the several coding types. For intra macroblock, there are two types of intra coding supported by H.264, which are denoted as Intra_4×4 and Intra_16×16. The Intra_4×4 mode is based on predicting each 4×4 luminance block separately and is suitable for coding the area that has significant details. On the other hand, the Intra_16×16 mode performs prediction of the whole 16×16 luminance block and is more suitable for coding very smooth area of a frame.

In most previous coding standards, such as H.263+ and MPEG-4, intra prediction has been conducted in the transform domain. In H.264, the intra prediction is conducted in both spatial domain and transform domain. By referring to the neighboring samples of previously encoded blocks that are left or above to the encoding block, the encoding intra block is predicted. To prevent the error propagation due to motion compensated prediction of inter coding macroblock, the intra coding block can only be predicted by the neighboring intra macroblocks.

Figure 1.16 (a) Intra_4x4 prediction that is conducted for pixel A-P using neighboring samples a-q. (b) Corresponding prediction directions for Intra_4x4 prediction.

For the Intra_4×4 mode, each 4×4 block is predicted by the spatially neighboring samples. Eight directional prediction modes are presented in Figure 1.16, except the DC prediction mode. For the DC prediction mode, the DC coefficients of adjacent blocks

are used to predict the entire 4×4 block. For other prediction modes, the 16 samples within the 4×4 block, which are denoted as A-P, are predicted using prior decoded samples in neighboring blocks labeled as a-q. For illustration, by using the prediction mode 0 (vertical prediction), the samples above the 4×4 block are used to predict the encoding block, as indicated by arrows in Figure 1.17. The differences between the pixels in block and the neighboring pixels in vertical direction are calculated and transformed into coefficients. The prediction mode 1 (horizontal prediction) operates in a similar manner to vertical prediction except that the samples at the left of the 4×4 block are used.

Figure 1. 17 Five examples of Intra_ 4×4 block prediction modes.

When utilizing the Intra_ 16×16 mode, the whole luminance component of a 16×16 macroblock is predicted. Different from Intra_ 4×4 , only four prediction modes are supported. They are prediction mode 0 (vertical prediction), mode 1 (horizontal prediction), mode 2 (DC prediction) and mode 4 (plane prediction) [7].

1.2.3 Flexible Slice Size and Ordering

Unlike the rigid slice structure that is used in previous coding standards, slice size and ordering in H.264 are highly flexible. Slice is a sequence of macroblocks. In the previous standards, the slice is processed in the order of a raster scan generally, as shown in Figure 1.18. A picture may be split into one or several slices. Each slice contains a synchronization marker that makes the slice can be decoded independently. The macroblock information contained in each slice should not be correlative. That means a macroblock cannot be predicted by the macroblock that belongs to different slice.

In H.264, the number of macroblocks in each slice is not fixed. The slices of one frame can be sent and received in any order relative to each other. And the concept of slice group is introduced. A slice group may contain several slices.

Figure 1. 18 General slice structure when not using FMO.

Figure 1. 19 Using FMO to divide a QCIF frame into slices.

In order to enhance the robustness to data losses, the design of H.264 also supports flexible macroblock ordering (FMO). By using the FMO, the locations of macroblocks assigned in one slice can be random. Then different slices can form a slice group. The information in each slice group must be independent, but the information of slices that are in the same slice group can be correlative. Figure 1.19 gives two examples of the slice groups, in which each small square presents a macroblock. In the first example, the macroblocks circled by thick line are assigned to the same slice group. In the second example, the grey macroblocks are assigned to one slice group, and the dark macroblocks are assigned to the other slice group.

1.3 Objective and Contribution

The compressed digital video is often transmitted over communication networks, such as Internet and wireless communication network. Most communication networks are error-prone, and the correct transmission of video cannot be guaranteed. The compressed video is very sensitive to the transmission error, since the temporal and spatial redundancies have been removed during compression. The spatial and temporal prediction schemes increase the dependency between samples of video. When the samples within a small region have error, the other relative samples may be affected, and the error region may become larger in following frames. The video quality can be degraded significantly by a single error.

To alleviate the effect of transmission error, many error control schemes have been developed. The error resilience methods that are implemented by encoder are referred to as pre-processing, including forward error correction, resynchronization, multiple description coding and so on. By using the pre-processing method, some artificial redundancies are introduced to the compressed bitstream to prevent the bitstream from error. The error control schemes that aim to correct the error area in decoded video are

titled as error concealment. There are mainly two types of error concealment methods. One type is spatial error concealment that uses the spatial correlation of video to recover the error macroblocks. Another type is temporal error concealment that exploits the temporal relation to speculate the information of error macroblocks. In this thesis, several efficient error control methods have been proposed, including pre-processing methods, spatial error concealment algorithm and temporal error concealment approaches.

Since the compressed video may be transmitted over a wide range of communication networks, different types of transmission error may be considered. For the wireless network, the error format is burst error, in which the error bits are successive. Another error type is random error that the distributions of error bits are discrete and random. This is used as default error format in the thesis.

1.4 Outline of the Thesis

This thesis is organized as follows:

Chapter 2 A review of the existing error control methods is presented. Most popular error control approaches are introduced in this chapter. This review can make following chapters in this thesis easier to understand, since it can provide some background knowledge relative to the assumptions and implementations of the proposed methods.

Chapter 3 A spatial error concealment scheme is introduced in this chapter. This method is based on the smooth property of image. A function that describes the smooth connection of lost block and its neighboring blocks is constructed in transform domain, and the lost block is recovered by this function.

Chapter 4 A temporal error concealment approach is developed. A first-order plane is built by the motion vectors of blocks, which are neighbors to the lost

macroblock. Since the objects within a small area of a frame are likely to move in the same direction, we can use the first-order plane to approximate the motion vector of the lost macroblock.

Chapter 5 In this chapter, the Lagrange interpolation function is used to describe the motion tendency within a small area in frame. When the small area contains lost macroblock, the Lagrange function can be used to recover the motion vectors of the lost macroblock.

Chapter 6 A polynomial model is formulated by the motion vectors that belong to the neighboring blocks of the lost macroblock. The motion vectors of the lost macroblock can be recovered by this polynomial model, since it can reflect the relation of neighboring motion vectors.

Chapter 7 A content-based error resilience method is proposed. By using this method, some macroblocks in every n^{th} frame are encoded as periodic macroblocks. The probability of error propagation can be efficiently alleviated by these periodic macroblocks.

Chapter 8 A side information generation scheme for the multiple description coding is presented. Multiple description coding is a widely used error resilience method. The main problem of this method is the mismatch between the reference frames in encoder and decoder. To solve this problem and achieve better error resilient performance, we propose a new approach to generate side information, which is transmitted with each description.

Chapter 9 A conclusion is presented in this chapter, and future research directions relative to this thesis are proposed.

Chapter 2

Review

With the advances of multimedia system technology and wireless mobile communications, there has been a growing need for the support of multimedia services such as mobile teleconferencing, mobile TV, distance learning. These services require the real time transmission of video data over communication channels. Most of the communication channels used by current applications are error-prone. Unfortunately, the coded video data is highly sensitive to information loss and channel bit errors. The decoded video quality suffers dramatically at high bit error ratio. This quality degradation is exacerbated when no error control mechanism is utilized to protect the coded video against the error-prone environments. Single bit error could lead to disastrous quality deterioration. Moreover, the temporal and spatial prediction schemes used in most video coding standards cause the coded video more vulnerable to channel errors.

In order to minimize the impact of transmission error, error resilience coding methods and error concealment algorithms are often adopted in the codec. An error resilience coding technique has the capability to increase the portion of a corrupted bitstream that can be correctly decoded. The larger the correctly decoded portion of data is, the more resilient the coding is. Because the compressed video is utilized in

various applications, the error resilience techniques must deal with different error conditions. Then many different error resilience techniques have been proposed [25-27].

The error concealment schemes are post-processing methods, and are implemented while decoding. The spatial and temporal relationships of video are often utilized by error concealment method to recover the corrupted area of video. In this chapter, some novel error resilience and error concealment algorithms are reviewed.

2.1 Forward Error Correction

Forward error correction (FEC) is often employed to reduce the effects of error on the decoded video [28-31]. This error resilience scheme introduces redundant bits in the encoded video, therefore the error detection and correction capability of FEC are carried out by the cost of bit rate overhead. Since the bandwidth of the communication network is limited, the video encoder has to reduce the bits used for source coding to accommodate the channel coding bits for error protection purposes. This process may impair the quality of the decoded video in error-free environment. The best compromise between the error resilient performance and the source video quality is to make the bit cost of a FEC scheme adaptable to varying network conditions. To achieve this compromise, rate-distortion optimization equation is often adopted in the FEC schemes [32].

Figure 2. 1 Reed-Solomon code.

Figure 2. 2 Multicast system over wired and wireless network.

Reed-Solomon (RS) is a widely used FEC code [33] [34]. For a package that is N bits length, if the length of source coding is equal to k , then the RS code is $N-k$ bits. When the number of error bits does not exceed $N-k$, the package will be decoded correctly. Figure 2.1 shows an example of Reed-Solomon coding.

(a)

(b)

Figure 2. 3 The data dependence of scalable video coding. (a) no enhancement layer in prediction. (b) using enhancement layer in prediction.

Frossard and Verscheure [35] proposed an optimal rate allocation scheme for the

MPEG-2 coding and FEC rates, under a total given bandwidth. The optimality is defined in terms of minimum visual distortion given a set of video and network parameters. Another bandwidth allocation strategy is presented by Lee *et al* in [36], which is designed for multicast video transmission over wired and wireless network, as shown in Figure 2.2. The server in a multicast system adapts the bandwidth and FEC code of each layer so as to maximize the overall video quality.

The simple FEC technique normally assigns equal amount FEC to each data, regardless of their different sensitivities to errors and their contributions to overall video quality. For example, when the equal FEC technique is used, all transformed coefficients of a macroblock will be assigned the same amount of FEC, although the DC coefficient is more important than the AC coefficients. To make the FEC more efficient, unequal FEC methods are proposed [37]. The basic idea of unequal FEC is that the important data will receive more protection than the less important data.

Kim *et al* [126] proposed a FEC-based bit-plane-wise unequal error protection algorithm. An unequal amount of FEC is assigned to each bit-plane. Considering the delay constraints of Internet video transmission, the unequal FEC algorithm must be simple and fast. The 2-D and 3-D SPIHT algorithm is used to generate the FEC in bitstream.

In order to implement the unequal FEC, Yap *et al* [139] proposed a modified H.263 encoder. In this modified encoder, the source significance information (SSI) is generated during the source coding procedure. The SSI is used to indicate the priority of data. The data with high priority will receive more protections. The SSI will not be transmitted to the decoder, therefore it will not increase the bitrate. Rate compatible punctured convolutional code is used as channel coding. According to the SSI, different amounts of redundancies are added to the source coding data. The entropy coding of the encoder is also modified. By reordering the variable length codewords into fixed length slots of data, the loss of resynchronization caused by variable length coding can be eliminated.

Unequal FEC is often adopted in the scalable video coding [38] [39]. For the scalable video transmission, the base layer is the most important layer, and it must be transmitted correctly. Otherwise, the encoded video cannot be reconstructed even all enhancement layers are correctly received. The relation between base layer and enhancement layer is presented in Figure 2.3. The layer 0 represents the base layer, which must be transmitted correctly. The layer 1 and layer 2 are the enhancement layers that are dependent on its bottom layer. The motion estimation is adopted in the scalable video coding. The encoding frame can be predicted by the previous frame, which is reconstructed by only base layer or all layers. The base layer will be assigned a large amount of FEC to make sure it can be correctly decoded. And fewer FEC will be assigned to the enhancement layers, since they contribute less to the overall video quality.

Figure 2. 4 Layered FEC for multicast framework.

In [38], a layered FEC is employed as an error-control mechanism in a layered multicast framework. The structure of this layered FEC is presented in Figure 2.4. An equation-based scheme is used to measure the network characteristics. The video transmitted through the channel that has high loss probability or high latency will have more protections. The FEC streams are only multicast to the receivers that need them.

Mohr *et al* [40] presented an unequal loss protection scheme, in which unequal amounts of forward error correction are applied to progressive data to provide graceful degradation of image quality as packet losses increase. The Set Partitioning in Hierarchical Trees is used in this scheme to find a good assignment of the FEC code.

Horn *et al* [138] proposed another unequal error protection approach for the scalable video, which adopts the Newton's interpolation. In the encoder, the bistreams of different layers are multiplex to create a block of packets (BOP) that contains n packets of bits. The bits used for source coding are filled into k packets in each BOP. Then the remaining $(n-k)$ packets are filled by Reed-Solomon channel coding. The symbol of channel coding is calculated by the Newton's interpolation. If less than k packets are received by the decoder, some simple error concealment schemes are utilized. In the enhancement layer, the lost area of image is replaced by relative image in the base layer. If there is error in base layer at this time instant, the lost part is copied from the last decoded enhancement layer frame.

The FEC can be used together with ARQ (automatic repeat request) to achieve better error resilient performance [41] [42]. In [43], Kallel proposed a method that combines the ARQ protocols with adaptive FEC. The basic idea of this method is to adjust the coding rate of error correction according to the system parameters, such as SNR, round trip delay and buffer size. Then the throughput efficiency can be maximized.

2.2 Resynchronization

In most video coding standards, variable length codeword is adopted to achieve higher coding efficiency. However, when transmission error occurs, this technique may cause the error propagate in both of spatial direction and temporal direction. To decrease such error propagation, resynchronization markers are inserted at certain positions in

bitstream. When the decoder detects an error, it can hunt for the next resynchronization marker and regain synchronization. In the coding standards H.261 and H.263, the location of resynchronization marker is fixed to the header of each slice, which is a row of macroblocks as shown in Figure 2.5. To enhance the error resilient performance of resynchronization, MPEG-4 supports flexible insertion of resynchronization marker. Then many algorithms about the insertion of the resynchronization marker are proposed [44-46].

Pascal and Verscheure [47] proposed a new error resilience scheme that adopts a scene-complexity adaptive mechanism, named Adaptive MPEG-2 Information Structure (AMIS). The AMIS modulates the number of resynchronization points in order to maximize the perceived video quality.

Figure 2. 5 Relation of synchronization markers and coded macroblocks in H.263 frame.

In [48], an error-robust and JPEG compliant progressive image compression scheme over wireless channel is presented. The use of the restart markers in the JPEG standard provides the resynchronization function for error handling. A restart marker regulation technique is exploited to preprocess restart markers at the decoding end. After decoding, the error between two restart makers can be detected and further processed by the error concealment to reduce image degradation.

The rate-distortion optimized mode selection scheme and the algorithm of synchronization marker insertion are introduced by Cote *et al* [49]. They describe an effective method for increasing error resilience of video transmission over bit error-prone networks. The proposed algorithm takes into account the channel condition and the error concealment method used by the decoder to optimize video coding mode selection and placement of synchronization markers.

In Yoo's paper [50], the resynchronisation marker positioning problem is investigated. A new cost function of the error propagation is formulated. Then the optimum positions of resynchronization markers are derived based on the cost function.

2.3 Error Concealment

The error resilience methods that we introduced in previous sections improve the error resilient performance of compressed video at the expense of bit rate. When the bandwidth of transmission channel is very limited, there may not have enough bits to be allocated to error protection code. To solve this problem, many post-processing schemes are proposed. These post-processing schemes are often referred to as error concealment methods, which are implemented in the decoder. Since the video has correlation in spatial and temporal domains, the error concealment method can exploit these correlations to reconstruct the lost or corrupted area in video. Of course, the reconstructed area may not be exactly the same as the original video, but the visual quality of the decoded video can be efficiently improved.

Most error concealment methods can be divided into two categories: spatial error concealment and temporal error concealment. For most natural images, the pixel values within a small area are very close. By making use of this property, the lost pixels or blocks can be approximated by the nearby pixels. The methods that exploit the

information of neighboring area to recover the lost information are referred to as spatial error concealment method.

The methods that use the temporal information to recover the lost part of compressed video, and the schemes that aim to recover the temporal information of compressed video are referred to as temporal error concealment methods. Most temporal error concealment methods are designed to recover the lost motion vector. For an inter macroblock, the information of motion vectors is more important than the residual information, since the residual information is useless without motion vectors. If the decoder only has the motion vectors but no residual information, the video still can be reconstructed with a degraded quality.

Before the introduction of error concealment methods, the definitions of the *lost block* and the *corrupted block* should be explicated. When the whole information of a block is missing, the block is referred to as *lost block*. If only partial information of a block is lost, and the rest information of this block is still decodable, the block is a *corrupted block*.

2.3.1 Interpolation

Interpolation is a widely used spatial error concealment method. The pixels in the lost blocks are calculated as a weighted summary of neighboring pixels, and this procedure is called as interpolation. A simple interpolation method is presented in Figure 2.6 [51]. The boundary pixels in neighboring blocks are used in interpolation. In this method, four neighboring pixels are used in interpolation, and the values of interpolated pixels are calculated as Equation (2.1). $b(i,k)$ is the interpolated pixel. $b_x(i,k)$ are the four neighboring pixels, and d_x are the distances between the neighboring pixels and the interpolated pixel. The nearer neighboring pixels contribute more to the interpolated pixel.

Figure 2. 6 Linear interpolation.

$$b(i, k) = \frac{1}{d_R + d_L + d_B + d_T} \times (d_R b_L(1, k) + d_L b_R(N, k) + d_B b_T(i, 1) + d_T b_B(i, N)) \quad (2.1)$$

Figure 2. 7 Edge-oriented spatial interpolation algorithm proposed by Hsia.

When the lost block contains a sharp edge, this interpolation method may blur the edge. To avoid this problem, many shape-based interpolation methods have been presented [52] [53]. Hsia [54] proposed an edge-oriented spatial interpolation method. The implementation of this method is shown in Figure 2.7. The first step of this algorithm is to find the edge direction of a lost block by using one-dimensional matching techniques from two boundaries of the neighboring blocks, which is denoted as best match boundary in Figure 2.7. Then, the error pixels that are encircled by the

match boundaries are recovered by weighting linear interpolation along the estimated edge direction. Afterwards, the median filter is used to recover the residual damaged pixels.

Figure 2. 8 Multiple edge oriented interpolation.

Kwok *et al* [55] proposed a multi-directional interpolation algorithm, in which more than one edge direction is considered in interpolation. Figure 2.8 gives an example of this method. For illustration, there are two different edges in one macroblock. Then, it is assumed that there are two different objects in this macroblock. The interpolation is conducted along the edge directions respectively. That means only the neighboring pixel that is considered as in the same object is utilized in the interpolation.

Li and Orchard [56] introduced another edge-directed interpolation algorithm that is based on the covariance coefficients. First, the local covariance is estimated from a low-resolution image, and then this covariance is used to adapt the interpolation at a higher resolution based on the geometric duality between the low-resolution covariance and the high-resolution covariance. The edge-directed property of

covariance-based adaptation makes it capable for adjusting the interpolation coefficients to match an oriented edge.

2.3.2 Recovery of DCT coefficients

The interpolation is a spatial error concealment method that is used to recover the lost pixel. Some other spatial error concealment schemes aim at recovering the lost DCT coefficients. The smoothness property of natural image is often utilized to recover the lost DCT coefficients. For most natural images the neighboring pixels are very close. And the neighboring blocks should connect smoothly with each other. By making use of this smoothness property, a constraint function that describes the connection of neighboring blocks can be constituted in DCT domain [57-59]. Equation (2.2) gives an example of the constraint function that describes the connection. b_i, t_i, l_i and r_i represent the boundary pixel vectors of the lost block. $r_{i-1}, l_{i+1}, b_{i-p}$ and t_{i+p} denote the neighboring pixel vectors. Then a set of DCT coefficients that minimize the differences between the recovered block and its neighboring blocks can be recovered by this constraint function, as shown in Figure 2.9.

Figure 2. 9 The location of lost block and its boundary vectors.

$$\psi = \|t_i - b_{i-P}\|^2 + \|b_i - t_{i+P}\|^2 + \|l_i - r_{i-1}\|^2 + \|r_i - l_{i+1}\|^2 \quad (2.2)$$

Figure 2.10 Smoothing directions for an 8×8 block.

In [60] and [61], the constraint function based on the smoothness property is formulated in pixel level rather than block level. Not only the differences between macroblock boundaries are minimized, but also the differences of neighboring pixels are minimized. The constraint function is shown in Equation (2.3). $\hat{a}_k$ are the coefficients in block B , k represents the index of the coefficient. L represents the set of lost coefficients. $\hat{f}_{m,n}$ indicates the recovered pixel in block B . $w_m^w, w_m^w, w_m^w, w_m^w$ are smoothing weights, which are either one or zero depending on whether the variations at the point (m, n) in corresponding directions should be minimized or not. The arrows in Figure 2.10 indicate the smooth directions.

$$\begin{aligned} \Psi(\hat{a}_k, k \in L) = & \frac{1}{2} \sum_{(m,n) \in B} [w_{m,n}^w (\hat{f}_{m,n} - \hat{f}_{m,n-1})^2 + w_{m,n}^e (\hat{f}_{m,n} - \hat{f}_{m,n+1})^2 \\ & + w_{m,n}^n (\hat{f}_{m,n} - \hat{f}_{m-1,n})^2 + w_{m,n}^s (\hat{f}_{m,n} - \hat{f}_{m+1,n})^2] \end{aligned} \quad (2.3)$$

An inherent drawback of the maximally smoothness method is that the edge information of lost block will be blurred. To solve this problem, Zhu *et al* [62] proposed a second-order derivative-based smoothness measure equation. A weighted

combination of the quadratic variation and the Laplacian operator is employed as a second-order smoothness measure, to enable smoothness and sharp rendition of edges in various directions. The quadratic variation measured over surface is presented in Equation (2.4).

$$\Psi = \iint \left(\left(\frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 f}{\partial y^2} \right)^2 \right) dx dy \quad (2.4)$$

When this equation is used to measure the pixel difference within a block B that contains $m \times m$ samples, Equation (2.4) can be written in discrete domain. $f_{(i,j)}$ represents the pixel in block.

$$\begin{aligned} \Psi = & \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} (f_{(i-1,j)} - 2f_{(i,j)} + f_{(i+1,j)})^2 + \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} (f_{(i,j-1)} - 2f_{(i,j)} + f_{(i,j+1)})^2 \\ & + 2 \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} (f_{(i,j)} - f_{(i+1,j)} - f_{(i,j+1)} + f_{(i+1,j+1)})^2 \end{aligned} \quad (2.5)$$

Unlike the quadratic variation measure, the Laplacian-based smoothness measure only involves the second-order derivatives in the horizontal and vertical directions. In the continuous domain, it is defined as Equation (2.6). The discrete version is presented in Equation (2.7).

$$\Psi = \iint \left(\left(\frac{\partial^2 f}{\partial x^2} \right) + \left(\frac{\partial^2 f}{\partial y^2} \right) \right)^2 dx dy \quad (2.6)$$

$$\Psi = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} (f_{(i+1,j)} + f_{(i-1,j)} + f_{(i,j+1)} + f_{(i,j-1)} - 4f_{(i,j)})^2 \quad (2.7)$$

The hybrid measure in discrete domain that combines the quadratic variation and the Laplacian is shown in Equation (2.8).

$$\begin{aligned} \Psi = & \left[\sum_{i=0}^{m-1} \sum_{j=0}^{m-1} (f_{(i-1,j)} - 2f_{(i,j)} + f_{(i+1,j)})^2 + \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} (f_{(i,j-1)} - 2f_{(i,j)} + f_{(i,j+1)})^2 \right. \\ & + 2 \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} (f_{(i,j)} - f_{(i+1,j)} - f_{(i,j+1)} + f_{(i+1,j+1)})^2 \\ & \left. + \lambda \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} (f_{(i+1,j)} + f_{(i-1,j)} + f_{(i,j+1)} + f_{(i,j-1)} - 4f_{(i,j)})^2 \right] \end{aligned} \quad (2.8)$$

λ is a balance parameter that controls the relative amount of smoothing applied in the horizontal and vertical directions versus in the two diagonal directions. By using this combined measurement, the differences of neighboring pixels in variable directions can be minimized, but the sharp edge still can be maintained.

2.3.3 Boundary Matching

Boundary matching is a popular algorithm that is used to recover the lost motion vectors. An early boundary matching algorithm proposed by Lam *et al* [63] selects a motion vector from a set of candidate motion vectors, which minimizes the total variation between the boundaries of the reconstructed block and its neighboring blocks, to replace the lost motion vector. For example, as shown in Figure 2.11, if the $N \times N$ block M_5 is lost, the motion vectors of its neighboring blocks can be used as candidate motion vectors. Each candidate motion vector points to a corresponding block in the previous frame. Then each corresponding block in the previous frame can be used to reconstruct the lost block M_5 . With a recovered motion vector and the residual information, the pixels in M_5 can be recovered by Equation (2.9). $f_R(x, y, n)$ represents the recovered pixel located in the n^{th} frame, x and y are the coordinates of pixel.

$$\hat{f}_R(x, y, n) = f_R(x + \hat{d}_x, y + \hat{d}_y, n - 1) + f_D(x, y, n) \quad (2.9)$$

And the total difference C between the reconstructed block and its neighboring blocks will be calculated. The total difference C is defined by Equation (2.10). C_A, C_L and C_B represent the differences in top, left and bottom boundaries. The candidate motion vector that results in minimum total difference C will be used to replace the lost motion vector of block M_5 . To achieve better concealment result, the

author proposed that the residual information of the lost macroblock can be set to zero, or replaced by the residual information of neighboring blocks. And this method is referred to as extended boundary matching algorithm.

$$\begin{aligned}
 C_A &= \sum_{x=x_0}^{x_0+N-1} \left(\hat{f}_R(x, y_0, n) - f_R(x, y_0 - 1, n) \right)^2 \\
 C_L &= \sum_{y=y_0}^{y_0+N-1} \left(\hat{f}_R(x_0, y, n) - f_R(x_0 - 1, y, n) \right)^2 \\
 C_B &= \sum_{x=x_0}^{x_0+N-1} \left(\hat{f}_R(x, y_0 + N - 1, n) - f_R(x, y_0 + N, n) \right)^2 \\
 C &= C_A + C_L + C_B
 \end{aligned} \tag{2.10}$$

Figure 2. 11 The lost block M_5 and its neighboring blocks.

Based on the previous method, Park *et al* [64] proposed an improved boundary matching algorithm. If there is a sharp edge existing in the lost macroblock, the previous method will yield very large variation. In the improvement method, the differences between diagonally connected pixels are taken into account. For example, in the n^{th} frame, the difference in top boundary is defined in Equation (2.11).

$$\begin{aligned}
 Diff_1(x, y_0) &= \hat{f}_R(x, y_0) - f_R(x - 1, y_0 - 1) \\
 Diff_2(x, y_0) &= \hat{f}_R(x, y_0) - f_R(x, y_0 - 1) \\
 Diff_3(x, y_0) &= \hat{f}_R(x, y_0) - f_R(x + 1, y_0 - 1) \\
 Diff_4(x, y_0) &= \hat{f}_R(x, y_0) - \bar{f}_R(x - 1, y_0) \\
 Diff_5(x, y_0) &= \hat{f}_R(x, y_0) - \bar{f}_R(x + 1, y_0)
 \end{aligned} \tag{2.11}$$

where $\bar{f}_R(x-1, y_0)$ and $\bar{f}_R(x+1, y_0)$ are sub-pixels, the values are calculated as:

$$\begin{aligned}\bar{f}_R(x-1, y_0) &= \frac{1}{4} \times (f_R(x-1, y_0-1) + f_R(x, y_0-1) \\ &\quad + \hat{f}_R(x-1, y_0) + \hat{f}_R(x, y_0)) \\ \bar{f}_R(x+1, y_0) &= \frac{1}{4} \times (f_R(x+1, y_0-1) + f_R(x, y_0-1) \\ &\quad + \hat{f}_R(x+1, y_0) + \hat{f}_R(x, y_0))\end{aligned}\quad (2.12)$$

Then the total difference is re-defined as:

$$\begin{aligned}C'_A &= \sum_{x=x_0}^{N-1+x_0} \text{Min}\{|Diff_1(x, y_0)|, |Diff_2(x, y_0)|, \\ &\quad |Diff_3(x, y_0)|, |Diff_4(x, y_0)|, |Diff_5(x, y_0)|\} \\ C' &= C'_A + C'_L + C'_B\end{aligned}\quad (2.13)$$

Jian *et al* [65] also proposed an algorithm to maintain the edge information of corrupted area. The boundary difference is adapted according to the different edge orientations. The pixels that are adjacent to the lost macroblock are used to determine the edge orientation, and then decide which boundary information is used in the estimation of the lost block.

Chen *et al* [66] presented a refined boundary matching algorithm. The previous boundary matching algorithms estimate only one motion vector for a damaged macroblock by minimizing the boundary difference. If there are two different objects existing in the lost area, these methods may generate edge fragmentation, object deformation and other noticeable artifacts. Then they present a framework that using various motion vectors for different regions within a damaged macroblock. The damaged macroblock is divided into small blocks, and each of the blocks is concealed with an estimated motion vector. The estimations of motion vectors are independent, but constrained.

In [67], an iterative weighted boundary matching method is introduced. For variable length coded data, a transmission error in a codeword affects not only the underlying codeword but also the subsequent codeword. The objective of this approach

is to minimize the total boundary error of all lost macroblocks in one inter frame. In the first recovery stage, extended boundary matching algorithm that is proposed in [63] is used. Then the recovered frame is used for the following stage iteratively. In order to avoid the dependent recovery problem, uncertain parameter is imposed on the boundary matching criterion. The error concealment stage will iterate until the total sum of boundary error in one frame is minimized.

Tsai *et al* [68] proposed a low complexity error concealment algorithm using progressive interpolation and boundary matching. B frame can be encoded by four different modes, as shown in Figure 2.12. It can be predicted backward, forward, or bi-directional. By using the bi-directional predicted frame as reference frame, the error propagation effect in inter frame can be reduced largely. The fourth mode is direct coding mode, in which the B frame is coding with bi-directional property. The motion vector in this coding mode is a linear relationship between the past P frame and the future P frame.

(a)

(b)

(c)

(d)

Figure 2. 12 Four coding modes of B frame. (a) Forward mode. (b) Backward mode.

(c) Interpolation mode. (d) Direct mode.

When a packet of P frame is lost, the macroblocks in the neighboring B frame that are co-located with the corrupted macroblocks will be found. Then the coding modes of these co-located macroblocks are identified. A corrupted macroblock can be recovered by interpolation or boundary matching algorithm according to its coding mode.

In [69] and [70], boundary matching algorithms that are based on multiple reference frames are presented. The search range of boundary matching is extended from single reference frame to multiple reference frames. Then it is more probable to find a similar macroblock to replace the lost macroblock.

2.3.4 Block Matching

The block matching algorithm is similar to the boundary matching algorithm. Instead of the boundary information, information of the blocks that are nearby the corrupted area is exploited.

$$MAD(dx, dy) = \frac{1}{P \times Q} \sum_{x=0}^{P-1} \sum_{y=0}^{Q-1} |B_{TOP}(x, y) - B'_{TOP}(x + dx, y + dy)| \quad (2.14)$$

Chen *et al* [71] proposed a recursive block-matching principle for error concealment. For video signal, there exists significant statistical redundancy between adjacent video frames in temporal domain. The proposed algorithm makes use of the temporal redundancy of a video sequence by a block matching approach. Based on the two-step block-matching principle, a recursive block-matching algorithm is presented. In the first step of algorithm, the upper part of lost macroblock is used to perform temporal matching for all neighboring candidates in a $M \times N$ searching window of a reference frame. The displacement leads to MAD minimization is used to conceal upper part of the lost macroblock.

In Equation (2.14), P and Q indicate the size of upper part, and B'_{TOP} is the candidate with displacement of (dx, dy) .

A weighted MAD (WMAD) is calculated to decide the displacement of bottom part in the lost macroblock, as the following equation:

$$\begin{aligned} WMAD(dx, dy) &= w_1 \times MAD_{TOP}(dx, dy) + w_2 \times MAD_{BOT}(dx, dy) \\ MAD_{TOP}(dx, dy) &= \frac{1}{P \times Q} \sum_{x=0}^{P-1} \sum_{y=0}^{Q-1} |C_{TOP}(x, y) - C'_{TOP}(x + dx, y + dy)| \\ MAD_{BOT}(dx, dy) &= \frac{1}{P \times Q} \sum_{x=0}^{P-1} \sum_{y=0}^{Q-1} |B_{BOTTOM}(x, y) - B'_{BOTTOM}(x + dx, y + dy)| \end{aligned} \quad (2.15)$$

where w_1 and w_2 are set to 0.5 and 1 respectively. C_{TOP} is the concealed upper half macroblock.

Then recursive calculations of the weighted MAD for the upper half-macroblock and bottom half-macroblock are executed by following equations.

$$\begin{aligned} WMAD(dx, dy) &= w_1 \times MAD_{TOP}(dx, dy) + \\ &w_2 \times MAD_{MED}(dx, dy) + w_3 \times MAD_{BOT}(dx, dy) \\ MAD_{TOP}(dx, dy) &= \\ &\frac{1}{I_P \times I_Q} \sum_{x=0}^{P-1} \sum_{y=0}^{Q-1} |B_{TOP}(x, y) - B'_{TOP}(x + dx, y + dy)| \end{aligned} \quad (2.16)$$

$$\begin{aligned} WMAD(dx, dy) &= w_1 \times MAD_{TOP}(dx, dy) + \\ &w_2 \times MAD_{MED}(dx, dy) + w_3 \times MAD_{BOT}(dx, dy) \\ MAD_{BOTTOM}(dx, dy) &= \\ &\frac{1}{I_P \times I_Q} \sum_{x=0}^{P-1} \sum_{y=0}^{Q-1} |B_{BOTTOM}(x, y) - B'_{BOTTOM}(x + dx, y + dy)| \end{aligned} \quad (2.17)$$

Region of $I_P \times I_Q$ is combined by the top region and the bottom region of concealed macroblock.

(c)

(d)

Figure 2. 13 Schematic presentation of the split-matching algorithm.

In [72], an error concealment scheme that is based on block-matching principles and spatial-temporal video redundancy is presented. For the intra frame, spatial split-match error concealment method is adopted. In order to deal with the problem of reconstructing lost regions with high spatial detail, texture or edges, spatial region similarity principles have been applied in existing neighboring regions and concealment has been performed in the direction defined by the detected similar regions. The flowchart of this algorithm is shown in Figure 2.13. MB_L is the lost macroblock, and MB_T, MB_B are the neighboring macroblocks.

In the first step, the largest vertically neighboring blocks b_1 and t_1 of size 16×8 pixels are compared. If these regions are considered similar, the lost macroblock is

displaced by the macroblock that locates at the same place of previous frame. Otherwise the neighboring blocks are split into smaller region of 8×8 and compared, as shown in Figure 2.13b. When similar blocks are found, the corresponding block in previous frame is used in concealment. In step 3, the blocks in step 2 are further split into the smallest block, whose size is 4×4 . The smallest blocks to be matched (b_i with t_i) slide inside the search region and all their combinations are considered for match determination. The concept of sliding window ensures edge or line reconstruction at any direction. In the final step, the search region of step 3 is enlarged, as shown in Figure 2.13d.

Mean absolute difference (MAD) can be used to evaluate the region similarity. For example, the MAD between neighboring blocks B_T and B_B of size $b_x \times b_y$ can be estimated as:

$$MAD = \frac{1}{b_x \times b_y} \sum_{x=0}^{b_x} \sum_{y=0}^{b_y} |B_T(x, y) - B_B(x, y)| \quad (2.18)$$

For the inter frame, temporal block matching error concealment method exploits the temporal redundancy of a video sequence. The assumption of a smooth and uniform motion for at least some adjacent blocks is adopted. Temporal block matching is performed between the top and bottom macroblocks, which is represented as MB_T and MB_B respectively, and the equally sized blocks locate in a predefined search region in the reference frame. The center of the search region is MB_L that is the lost macroblock, as shown in Figure 2.14. Neighboring candidates for block matching are either the available top neighboring macroblock MB_T , the available bottom macroblock MB_B , or the combination of both. In the latter case, their vertical distance is restricted to 16 pixels, and the movement of MB_T and MB_B in the search region is identical. Temporal matching is performed for all neighboring candidates in a predefined search region of size $N \times M$, and the one that leads to minimum MAD is

used to estimate the lost macroblock content. Concealment is performed by replacing the lost macroblock area by the image content of the appropriate neighbor in temporal.

Figure 2. 14 Temporal matching of top and/or bottom available macroblock.

2.3.5 Other Error Concealment Methods

2.3.5.1 Spatial Error Concealment Methods

Zhang *et al* [73] proposed a content-adaptive spatial error concealment method. By using the edge information extracted from the surrounding blocks, each error block is classified into one of three categories with different contents. The three categories are:

- Uniform block: the gray level of error block may be constant or nearly so.
- Edge block: the block locates on the boundary of two or more objects with different gray level.
- Texture block: both gray level and edge direction vary significantly in the block.

Figure 2. 15 Content-based spatial error concealment.

Figure 2. 16 Conceal the edge block by directional interpolation

Then the error block is reconstructed by appropriate methods according to the block category, as show in Figure 2.15. The uniform block is concealed by bilinear interpolation. The texture block is recovered by neighboring matching error concealment. And edge block is concealed by directional interpolation, as shown in Figure 2.16. For example, the interpolation of pixel p along the given direction θ is:

$$f_{\theta}(p) = \left[\sum_{x_n \in M} \frac{f_{\theta}(x_n)}{D_n^2} \right] / \sum_{x_n \in M} \frac{1}{D_n^2} \quad (2.19)$$

where M is the set of available neighboring pixels on the scanning line. x_n is the n^{th} pixel of M . D_n represents the distance from x_n to p .

Lee *et al* [74] introduced a new method that uses motion estimation to consider actual motions, such as rotation, magnification, reduction and parallel motion. Since many videos include a variety of three-dimensional motions, the proposed method established an affine transform to estimate the motion of lost data more accurately. The affine transform is based on geometric transform that is applicable to a list of match points from two images. The transform equation that includes complex three-dimensional motions can be expressed as:

$$\begin{aligned} x' &= (x \cos \theta + y \sin \theta)S_x + T_x \\ &= (S_x \cos \theta)x + (S_x \sin \theta)y + T_x \\ y' &= (-x \sin \theta + y \cos \theta)S_y + T_y \\ &= (-S_y \sin \theta)x + (S_y \cos \theta)y + T_y \end{aligned} \quad (2.20)$$

Shirani *et al* [75][76] exploited a maximum a posteriori (MAP) estimator with an adaptive Markov random field (MRF) as a priori model of the image, to improve the video reconstruction quality. The adaptive model enables the estimation to incorporate information embedded not only in the immediate neighboring pixels but also in a wider neighborhood into the reconstruction procedure without increasing the order of the MRF model.

Zhang *et al* [77] also used the MAP to recover the lost information of encoded digital video. In their method, both the spatial and temporal textual features in video

signals are separately modeled using the dual Markov random field (DMMRF). The lost information is then estimated using maximum a posteriori (MAP) probabilistic approach, which is based on the spatial and temporal MMRF models. To reconstruct the high frequency information of the damaged area, a new adaptive potential function is also introduced. The simulation results of MMRF are presented in Table 2.1. Compared with other methods, such as average motion vector (AVMV), boundary matching approach (BMA) and temporal-spatial Markov random field (TSMRF), the proposed method (DMMRF) can achieve better error concealment results.

Table 2. 1 Compare the performance of MMRF with other concealment methods

Method	PSNR (dB)
AVMV	30.4
BMA	30.0
TSMRF	30.5
DMMRF	31.2

Belfiore *et al* [78] proposed an error concealment method to preserve the edge and texture information. They developed a spatial error concealment technique, which combines edge-preserving interpolation and texture analysis, to provide a reconstruction of lost macroblocks optimized for visual perception. The edge information can be recovered by coarse-to-fine MAP estimation with a Markov random field prior, and the lost areas can be replenished with a texture synthesized by neighboring macroblocks.

Salama *et al* [79] proposed an error concealment scheme for the compressed video that is transmitted over ATM network. In ATM networks, channel errors or congestion cause data to be dropped, which results in the loss of entire macroblocks. In order to find the locations of lost macroblocks, a technique for packing the compressed data as ATM cell is presented. Then error concealment techniques based on a Markov random field (MRF) model are used to recover the lost macroblocks. The error concealment algorithms are categorized as being either spatial (utilizing pixel data) or temporal

(utilizing motion vectors) in nature. The spatial techniques rely on pixel data within a current damaged frame and a MRF model of frame to restore damaged areas. Similarly, the missing motion vectors are estimated by modeling the motion field as an MRF, and find the maximum a posteriori (MAP) estimate of each missing motion vector by giving its neighboring motion vectors. This temporal-spatial error concealment can efficiently improve the video quality, as shown in Table 2.2.

Table 2. 2 Average PSNR values for different error concealment schemes for the Flower Garden, Football, and Hockey sequences

(a) 2% ATM cell loss rate

Error Concealment Technique	Flower garden	Football	Hockey
Temporal Replacement	31.37	33.07	34.92
Average Motion Vector	33.32	33.58	36.07
Median Motion Vector	34.13	34.13	36.78
MAP estimation of Motion Vector	34.86	34.15	36.74
Temporal-Spatial Approach	35.33	34.96	38.73

(b) 5% ATM cell loss rate

Error Concealment Technique	Flower garden	Football	Hockey
Temporal Replacement	28.45	29.83	31.94
Average Motion Vector	30.27	30.26	33.19
Median Motion Vector	31.16	30.88	33.92
MAP estimation of Motion Vector	31.64	30.62	33.53
Temporal-Spatial Approach	32.10	31.57	35.64

Figure 2. 17 Error resilient image coding bitstream structure.

Figure 2. 18 Post-processing error-concealment decoding flow.

To achieve better concealment performance, Lee *et al* [80] developed an algorithm that combines the pre-processing and post-processing methods. In their method, an error resilient image coding for DCT based image compression is used in encoder. The detail of this error resilience method is shown in Figure 2.17. Block align information (BAI) is used to specify the bit count of each encoded block. Each block is encoded by the similar manner of JPEG, except for using fixed length DC and removing the end of block symbol. This method can successfully prevent errors from propagating across image block boundaries with little overhead.

In the decoder, a novel error concealment scheme is proposed to retain low-frequency information and discard suspicious high-frequency information. Low-resolution information, rather than total corruption, can be obtained during the decoding process. Figure 2.18 is the flowchart of this error concealment method.

Zhang and Wang [81] proposed an information-restoration algorithm. In their method, not only the local information but also the information in the remote regions of image is used in concealment. The method originates from the idea that there exists abundant long-range correlation within natural image, and the human vision system composed of our eyes and brains can sufficiently utilize such types of information redundancy to implement the functions of image interpretation, representation, restoration, enhancement and error concealment. Their method can be summarized as

five basic steps: fetching, searching, matching, competing and recovering, as presented in Figure 2.19.

Figure 2. 19 Demonstration of the general approach.

Figure 2. 20 An example of the recovery of shape information in [83]. (a) α -plane of frame 0. (b) Received α -plane with one slice missing. (c) One possible reconstructed boundary for figure.

In [82], model-based schemes for error concealment of video are presented.

Appearance models for specific objects in the scene are built, and used to replenish any lost information. Due to the models being designed specially for the object, they are able to capture the statistical variations in the object appearance more effectively, thereby leading to better error concealment performance.

Most error concealment methods are focused on the recoveries of texture and motion of video. Schuster *et al* [83] introduced a shape error concealment technique that uses geometric boundary information of the received α -plane. Video object coding is one of the innovations of MPEG-4. The α -plane of a video object defines its shape at a given instance in time and hence determines the boundary of its texture. When error occurs during video transmission, the second-order Hermite splines are used to model the received boundary in the neighboring blocks, while the third-order Hermite splines are used to model the missing boundary. Figure 2.20 gives an example of this shape recovery algorithm. Figure 2.20a is the original α -plane. In Figure 2.20b, a row of this α -plane is lost. The velocities of splines are matched at the boundary point closest to the missing block. There exist several possible concealing splines of the lost boundary, and all these splines are kept until the end. At the end, the concealment solution that results in one closed boundary is selected.

Another spatial shape error concealment technique is proposed for the object-based video coding schemes in [84]. In this technique, it is also assumed that the shape of the corrupted object is presented in the terms of a binary α -plane, in which some of the shape data is missing due to channel errors. From this α -plane, a contour that is corresponding to the boundary of the object can be extracted. If any part of the contour is lost due to transmission error, the proposed technique can interpolate the missing contours by the available surrounding contours. After all missing parts of the contour have been interpolated, the concealed α -plane can be easily reconstructed from the fully recovered contour.

2.3.5.2 Temporal Error Concealment Methods

Most temporal error concealment methods can achieve good recovery result in a motion-smooth area. When a scene has changed between neighboring frames, these methods may cause artifact. In [85], the impact of scene change on the conventional error concealment method is addressed, as shown in Figure 2.21. A novel error concealment method is proposed to improve the insufficiency of conventional temporal error concealment algorithm due to the occurrence of scene change. Combining with the low complexity scene change detection algorithm using macroblock type information, the corrupted blocks resulting from bit errors are concealed either temporally or spatially depending on whether or not an abrupt scene changes is found. In the case of gradual scene change, interpolation and extrapolation are proposed to utilize the linear property of gradual scene change sequence, and effectively reduce the concealment error in comparison with conventional algorithm.

Figure 2. 21 Illustration of impact of scene change on conventional temporal error concealment.

Another drawback of most conventional temporal error concealment methods is that the information of neighboring motion vectors is often used in the recoveries of lost motion vectors. When the video is transmitted through a burst-packet-loss network, the motion vectors in a large area may be lost together. And the lost area may not be

recovered efficiently by conventional error concealment method. To solve this problem, Pyun *et al* [86] introduced a robust error concealment method to estimate the true motion vector of the missing block by using the bi-directional motion tracking technique within neighboring frames of a damaged frame. This motion tracking technique uses the linear trajectory feature of the image block in the forward and backward prediction directions.

Chen *et al* [87] presented an algorithm that combines the overlapped motion compensation and the side matching criterion to make the effects of the lost motion vectors subjectively imperceptible. In the first step of the algorithm, boundary matching is exploited to find an appropriate motion vector $\hat{V}(p, q)$. Then, the block compensated according to $\hat{V}(p, q)$ is subdivided into four $n/2 \times n/2$ subblocks. Each subblock is synthesized by the weighted average of three predicted subblocks: one according to the recovered vector $\hat{V}(p, q)$ chosen by boundary matching criterion, the second one according to the motion vector of the horizontally neighboring $n \times n$ block, and the third one according to the motion vector of the vertically neighboring $n \times n$ block. The weight matrices are shown in Equation (2.21). Matrix in 2.21a is the weighting value for the recovery with one of the motion vectors of adjacent blocks using boundary matching criterion. Matrix 2.21b is used for the recovery with the motion vector of top or bottom block. And matrix 2.21c is utilized for the reconstruction with the motion vector of left or right block. For example, when the motion vector of a 16×16 macroblock is lost, it will be divided into 8×8 blocks for error concealment. If the block is recovered by boundary matching criterion, the reconstructed block will be M_r^s . If the lost motion vector is replaced by the motion vector of neighboring macroblocks in horizontal and vertical directions, the reconstructed blocks will be M_r^v and M_r^h . The final reconstructed block is a weighted summary of these reconstructed blocks.

$$W_1 = \begin{vmatrix} 4 & 5 & 5 & 5 & 5 & 5 & 5 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 4 & 5 & 5 & 5 & 5 & 5 & 5 & 4 \end{vmatrix} \quad (2.21a)$$

$$W_2 = \begin{vmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \end{vmatrix} \quad (2.21b)$$

$$W_3 = \begin{vmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \end{vmatrix} \quad (2.21c)$$

In [88], a temporal error concealment method using motion field interpolation is proposed. For each pixel in a damaged block, the algorithm recovers a motion vector using bilinear interpolation of neighboring motion vectors, as shown in Equation (2.22). V_T, V_R, V_B and V_L represent the motion vectors of neighboring blocks. (x_n, y_n) is the normalized spatial coordinate of the pixel within block. The vector is then used to conceal the damaged pixel. Overlapped motion compensation is used to combine this algorithm with a boundary matching error concealment algorithm.

$$\hat{b}(x_n, y_n) = \frac{(1 - x_n)V_L + x_nV_R + (1 - y_n)V_T + y_nV_B}{2} \quad (2.22)$$

Lee *et al* [89] proposed a joint spatial temporal error concealment method. To do that they model video sequences as mixtures of simple Markov processes and utilize the model for concealment of portions of video, which are missing. This model combines spatial and temporal information in a seamless manner when performing concealment on the missing blocks.

Table 2. 3 Average PSNR comparisons for various schemes.

Sequence		Flower Garden						
f/s	Search Range	Error Free	ZR	Average	SI	SM	SM+OMC	GA
30	[-8,+7]	26.89	16.13	24.25	16.62	22.02	24.07	24.35
15	[-16,+16]	24.59	14.71	21.16	16.11	19.25	21.35	21.92
10	[-24,24]	23.28	14.09	19.84	15.45	17.64	19.68	20.34
Sequence		Table Tennis						
f/s	Search Range	Error Free	ZR	Average	SI	SM	SM+OMC	GA
30	[-8,+7]	30.05	22.84	25.78	21.98	25.37	26.98	27.32
15	[-16,+16]	28.98	21.22	24.51	21.28	23.10	24.83	25.35
10	[-24,24]	28.32	20.59	23.55	20.84	21.81	23.64	24.08
Sequence		Foot ball						
f/s	Search Range	Error Free	ZR	Average	SI	SM	SM+OMC	GA
30	[-8,+7]	22.10	17.04	18.64	17.37	17.79	19.19	19.63
15	[-16,+16]	20.51	15.33	16.49	16.28	15.91	17.28	17.62
10	[-24,24]	19.83	14.45	15.54	15.71	14.99	16.37	16.92

In [87], an error concealment technique, which combines the side matching criterion and overlapped motion compensation to make the effect of lost motion vectors subjectively imperceptible, is presented. In order to improve the performance of this method, Lee and Chen [90] proposed a robust and effective genetic algorithm (**GA**) used with the side matching criterion to get the best replacement of the lost image. The error concealment results of this technique are shown in Table 2.3. **ZR** is the zero vector reconstruction. **Average** is the average of motion vectors of four neighboring macroblocks. **SM** is side matching criterion. **OMC** is overlapped motion compensation.

SI is spatial interpolation.

Tsekeridou *et al* [91] made a study on the use of vector rational interpolation for the estimations of erroneously received motion fields of MPEG-2. Various rational interpolation schemes have been investigated, some of which are applied to different interpolation directions. One scheme additionally uses the boundary matching, and another one attempts to locate the direction of minimal or maximal change in the local motion field. Another one further adopts bilinear interpolation principles, whereas a last one additionally exploits available coding mode information. The main advantages of these methods are their capabilities to adapt their behavior with respect to the neighboring motion information.

Table 2. 4 PSNR comparison of different temporal error concealment methods for each P frame in the test sequence: table tennis.

	Frame 63	Frame 75	Frame 87	Frame 99	Frame 111	Frame 123
No EC	13.90	13.06	13.10	13.67	13.72	13.60
Zero Vector	21.03	20.76	20.17	20.10	20.78	20.66
BMA	24.68	26.59	23.74	25.98	26.09	25.66
p=1	27.07	27.42	23.74	26.03	26.04	25.51
p=2	27.74	28.34	25.72	27.26	27.72	26.58
p=3	27.79	28.53	25.78	27.93	27.81	26.59
p=4	27.88	28.30	25.78	27.97	27.64	26.58
p=5	27.70	28.51	25.77	27.97	27.58	26.55
p=6	27.87	28.58	25.77	27.97	27.74	26.37

Lee *et al* [92] presented a multiframe error concealment method. Most error concealment methods only consider spatial and/or temporal correlation in a single frame, which limits their abilities to obtain an optimal recovery. Since the transmission error in one frame may transmit to the following frames, error concealment should simultaneously minimize the errors not only in the currently decoded frame but also in the succeeding B and P frames that depend on the corrupted frame. The simulation

results are presented in Table 2.4, p is the number of frames used in error concealment. BMA represents the boundary matching algorithm. It is proved that extending the search range can improve the error concealment performance.

In the presence of transmission error, the video quality will degrade as the number of frames increases within a GOP. This is due to the propagation and accumulation of errors of the previous frames. To improve the qualities of frames in the rear of a GOP, Tang and Yu [93] developed a unique error concealment framework. The framework is composed of two phases: forward decoding and backward frame update. In the first phase, frames are decoded and concealed when error occurs. Many conventional error concealment algorithms, such as boundary matching method, can be used to recover the lost block, and the error level of each pixel is calculated by Equation (2.23).

$$\sigma^2(x, y, t) = \begin{cases} \sigma^2(x + V_x, y + V_y, t - 1), & \text{if pixel } (x, y) \text{ is not lost} \\ \sigma^2(x + V_x, y + V_y, t - 1) + \sigma_{cf}^2, & \text{if pixel } (x, y) \text{ is lost} \end{cases} \quad (2.23)$$

In the second phase, the frames of the previous decoded GOP may be updated. The update process checks the error level of each pixel in the previous GOP, and sees if the pixel can be replaced by a pixel in the following frame with reduced error level. If the error level of a pixel can be reduced when it is replaced by a pixel in the following frames, it will be updated.

In [94], many conventional error concealment methods are analyzed, and several modification schemes for these conventional methods are proposed. For example, average value algorithm is a very simple error concealment method, in which the motion vector of the lost block can be got by averaging the motion vectors of vertical neighboring blocks. But this method cannot work, when the vertical neighboring motion vectors are not available. To solve this problem, Suh and Ho use the boundary matching algorithm to estimate the motion vector of TB, as denoted in Figure 2.22a. Then the motion vector of the lost block can be recovered. The TB can be further divided into smaller area, which is presented as STB in Figure 2.22b. In order to

reduce the computational complexity, the location of STB is across the boundaries of macroblocks. Then the motion vector of ASTB can be used to estimate the motion vector of neighboring macroblock.

Figure 2. 22 Modified average algorithm. (a) TB. (b) STB. (c) ASTB.

2.4 Multiple Description Coding

Multiple description coding is an efficient error resilience coding method. By using this scheme, the original video will be encoded into several bitstreams, which are referred to as descriptions. And these descriptions will be transmitted over different channels. When all descriptions are correctly received by the decoder, the decoded video will have best quality. If any description is lost or corrupted during transmission, the decoder still can reconstruct the video with degraded quality [99]. Figure 2.23 gives

an example of the multiple description coding.

Many various multiple description coding approaches have been proposed. In [95], a simple method that is based on pixel interleaving is introduced. This method takes advantage of the intrinsic correlation of the adjacent pixels to generate the descriptions, as shown in Figure 2.24.

Figure 2. 23 Example of a multiple description coding communication system.

The motion vector is one of the most important data in the compressed bitstream, and its loss can lead to severe degradation in the decoded picture quality. Kim *et al* [96] proposed a multiple description motion coding algorithm to enhance the robustness of the motion vector against transmission errors. The motion vector is encoded into two descriptions, which are transmitted over distinct channels. If one of the descriptions is lost during transmission, the decoder still can provide an acceptable video quality. A complete multiple description video coder is implemented by modifying the syntax of H.263 standard, and tested intensively in a realistic error-prone environment.

Figure 2. 24 Multiple description image coding using pixel interleaving.

In Wang *et al*'s paper [97], a multiple description generation scheme that produces two descriptions is proposed. This scheme is based on the pairwise correlating transforms, and it can be implemented under general image coding framework. The correlation introduced by the transform helps to reduce the distortion when only a single description is received. Karhunen-Loeve transform (KLT) that can introduce controlled correlation to the bitstream is used in the proposed method. If one description is lost during transmission, it still can be estimated from the other description with certain accuracy. The flowchart of this multiple description encoder is shown in Figure 2.25.

Figure 2. 25 Multiple description transform coding using a cascade of KL transform and a correlating transform.

Wang [98] proposed another multiple description video coder, which uses redundancy-rate-distortion criterion to split a one-layer stream generated by a standard video coder into two correlated streams. For each encoding block, a certain threshold is assigned. The non-zero coefficients that are larger than this threshold are duplicated into each description. The coefficients that are smaller than the threshold will be alternated between descriptions according to their magnitudes. The threshold is calculated by the redundancy-rate-distortion criterion.

2.5 Intra Refresh

In order to achieve better coding efficiency, the temporal redundancy of natural video will be removed by the motion compensated prediction during coding procedure. Unfortunately, this technique will cause error propagation between inter frames. The error in one inter frame may transmit to successive frames, until an intra frame. Of course, inserting more intra frame will increase the error resilient performance of the compressed video, but the bit cost of intra frame is much higher than inter frame. The bandwidth limitation of transmission channel makes it impossible to insert many intra frames in a bitstream.

Instead of encoding the whole frame as intra frame, some macroblocks in an inter frame can be encoded as intra macroblocks to enhance the error resilient performance of bitstream. This method is referred to as intra refresh, which is an efficient method to stop the error propagation.

Liao and Villasenor [100] proposed an adaptive block-based intra refresh algorithm for increasing the error resilience. The intra update rates can be adjusted by this algorithm, according to the different image characteristics and various channel conditions. The update scheme is based on an “error-sensitivity metric”, accumulated at the encoder, representing the vulnerability of each coded block to channel errors. When frame is encoded, the accumulated metric for each block is examined. Those blocks that have unacceptable high metric will be encoded as intra block.

Dogan *et al* [101] developed an error resilient transcoding method, which adopts adaptive intra refresh scheme. The proposed algorithm implements the transcoding at proxy, which can provide the necessary output transmission rates with the required robustness. The intra refresh can be added to an already compressed video stream. Chiou *et al* [102] also proposed to adopt the intra refresh in transcoding. A two-pass transcoding scheme for enhancing the error resilience in the media gateway of a

three-tier streaming system is presented. The transcoder can adapt intra refresh rate according to a channel's packet loss rate, and protect the most important macroblocks from packet loss.

Zhao *et al* [103] introduced an intra refresh scheme that is based on pixel level inter-frame dependence history. The coding mode of each pixel is tracked by the encoder. If a pixel has a long inter coding history, it means this pixel is very likely to be affected by the error propagation. Then the macroblock that contains this pixel will be encoded as intra macroblock.

Chang and Lee [104] proposed an intra refresh method that based on a precise error-tracking scheme for robust transmission of real-time video. By using the feedback channel, the decoder reports the addresses of corrupted blocks back to the encoder. With these negative acknowledgements, the encoder can precisely calculate and track the propagated errors by examining the motion dependency of each pixel in the currently encoding frame, as described in Figure 2.26. Then the error-propagation effects can be terminated completely by intra refreshing the affected macroblocks.

Figure 2. 26 Linear forward error tracking.

The effect of data loss on video often depends on the motion properties of frames. If the motion is slow, the effect of data loss will be limited, since there are little changes between subsequent frames. If the frame that contains high motion has data loss, there will be more serious consequence on the quality of decoded video. Taking this into account, it is obvious that the frame with high motion should be treated differently as the low motion frame. Worrall *et al* [105] proposed a scheme that attempts to optimize error robustness by adjusting the intra refresh rate according to the

motion properties of frames. The estimation of motion property is obtained by exploiting the data-partition mode of MPEG-4.

In He *et al*'s paper [106], a rate-distortion mode for DCT-based video coding incorporating the macroblock intra refresh rate is developed. For any given bit rate and intra refresh rate, this mode is capable to estimate the corresponding coding distortion. Then they presented a theoretical analysis of the distortion that is caused by channel errors and error propagations of inter frame. By making use of this analysis, a statistical model can be constructed to estimate the distortion induced by different channel conditions and encoding settings. This model also can mathematically describe the complex behavior of channel errors in a video coding and transmission system. Based on the rate-distortion model and the statistical model, a solution for adaptive intra refresh mode selection can be derived.

The intra refresh schemes mentioned before are designed for single-frame motion compensated prediction and integer-pixel precision. These methods are not suitable for H.26L, which adopts long-term memory motion compensated prediction and sub-pixel precision. Ma *et al* [107] introduced an intra macroblock refresh scheme for H.26L, which is based on the rate-distortion optimization. This scheme takes account of the error propagation effect into the encoding distortion evaluation and has the precision of 1/4 pixel position. The distortion of a pixel can be calculated as:

$$\|d_n^i\|^2 = (1 - P_e) \left\| (\bar{X}_{n-1}^i + \bar{e}_n^i) - (\hat{X}_{n-1}^i + \bar{e}_n^i) \right\|^2 + P_e \left\| \tilde{X}_n^i - \hat{X}_{n-1}^i \right\|^2 \quad (2.24)$$

where P_e is the error probability. $\bar{X}_n^i$ is the reconstructed value of pixel i in the n^{th} frame at the encoder. $\bar{e}_n^i$ is the encoded residual of this pixel. $\hat{X}_n^i$ is the reconstructed pixel value in the decoder. $\tilde{X}_n^i$ represents the recovered pixel when error occurs.

Then the rate-distortion optimization equation of a macroblock can be written as:

$$J_{\min} = \sum_{i \in MB} \|X_n^i - \bar{X}_n^i\|^2 + \sum_{i \in MB} \|d_n^i\|^2 + \lambda_{MODE} R_{MB}(MODE) \quad (2.25)$$

X_n^i is the original pixel value, and R_{MB} is the rate of this macroblock. The macroblock that has largest J will be encoded as intra macroblock.

2.6 Conclusion

In this chapter, many error control schemes proposed by previous work are introduced. In generally, the post-processing methods are more suitable for the video transmission over Internet. Since the error mode of Internet is random, the neighboring information of a lost frame or macroblock is often available. Then the correlation of natural video in spatial and temporal domains can be used to recover the lost area. For the wireless and mobile networks, the error format is burst, that means the error is very likely to occur successively. In this case, the information needed by error concealment approaches may be insufficient. The pre-processing methods, such as FEC, are often utilized to deal with the burst error. When enough protection is added on the bitstream, the video can be decoded correctly even under a high error ratio. It should be noted that the protection provided by the pre-processing methods is at the cost of bitrate. This makes the pre-processing methods not suitable for the situation that the bandwidth of transmission channel is very limited.

Chapter 3

An Efficient Maximally Smooth Error Concealment Method

3.1 Introduction

In this chapter, an efficient spatial error concealment method for H.264 is presented.

Within a small area in a natural image, adjacent blocks are smoothly connected. Based on this smoothness property, a linear constraint function that describes the connection of lost block and its neighboring blocks can be formulated. The difference between the recovered block and its nearby blocks should be minimized. The transform coefficients of lost block can be recovered by finding the minimal value of the constraint equation. Since the smooth connection constraint is only imposed on the boundary pixels, the recovered center pixels may have significant error. To achieve better recovery results,

the recovered pixels that are on the boundaries of the lost block and average pixel difference are used for the interpolation of center pixels. The implementation is simple, and it is suitable for real-time video application. Experimental results show that our method can efficiently improve the quality of video.

H.264 is a new video coding standard. Previous video coding standards perform the 8×8 Discrete Cosine Transform (DCT) on residual in an intra block. H.264, however, adopts a 4×4 integer transform, a transform with the same coding efficiency as a 4×4 DCT [7] [109]. Although some spatial error concealment methods had been proposed for the existing video coding standard, these methods are optimized for the 8×8 block. Therefore, the development of error concealment method for H.264 compressed video becomes more important.

3.2 Relative Spatial Error Concealment Methods

Projection onto convex sets (POCS), temporal predictive concealment (TPC) and spatial predictive concealment (SPC) are examples of error concealment methods used to recover lost blocks. The POCS can efficiently prevent error propagation [110] [111]. It uses the overlapping information from several frames and blocks to generate relations among pixels in the damaged block. TPC exploits information in the previous frames to recover the corrupted areas. And SPC uses the spatial redundancy to recover damaged blocks.

As we introduced in previous chapter, many algorithms of SPC that were proposed for MPEG video rely on the smoothness property of image [57-59]. In these methods, the objective functions are formulated according to different constraints. By solving these objective functions, the coefficients of lost or corrupted block can be recovered.

The technique proposed by Park *et al* [57] illustrates how the objective function is formulated and the lost blocks are recovered. A lost block can be denoted as B_i in spatial domain. The boundary vectors of block B_i in four directions, top, bottom, left and right, can be represented as:

$$t_i(m) = B_i(0, m), \quad m = 0, \dots, N-1 \quad (3.1a)$$

$$b_i(m) = B_i(N-1, m), \quad m = 0, \dots, N-1 \quad (3.1b)$$

$$l_i(n) = B_i(n, 0), \quad n = 0, \dots, N-1 \quad (3.1c)$$

$$r_i(n) = B_i(n, N-1), \quad n = 0, \dots, N-1 \quad (3.1d)$$

The corresponding locations of B_i and its adjacent blocks are shown in Figure 3.1. There are P blocks in each row of frame.

N denotes the size of lost block, and $B_i(n, m)$ represents the horizontally m^{th} pixel of n^{th} row. The pixel in lost block is represented in terms of DCT coefficients:

$$B_i(n, m) = \phi^T(n, m) X_i \quad (3.2)$$

where X_i is the matrix of DCT coefficients of block B_i at frequency domain, and $\phi(n, m)$ is the DCT kernel function.

A constraint function utilized for the estimation of lost DCT coefficients is defined in Equation (3.3). This constraint function is based on the smoothness

property of natural image. It is assumed that the recovered block should be smoothly connected with its neighboring blocks on the boundaries.

$$\psi = \|t_i - b_{i-p}\|^2 + \|b_i - t_{i+p}\|^2 + \|l_i - r_{i-1}\|^2 + \|r_i - l_{i+1}\|^2 \quad (3.3)$$

Figure 3. 1 The lost block B_i and its neighboring blocks.

By using Equation (3.2), the constraint function presented in Equation (3.3) can be rewritten in terms of DCT coefficients.

$$\begin{aligned} \Psi(X_i) = & \left\| [\phi(0, \cdot)]^T X_i - [\phi(\cdot)]^T b_{i-p} \right\|^2 \\ & + \left\| [\phi(N-1, \cdot)]^T X_i - [\phi(\cdot)]^T t_{i+p} \right\|^2 \\ & + \left\| [\phi(\cdot, 0)]^T X_i - [\phi(\cdot)]^T r_{i-1} \right\|^2 \\ & + \left\| [\phi(\cdot, N-1)]^T X_i - [\phi(\cdot)]^T l_{i+1} \right\|^2 \end{aligned} \quad (3.4)$$

In order to minimize the constraint function, four sub-equations can be derived from Equation (3.4). For an 8×8 block, a maximum of 28 coefficients can be estimated from the four sub-equations. There are a total of 64 coefficients in an 8×8 block, therefore only the low frequency coefficients can be reconstructed. Figure 3.2

shows the locations of recovered coefficients.

a_{i00}	a_{i01}	a_{i02}	a_{i03}	a_{i04}	a_{i05}	a_{i06}	a_{i07}
a_{i10}	a_{i11}	a_{i12}	a_{i13}	a_{i14}	a_{i15}	a_{i16}	a_{i17}
a_{i20}	a_{i21}						
a_{i30}	a_{i31}						
a_{i40}	a_{i41}						
a_{i50}	a_{i51}						
a_{i60}	a_{i61}						
a_{i70}	a_{i71}						

Figure 3. 2 The locations of maximally obtainable DCT coefficients.

The approach presented by Wang *et al* [61] imposes the smoothness constraint to each pixel of the corrupted blocks, and exploits the correctly received coefficients as well as the boundary pixels of adjacent blocks to calculate the lost DCT coefficients. Their recovery results are better than the results presented in [57] and [59], but the computation complexity is increased.

Unfortunately, all these methods are not suitable for the H.264 video. The constraint functions formed by these methods are designed for 8×8 blocks, and 8×8 DCT is used in the solutions of the constraint functions. Since these constraint functions are uncertain functions, which cannot result in unique solution, only the low frequency coefficients are recovered and the unreliable high frequency coefficients are discarded. In H.264, as 4×4 block is used, the constraint function will have a unique solution, and all coefficients of the lost blocks can be recovered. But directly recovering the lost blocks by the similar constraint function will result in obvious artifact.

In this chapter, we utilize the concept of spatial predictive concealment to reconstruct the corrupted H.264 video. The main difference between the previous methods designed for MPEG video and our proposed method is the presence of a unique solution to the objective function in proposed method. In MPEG video, where the smoothness connection constraint is imposed only on the boundary of the lost block, the objective function is uncertain [57] [59]. Unique solution for all 64 coefficients cannot be obtained, only low frequency coefficients are recoverable. In H.264 video, since the block size is reduced to 4×4 , the constraint function defined by us has unique solution, and all 16 coefficients of a damaged block can be recovered. Since the reconstructed high frequency coefficients may be inaccurate and we only impose the smoothness constraint on the boundaries of damaged blocks, the reconstructed center pixels may be erroneous. Imposing smoothness constraint to each pixel can reduce the error. This method will inevitably increase the computational complexity, to avoid this drawback, we propose a simple interpolation method that makes use of the recovered boundary pixels to compute the center pixels. Average pixel differences are taken into account to enhance the video quality.

The computational cost of the proposed method is slightly higher than the approach presented in [59], as the center pixels are recovered twice. However, since the interpolation method used for the second time estimation of center pixels is very simple, the increment of computational cost is negligible. Compared with the scheme in [61], the proposed method is much easier to be implemented.

3.3 The Recovery of Lost Block

In this section, the details of an efficient coefficient recovery algorithm for H.264 video are provided. Based on error detection technology [141] [142] and interlace method [143] that packs the neighboring blocks into different packets, we can assume that the location of the lost block is detected, and the four neighboring blocks are correctly decoded.

The lost block is denoted as $B_{i,j}$, and the four neighboring blocks are represented as $B_{i-1,j}$, $B_{i+1,j}$, $B_{i,j-1}$ and $B_{i,j+1}$ respectively, as shown in Figure 3.3. If any neighboring block is corrupted, we can use the co-located block in previous frame instead, the recovery results will be slightly degraded. The four boundary pixel vectors of block $B_{i,j}$ are top , lef , rig and bot respectively. top_B , lef_B , rig_B and bot_B are the four neighboring boundary pixel vectors, their locations are described in Figure 3.3.

Thus, we have:

$$top(n) = B_{i,j}(n,0), \quad n = 0, \dots, 3 \quad (3.5a)$$

$$bot(n) = B_{i,j}(n,3), \quad n = 0, \dots, 3 \quad (3.5b)$$

$$lef(m) = B_{i,j}(0,m), \quad m = 0, \dots, 3 \quad (3.5c)$$

$$rig(m) = B_{i,j}(3,m), \quad m = 0, \dots, 3 \quad (3.5d)$$

where $B_{i,j}(n,m)$ represents the horizontal n^{th} pixel of the m^{th} row in lost block.

Firstly, we use the boundary pixel vectors to define a function that describes the connection of the damaged block and its four neighboring blocks. Because of the

smoothness property of image, we can assume that the luminance components change relatively slowly within a small area. That means the boundary pixel values of two neighboring blocks are close. Then we defined a constraint function φ , which describes the smooth connection. The recovered boundary pixel vectors should minimize the functions φ .

$$\varphi = \|top - top_B\|^2 + \|bot - bot_B\|^2 + \|lef - lef_B\|^2 + \|rig - rig_B\|^2 \quad (3.6)$$

Figure 3. 3 The lost block and the four boundary pixel vectors.

Next, we use the coefficients to represent the boundary pixel vectors of the damaged block. There are two different transform matrices that have been defined in H.264. The one that is presented in Equation (3.7) is defined in the old version of H.264, which is referred to as H.26L. The matrices presented in Equation (3.8) are adopted in the new version of H.264 to reduce the computational cost.

$$C_{transform} = \begin{bmatrix} 13 & 13 & 13 & 13 \\ 17 & 7 & -7 & -17 \\ 13 & -13 & -13 & 13 \\ 7 & -17 & 17 & -7 \end{bmatrix} \quad (3.7a)$$

$$C_{inverse-transform} = \begin{bmatrix} 13 & 17 & 13 & 7 \\ 13 & 7 & -13 & -17 \\ 13 & -7 & -13 & 17 \\ 13 & -17 & 13 & -7 \end{bmatrix} \quad (3.7b)$$

$$C_{transform} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \quad (3.8a)$$

$$C_{inverse-transform} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix} \quad (3.8b)$$

These two transform matrices have similar properties and efficiencies. Both of them can be used to convert the pixel vectors into coefficient domain. We use the matrix defined in Equation (3.7) as example to illustrate our algorithm. From Equation (3.7), we can form a transform equation, as shown in Equation (3.9).

$$\begin{bmatrix} a' \\ b' \\ c' \\ d' \end{bmatrix} = C_{inverse-transform} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \quad (3.9)$$

where a', b', c' , and d' are inverse transformed pixel values. A, B, C , and D are transformed coefficients of blocks. When the inverse-transform matrix presented in Equation (3.7) is used, the relation between a' and the original pixel a is $a' = 676a$, similarly $b' = 676b$ and so on. The inverse transform is computed by the row-column decomposition method [109]. From Equation (3.9), we can define

the one-dimensional inverse transform matrices $h(n)$ and $h(m)$ in Equation (3.10). n and m denote the horizontal and vertical locations in pixel domain.

$$h(0) = [13 \quad 17 \quad 13 \quad 7] \quad (3.10a)$$

$$h(1) = [13 \quad 7 \quad -13 \quad -17] \quad (3.10b)$$

$$h(2) = [13 \quad -7 \quad -13 \quad 17] \quad (3.10c)$$

$$h(3) = [13 \quad -17 \quad 13 \quad -7] \quad (3.10d)$$

Then, we can build a two-dimensional inverse transform vector $h_{n,m}$.

$$h_{n,m} = h(n) \otimes h(m) \quad (3.11)$$

where $\otimes$ is the Kronecker-Delta product. The inverse transform vector $h_{n,m}$ is a 1×16 vector. By using the inverse transform matrices in Equation (3.10), each pixel of the lost block can be represented by coefficient vector $X_{i,j}$.

$$B_{i,j}(n,m) = h_{n,m} X_{i,j} / 676^2 \quad (3.12)$$

In Equation (3.12), $X_{i,j}$ is a 16×1 coefficient vector of the block $B_{i,j}$. $B_{i,j}(n,m)$ indicates a pixel, whose coordinate is (n,m) . We can rewrite the boundary pixel vectors of $B_{i,j}$ in the following formats:

$$top(n) = h_{n,m}^C X_{i,j} / 676^2 \quad 0 \leq n \leq 3 \quad (3.13a)$$

$$bot(n) = h_{n,m}^B X_{i,j} / 676^2 \quad 0 \leq n \leq 3 \quad (3.13b)$$

$$lef(m) = h_{n,m}^L X_{i,j} / 676^2 \quad 0 \leq m \leq 3 \quad (3.13c)$$

$$rig(m) = h_{n,m}^R X_{i,j} / 676^2 \quad 0 \leq m \leq 3 \quad (3.13d)$$

where $h_{n,m}^C$, $h_{n,m}^B$, $h_{n,m}^L$ and $h_{n,m}^R$ are the inverse transform matrices corresponding to the block boundaries in different directions. The definitions of these matrices are presented in Equation (3.14).

$$h_{n,m}^C = \begin{bmatrix} h_{0,0} \\ h_{1,0} \\ h_{2,0} \\ h_{3,0} \end{bmatrix}, \quad h_{n,m}^B = \begin{bmatrix} h_{0,3} \\ h_{1,3} \\ h_{2,3} \\ h_{3,3} \end{bmatrix}, \quad h_{n,m}^L = \begin{bmatrix} h_{0,0} \\ h_{0,1} \\ h_{0,2} \\ h_{0,3} \end{bmatrix}, \quad h_{n,m}^R = \begin{bmatrix} h_{3,0} \\ h_{3,1} \\ h_{3,2} \\ h_{3,3} \end{bmatrix} \quad (3.14)$$

Similarly, we can write the boundary pixel vectors within four neighboring blocks in coefficient format by using Equation (3.15).

$$top_B = h(n)coeff_{Top} / 676 \quad (3.15a)$$

$$bot_B = h(n)coeff_{Bottom} / 676 \quad (3.15b)$$

$$rig_B = h(m)coeff_{Right} / 676 \quad (3.15c)$$

$$lef_B = h(m)coeff_{Left} / 676 \quad (3.15d)$$

where $coeff_{Right}$, $coeff_{Left}$, $coeff_{Bottom}$ and $coeff_{Top}$ are the transformed coefficients corresponding to the four boundary pixel vectors, rig_B , lef_B , bot_B and top_B , respectively. All of them are 4×1 vectors. Because Equations (3.13) and (3.15) are linear equations, the constraint Equation (3.6) can be rewritten as a coefficient equation:

$$\begin{aligned} \Psi = & \left\| h_{n,m}^B X_{i,j} / 676^2 - h(n)coeff_{Bottom} / 676 \right\|^2 \\ & + \left\| h_{n,m}^C X_{i,j} / 676^2 - h(n)coeff_{Top} / 676 \right\|^2 \\ & + \left\| h_{n,m}^R X_{i,j} / 676^2 - h(m)coeff_{Right} / 676 \right\|^2 \\ & + \left\| h_{n,m}^L X_{i,j} / 676^2 - h(m)coeff_{Left} / 676 \right\|^2 \end{aligned} \quad (3.16)$$

Because of the smooth connection property, the recovered coefficients of the lost block should minimize Ψ . Since Ψ is a quadratic function of $X_{i,j}$, its minimal value is where the gradient of Ψ vanishes.

$$\begin{aligned} \frac{\partial \Psi}{\partial X_{i,j}} = & [(h_{n,m}^B)^T h_{n,m}^B / 676^4 + (h_{n,m}^C)^T h_{n,m}^C / 676^4 + (h_{n,m}^L)^T h_{n,m}^L / 676^4 + (h_{n,m}^R)^T h_{n,m}^R / 676^4] X_{i,j} \\ & - [(h_{n,m}^B / 676^2)^T h(n)coeff_{Bottom} + (h_{n,m}^C / 676^2)^T h(n)coeff_{Top} + (h_{n,m}^L / 676^2)^T h(m)coeff_{Left} \end{aligned}$$

$$+ (h_{n,m}^R / 676^2)^T h(m) \text{coeff}_{Right}] / 676 = 0 \quad (3.17)$$

In Equation (3.17), the “ T ” represents transpose. $(h_{n,m}^B)^T h_{n,m}^B$, $(h_{n,m}^C)^T h_{n,m}^C$, $(h_{n,m}^L)^T h_{n,m}^L$, and $(h_{n,m}^R)^T h_{n,m}^R$ are 16×16 matrices, and $(h_{n,m}^B / 676^2)^T h(n) \text{coeff}_{Bottom}$ is 16×1 matrix. We can obtain a unique solution of the lost coefficient vector $X_{i,j}$ from Equation (3.17). By using this recovered coefficient vector $X_{i,j}$, we can reconstruct the pixels of the lost block. Since we only impose the smoothness constraint on the boundary of the lost block, the recovered center pixels may be negative. To solve this problem, some error concealment algorithms for MPEG video [61] impose the smoothness constraint to each pixel within lost blocks. As mentioned in previous chapter, the recovery results can be better, but the computational complexity is significantly increased.

3.4 Interpolation of Inner Pixels

For H.264, since the block size is 4×4 , the center pixels are directly connected to the boundary pixels of block. Center pixels can be estimated by averaging the adjacent boundary pixels. To reduce the artifact of the recovered block, we use the first-order linear interpolation to calculate the average differences between each pixel in vertical and horizontal directions, and use these average differences to improve the estimation of center pixels, as described in Figure 3.4.

$P(-1,-1)$	$P(0,-1)$	$P(1,-1)$	$P(2,-1)$	$P(3,-1)$	$P(4,-1)$
$P(-1,0)$	$P(0,0)$	$P(1,0)$	$P(2,0)$	$P(3,0)$	$P(4,0)$
$P(-1,1)$	$P(0,1)$	$P(1,1)$	$P(2,1)$	$P(3,1)$	$P(4,1)$
$P(-1,2)$	$P(0,2)$	$P(1,2)$	$P(2,2)$	$P(3,2)$	$P(4,2)$
$P(-1,3)$	$P(0,3)$	$P(1,3)$	$P(2,3)$	$P(3,3)$	$P(4,3)$
$P(-1,4)$	$P(0,4)$	$P(1,4)$	$P(2,4)$	$P(3,4)$	$P(4,4)$

Figure 3. 4×4 block and its neighboring pixels.

$$D_v = [P(-1,3) - P(-1,-1)] / 4 \quad (3.18a)$$

$$D_h = [P(3,-1) - P(-1,-1)] / 4 \quad (3.18b)$$

Consider the $[P(0,0).....P(3,3)]$ is the recovered pixel matrix of the damaged block $B_{i,j}$. D_v and D_h are the average differences in vertical and horizontal directions. Then, we use them to set the center pixels, as presented in Equation (3.19).

$$P'(1,1) = [P(0,1) + P(1,0)] / 2 + (D_v + D_h) / 2 \quad (3.19a)$$

$$P'(2,1) = [P(2,0) + P(3,1)] / 2 + (D_v + D_h) / 2 \quad (3.19b)$$

$$P'(1,2) = [P(0,2) + P(1,3)] / 2 + (D_v + D_h) / 2 \quad (3.19c)$$

$$P'(2,2) = [P(3,2) + P(2,3)] / 2 + (D_v + D_h) / 2 \quad (3.19d)$$

We use these modified center pixels to replace the prior recovered center pixels for better recovery results. Through our experiment, it has been proven that the modification of center pixels can significantly improve the PSNR of recovered frame.

3.5 Performance Evaluation of the Maximally Smooth Error Concealment

We select three different QCIF video sequences, Akiyo, Foreman and Stefan, to conduct the experiments. Because our algorithm is designed for intra block, each video is encoded as intra frames by TML 84, which is a standard codec of H.264. By changing the quantization factor of the encoder, we encoded each sequence at different bitrates. In each frame, we randomly select the location of the lost block, and set the lost block ratio as 1%, 5%, 10% and 15%, respectively.

Since H.264 is a new video coding standard, it is difficult to find other spatial error concealment methods that are designed for H.264. We only compare our proposed method with the most common error concealment method, temporal replacement method.

Results are presented in Table 3.1. Lost blocks are reconstructed by the whole sixteen coefficients, which are recovered by Equation (3.17). The PSNR of the reconstructed frame is denoted as $PSNR_r$. This method cannot achieve satisfied recovery results, since the center pixels may have negative values. Negative center pixels can be eliminated by setting the recovered high frequency coefficients to zero. Then the differences between pixels within lost blocks can be reduced, and the values of center pixels will be close to boundary pixels. We selected seven low frequency coefficients to recover the lost block, while the remaining high frequency coefficients are set to zero. The PSNR of the frame recovered by this method is denoted as $PSNR_p$.

Compared with $PSNR_r$, the value of $PSNR_p$ can be improved by about 2 dB. $PSNR_c$ denotes the PSNR of video recovered by temporal replacement method, which uses the co-located blocks in previous frame to replace the lost blocks. $PSNR_o$ represents the PSNR of the proposed method presented in Section 3.3 and 3.4. The experiment result shows that our method can obtain better recovery results.

Table 3. 1 Comparison of the PSNR in different situations

(a) Akiyo

Bitrate	PSNR (without loss)	Block Loss Rate	$PSNR_L$	$PSNR_r$	$PSNR_p$	$PSNR_c$	$PSNR_o$
321.84	37.39	1%	25.96	28.48	30.30	34.11	36.95
		5%	20.27	22.61	24.77	32.46	35.85
		10%	18.05	20.21	22.44	30.80	35.04
		15%	12.15	14.19	16.44	29.48	31.74
265.43	35.96	1%	25.63	28.03	29.79	33.41	35.61
		5%	20.25	22.57	24.66	31.95	34.79
		10%	18.04	20.16	22.35	30.43	34.17
		15%	12.27	14.44	16.69	29.15	31.43

(b) Foreman

Bitrate	PSNR (without loss)	Block Loss Rate	$PSNR_L$	$PSNR_r$	$PSNR_p$	$PSNR_c$	$PSNR_o$
545.58	34.07	1%	24.21	25.45	27.31	29.75	33.78
		5%	18.70	19.84	21.99	26.78	33.10
		10%	16.73	17.92	20.10	25.46	32.56
		15%	10.85	11.99	14.22	22.72	30.05
439.54	32.68	1%	24.04	25.12	26.87	29.16	32.47
		5%	18.61	19.87	21.98	26.40	31.97
		10%	16.70	17.91	20.07	25.05	31.58
		15%	10.77	12.00	14.23	22.63	29.52

(c) Stefan

Bitrate	PSNR (without loss)	Block Loss Rate	PSNR _L	PSNR _r	PSNR _p	PSNR _c	PSNR _o
1230.53	32.30	1%	25.14	26.27	27.89	29.72	31.71
		5%	20.06	21.27	23.43	25.38	30.52
		10%	17.92	19.13	21.43	22.99	29.66
		15%	12.14	13.41	15.82	21.12	26.05
1008.56	30.52	1%	24.65	25.54	27.06	28.75	30.13
		5%	19.75	20.99	23.08	24.57	29.27
		10%	17.86	19.11	21.32	22.56	28.64
		15%	12.14	13.45	15.85	20.86	25.64

Notes:

$PSNR_L$: without any error concealment.

$PSNR_r$: using the whole 16 recovered coefficients to calculate the lost pixels.

$PSNR_p$: using the 7 low frequency coefficients to recover the lost pixels.

$PSNR_c$: using temporal replacement method to recover the lost pixels.

$PSNR_o$: using proposed method.

(a)

Figure 3.5 PSNR of different bitrate. (a) Stefan: 5% blocks lost. (b) Foreman: 5%

blocks lost.

Figure 3.5 indicates the change tendency of PSNR along different bitrates. The simulation results of proposed method are satisfactory. When the bitrate is low, the quality of recovered video is very close to the error-free video.

3.6 Summary

In this chapter, we propose a new algorithm of error concealment for the new video coding standard H.264. We use the boundary pixels of lost block and its four neighboring blocks to formulate a constraint equation that is based on first-order derivative-based smoothness measure. Then the first-order linear interpolation is exploited to further improve the error concealment performance of the proposed

method. The experiment results have shown that our approach is efficient in different transmission bitrates for various video sequences over a wide range of block loss rates.

Chapter 4

First-Order Plane Estimation for Motion Vector Recovery

4.1 Introduction

The information of motion vector is very important for the compressed video. In this chapter, we propose a new error concealment algorithm for the new video coding standard H.264, which makes use of a first-order plane to estimate motion vectors.

The standard H.26L that we mentioned in previous chapter is an original version of H.264. A 16×16 macroblock in H.26L can also be divided into different block shapes for motion estimation. As shown in Figure 1.12, a 16×16 macroblock can be divided into 16×8 , 8×16 or 8×8 blocks. If the macroblock is divided into 8×8 blocks, each 8×8 block can be further divided into 4×8 , 8×4 or 4×4 blocks. The

division modes used in all 8×8 blocks must be the same. The number of motion vectors in one macroblock is changed from 1 to 16. In H.264, a more flexible macroblock division scheme is used [109] [118], the details of the macroblock division modes are presented in Figure 1.13. Unlike H.26L, the division of each 8×8 block is independent. That means different division modes can be used for different 8×8 blocks. Each block has a corresponding motion vector. Obviously, an inter macroblock in H.264 will contain more motion vectors than previous coding standards.

For natural image, the motions within a small area are likely to move in the same direction. Based on this property, we can assume that the motion vectors of neighboring macroblocks have similar motion tendencies. Since motion vectors in H.264 covers smaller region than previous coding standards, the correlation between neighboring motion vectors increases. By using the motion vectors that are next to the vertices of lost macroblock, we can constitute a first-order plane that indicates the movement tendency in this small area, and estimate the motion vectors of the vertices. The motion vectors of neighboring 4×4 blocks are used to establish the first-order plane.

From these motion vectors, the motion vector of each pixel in the lost macroblock is estimated through interpolation. A typical linear interpolation function is selected to calculate the motion vector of each pixel [112]. This interpolation function not only ensures the smooth motion change within the recovered macroblock, it also makes the recovered macroblock connect smoothly with its

neighboring macroblocks. The simulation results show that the proposed method can efficiently improve the video quality over different macroblock loss rates.

4.2 Motion Vector Recovery

Many efficient temporal error concealment methods have been proposed. The most commonly used technique is to replace the damaged motion vector with (0,0) [113], this approach is usually referred to as temporal replacement.

As mentioned in Chapter 2, boundary matching is another popular error concealment technique. Lee *et al* [69] presented a method that uses multi-frame boundary matching algorithm to compare each candidate block and select the best matching block to mask the corrupted block. Atzori *et al* [114] proposed an algorithm that combines boundary matching algorithm and mesh-based warping.

The neighboring blocks are often used to estimate the lost motion vectors. Chen *et al* [78] proposed an overlapped motion compensation algorithm to recover the lost blocks. Wang *et al* [115] used the best neighborhood matching theory to recover the lost block. Teskeridou *et al* [72] proposed an approach that makes use of neighboring blocks. They compared each candidate block with the neighboring block of the corrupted block. After finding the best matching block, they used the block at corresponding location to mask the corrupted block.

The main drawback of these methods is that incorrect estimation of the concealment displacement can lead to poor recovery result of the corrupted block. To

avoid this problem, many approaches that use motion field interpolation had been presented [116]. Mualla *et al* [88] presented a motion field interpolation method that uses a bilinear motion field interpolation equation to calculate the motion vector of each pixel. Park *et al* [117] proposed an algorithm that is based on NURBS interpolation. Turaga *et al* [84] introduced a model-based scheme for error concealment. In these interpolation methods, the motion vector of each pixel is estimated individually, and incorrect estimation of motion vector affects only individual pixel, but not the whole block.

4.3 Algorithm of Linear Interpolation

In this section we proposed a linear interpolation error concealment algorithm for H.264. For natural image, the blocks within a small area will likely move in the same direction. The motion vectors of neighboring blocks are highly correlated. For the video encoded by H.264, the motion vector covers smaller region than the previous coding standards, hence the correlation between neighboring motion vectors increases. We make use of this encoding property to develop our new algorithm in the following subsections.

4.3.1 Estimate Lost Motion Vectors by Corner Motion Vectors

Our approach can be divided into two steps. First, we estimate the motion vectors of four vertices of the lost macroblock. We assume that the location of lost macroblock has been detected, and all neighboring macroblocks are correctly received and decoded. The corresponding positions of the lost macroblock and its neighboring macroblocks are shown in Figure 4.1, i and j denote the spatial location of macroblocks. Mv_1, Mv_2, Mv_3 and Mv_4 indicate the motion vectors of four vertices located at the corners of the lost macroblock. The motion vectors that cover 4×4 blocks at the corners of the neighboring macroblocks are used to estimate the motion vectors of four vertices. For simplicity, we only present how to calculate Mv_1 . v_1, v_2 and v_3 represent the motion vectors covering neighboring 4×4 blocks respectively, as shown in Figure 4.1. That does not mean that the neighboring macroblocks must be divided into 4×4 blocks for motion estimation. If the neighboring macroblock mode is 16×16 , we will consider that the motion vectors of all 4×4 blocks in this macroblock are the same. And one of them is used to estimate the value of Mv_1 . The reason why we select v_1, v_2 and v_3 to estimate Mv_1 is because they are located nearest to the vertex. Thus they have the highest correlation with the motion vector of vertex.

If the motion vectors of the nearest neighboring blocks are not available, we can select other motion vectors that are near to the corners of lost macroblock to estimate the motion vectors of vertices. The estimation accuracy will decrease.

Figure 4. 1 The estimation of the vertices motion vectors.

We can assume that the location of the pixel located at the vertex of lost macroblock is $(0, 0)$ in the two dimensional reference frame, and each pixel size is one unit. The coordinates of neighboring block centers are $(1.5, 2.5)$, $(-2.5, 2.5)$ and $(-2.5, -1.5)$ respectively, as shown in Figure 4.2(b). The motion vectors of neighboring blocks are two dimensional vectors, and the values of the two dimensions are independent. v_1, v_2 and v_3 represent the values of motion vectors in vertical or horizontal directions. We use the coordinates of neighboring block centers and the corresponding motion vectors to build a first-order plane P as described in Equation (4.1). Equation (4.1) is the common equation that defines a plane in three- dimensions. x, y and z indicate the coordinates of a point belonging to the plane. a, b and c are constants for a certain plane.

$$\frac{x}{a} + \frac{y}{b} + \frac{z}{c} = 1 \quad (4.1)$$

In the proposed algorithm, x and y indicate the center coordinates of neighboring 4×4 blocks in two dimensional reference frame, and z is the corresponding motion vector value in horizontal or vertical direction, as shown in Figure 4.2(a).

Figure 4. 2 The plane built by neighboring motion vectors.

Then we can use v_1, v_2 and v_3 to compute the coefficients in Equation (4.1), the results are represented in Equation (4.2).

$$a = \frac{2.5v_1 - v_2 + 2.5v_3}{v_2 - v_3} \quad (4.2a)$$

$$b = \frac{2.5v_1 - v_2 + 2.5v_3}{v_1 - v_2} \quad (4.2b)$$

$$c = \frac{2.5v_1 - v_2 + 2.5v_3}{4} \quad (4.2c)$$

Because the motion vector used in H.264 covers smaller region than previous coding standards, the motion vectors that cover the corner region in neighboring macroblocks have higher correlation with the motion vectors of the corrupted macroblock. Therefore, we can assume that Mv_1 should be located on the first-order plane P in Figure 4.2(a). The value of Mv_1 can be calculated by Equation (4.1). Then Mv_1 can be estimated by v_1, v_2 and v_3 , as represented in Equation (4.3). Mv_2, Mv_3 and Mv_4 can be estimated by similar approach.

$$Mv_1 = \frac{2.5v_1 - v_2 + 2.5v_3}{4} \quad (4.3)$$

4.3.2 Interpolation of Motion Vector for Each Pixel

Then we use the motion vectors of four vertices to interpolate the motion vector of each pixel within the lost macroblock. To reduce the computational cost, we select a simple linear Equation (4.4) to interpolate the motion vector of each pixel.

$$Mv_{\tilde{x},\tilde{y}} = Mv_1[\tilde{y} \cdot (1 - \tilde{x})] + Mv_2(\tilde{x} \cdot \tilde{y}) + Mv_3[(1 - \tilde{x}) \cdot (1 - \tilde{y})] + Mv_4[\tilde{x} \cdot (1 - \tilde{y})] \quad (4.4)$$

where $\tilde{x}$ and $\tilde{y}$ represent the normalized pixel coordinates that are calculated by Equation (4.5). In Equation (4.5), x_0, x_1, y_0 and y_1 represent the edge coordinates of lost macroblock in four different directions respectively, as shown in Figure 4.3.

Figure 4. 3 Interpolation of motion vector for each pixel.

$$\tilde{x} = \frac{x - x_0}{x_1 - x_0} \quad (4.5a)$$

$$\tilde{y} = \frac{y - y_0}{y_1 - y_0} \quad (4.5b)$$

By using this interpolation algorithm, motion discontinuity is avoided. The motion vectors change smoothly over the frame, and the recovered macroblock will be connected smoothly with its neighboring macroblocks. For example, when $\tilde{x} = 1$, it means that the pixel is on the right side of macroblock. In this case, only Mv_2 and Mv_4 contribute to $Mv_{\tilde{x}, \tilde{y}}$.

Through the interpolated motion vectors, we can get the corresponding pixel intensity in the reference frame. And we can use these pixels to displace the lost pixels, as presented by Equation (4.6).

$$I^{rec}(x, y, k) = I^{ref}(x + Mv_{\tilde{x}, \tilde{y}}^H, y + Mv_{\tilde{x}, \tilde{y}}^V, k - 1) \quad (4.6)$$

where I^{rec} and I^{ref} denote the pixel intensity in current frame and reference frame respectively. k indicates the current frame index that includes the lost macroblock, and $k - 1$ indicates the index of last decoded frame that is used as reference frame.

$Mv_{\tilde{x}, \tilde{y}}^H$ and $Mv_{\tilde{x}, \tilde{y}}^V$ denote the horizontal and vertical part of $Mv_{\tilde{x}, \tilde{y}}$ respectively.

By using these corresponding pixels, the lost macroblock can be reconstructed. The simulation results prove that our method can improve the video quality obviously.

4.4 Simulation Results of the Motion Vector Recovery Method

We use five video sequences, Stefan, Foreman, Coastguard, Mobile and Tempete, in our experiment. The videos are encoded and decoded by H.26L-PFGS program. In our simulation, the frame structure of the encoded video is IPPPIPP..... By changing the

quantization factor, the videos are encoded at different bitrates. The location of lost macroblock is selected randomly. Since H.264 is a new video coding standard, there are not many error concealment approaches that are designed for this new coding standard. Moreover, the motion estimation schemes adopted in H.264 are different with other coding standards. The video compressed by H.264 cannot be concealed efficiently by the error concealment schemes, which are based on 16×16 macroblock motion estimation. We only compare our algorithm with two of the most common error concealment methods--temporal replacement and boundary matching algorithm. The simulation results are presented in Table 4.1 and 4.2. The macroblock loss rates we used are 2%, 5%, 10% and 15%. In Table 4.1, $PSNR_L$ denotes the PSNR of corrupted video with different macroblock loss rates. $PSNR_T$ denotes the PSNR of video concealed by temporal replacement. $PSNR_C$ represents the PSNR of video that is recovered by our algorithm. The simulation results show that the video quality is obviously improved by the proposed method. Despite the fast motion in Stefan, the proposed method is able to competently recover the motion information.

Figure 4.4 shows the change of PSNR with bitrates. In Figure 4.4(a), when the macroblock loss rate is 5%, the quality of video recovered by proposed algorithm is closed to the video without any error. In Figure 4.4(b), 10% macroblocks are lost in each frame. The performance of the proposed algorithm is obviously better than the temporal replacement.

Table 4. 1 The simulation results of our algorithm. (a) Foreman. (b) Stefan. (c) Coastguard. (d) Mobile. (e) Tempete.

(a)

Original PSNR (dB)	QP	Bitrate (Kbit/s)	PSNR (dB)	PSNR of Different Macroblock Loss Rates(dB)			
				2%	5%	10%	15%
28.78	28	79.89	PSNR _L	21.75	17.66	15.19	14.78
			PSNR _T	28.02	27.16	25.98	24.98
			PSNR _C	28.61	27.75	26.72	25.45
31.42	24	119.42	PSNR _L	22.18	17.82	15.28	14.86
			PSNR _T	30.22	28.95	27.69	25.22
			PSNR _C	31.07	29.84	28.25	26.66
33.86	20	191.26	PSNR _L	22.42	17.91	15.33	14.90
			PSNR _T	32.39	30.56	28.43	26.58
			PSNR _C	33.30	31.72	29.72	27.65
36.28	16	302.97	PSNR _L	22.57	17.96	15.36	14.92
			PSNR _T	33.55	31.20	29.51	26.44
			PSNR _C	35.39	33.47	30.72	28.38

(b)

Original PSNR (dB)	QP	Bitrate (Kbit/s)	PSNR (dB)	PSNR of Different Macroblock Loss Rates(dB)			
				2%	5%	10%	15%
25.74	28	175.83	PSNR _L	21.05	17.17	15.37	14.65
			PSNR _T	24.85	23.73	22.75	22.06
			PSNR _C	25.34	24.74	24.15	23.87
28.75	24	296.28	PSNR _L	21.10	17.49	15.56	14.67
			PSNR _T	27.06	26.06	23.84	22.53
			PSNR _C	28.17	27.33	25.21	24.44
31.53	20	459.48	PSNR _L	21.53	18.07	15.86	14.84
			PSNR _T	28.97	27.28	24.35	23.69
			PSNR _C	30.21	28.71	26.80	25.55
34.72	16	736.32	PSNR _L	22.63	18.58	16.23	15.30
			PSNR _T	30.43	27.16	24.92	23.31
			PSNR _C	32.97	31.23	27.27	26.02

(c)

Original PSNR (dB)	QP	Bitrate (Kbit/s)	PSNR (dB)	PSNR of Different Macroblock Loss Rates(dB)			
				2%	5%	10%	15%
27.78	28	53.56	PSNR _L	21.29	19.21	16.34	14.04
			PSNR _T	27.12	25.96	25.80	24.09
			PSNR _C	27.66	26.76	26.45	24.83
29.64	24	101.76	PSNR _L	22.10	19.23	16.35	14.70
			PSNR _T	28.88	26.82	25.52	25.09
			PSNR _C	29.31	28.70	27.03	25.95
31.83	20	191.43	PSNR _L	22.61	19.38	16.45	14.98
			PSNR _T	30.07	29.47	25.73	25.84
			PSNR _C	31.49	30.49	28.85	27.04
34.46	16	374.24	PSNR _L	23.76	19.52	16.48	16.22
			PSNR _T	33.06	31.20	28.53	27.19
			PSNR _C	33.37	32.02	31.35	28.19

(d)

Original PSNR (dB)	QP	Bitrate (Kbit/s)	PSNR (dB)	PSNR of Different Macroblock Loss Rates(dB)			
				2%	5%	10%	15%
24.69	28	213.00	PSNR _L	19.01	16.01	13.86	12.52
			PSNR _T	23.77	23.27	22.48	21.32
			PSNR _C	24.16	23.80	23.00	22.26
27.37	24	357.60	PSNR _L	19.58	16.28	14.03	12.64
			PSNR _T	25.87	24.87	24.19	22.80
			PSNR _C	26.50	25.98	24.71	23.67
30.20	20	576.13	PSNR _L	20.05	16.96	13.94	12.68
			PSNR _T	28.20	27.02	25.03	24.14
			PSNR _C	28.79	28.02	25.74	25.15
33.48	16	945.11	PSNR _L	20.15	16.54	14.18	12.76
			PSNR _T	30.71	29.34	26.57	25.17
			PSNR _C	31.15	30.23	27.44	25.74

(e)

Original PSNR (dB)	QP	Bitrate (Kbit/s)	PSNR (dB)	PSNR of Different Macroblock Loss Rates(dB)			
				2%	5%	10%	15%
25.07	28	145.13	PSNR _L	22.56	20.36	18.72	17.08
			PSNR _T	23.73	23.25	22.71	22.08
			PSNR _C	24.67	24.37	23.81	22.82
27.64	24	271.56	PSNR _L	23.70	21.43	19.82	17.25
			PSNR _T	26.07	25.85	25.04	23.84
			PSNR _C	26.82	26.57	25.54	24.12
30.42	20	483.18	PSNR _L	24.95	21.93	19.30	17.72
			PSNR _T	28.92	27.80	25.14	25.01
			PSNR _C	29.53	28.47	26.23	25.64
33.70	16	817.70	PSNR _L	24.81	21.16	19.92	18.38
			PSNR _T	30.82	28.97	27.18	26.59
			PSNR _C	31.17	30.76	28.97	27.14

Notes:

QP: The quantization parameter used to encode the video sequences.

PSNR_L: The PSNR of corrupted video without any concealment.

PSNR_T: The PSNR of video recovered by temporal replacement

PSNR_C: The PSNR of video recovered by our proposed algorithm.

Table 4. 2 Comparison between first-order plane and boundary matching algorithm.

QP	Method	PSNR of Different Macroblock Loss Rates			
		2%	5%	10%	15%
24	First-Order Plane	31.07	29.84	28.25	26.66
	Boundary Matching	29.58	29.02	27.83	25.65
20	First-Order Plane	33.30	31.72	29.72	27.65
	Boundary Matching	31.95	30.70	29.19	27.11

In Table 4.2, a comparison between the proposed method and the boundary matching algorithm introduced in [120] is presented. In the boundary matching algorithm, the motion vectors of adjacent 8×8 blocks are utilized as the candidate motion vectors for the lost macroblocks. The motion vector that can minimize the

sum of absolute boundary difference between the reconstructed macroblock and its neighboring macroblocks will be used in recovery. The QCIF video sequence, Foreman, is selected to conduct the comparison.

Figure 4. 4 Comparison. (a) Coastguard: 5% macroblocks lost, (b) Coastguard: 10%

macroblocks lost.

4.5 Conclusion

A new temporal error concealment method designed for the new coding standard H.264 is presented in this chapter. In H.264, there are more motion vectors in one macroblock since motion vector covers smaller region than the previous coding standards. This new coding scheme increases the correlation between neighboring motion vectors. By using the motion vectors of neighboring 4×4 blocks that are next to the vertices of the lost macroblock, a first-order plane that can describe the relations of motion vectors within a small area can be constructed. Based on this plane, the motion vector of the vertex located at the corner of lost macroblock can be estimated. We select a linear interpolation equation to compute the motion vector of each pixel belonging to the lost macroblock. The simulation results show that the proposed method can efficiently improve the qualities of different video sequences.

Chapter 5

A Motion Vector Recovery Algorithm Using Lagrange Interpolation

5.1 Introduction

In this chapter, another algorithm that is used to recover the lost motion vectors of H.264 video is introduced. In the previous chapters, the motion estimation scheme used in H.264 has been presented. Compared with previous coding standards, a macroblock encoded by H.264 will contain more motion vectors, and the correlations between neighboring motion vectors increase.

We will present an efficient motion vector recovery algorithm that makes use of the Lagrange interpolation formula in the following sections. Based on the correlations of neighboring motion vectors, we can assume that the lost motion vector has similar

motion tendency with its neighboring motion vectors. Then we can use the Lagrange interpolation formula to construct a polynomial that describes the motion tendency of motion vectors, which are next to the lost motion vector, and use this polynomial to recover the lost motion vector. The simulation results show that our algorithm can efficiently improve the visual quality of corrupted video.

5.2 Approximation

In order to find the motion relation among neighboring macroblocks, we need to formulate a function to describe the change tendency of motion vectors. The approximation theory in numerical analysis can be used to form this described function. In this section, we will introduce the concept of approximation, and how to implement the approximation [119].

5.2.1 Goals and Strategy

A very important class of numerical problems involves computing the values related to a function, not analytically, but as a set of data points. For example, one often wishes to evaluate a function at some argument values other than an argument at which a data value is given.

A function is specified as a set of values at a discrete set of points for two reasons. First, this situation occurs when the function comes from some physical process and is

only known by measuring the value of the function at various points, or when the data values are the results of previous computation. Second, it arises when the computer cannot be used to efficiently evaluate an analytically definable function at arbitrary argument values in a direct way. An example of such a function is $f(x) = \int_0^x \sin(1/(u^2 + 2)) du$. In this case the strategy is in two-parts: (1) evaluate the function in some inefficient way at a set of data points, and (2) for routine evaluation, use techniques of approximation to calculate the function from the discrete number of data points tabulated.

The approximation problem can be stated as follows: (i) given a discrete set of argument values and a corresponding set of function values, each with some errors due to computation or measurement, (ii) find a function defined on a desired real interval such that this function is an appropriately close approximation to the function which produced the data points. With respect to operations on the function such as differentiation, integration, and transforms, the strategy is to apply these operations to the approximating function after the approximation has been carried out.

The simplest criterion for approximation is that the approximating function must pass through all data points. This criterion, which we will call exact matching, is commonly used, though there are many cases where it is not the best criterion. Other criteria depend on evaluating the approximating function $\hat{f}$ at each of the argument values x_i and considering the vector of differences, $\hat{f}(x_i) - y_i$, where y_i is the data value at x_i . That is, consider three vectors: x , the vector of argument values; y , the vector of corresponding data values; and $\hat{f}$, the vector of values of

the function evaluated at the argument values. Then we produce a class of criteria by trying to minimize the norm of $\hat{f} - y$.

Which criterion should be used? If we know that the data values are very accurate and that the selected form of approximating function is very good for the data we have, it makes sense to use the exact-matching criterion. If the data values have significant errors, it does not make sense to force the approximating function to pass through the data points, only near them.

5.2.2 Interpolation Methods

The most common approximating functions are the polynomials. The most common criterion for approximation is exact matching. In both cases, the popularity is due to the mathematical tractability of analysis and is not due to the quality of the approximation thus produced.

Interpolation consists of the evaluation of a function at some argument values other than the points at which it is tabulated. We accomplish this by fitting a function through some set of the tabulated data points, whose argument values are close to the arguments at which we desire to evaluate the function. Then we evaluate the fitted function at the argument in question. Fitting a polynomial which passes through $N + 1$ data points depends on the following theorem.

Theorem: Given a set (x_i, y_i) , $i = 0, \dots, N$, where the x_i are distinct, there

exists a unique polynomial $p(x)$ of degree N or less for which

$$p(x_i) = y_i, \quad 0 \leq i \leq N.$$

The importance of this theorem is twofold. First, given a set of data points with distinct arguments, it states that there are some polynomials passing through the data points; second, it states that no matter what method we use to compute the polynomial, we will always get the same polynomial, except for computational error. Many methods for finding this polynomial can be developed. It is important to recognize that for a given set of data points these methods differ only in their computational errors and efficiencies.

In the following parts, we prove the uniqueness of the polynomial passing through (x_i, y_i) . Then the format of the polynomial is described. That explains the existence of the polynomial.

Proof: For the moment, assume there are polynomials that can passing through all points, and let us prove uniqueness. Assume there exists two polynomials $p(x)$ and $q(x)$, both of degree N or less and both passing through all $N + 1$ data points:

$$p(x_i) = y_i, \quad i = 0, 1, \dots, N \quad (5.1)$$

and
$$q(x_i) = y_i, \quad i = 0, 1, \dots, N. \quad (5.2)$$

Consider
$$s(x) = p(x) - q(x) \quad (5.3)$$

We know that $s(x)$ is a polynomial of degree N or less, because it is the difference of two such polynomials. But note that

$$s(x_i) = 0, \quad i = 0, 1, \dots, N \quad (5.4)$$

The fundamental theorem of algebra states that a N th-degree polynomial has only N distinct zeros unless the polynomial is identically equal to zero. Here we have the polynomial $s(x)$ with $N + 1$ zeros, so

$$s(x) \equiv 0 \quad (5.5)$$

which implies $p(x) \equiv q(x)$ (5.6)

Equation (5.6) contradicts our assumption that p and q are distinct polynomials, so we know there can exist only one polynomial with the required properties.

To complete the proof, we will construct the required polynomial. Note that a N th-degree polynomial has $N + 1$ parameters, and we have $N + 1$ independent constraints given by the exact matching criterion to apply at each data point. Thus, we should not be surprised at the existence and uniqueness of the solution. Put more precisely, the constraints are of the form

$$y_i = \sum_{j=0}^N a_j x_i^j, \quad i = 0, 1, \dots, N \quad (5.7)$$

Note that each equation is a linear equation of a_j . For the existence and uniqueness of the solution, we would require that matrix whose ij th element is $(x_i)^j$ to be nonsingular.

5.2.3 Lagrange Polynomials

In the previous subsection, the uniqueness of the polynomial passing through all $N + 1$ points had been proven. Now, the existence of the polynomial will be proven.

Proof: The basis of the construction of the exact matching polynomial is to write it in the form

$$p(x) = \sum_{j=0}^N y_j L_j(x) \quad (5.8)$$

where each $L_j(x)$ is a polynomial of degree N . Equation (5.1) will be satisfied if $L_j(x)$ is 0 at every tabular argument except x_j and is 1 at x_j . If each $L_j(x)$ has the above properties, $p(x_i)$ will have the value y_i , because the fact that $L_j(x_i) = 0$ for $j \neq i$ will cause the y_i for $j \neq i$ to be multiplied by 0 and y_i to be multiplied by 1. Thus, we have reduced our problem to finding a set of polynomials $L_j(x)$, called the **Lagrange polynomials**, such that for all j

$$L_j(x_i) = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases} \quad (5.9)$$

Since $L_j(x)$ is a polynomial, to have a zero at $x = x_i$, it must have a factor $(x - x_i)$. Thus, we have

$$L_j(x) = q(x) \prod_{\substack{i=0 \\ i \neq j}}^N (x - x_i) \quad (5.10)$$

where $q(x)$ is a polynomial. But $L_j(x)$ is a polynomial of degree N or less, so $q(x)$ must be constant. The constant is determined by the one constraint remaining to be satisfied, namely,

$$L_j(x_j) = 1 \quad (5.11)$$

producing

$$1 = q \prod_{\substack{i=0 \\ i \neq j}}^N (x_j - x_i) \quad (5.12)$$

from which follows

$$L_j(x) = \left(\prod_{\substack{i=0 \\ i \neq j}}^N (x - x_i) \right) / \left(\prod_{\substack{i=0 \\ i \neq j}}^N (x_j - x_i) \right) \quad (5.13)$$

We have now constructed the polynomial $p(x)$ with the required properties.

Note that the $L_j(x)$ depends only on the x_i , not on the y_i . A method that writing the fitting polynomial as a linear combination of polynomials depending only on the x_i values, can be used to find polynomials satisfying more general constraints that exact matching at function values. In particular, the method can be used to find a polynomial $p(x)$ which agrees with the input function f in value and in the first k_i of its derivatives at x_i , for $0 \leq i \leq N$, where k_i may vary with i (from point to point):

$$p^{(m)}(x_i) = y_i^{(m)}, \quad m = 0, 1, \dots, k_i; \quad i = 0, 1, \dots, N \quad (5.14)$$

where $p(x)$ is a polynomial of degree $1 + \sum_{i=0}^N (k_i + 1)$.

Let us illustrate the use of Lagrange polynomials by fitting a first-degree polynomial, a straight line, through the two points (x_0, y_0) and (x_1, y_1) . The Lagrange polynomials are

$$L_0(x) = (x - x_1)/(x_0 - x_1) \quad (5.15)$$

and

$$L_1(x) = (x - x_0)/(x_1 - x_0) \quad (5.16)$$

Thus, the approximating line is

$$p(x) = y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0} \quad (5.17)$$

which can be written in its more customary form as

$$p(x) = \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1 \quad (5.18)$$

A weighted average y_i of y_0 and y_1 , where if x is between x_0 and x_1 , is the complement of the proportion of the distance between x_0 and x_1 that is between x and x_1 . Equation (5.17) can also be written as

$$p(x) = y_0 + \frac{x - x_0}{x_1 - x_0} (y_1 - y_0) \quad (5.19)$$

where the interpolated value is written as the value at the beginning of the interval plus the product of the tabular y interval and the fraction of the full tabular x interval taken by $x - x_0$.

5.3 Described Function of Lost Motion Vectors

In order to present the change tendency of motion vector, we use the Lagrange Interpolation function to formulate a describe function.

The lost macroblock is represented as $M_{i,j}$, i and j denote the spatial location of macroblocks. Based on the interleaving technique that packs neighboring macroblocks into different packets, we can assume that the four neighboring macroblocks are correctly decoded. The corresponding locations of the lost macroblock and its neighboring macroblocks are shown in Figure 5.1. $M_{v_0}, \dots, M_{v_3}$ denote the neighboring motion vectors that cover neighboring 4×4 blocks. $V_0, \dots, V_3$ present the motion vectors that we want to recover, each of them covers a 4×4 block. For simplicity, we only present how to use the horizontal neighboring motion vectors to recover the $V_0, \dots, V_3$. Other motion vectors in macroblock $M_{i,j}$ can be recovered

in the same way.

Figure 5. 1 The lost macroblock and its neighboring macroblocks.

Table 5. 1 The corresponding coordinates of each motion vector.

y_i	Mv_0	Mv_1	V_0	V_1	V_2	V_3	Mv_2	Mv_3
x_i	-3.5	-2.5	-1.5	-0.5	0.5	1.5	2.5	3.5

0

In the new coding standard H.264, a 16×16 macroblock can be divided into different block shapes for motion estimation. The number of motion vector changes from 1 to 16 in one macroblock. Based on the correlation of neighboring motion vectors, the Lagrange interpolation formula can be used to estimate the motion vectors in the lost macroblocks. To build the polynomial, we establish a horizontal reference frame in the lost macroblock. The center of the lost macroblock is the origin of the reference frame, and the size of a 4×4 block is one unit. Then the centre of each block in the lost macroblock has a corresponding coordinate. We can have a set of given points as presented in Table 5.1.

Based on the Lagrange interpolation formula, we can use the neighboring motion vectors to construct a polynomial. To save the computational cost, we only select four nearest motion vectors to establish the polynomial. The parameters of this polynomial can be computed by Equation (5.20).

$$L_0^{(3)}(x) = -\frac{(x+2.5)(x-2.5)(x-3.5)}{42} \quad (5.20a)$$

$$L_1^{(3)}(x) = \frac{(x+3.5)(x-2.5)(x-3.5)}{30} \quad (5.20b)$$

$$L_2^{(3)}(x) = -\frac{(x+3.5)(x+2.5)(x-3.5)}{30} \quad (5.20c)$$

$$L_3^{(3)}(x) = \frac{(x+3.5)(x+2.5)(x-2.5)}{42} \quad (5.20d)$$

Then the polynomial can be written as Equation (5.21). By using the predefined coordinate of each block, the $L_3^{(0)} \cdots L_3^{(3)}$ can be considered as constants that are corresponding to the locations of blocks.

$$\begin{aligned} I(x) = & Mv_0 \left[-\frac{(x+2.5)(x-2.5)(x-3.5)}{42} \right] \\ & + Mv_1 \left[\frac{(x+3.5)(x-2.5)(x-3.5)}{30} \right] \\ & + Mv_2 \left[-\frac{(x+3.5)(x+2.5)(x-3.5)}{30} \right] \\ & + Mv_3 \left[\frac{(x+3.5)(x+2.5)(x-2.5)}{42} \right] \end{aligned} \quad (5.21)$$

5.4 Recover the Motion Vectors of Lost Macroblock

By using Equation (5.21), we can recover the motion vectors in the lost macroblock respectively. To make the movement change smoothly in the lost macroblock, we use Equation (5.21) to estimate the V_0, V_3 , and utilize them with the neighboring motion

vectors to compute V_1 and V_2 . By iterating Lagrange interpolation formula, we construct other two polynomials to recover V_1 and V_2 . For V_1 , we use $(-2.5, Mv_1)$ and $(-1.5, V_0)$ to form a linear polynomial that is presented by Equation (5.22a). And for V_2 , we use $(1.5, V_3)$ and $(2.5, Mv_2)$ to build the polynomial, as shown in Equation (5.22b). By using Equation (5.22), we can obtain better recovery results of V_1 and V_2 .

$$I(x) = Mv_1[-(x+1.5)] + V_0[x+2.5] \quad (5.22a)$$

$$I(x) = V_3[-(x-2.5)] + Mv_2[x-1.5] \quad (5.22b)$$

Then we have the recovered motion vectors $V_0, \dots, V_3$ that are predicted by horizontal neighboring motion vectors. By iterating this method, all motion vectors that cover the 4×4 blocks in the lost macroblock can be recovered by the neighboring motion vectors in horizontal direction. We denote this set of motion vectors that are predicted by horizontal neighboring motion vectors as $V_0^h, \dots, V_{15}^h$.

By using the same recovery approach, we can obtain another set of recovered motion vectors that are predicted by the neighboring motion vectors in vertical direction. This set of recovered motion vectors can be presented as $V_0^e, \dots, V_{15}^e$. In [78], the author had demonstrated that a more robust performance can be achieved if the concealed block is a weighted sum of a number of candidate concealments. Based on this statement, we use the average of motion vectors that are recovered by neighboring motion vectors in horizontal and vertical directions to reconstruct the corrupted macroblock.

$$V_i^{rec} = \frac{V_i^h + V_i^e}{2}, \quad i = 0, \dots, 15 \quad (5.23)$$

If the horizontal or vertical neighboring macroblocks are not available, we can only use the neighboring motion vectors in a single direction to recover the lost motion vectors. The quality of video that is reconstructed by neighboring motion vectors in single direction is worse than the video that is reconstructed by neighboring motion vectors in both directions.

5.5 Recovery Performance of Lagrange Interpolation

We select four video sequences, Foreman, Stefan, Mobile and Tempete, to conduct the experiment. The videos are encoded and decoded by TML-90 program. The frame structure in our simulation is IPPPIPP..... By changing the quantization factor, the videos are encoded at different bitrates. To prove the performance of the proposed method, the temporal replacement and boundary matching algorithm are selected to compare with the proposed method. By using the temporal replacement [113], the lost motion vectors will be set as (0,0). The boundary matching algorithm used in simulation is presented by Wang *et al* [120], which is designed for the H.26L test model (previous version of H.264). In their algorithm, the motion vectors of 8×8 blocks that are adjacent to the lost macroblock are calculated as the average of the motion vectors of the spatially corresponding 4×4 or other shaped (e.g. 4×8) blocks. And each of the candidate motion vectors that cover neighboring 8×8 blocks will be used to reconstruct the lost macroblock. The motion vector that can minimize the sum

of absolute boundary differences between the reconstructed macroblock and its neighboring macroblocks will be used to replace the motion vector of the lost macroblock.

Based on the interleaving technique, we can assume that the macroblocks that are adjacent to the lost macroblock are correctly decoded, and the motion vectors of four neighboring macroblocks are available. In this situation, we can use the neighboring motion vectors in horizontal and vertical directions to recover the lost motion vectors. The location of the lost macroblock is selected randomly. The macroblock loss rates we used to conduct the simulation are 2%, 5%, 10% and 15% respectively. The PSNRs of each scenario are presented in Table 5.2. QP is the quantization parameter used in encoder. $PSNR_{NE}$ denotes the PSNR of corrupted video with different macroblock loss rates. $PSNR_{TR}$ denotes the PSNR of video concealed by temporal replacement. $PSNR_{BM}$ represents the PSNR of video recovered by boundary matching algorithm. $PSNR_{LA}$ represents the PSNR of video that is reconstructed by the proposed algorithm. The simulation results show that the proposed algorithm can efficiently improve the quality of corrupted video. Compared with the temporal replacement and boundary matching algorithm, the proposed algorithm can achieve better visual quality. And the computational cost of the proposed algorithm is less than the boundary matching algorithm. For example, when the macroblock loss rate is 10% in each frame, the time used by the proposed method to calculate the lost motion vectors of 50 frames is 0.422 second, and the time needed by boundary matching algorithm is 1.438 second. This makes the proposed method more suitable for the real-time video transmission.

Figure 5. 2 Recovered by motion vectors in both directions. (a) Tempete: 5% macroblocks lost. (b) Tempete: 10% macroblocks lost.

Figure 5.2 shows the change tendency of PSNR along different bitrates. In Figure 5.2(a), the macroblock loss rate is 5%. As the bitrate increases, the improvement of PSNR achieved by the proposed algorithm becomes larger. In Figure 5.2(b), 10% macroblocks are lost in each frame. The simulation results prove that the proposed method can obtain better reconstruction quality in high macroblock loss rate.

Table 5. 2 The simulation results of proposed algorithm with interleaving. (a) Foreman with interleaving. (b) Stefan with interleaving. (c) Mobile with interleaving. (d) Tempete with interleaving.

(a)

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Macroblock Loss Rates			
				2%	5%	10%	15%
24	119.42	31.42	PSNR _{NE}	21.32	17.36	15.07	14.03
			PSNR _{TR}	29.39	28.76	27.19	25.00
			PSNR _{BM}	29.58	29.02	27.83	25.65
			PSNR _{LA}	30.68	29.97	28.77	27.19
20	191.26	33.86	PSNR _{NE}	22.58	17.51	15.73	14.13
			PSNR _{TR}	31.61	30.30	28.96	26.64
			PSNR _{BM}	31.95	30.70	29.19	27.11
			PSNR _{LA}	33.23	32.10	30.24	28.15
16	302.97	36.28	PSNR _{NE}	23.04	17.70	15.97	14.43
			PSNR _{TR}	33.34	30.93	29.73	27.59
			PSNR _{BM}	34.23	33.25	30.23	28.15
			PSNR _{LA}	35.11	33.55	31.58	29.36

(b)

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Macroblock Loss Rates			
				2%	5%	10%	15%
24	296.28	28.75	PSNR _{NE}	21.54	18.39	16.07	14.53
			PSNR _{TR}	27.39	25.49	23.69	23.33
			PSNR _{BM}	27.59	26.79	25.40	24.10
			PSNR _{LA}	28.04	27.37	26.14	25.11
20	459.48	31.53	PSNR _{NE}	22.01	18.49	16.12	14.93
			PSNR _{TR}	29.30	27.37	25.07	23.64
			PSNR _{BM}	30.14	28.25	26.91	25.46
			PSNR _{LA}	30.28	28.74	27.53	25.56
16	736.32	34.72	PSNR _{NE}	22.29	18.98	16.15	14.96
			PSNR _{TR}	30.50	27.05	25.19	24.02
			PSNR _{BM}	32.22	30.52	27.00	25.92
			PSNR _{LA}	32.95	30.70	27.62	26.75

(c)

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Macroblock Loss Rates			
				2%	5%	10%	15%
24	357.60	27.37	PSNR _{NE}	19.79	16.49	13.86	12.48
			PSNR _{TR}	26.53	25.65	24.14	23.48
			PSNR _{BM}	26.70	25.88	24.45	23.69
			PSNR _{LA}	26.78	26.08	25.10	24.08
20	576.13	30.20	PSNR _{NE}	20.00	16.97	13.96	12.79
			PSNR _{TR}	27.99	27.09	25.77	24.37
			PSNR _{BM}	28.29	28.16	26.07	24.47
			PSNR _{LA}	29.00	28.28	26.91	25.79
16	945.11	33.48	PSNR _{NE}	20.26	16.99	14.04	12.82
			PSNR _{TR}	30.14	29.37	26.34	25.19
			PSNR _{BM}	30.72	29.91	26.52	25.74
			PSNR _{LA}	30.91	30.33	28.45	26.69

(d)

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Macroblock Loss Rates			
				2%	5%	10%	15%
24	217.56	27.64	PSNR _{NE}	23.20	20.77	19.09	17.98
			PSNR _{TR}	26.18	25.77	24.22	23.50
			PSNR _{BM}	26.81	26.23	24.90	24.42
			PSNR _{LA}	27.12	26.46	25.76	25.09
20	483.18	30.42	PSNR _{NE}	24.11	22.35	19.60	18.09
			PSNR _{TR}	28.77	27.27	25.07	25.25
			PSNR _{BM}	29.24	27.98	25.90	25.65
			PSNR _{LA}	29.29	28.67	27.45	26.35
16	817.70	33.70	PSNR _{NE}	24.32	22.51	20.06	18.66
			PSNR _{TR}	30.60	28.87	27.16	26.48
			PSNR _{BM}	31.52	29.43	27.54	26.04
			PSNR _{LA}	31.85	30.36	29.16	27.24

If the encoder does not adopt interleaving technique, the macroblocks in a row will be packed into the same slice. When a packet loses, we will only have the neighboring motion vectors in vertical direction to recover the lost motion vectors. In the simulation, each slice contains fixed number of macroblocks. We have tested the performance of the proposed method in different bitrates. The simulation data in this scenario are presented in Table 5.3. Figure 5.3 shows the performance of the tested

error concealment methods with different bitrates. The proposed method can improve the quality of corrupted video obviously.

Figure 5. 3 Recovered by motion vectors in one direction. (a) Tempete: 5% packets lost. (b) Tempete: 10% packets lost.

Table 5. 3 The simulation result of proposed algorithm without interleaving. (a)

Foreman without interleaving. (b) Stefan without interleaving. (c) Mobile without interleaving. (d) Tempete without interleaving.

(a)

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Macroblock Loss Rates			
				2%	5%	10%	15%
24	119.42	31.42	PSNR _{NE}	22.06	17.87	15.16	13.89
			PSNR _{TR}	29.52	29.30	26.86	25.08
			PSNR _{BM}	28.87	28.54	27.63	25.45
			PSNR _{LA}	30.30	29.88	28.23	26.63
20	191.26	33.86	PSNR _{NE}	22.17	17.94	14.90	14.00
			PSNR _{TR}	30.68	30.56	28.61	26.22
			PSNR _{BM}	31.64	30.66	28.92	27.03
			PSNR _{LA}	32.79	31.50	29.68	27.87
16	302.97	36.28	PSNR _{NE}	22.52	17.36	15.62	13.91
			PSNR _{TR}	33.33	31.81	29.41	27.69
			PSNR _{BM}	34.00	32.88	30.14	27.94
			PSNR _{LA}	34.84	32.79	30.90	28.88

(b)

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Macroblock Loss Rates			
				2%	5%	10%	15%
24	296.28	28.75	PSNR _{NE}	22.43	18.03	15.45	14.58
			PSNR _{TR}	27.26	25.87	23.62	22.75
			PSNR _{BM}	27.41	26.73	25.13	23.98
			PSNR _{LA}	28.02	27.25	25.84	24.76
20	459.48	31.53	PSNR _{NE}	22.63	18.47	16.07	15.06
			PSNR _{TR}	29.44	26.92	25.18	24.14
			PSNR _{BM}	29.86	28.19	26.48	25.05
			PSNR _{LA}	30.17	28.66	26.74	25.82
16	736.32	34.72	PSNR _{NE}	22.95	19.00	16.12	15.29
			PSNR _{TR}	30.85	27.63	25.01	24.20
			PSNR _{BM}	31.94	30.51	26.76	25.88
			PSNR _{LA}	32.71	30.63	27.49	26.39

(c)

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Macroblock Loss Rates			
				2%	5%	10%	15%
24	357.60	27.37	PSNR _{NE}	20.19	16.61	14.27	12.57
			PSNR _{TR}	26.38	25.43	23.98	23.61
			PSNR _{BM}	26.57	25.72	24.31	23.56
			PSNR _{LA}	26.69	25.96	24.51	23.82
20	576.13	30.20	PSNR _{NE}	20.68	16.95	14.38	13.05
			PSNR _{TR}	27.79	27.58	25.44	24.38
			PSNR _{BM}	28.15	27.84	26.01	24.65
			PSNR _{LA}	28.79	28.07	26.72	25.40
16	945.11	33.48	PSNR _{NE}	20.85	17.53	14.72	13.41
			PSNR _{TR}	30.10	29.09	26.18	25.02
			PSNR _{BM}	30.75	29.74	26.25	25.57
			PSNR _{LA}	30.56	29.66	28.02	26.33

(d)

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Macroblock Loss Rates			
				2%	5%	10%	15%
24	217.56	27.64	PSNR _{NE}	24.73	21.37	17.92	17.41
			PSNR _{TR}	26.19	25.74	24.12	23.82
			PSNR _{BM}	26.79	26.07	24.29	24.16
			PSNR _{LA}	26.98	26.31	25.06	24.33
20	483.18	30.42	PSNR _{NE}	23.98	21.82	19.66	17.48
			PSNR _{TR}	28.67	27.94	25.63	24.65
			PSNR _{BM}	28.97	28.20	25.78	25.62
			PSNR _{LA}	29.01	28.45	26.90	26.01
16	817.70	33.70	PSNR _{NE}	25.38	22.02	19.41	17.66
			PSNR _{TR}	30.96	28.96	26.85	26.40
			PSNR _{BM}	31.58	29.37	27.23	26.33
			PSNR _{LA}	31.67	30.19	28.93	27.02

Figure 5. 4 The subjective performance of the three algorithms. (a) Corrupted Video, PSNR = 12.2321. (b) Temporal Replacement, PSNR = 30.3096. (c) Boundary matching, PSNR = 30.9625. (d) Proposed Method, PSNR = 34.0804.

In Figure 5.4, we give an example to illustrate the visual quality of the proposed algorithm. 10% packets are lost, and the motion vectors in the lost macroblocks are estimated by the neighboring motion vectors in vertical direction. The results show that the proposed method can produce reasonable visual quality.

5.6 Conclusion

In this chapter, we present an efficient motion vector recovery algorithm for H.264. Compared with previous coding standards, the neighboring motion vectors in H.264 have higher correlation. By exploiting this property of H.264, we can use the motion

vectors that are adjacent to the lost motion vector to build a polynomial, which can reflect the motion tendency of these neighboring motion vectors. The parameter in this polynomial can be computed by Lagrange interpolation. Based on the constructed polynomial, the lost motion vectors can be estimated. The simulation results show that the proposed approach can efficiently improve the visual quality of corrupted video.

Chapter 6

Efficient Temporal Error Concealment Based on a Polynomial Model

6.1 Introduction

In the previous chapter, we have presented an error concealment method that is based on Lagrange interpolation formula. Exact matching criterion is used to formulate the objective function of motion vectors. In this chapter, we use the least-squares criterion to construct a polynomial that can also mimic the change tendency of motion vectors. We choose a norm, which measures the difference between the approximating function and the data, and we formulate a polynomial by minimizing that norm [119].

The compressed video bitstream is very sensitive to the channel errors. A single bit error may cause serious degradation of video quality. To eliminate the effect of bit error,

many efficient error concealment algorithms have been proposed in recent years. In the new coding standard H.264, the motion estimation scheme is based on different block shapes. Then most previous motion vector recovery algorithms that are designed for H.263 or MPEG-4 cannot perform well for H.264. Wang *et al* [120] proposed a modified boundary matching algorithm for H.26L, which is the previous version of H.264.

In this chapter, we introduce an efficient error concealment algorithm for H.264 that is based on a polynomial model. As the number of motion vectors in a macroblock increases, the correlation between neighboring motion vectors increases. Then we can use a polynomial model to reflect the correlation between neighboring motion vectors. This polynomial model is built by the least-squares approximation theory, and the model results in an approximation function that can describe the change tendency of the motion vectors within a small area. The related mathematical background is presented in next section. Based on the property of this polynomial model, we can obtain an approximation of the lost motion vector from the neighboring motion vectors. Then the lost macroblock can be reconstructed. The simulation results prove that the proposed algorithm can efficiently improve the quality of corrupted video.

6.2 Least-Squares Approximation

In the following subsections, we assume that the measured function values y_i are in error but the argument values x_i are error free. In other words, y_i are random

variables while x_i are deterministic; each x_i is determined by the experimenter as an independent variable, by which experimenter chooses to measure the corresponding y_i .

6.2.1 Fitting a Straight Line

Perhaps Least Squares (LS) is the most widely used technique in geophysical data analysis. In least squares, the parameters to be estimated must arise in expressions that are relative to the observations [137]. When the parameters appear linearly in these expressions, the least squares estimation problem can be solved in closed form, and it is relatively straightforward to derive the statistical properties of the resulting parameters.

In order to illustrate the more general problem, we will treat one very simple example, fitting a straight line to a collection of pairs of observations (x_i, y_i) , $i = 1, 2, \dots, n$, in some details. We suppose that a reasonable model is

$$y = a_0 + a_1 x \quad (6.1)$$

and we need a mechanism to determine a_0 and a_1 . This is just a special case of the more general problem of fitting a polynomial with order n , for which one would need to find $n+1$ coefficients. The most commonly used method for finding a model is the least squares estimation. It is supposed that x is an independent variable, which is known exactly, while y is a dependent variable. The least squares (LS) estimations for a_0 and a_1 are to find the predicted value of the curve minimizing the sum of the squared deviations from the observation. In other words, the problem is to find the

values of a_0 and a_1 that can minimize the residual sum of squares:

$$F(a_0, a_1) = \sum_{i=1}^n (y_i - a_0 - a_1 x_i)^2 \quad (6.2)$$

Note that this involves the minimization of vertical deviations from the line, so it is not symmetric in y and x . In other words, if x is treated as the dependent variable instead of y , one might well expect a different result.

To find the minimal values of F in Equation (6.2), we can solve the equation resulting from setting

$$\frac{\partial F}{\partial a_0} = 0, \quad \frac{\partial F}{\partial a_1} = 0 \quad (6.3)$$

namely

$$\sum_i y_i = n\hat{a}_0 + \hat{a}_1 \sum_i x_i \quad (6.4a)$$

$$\sum_i x_i y_i = \hat{a}_0 \sum_i x_i + \hat{a}_1 \sum_i x_i^2 \quad (6.4b)$$

where $\hat{a}_i$ is the estimate of a_i . Solving for the $\hat{a}_i$ yields the least squares parameter estimates:

$$\hat{a}_0 = \frac{\sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i}{n \sum x_i^2 - (\sum x_i)^2} \quad (6.5a)$$

$$\hat{a}_1 = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2} \quad (6.5b)$$

where $\sum$'s are implicitly from $i=1$ to n in each case. After generating these estimates, it is natural to wonder how much faith we should have in $\hat{a}_0$ and $\hat{a}_1$, and whether the fit to the data is reasonable. In order to assess the goodness of fitting in the resulting mode, we can consider the statistical properties of the least squares estimates, and find how regression is related to correlation.

6.2.2 Assess Fitting

In order to assess the goodness of fitting, we need to adopt some kinds of statistical models for the noise associated with the measurements. In the *Standard Statistical Model* (SSM), which is a statistics collection module, we suppose that y is a linear function of x plus some random noise,

$$y_i = a_0 + a_1 x_i + e_i \quad i = 1, \dots, n \quad (6.6)$$

In Equation (6.6), the values of x_i are taken to be fixed, while e_i are independent random variables with $E(e_i) = 0$ and $Var(e_i) = \sigma^2$, but for the time being we make no further assumption about the exact distribution of e_i .

Under the SSM, it is straightforward to show that the least squares estimate for a straight line is unbiased, which means $E[\hat{a}_i] = a_i$. To do this for a_0 , we make use of the fact that $E[y_i] = a_0 + a_1 x_i$ and take the expected value of a_0 in Equation (6.5).

This yields:

$$\begin{aligned} E[\hat{a}_0] &= \frac{\sum x_i^2 \sum E[y_i] - \sum x_i \sum x_i E[y_i]}{n \sum x_i^2 - (\sum x_i)^2} \\ &= \frac{\sum x_i^2 (na_0 + a_1 \sum x_i) - \sum x_i (a_0 \sum x_i + a_1 \sum x_i^2)}{n \sum x_i^2 - (\sum x_i)^2} \\ &= a_0 \end{aligned} \quad (6.7)$$

A similar proof establishes that $E[\hat{a}_1] = a_1$. It should be noted that this proof only uses $E[e_i] = 0$ and the fact that the errors are additive.

Using the SSM, we have that $Var[y_i] = \sigma^2$ and $Cov[y_i, y_j] = 0$ for $i \neq j$. By making use of these properties, it is possible to calculate the variances for $\hat{a}_i$ as

$$Var[\hat{a}_0] = \frac{\sigma^2 \sum x_i^2}{n \sum x_i^2 - (\sum x_i)^2} \quad (6.8a)$$

$$\text{Var}[\hat{a}_1] = \frac{n\sigma^2}{n\sum x_i^2 - (\sum x_i)^2} \quad (6.8b)$$

$$\text{Cov}[\hat{a}_0, \hat{a}_1] = \frac{-\sigma^2 \sum x_i^2}{n\sum x_i^2 - (\sum x_i)^2} \quad (6.8c)$$

To show this we make use of the fact that Equation (6.5) can be rewritten in the form:

$$a_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = \frac{\sum (x_i - \bar{x})(y_i)}{\sum (x_i - \bar{x})^2} \quad (6.9)$$

where $\bar{x}$ is the average of $x_i, i = 1, \dots, n$. Then

$$\text{Var}[a_1] = \frac{\sigma^2}{\sum (x_i - \bar{x})^2} \quad (6.10)$$

Similar calculation can be done for the other expressions in Equation (6.8).

From Equation (6.8), we can see that the variances of the slope and the interception depend on x_i and σ^2 . Since x_i are known, we just need a means to find σ^2 . In the SSM, $\sigma^2 = E[y_i - a_0 - a_1 x_i]^2$. So we can estimate σ^2 from the average squared deviations of data about the fitted line:

$$RSS = \sum_i (y_i - \hat{a}_0 - \hat{a}_1 x_i)^2 \quad (6.11)$$

Later we will see that

$$s = \frac{RSS}{n-2} \quad (6.12)$$

is an unbiased estimation of σ^2 . The number of degrees of freedom is $n-2$, because 2 parameters have been estimated from the data. So our recipe for estimating $\text{Var}[\hat{a}_0]$ and $\text{Var}[\hat{a}_1]$ simply involves substituting s^2 for σ^2 in Equation (6.8).

We call these estimates $s_{\hat{a}_0}^2$ and $s_{\hat{a}_1}^2$, respectively.

When e_i are independent random variables that distribute normally, $\hat{a}_0$ and $\hat{a}_1$

will be too, since they are just linear combinations of independent normal random variables. More generally, if e_i are independent and satisfy some assumptions, a version of the Central Limit Theorem will apply, and for large n , $\hat{a}_0$ and $\hat{a}_1$ are approximately normal random variables.

An immediate and important consequence of this is that we can invoke exact/approximate confidence intervals and hypothesis tests based on the distribution properties of $\hat{a}_i$. It can be shown that

$$\frac{\hat{a}_i - a_i}{s_{\hat{a}_i}} \sim t_{n-2} \quad (6.13)$$

and we can use the t -distribution to establish the confidence intervals and hypothesis testing.

The most basic thing to do in assessing the fitting is to use the residual from the model, in this case:

$$\hat{e}_i = y_i - \hat{a}_0 - \hat{a}_1 x_i \quad (6.14)$$

They should be plotted as a function of x , which allows one to see systematic misfit or departures from the SSM. These may indicate the need for a more complex model or transformation of variables.

When the variance of errors is an independent constant of x , the errors are said to be *homoscedastic*. When the opposite is true, they are *heteroscedastic*. When the variance varies with x , sometimes it is possible to find a transformation to correct the problem. For example, instead of $y = ax$, one could try $\sqrt{y} = \gamma\sqrt{x}$. Then $\hat{a} = \hat{\gamma}^2, \dots$

A common scenario one might wish to test is whether the interception is zero. This

can be done by calculating both slope and interception, and finding $s_{\hat{\gamma}_0}$. Then one could use the t -test on the hypothesis $H_0 : \gamma_0 = 0$ with

$$t = \frac{\hat{\gamma}_0}{s_{\hat{\gamma}_0}} \quad (6.15)$$

Another strategy in assessing fit is to look at the sample distribution of residuals, compared to a normal probability plot. Q-Q plots of the residuals can provide a visual means of assessing things like gross departures from normality or identifying outliers. Least squares estimates are not robust against outliers, which can have a large effect on the estimated coefficients, their standard errors and s . This is especially true if outliers correspond to the extreme values of x .

6.3 Motion Vector Recovery Algorithm Using Polynomial Model

6.3.1 The Lost Macroblock and Its Neighbors

In this section, we present an efficient motion vector recovery algorithm. In H.264, there are two methods to assign macroblocks to slice [118], as shown in Figure 6.1. In the first method, the neighboring macroblocks in horizontal direction will be assigned to the same slice. If a packet lost, we can only use the neighboring macroblocks in vertical direction to reconstruct the lost macroblocks, as the neighboring macroblocks in horizontal direction are not available. In the other assignment method, the

interleaving technique is adopted [121] [122]. The neighboring macroblocks in horizontal and vertical directions will be packed into different packets. Based on the interleaving technique, we can assume that the four macroblocks, which are adjacent to the lost macroblock, are correctly received and decoded. And we can obtain more information from the neighboring macroblocks to estimate the motion vectors in the lost macroblock.

Figure 6. 1 Two examples for macroblock assignment to a slice in H.264.

Figure 6. 2 The location of a lost macroblock and its neighboring macroblocks. (a) A slice lost. (b) A macroblock lost.

In the decoder of H.264, a record will be generated during the decoding procedure. When a slice is available for decoding, the macroblocks residing in this slice will be denoted as “correctly received”. Otherwise, the macroblocks will be labeled as “lost”. By using this record, we can find the location of the lost macroblock easily. We denote the lost macroblock as $M_{i,j}$. In Figure 6.2, the lost macroblocks are indicated by the dark squares, and the gray squares indicate the macroblocks whose motion vectors will be used to estimate the lost motion vectors.

6.3.2 Polynomial of Neighboring Motion Vectors

As shown in Figure 6.3, $M_{v_1}, \dots, M_{v_4}$ represent the motion vectors that cover 4×4 blocks in a lost macroblock, and $V_1^l, \dots, V_4^l$ denote the motion vectors in the left neighboring macroblock $M_{i-1,j}$. As the distance between two motion vectors increases, the correlation between them will decrease. Then we select the motion vectors in the left neighboring macroblock to recover the motion vectors that cover left part of the lost macroblock. For simplicity, we only present how to use $V_1^l, \dots, V_4^l$ to recover M_{v_1} and M_{v_2} . The other motion vectors in the lost macroblock can be recovered by reiterating the same approach.

First, we form a one-dimensional frame in the lost macroblock. The origin of this one-dimensional frame is located at the center of lost macroblock. We assume that a pixel is one unit in size, and the centers of each 4×4 block in the lost macroblock and

the neighboring macroblocks have corresponding coordinates. From this one-dimensional frame, we can obtain a set of given points (x_i, y_i) as described in Table 6.1. x_i is the coordinate of block center, and y_i is the vertical or horizontal part of the corresponding motion vectors.

V_1^l	V_2^l	V_3^l	V_4^l	M_{v_1}	M_{v_2}	M_{v_3}	M_{v_4}	V_1^r	V_2^r	V_3^r	V_4^r
	$M_{i-1,j}$				$M_{i,j}$				$M_{i+1,j}$		

Figure 6. 3 The motion vectors in the lost macroblock and neighboring macroblock.

Table 6. 1 The corresponding coordinates of motion vectors.

y_i	V_1^l	V_2^l	V_3^l	V_4^l	M_{v_1}	M_{v_2}	M_{v_3}	M_{v_4}	V_1^r	V_2^r	V_3^r	V_4^r
x_i	-22	-18	-14	-10	-6	-2	2	6	10	14	18	22
	0											

Then, we can use a polynomial model based on least squares estimate to describe the correlations between motion vectors in the neighboring macroblocks. The polynomial is presented in Equation (6.16).

$$P_m(x) = a_0 + a_1x + \dots + a_mx^m \quad (6.16)$$

where $a_0, a_1, \dots, a_m$ are a set of unknown coefficients that can be calculated by the least squares estimation. And this set of coefficients should minimize the squares of differences between $P_m(x)$ and y_i . Then we can represent the squares of differences between $P_m(x)$ and y_i as a function of the independent variables $a_0, a_1, \dots, a_m$, as described in Equation (6.17).

$$\sum_{i=1}^n \delta_i^2 = \sum_{i=1}^n [y_i - P_m(x_i)]^2 = F(a_0, a_1, \dots, a_m) \quad (6.17)$$

To obtain the minimum of $F(a_0, a_1, \dots, a_m)$, the set of coefficients $a_0, a_1, \dots, a_m$ should satisfy Equation (6.18).

$$\frac{\partial F}{\partial a_i} = -2 \sum_{i=1}^n [y_i - \sum_{k=0}^m a_k x_i^k] x_i^j = 0 \quad j = 0, 1, \dots, m \quad (6.18)$$

From Equation (6.18), we can get a set of functions to calculate the coefficients, as presented in Equation (6.19).

$$\begin{aligned} na_0 + a_1 \sum_{i=1}^n x_i + a_2 \sum_{i=1}^n x_i^2 + \dots + a_m \sum_{i=1}^n x_i^m &= \sum_{i=1}^n y_i \\ a_0 \sum_{i=1}^n x_i + a_1 \sum_{i=1}^n x_i^2 + a_2 \sum_{i=1}^n x_i^3 + \dots + a_m \sum_{i=1}^n x_i^{m+1} &= \sum_{i=1}^n y_i x_i \\ \vdots \\ a_0 \sum_{i=1}^n x_i^m + a_1 \sum_{i=1}^n x_i^{m+1} + a_2 \sum_{i=1}^n x_i^{m+2} + \dots + a_m \sum_{i=1}^n x_i^{2m} &= \sum_{i=1}^n y_i x_i^m \end{aligned} \quad (6.19)$$

As we have four motion vectors in the neighboring macroblock, we can use up to third-order polynomial for this interpolation, however the first-order polynomial cannot accurately describe nonlinear movement. The third-order polynomial often results in an oscillatory curve, and it is suitable for the interpolation of data that change quickly. Since the motion vectors we use in interpolation are within a small area, the values of these motion vectors will be closed. The third-order polynomial is not necessary. Besides, the computational cost of the third-order polynomial is higher than the lower order polynomials. The second-order polynomial can represent a smooth curve. Compared with the other two polynomials, the second-order polynomial is more suitable for this application. When the movement in the neighboring macroblocks is linear, the coefficient of the second-order item in the second-order polynomial will

approach to zero, and the second-order polynomial will approximate to the first-order polynomial. Based on the correlation of neighboring motion vectors, we can use the second-order polynomial to estimate the motion vectors, Mv_1 and Mv_2 , in the lost macroblock.

Furthermore, we can extend the polynomial model to two dimensions. The motion vectors in vertical neighboring macroblocks can also be used to estimate the motion vectors in the lost macroblocks. The Mv_1 and Mv_2 , which are estimated by the horizontal neighboring motion vectors, $V_1^l, \dots, V_4^l$, are denoted as Mv_1^h and Mv_2^h . By applying the same method on the neighboring motion vectors in the top neighboring macroblock $M_{i,j-1}$, we can achieve another set of estimated values for Mv_1 and Mv_2 , which are represented as Mv_1^c and Mv_2^c . We use the average of the estimated values that are predicted from horizontal and vertical directions to reconstruct the lost macroblock, as presented in Equation (6.20). The simulation results show that the proposed method can efficiently improve the quality of the corrupted video.

$$MV_i^r = \frac{Mv_i^r + Mv_i^c}{2} \quad (6.20)$$

If the encoder does not adopt interleaving technique, the horizontal macroblocks that are adjacent to the lost macroblock are not available. Then we shall only construct the polynomial model by using the vertical neighboring motion vectors, and obtain one set of estimations for the motion vectors in lost macroblock. The quality of the recovered video may degrade slightly. The detailed simulation results are given in the next section.

6.4 Simulation Results of Polynomial Error Concealment

Four QCIF video sequences, Tempete, Foreman, Mobile and Stefan, are used to evaluate the performance of the proposed algorithm. Each video is encoded and decoded by TML-90 program. The frame structure that we used in simulation is IPPPIPPP..... Changing the quantization factor, we can encode the videos at different bitrates. Because the motion estimation scheme used in H.264 differs from previous coding standards, most motion vector recovery algorithms that are designed for previous coding standards cannot perform optimally for H.264. In order to show the performance of the proposed method, we select temporal replacement and boundary matching algorithm to compare with the proposed algorithm. By using the temporal replacement, the lost motion vectors will be replaced by (0,0). The boundary matching algorithm used in simulation was presented by Wang *et al* [120], which is designed for the H.26L test model (previous version of H.264). In their algorithm, the motion vector of each 8×8 block that is adjacent to the lost macroblock is calculated as the average of the motion vectors of the spatially corresponding 4×4 or other shaped (e.g. 4×8) blocks. And each of the candidate motion vectors that cover the neighboring 8×8 block will be used to reconstruct the lost macroblock. The motion vector, which can minimize the sum of absolute boundary differences between the reconstructed macroblock and its neighboring macroblocks, will be used as the motion vector of the lost macroblock.

Figure 6. 4 PSNR of different macroblock loss rates. (a) Tempete: 5% macroblocks

lost. (b) Tempete: 10% macroblocks lost.

Table 6. 2 Simulation results of different macroblock loss rates. (a) Tempete. (b)

Foreman. (c) Mobile. (d) Stefan.

(a)

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Macroblock Loss Rates			
				2%	5%	10%	15%
24	217.56	27.64	PSNR _L	22.47	21.25	18.70	17.65
			PSNR _C	26.29	25.03	24.86	23.92
			PSNR _B	26.88	26.28	25.85	24.81
			PSNR _O	27.31	26.99	26.47	26.39
20	483.18	30.42	PSNR _L	22.99	21.60	19.46	18.27
			PSNR _C	28.83	27.61	26.06	24.77
			PSNR _B	29.15	28.31	26.82	25.69
			PSNR _O	29.75	29.26	28.33	27.60
16	817.70	33.70	PSNR _L	25.46	22.07	20.00	18.60
			PSNR _C	30.91	30.29	27.35	26.06
			PSNR _B	31.56	30.47	27.71	26.83
			PSNR _O	32.28	31.18	29.73	28.56

(b)

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Macroblock Loss Rates			
				2%	5%	10%	15%
24	119.42	31.42	PSNR _L	21.85	17.59	14.40	13.19
			PSNR _C	29.34	28.80	27.82	25.83
			PSNR _B	29.90	29.54	28.16	26.56
			PSNR _O	31.31	30.98	30.78	30.45
20	191.26	33.86	PSNR _L	21.91	17.62	14.77	13.73
			PSNR _C	32.23	31.18	28.64	26.78
			PSNR _B	32.67	31.73	29.29	27.15
			PSNR _O	33.66	33.23	33.04	32.72
16	302.97	36.28	PSNR _L	22.82	17.71	14.92	13.75
			PSNR _C	33.71	31.65	30.06	27.32
			PSNR _B	34.31	33.46	30.41	28.37
			PSNR _O	35.71	35.37	34.53	33.61

(c)

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Macroblock Loss Rates			
				2%	5%	10%	15%
24	357.60	27.37	PSNR _L	19.69	16.62	13.93	11.93
			PSNR _C	26.04	25.42	24.08	23.08
			PSNR _B	26.66	26.18	25.12	24.41
			PSNR _O	27.18	26.99	26.60	26.19
20	576.13	30.20	PSNR _L	20.00	16.65	14.23	12.75
			PSNR _C	28.55	27.59	25.67	24.50
			PSNR _B	29.45	28.03	26.24	25.68
			PSNR _O	29.72	29.30	28.78	27.92
16	945.11	33.48	PSNR _L	20.35	16.85	14.38	12.85
			PSNR _C	30.52	30.17	26.45	24.89
			PSNR _B	31.65	30.35	27.53	26.25
			PSNR _O	32.21	30.68	29.62	28.80

(d)

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Macroblock Loss Rates			
				2%	5%	10%	15%
24	296.28	28.75	PSNR _L	21.55	18.30	15.19	14.36
			PSNR _C	27.41	25.80	23.66	22.73
			PSNR _B	28.07	27.08	25.76	24.32
			PSNR _O	28.34	27.49	26.37	25.91
20	459.48	31.53	PSNR _L	21.69	18.48	15.55	14.49
			PSNR _C	28.87	27.00	24.93	23.39
			PSNR _B	29.50	28.23	26.23	25.39
			PSNR _O	30.20	29.16	27.26	26.66
16	736.32	34.72	PSNR _L	22.30	18.55	16.08	14.58
			PSNR _C	30.30	27.30	25.28	23.12
			PSNR _B	32.30	29.19	27.14	26.35
			PSNR _O	33.08	30.03	28.02	27.46

Notes:

$PSNR_L$: The PSNR of corrupted video without error concealment.

$PSNR_C$: The PSNR of video recovered by the temporal replacement.

$PSNR_B$: The PSNR of video recovered by the boundary matching algorithm.

$PSNR_O$: The PSNR of video recovered by the proposed method.

We tested the performance of the proposed method in two different scenarios. First we assume that the encoder adopts interleaving technique. In this case, the neighboring macroblocks are packed into different packets. The motion vectors of four neighboring macroblocks are available. Then we can form the polynomial model in horizontal and vertical directions respectively and reconstruct the lost macroblocks.

The macroblock loss rates, which we used to conduct the simulation, are 2%, 5%, 10% and 15%. The simulation results of different macroblock loss rates are presented in Table 6.2. QP is the quantization parameter used in encoder. $PSNR_L$ denotes the PSNR of corrupted video without any error concealment. $PSNR_C$ denotes the PSNR of the video concealed by the temporal replacement. $PSNR_B$ indicates the PSNR of the video reconstructed by the boundary matching algorithm. $PSNR_O$ represents the PSNR of the video that is reconstructed by the proposed algorithm. Compared with the other two error concealment approaches, the proposed algorithm can achieve obviously better concealment results. Figure 6.4 shows the performance of the proposed method for different bitrates. In low bitrate, the PSNR of video recovered by the proposed method is close to the original PSNR.

Figure 6. 5 PSNR of different slice loss rates. (a) Tempete: 5% slices lost. (b) Tempete: 10% slices lost.

In the other scenario, the interleaving technique is not used in encoder. When a packet corrupted, the entire row of macroblocks will be lost. Then we can only use the neighboring motion vectors in vertical direction to formulate the polynomial model and

reconstruct the lost macroblock. In the simulation, each slice contains a fixed number of macroblocks. The simulation results of this scenario are presented in Table 6.3.

Compared with the simulation results that are recovered by neighboring motion vectors in two directions, under the same macroblock loss rate, the simulation results that are reconstructed by motion vectors in one direction is slightly worse. Figure 6.5 shows the performance of the proposed method in this scenario for different bitrates.

Table 6. 3 Simulation results of different slices loss rates. (a) Tempete. (b) Foreman. (c)

Mobile. (d) Stefan.

(a)

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Slices Loss Rates			
				2%	5%	10%	15%
24	217.56	27.64	PSNR _L	23.74	19.96	18.34	17.20
			PSNR _C	26.16	25.50	25.13	24.12
			PSNR _B	26.71	26.18	25.50	24.23
			PSNR _O	26.94	26.89	26.32	26.22
20	483.18	30.42	PSNR _L	24.51	21.27	18.54	17.67
			PSNR _C	29.06	27.69	26.18	24.62
			PSNR _B	29.00	27.99	26.42	25.55
			PSNR _O	29.49	29.14	28.07	27.46
16	817.70	33.70	PSNR _L	24.72	21.98	19.92	17.90
			PSNR _C	31.16	29.58	27.15	26.04
			PSNR _B	31.33	29.97	27.40	26.57
			PSNR _O	32.22	30.94	29.68	28.21

(b)

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Slices Loss Rates			
				2%	5%	10%	15%
24	119.42	31.42	PSNR _L	20.34	17.05	14.26	12.89
			PSNR _C	29.44	28.98	27.64	25.89
			PSNR _B	29.68	29.23	27.89	26.25
			PSNR _O	31.25	30.86	30.59	30.27
20	191.26	33.86	PSNR _L	20.51	17.17	14.39	12.97
			PSNR _C	31.87	30.67	28.11	26.97
			PSNR _B	32.31	31.66	28.90	27.04
			PSNR _O	33.47	33.12	33.01	32.51
16	302.97	36.28	PSNR _L	20.69	17.31	14.66	13.03
			PSNR _C	33.01	30.43	28.97	25.72
			PSNR _B	33.93	33.08	30.13	27.88
			PSNR _O	35.53	35.02	34.42	33.19

(c) Mobile

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Slices Loss Rates			
				2%	5%	10%	15%
24	357.60	27.37	PSNR _L	20.98	16.63	14.28	12.71
			PSNR _C	26.30	25.81	24.54	23.49
			PSNR _B	26.30	25.98	24.94	24.08
			PSNR _O	27.17	26.83	26.46	25.99
20	576.13	30.20	PSNR _L	21.58	16.76	15.52	12.88
			PSNR _C	28.60	27.57	25.43	24.68
			PSNR _B	29.32	28.06	25.98	25.32
			PSNR _O	29.66	29.20	28.57	27.79
16	945.11	33.48	PSNR _L	21.71	16.92	15.69	13.03
			PSNR _C	30.67	29.62	26.47	24.76
			PSNR _B	31.48	29.84	27.24	25.49
			PSNR _O	32.19	30.50	29.35	28.65

(d) Stefan

QP	Bitrate (Kbit/s)	Original PSNR	PSNR (dB)	PSNR of Different Slices Loss Rates			
				2%	5%	10%	15%
24	296.28	28.75	PSNR _L	21.12	17.87	15.43	14.36
			PSNR _C	27.54	25.83	23.96	22.58
			PSNR _B	27.96	26.62	25.44	24.21
			PSNR _O	28.30	27.09	26.17	25.71
20	459.48	31.53	PSNR _L	21.87	18.19	15.86	14.46
			PSNR _C	29.11	27.18	25.13	23.07
			PSNR _B	29.41	28.12	26.16	25.10
			PSNR _O	30.16	29.05	27.02	26.57
16	736.32	34.72	PSNR _L	22.67	18.38	16.44	14.86
			PSNR _C	30.22	27.02	25.07	23.51
			PSNR _B	32.26	29.84	26.87	26.10
			PSNR _O	32.91	30.00	27.57	26.95

Notes:

PSNR_L: The PSNR of corrupted video without error concealment.

PSNR_C: The PSNR of video recovered by the temporal replacement.

PSNR_B: The PSNR of video recovered by the boundary matching algorithm.

PSNR_O: The PSNR of video recovered by the proposed method.

In both scenarios, the proposed method can efficiently improve the quality of corrupted video. Compared with the other two approaches, the performance of proposed method is obviously better. For the low macroblock loss rate or slice loss rate, the proposed method can improve PSNR by 1 dB, compared with the temporal replacement. When the macroblock loss rate or slice loss rate is higher, the proposed method can achieve better PSNR improvement.

6.5 Conclusion

In this chapter, we present an efficient temporal error concealment algorithm that is based on a polynomial model for the new coding standard H.264. In H.264, the neighboring motion vectors have higher correlation than previous coding standards. By making use of the correlations between neighboring motion vectors, we can fit the neighboring motion vectors into a polynomial model, which can create an approximate function to describe the change tendency of motion vectors within a small area. At pre-defined points, this approximate function should minimize the square error between the given motion vectors and the function values. By utilizing the function that is formulated by the motion vectors adjacent to the lost macroblock, we can estimate the motion vectors that cover 4×4 blocks in the lost macroblock. Consequently, the lost macroblock can be reconstructed. The computational cost of the proposed method is low, and it is suitable for real-time video transmission.

Chapter 7

Periodic Macroblock for Error Resilient Video Transmission

7.1 Introduction

To increase the coding efficiency, in most video compression coding standards, motion compensated prediction is used to eliminate the temporal redundancy. This technique increases the dependence of neighboring frames. Each inter-frame depends on its reference frames. When one macroblock is corrupted, the macroblocks in successive frames that refer to the corrupted macroblock will be affected. And the affected area will increase with the error propagation effect. Thus a lost or corrupted macroblock may will cause disaster to the compressed video.

To alleviate the effect of macroblock loss, many error resilience approaches have been presented. Multiple description is a widely used error resilience concept. Wang *et al* presented a multiple description motion compensation method in [123]. In their approach, a second-order predictor is used for motion-compensation, and two descriptions, which contain the coded even and odd frames respectively, are generated by the encoder. Kim and Cho [124] proposed a multiple description of DCT coefficients. They use the redundancy rate-distortion criterion to split a one-layer compressed video stream into two correlated streams or descriptions.

Inserting more resynchronization signals can efficiently stop the error propagation. As shown in [125], variable-length resynchronizing codes can help to avoid the collapse of the video. Forward error correction is another efficient error resilient scheme. Mohr *et al* [40] presented unequal loss protection framework, in which unequal amounts of forward error correction are applied to progressive data to provide graceful degradation of image quality as packet losses increase. Kim *et al* [126] proposed a bit-plane-wise unequal error protection algorithm. This algorithm protects a compressed embedded bitstream generated by the 3-D SPIHT algorithm by assigning an unequal amount of forward error correction. Wang *et al* [127] introduced a novel video transmission architecture, which uses forward error protection and automatic repeat request (ARQ) protocol, to meet the low-delay and error robust requirements of wireless video communications. Frossard *et al* [47] addressed a new scene-complexity adaptive mechanism to determine the best compromise between the rate allocated to encode pure video information and the rate aiming at reducing the sensitivity to packet

loss. Wiegand *et al* [128] presented a framework that incorporates an estimated error into rate-constrained motion estimation and mode decision. Chang and Lee [104] proposed a precise error-tracking scheme, in which the decoder reports the addresses of the corrupted blocks to the encoder through a feedback channel. Then the encoder can precisely calculate and track the propagated error by examining the backward motion dependency. The error-propagation effects can be terminated completely by INTRA refreshing. Lee *et al* [81] proposed an error resilient image coding for DCT-based image compression. Their method can successfully prevent errors from propagating across image block boundaries with little overhead. Rhee and Joshi [129] presented a new error recovery scheme, called RESCU. The main benefit of the RESCU scheme is that it allows more time for transport-level recovery such as retransmission and forward error correction. Gallant and Kossentini [130] proposed a rate-distortion optimized mode-selection algorithm to increase the error resilience of low bitrate video communications.

Figure 7. 1 Multiple reference frames used in H.264.

In this chapter, we present a new error resilient scheme for the new coding standard H.264. To achieve better coding efficiency, this coding standard supports several new coding schemes. In H.264, not only the inter-block has prediction, but also

the intra-block is predicted from neighboring blocks. A 16x16 macroblock in H.264 can be divided into different block shapes for motion compensation. The smallest block size in H.264 is 4x4, and an integer 4x4 transform is used in encoding. Multiple reference frame scheme [20] [118] is adopted during the search process of motion vector. More than one encoded frame can be used as reference frame for the encoding frame. The blocks in the same macroblock can select the best matching encoded frames as reference frame to obtain minimum residual, as shown in Figure 7.1. The syntax of every block will indicate which encoded frame is selected as its reference frame. By making use of the reference frame buffer in the encoder, we can make some macroblocks in every n^{th} inter-frame refer to the frame that is n frames away, and these macroblocks are named as periodic macroblocks. The periodic macroblock can efficiently alleviate the error propagation between the frames that contain periodic macroblocks. The number of periodic macroblock in every n^{th} frame can be adjusted according to the available transmission bandwidth, as the periodic macroblock will consume more bits. The selection of periodic macroblock location is based on the distortion expectation that incorporates the distortions due to both transmission error and quantization distortion. The simulation results prove that the video quality can be significantly improved by the proposed scheme.

7.2 Multiple Reference Frames

Multiple frame motion compensated prediction, which also can be denoted as long-term memory motion-compensated prediction, is a unique property of H.264. The use of multiple frames for motion compensation in most cases provides significant prediction gain.

A repeating scene change is a common feature found in many video applications. Such a change in the video scene will cause the scene cut frame to be coded in the intra mode, which costs more bits than inter mode. By extending the search range of motion vectors, the encoding macroblock can select a frame that is not close to be the reference frame. Then it is more probable for the encoding macroblock to find a matching area in the reference frame, and the coding efficiency can be improved [131].

In Figure 7.2, an example of the multiple reference frame coder has been shown.

Figure 7. 2 Multiple reference frame motion compensated predictor.

In H.263, which does not support multiple reference frames, the variable time delay, which is caused by the using of more frames than the previously decoded one for motion compensated prediction, has to be transmitted as the side information. In

order to control the bit rate of the motion information, rate-constrained motion estimation is exploited to decide the reference frame [24] [132].

Another advantage of using multiple reference frames in motion compensation is that it has an inherent ability of overcoming some transmission errors and is thus more robust than single frame block match coding [133].

Figure 7.3 Single reference frame Markov chain.

Figure 7.4 Two reference frames Markov chain.

Let us assume that q_0 is the probability that a macroblock is encoded as intra macroblock. $q_1, \dots, q_N$ are the probabilities that this macroblock is encoded as inter macroblock and the previous $1, \dots, N^{th}$ frames is used in the prediction. p_{error} is the

macroblock error probability. The reconstruction state of a macroblock in decoder is denoted by S_t , with $S_t=0$ denoting error-free reconstruction and $S_t=1$ denoting erroneous reconstruction. Then we can use the Markov chain to present the reconstruction state of a macroblock, and calculate the probability of each scenario. For example, when single frame is utilized in motion prediction, if the reference frame is correctly reconstructed, the probability that a macroblock can be decoded without error is $1 - p_{error}$. If the reference frame has error, the probability of error-free reconstruction is $q_0(1 - p_{error})$. That means that the macroblock can be reconstructed correctly only when it is intra coded. Figure 7.3 shows the Markov chain for single frame motion compensated prediction, and Figure 7.4 presents the Markov chain for two frame motion compensated prediction. Comparing the two figures, it is obviously that the using of multiple reference frames can increase the correct receipt probability of a macroblock.

7.3 Content-Based Periodic Macroblock

In this section, we present a new error resilient coding method that is based on multiple reference frames for H.264. As we mentioned before, H.264 encoder can support multiple reference frames. By making use of this property, we can make some macroblocks, namely periodic macroblocks, to reference the frame that is n frames away. These periodic macroblocks only use the previous n^{th} frame as the reference frame. A periodic macroblock is not necessary to reference another periodic

macroblock, as shown in Figure 7.5. In the reference code of H.264, JM6.0, up to five previously encoded frames can be used as reference frame for the currently encoding frame. As long as the interval between two frames that contain periodic macroblocks does not exceed five, no additional reference frame buffer will be needed. The error propagation effect along inter-frames can be alleviated or partially terminated by the periodic macroblocks.

Figure 7.5 Periodic Macroblock.

As we described before, since an inter-block depends on its reference frame, it may be affected by the error in its reference frame. And the blocks that reference this inter-block may also be affected. Hence, when we calculate the pixel loss probability in an inter-frame, the loss probabilities of relative previous inter-frames should be considered. We can prove in theory that encoding some periodic macroblocks can reduce the loss probability of pixel. For simplicity, we assume that a frame is transmitted in one packet, and each packet is lost with probability p . The receipt probability of each packet is q , $q=1-p$. The pixels in the same frame have the same loss probability p . Further, we can assume that only the immediately previous frame is used as reference frame in the encoder. The first frame is intra-frame and is correctly received by the decoder. For a bitstream that is without periodic macroblocks, the

receipt probability of each pixel in the s^{th} frame is equal to q^s , as shown in Figure 7.6.

Figure 7. 6 The receipt probability of macroblock for single reference frame.

When some macroblocks in every n^{th} frame are encoded as periodic macroblocks, the pixel receipt probability may vary in frames that are posterior to a frame containing periodic macroblock. Since the receipt probability of referenced area should be considered, the receipt probabilities of pixels in periodic macroblocks and non-periodic macroblocks are different. When a pixel references the periodic area, its receipt probability will increase. To make the situation easy to understand, we discuss how to calculate the receipt probability of one pixel first. We assume that the immediately previous frame is used as reference frame for non-periodic macroblock. The receipt probabilities of different situations are presented in Figure 7.7. u is the frame index in bitstream, n is the interval between two frames that contain periodic macroblocks. k equals to u/n , which indicates the index of frame that contains periodic macroblocks. The items without grey background present the receipt probabilities of pixels within periodic macroblock in every n^{th} frames, and the others indicate the receipt probabilities of pixels within non-periodic macroblock. For the n^{th} frame ($k=1$), some macroblocks will be encoded as periodic macroblocks. If a macroblock is encoded as a periodic macroblock, the pixel receipt probability within this macroblock is q ,

otherwise the receipt probability is q^n . For the pixel in the $n+1^{\text{th}}$ frame, the receipt probability will be q^2 , if the pixel references the periodic area in the n^{th} frame. Otherwise, its receipt probability will be q^{n+1} .

Figure 7.7 The receipt probability of pixel in different scenarios.

For the second frame that contains periodic macroblocks ($k=2$), the receipt probability of a pixel has four different situations. In the first case, the pixel belongs to a periodic macroblock, and it references the periodic area in the n^{th} frame. The receipt probability of this pixel is q^2 . In the second case, the pixel is within the non-periodic macroblock, but it references the periodic area in the n^{th} frame. The receipt probability of this situation can be represented as $q \times q^n$. In the third case, the macroblock that contains this pixel is encoded as periodic macroblock, but it references the non-periodic area in the n^{th} frame. The receipt probability of the third situation is the same as the second situation. In the last case, the pixel is within a non-periodic macroblock, and there is no periodic area involving its relative references. This situation is the same as the general encoding situation that no periodic macroblock

exists. The receipt probability of this case is q^{2n} , which is smaller than other cases. Through the statistical derivation, we can prove that the periodic macroblock can increase the receipt probability of pixel.

For a QCIF video, each frame contains 99 macroblocks. If there are T macroblocks encoded as periodic macroblocks in every n^{th} frame and the locations of periodic macroblocks are selected randomly, the probability that a pixel is within a periodic macroblock in every n^{th} frame will be $\frac{T}{99}$. The receipt probability of a pixel in every n^{th} frame can be calculated by Equation (7.1). Compared with the pixel receipt probability of ordinary bitstream without periodic macroblock, the periodic macroblock can make the bitstream obtain larger pixel receipt probability P_k . For example, if the pixel loss probability is 10% in every frame and there are 30 periodic macroblocks in every 5th frame ($n=5$). In the 20th frame ($k=4$), the receipt probability of each pixel in a bitstream without periodic macroblock is equal to 12.16% (0.9^{20}), and the receipt probability in a bitstream with periodic macroblock is 21.8%.

$$P_k = \sum_{l=0}^k \binom{k}{l} \left(\frac{T}{99} \times q \right)^l \left(1 - \frac{T}{99} \right)^{(k-l)} q^{(k-l)n} \quad (7.1)$$

It is obvious that selecting the location of periodic macroblock randomly is not a wise choice. To achieve better error resilient performance, we use the distortion expectation of macroblock as the selection criterion of periodic macroblock. The distortion of each macroblock in the decoder can be represented in two parts. One part of the distortion is caused by quantization, as described in Equation (7.2). f indicates the original pixel value in a macroblock M , $\tilde{f}$ is the reconstructed pixel in the encoder, and u is the frame index.

$$D_q = \sum_{x,y \in M} (f_u[x,y] - \tilde{f}_u[x,y])^2 \quad (7.2)$$

Another part of distortion is caused by the transmission error. In the decoder of JM6.0, when a macroblock is lost or corrupted, the decoder replaces this macroblock by co-located macroblock in immediately previous frame. The distortion due to transmission error can be written as Equation (7.3). $\hat{f}$ denotes the decoded pixel, which may be different with $\tilde{f}$.

$$D_t = \sum_{x,y \in M} (\tilde{f}_u[x,y] - \hat{f}_{u-1}[x,y])^2 \quad (7.3)$$

Since we cannot know the pixel value in decoder, we use the reconstructed pixel in encoder to replace the decoded pixel, and calculate the distortion.

$$D_t = \sum_{x,y \in M} (\tilde{f}_u[x,y] - \tilde{f}_{u-1}[x,y])^2 \quad (7.4)$$

For the encoder, the distortion of each macroblock is a variable, since the encoder cannot make sure whether the macroblock can be received correctly or not. As such, we use the expectation of distortion as the selection criterion of periodic macroblock. The expectation of distortion can be calculated by Equation (7.5). p is the loss probability, it can be easily obtained through the feedback channel that is supported by most transmission protocols. The macroblocks that have maximum distortions will be selected as periodic macroblocks.

$$D_{\text{exp}} = (1-p)D_q + pD_t \quad (7.5)$$

The main purpose of error resilient scheme is to use little overhead to achieve good error resilient performance. The criterion we used to select the periodic macroblock can meet the requirement. Large quantization distortion indicates that the macroblock is not best match with any reference frame. When we encode this macroblock as

periodic macroblock, compared with other macroblocks that have small quantization distortions, fewer additional bits will be introduced. If the macroblock that has large distortion is lost due to transmission error, the video quality will obviously be degraded. Since the periodic macroblock will have more protection during the transmission, selecting the macroblocks having large distortion due to transmission error as periodic macroblocks will enhance the robustness. The distortion expectation can incorporate both of quantization distortion and distortion due to transmission error. The computation of distortion expectation is not complicated, and it can be implemented in real-time video transmission.

Another problem should be considered is the number of periodic macroblocks encoded in every n^{th} frame. For most frames, the immediately previous frame is the best reference frame. For instance, multiple reference frames is supported in H.264, but more than 90% blocks will select the immediately previous frame as reference frame. Because the periodic macroblock uses the frame that is n frames away as reference frame, it may cost more bits than ordinary inter-macroblocks. We propose to adjust the number of periodic macroblocks in bitstream according to the available transmission bandwidth. The more bandwidth is available, the more periodic macroblocks can be encoded to achieve better error resilient performance.

7.4 Simulation Results

We use QCIF video sequences, Foreman, Stefan and Mobile, to conduct our simulation. The videos are encoded and decoded by JM6.0, which is a standard codec of H.264. The frame structure in our simulation is IPPPP....., there are 49 P frames between two I frames. By changing the quantization factor, the sequences can be encoded into different bitrates. To test the error resilient performance of the proposed method, different macroblock loss rates are used in experiment. For each macroblock loss rate, we tested the error resilient performance of the proposed method in 20 different loss scenarios, and use the average PSNR values of the 20 loss situations as the final simulation results. H.264 supports flexible slice size and flexible macroblock ordering to enhance the robustness to data loss. So in our experiment we select the location of lost macroblock based on these properties. The error concealment method used in JM6.0 is temporal replacement in which the lost macroblock will be replaced by co-located macroblock in the immediately previous frame.

In JM6.0, the reference frame buffer stores 5 encoded frames as the reference frame for motion estimation. To avoid adding additional reference frame buffer, we select five frames as the interval distance. In other words, the periodic macroblocks in our simulation will use the fifth previous frame as reference frame.

We compare the error resilient performance of the bitstreams that contain different numbers of periodic macroblocks. In the simulation, some macroblocks are encoded as periodic macroblocks in every 5th frame.

Figure 7. 8 Comparison of PSNR. (a) Foreman: 5% macroblocks lost. (b) Foreman: 10% macroblocks lost. (c) Foreman: 15% macroblocks lost.

In Figure 7.8 and Table 7.1, we compare the video quality of different bitstreams. The simulation results prove that the video quality can be improved significantly by the proposed method. As shown in Figure 7.8, for the video sequence Foreman, when 10 periodic macroblocks in every 5th frame, the PSNR of video can be improved by up to 3.55 dB with different macroblock loss rates. As the number of periodic macroblocks increases, better PSNR improvement can be achieved. When there are 30 periodic macroblocks contained in every 5th frame, the PSNR can be improved more than 4 dB. The simulation results of Mobile and Stefan are presented in Table 7.1. The QP in Table 7.1 denotes the Quantization Parameter used in encoder. For these videos, the video quality also can be obviously improved by the proposed method. The proposed method can be easily implemented in practical situation on the current transmission

channel.

Table 7. 1 The PSNR of different macroblock loss rates.

(a)

Video Sequence: Mobile					
QP	Loss Rate	Without Protection	The Number of Periodic Macroblocks		
			10	20	30
32	5%	25.80	26.65	26.87	27.21
	10%	23.44	24.41	24.83	25.29
	15%	21.79	22.90	23.50	23.80
30	5%	26.43	27.24	27.56	27.89
	10%	23.62	24.68	25.17	25.47
	15%	21.85	23.11	23.77	24.08
28	5%	26.78	27.55	28.03	28.52
	10%	24.03	25.12	25.61	26.17
	15%	22.21	23.34	23.98	24.44

(b)

Video Sequence: Stefan					
QP	Loss Rate	Without Protection	The Number of Periodic Macroblocks		
			10	20	30
32	5%	22.59	23.92	24.60	25.25
	10%	20.21	21.51	22.46	23.83
	15%	19.00	20.39	21.06	22.05
30	5%	22.84	24.15	25.22	26.20
	10%	20.47	22.02	22.90	24.15
	15%	19.12	20.75	21.41	22.33
28	5%	23.16	24.85	25.49	26.62
	10%	20.78	22.19	23.03	24.43
	15%	19.49	21.18	22.04	22.72

In Table 7.2, we compare the proposed method with the intra refresh scheme. In H.264, in order to stop the error propagation in inter frame, some macroblocks in inter frame can be encoded as intra macroblocks. To compare the error resilience of periodic macroblock and intra macroblock, we encode all periodic macroblocks as intra macroblocks, and test the performance. The QCIF video sequence used in comparison is Foreman, and the Quantization Paramter is 30. The error resilience of intra

macroblock is slightly better than periodic macroblock, but the bitrate is increased significantly by the intra macroblock.

Table 7. 2 Comparison between periodic macroblock and intra macroblock

Methods	Number of Macroblock	Bitrate (Kbits/s)	PSNRs of Different Macroblock loss Rates		
			5%	10%	15%
Periodic Macroblock	10	87.14	28.47	28.15	26.24
	20	93.29	29.77	28.53	26.52
	30	95.83	30.32	29.08	27.08
Intra Macroblock	10	91.30	28.51	28.26	26.38
	20	101.53	29.87	28.61	26.74
	30	110.69	30.35	29.14	27.23

7.5 Conclusion

In this chapter, we present a content-based error resilient coding scheme for H.264.

The periodic macroblock can increase the receipt probability of pixel by alleviating the error propagation effect. Since multiple reference frames is supported in H.264, the proposed method will not introduce additional reference frame buffer to the encoder.

The distortion expectation of macroblock is used as the criterion to select periodic macroblocks. We would like to stress that the proposed method aims at increasing the robustness of the video transmission over the communication network, however we still compare the bitrate requirement for different numbers of periodic macroblocks.

According to the simulation results, the video quality can be significantly improved by up to 3.55 dB with only increase the bitrate by 5.51% in error prone channel.

Chapter 8

A Drift-Reduction Multiple Description Coder Using Multiple Reference Frames

8.1 Introduction

Multiple description coding is a widely used error resilient video coding scheme. In the review chapter, we have introduced some multiple description coding methods. By using these methods, a video sequence can be encoded into two or more bitstreams, each of them contains some information of the compressed video. Generally, the important information, such as motion vector, is duplicated in all bitstreams. And the less important information may be encoded in only one bitstream. These bitstreams are referred to as descriptions, and transmitted over different channels. The descriptions can be generated independently or correlatively. When all descriptions are correctly

received, the decoder can reconstruct the video with best quality. If any description is lost during transmission, the decoder can still reconstruct the video but with a lower, acceptable quality.

The main problem of multiple description coding is the mismatch of reference frames between the encoder and the decoder. For most multiple description coders, motion prediction is used to generate descriptions. The encoder of multiple description coding generally uses the frame that is reconstructed by all descriptions as reference frame in motion prediction. When any description is lost during transmission, the reference frame used in the decoder will be different from that is used in the encoder. Thus the mismatch, also referred to as drift, will appear in the reference frame and propagate into following frames. To reduce the drift, the differences of reference frames between the encoder and the decoder can be encoded as side information and transmitted along with descriptions. Reibman *et al* [134] presented a three-loop motion compensation encoder to mimic the three possible scenarios at the decoder: both descriptions received or either of the descriptions received. Tang and Zakhor [135] developed a multiple description video coding scheme that modifies the discrete cosine transform structure to the matching pursuits framework. Lee *et al* [136] proposed an enhanced two-stage multiple description codec that is based on Reibman's three-prediction-loop structure to alleviate the drift. The details of these methods will be introduced in next section.

Most of the presented methods used to generate side information are focused on the mismatch of reference frames between the encoder and the decoder. The differences of

frames that are reconstructed by all descriptions or by some descriptions are encoded as side information.

In this chapter, we propose a new method, which adopts multiple reference frames, to generate the side information. For each encoding frame, two previously encoded frames are used as reference frames in motion prediction. And the prediction results of the two reference frames will be encoded as descriptions and side information respectively. When transmission error occurs in either reference frames, the decoder can use the correctly received reference frame for decoding. The simulation results show that this method can improve video quality obviously along different macroblock loss rates.

8.2 Side Information

In this section, two methods designed for side information generation are introduced.

Reibman *et al* [134] designed a two-description codec with three separate prediction paths in the encoder to mimic three possible scenarios at the decoder: either description is lost or both descriptions are received. The structure of the three-loop encoder is shown in Figure 8.1. The prediction error F_0 , ($F_0 = X - P_0$), which assumes both descriptions are available, is generated by the central prediction loop, where X is the original video data and P_0 is the prediction frame reconstructed by all descriptions. F_0 is encoded into two descriptions, $\tilde{F}_1$ and $\tilde{F}_2$. Many different multiple description coding methods can be used in the central prediction loop to code $\tilde{F}_1$ and

$\tilde{F}_2$. In [134], multiple description transform coding (MDTC) is adopted. If descriptions, $\tilde{F}_1$ and $\tilde{F}_2$, are available to the decoder, $\hat{F}_0$ can be recovered from both descriptions. If the decoder receives only one description i , the prediction error $\hat{F}_i$ can be generated, and the reconstructed frame will be $P_i + \hat{F}_i$. In order to reduce the future mismatch between the prediction at the encoder and the decoder, the encoder also generates and codes the side prediction error $G_i = X - P_i - \hat{F}_i$, and transmits it along with the descriptions.

Figure 8. 1 The structure of the three-prediction-loop multiple description coder.

When both descriptions are received, the bits used for G_i are primarily redundancy. The bits used to encode the side information are wasted. In order to make the utility of side information more efficiently, Lee *et al* [136] proposed an enhanced two-stage multiple description video coder, which is based on Reibman's three-prediction-loop structure. In their design, an additional multiple description

scalar quantizer (MDSQ) is introduced into the central prediction loop. MDSQ is a scalar quantizer designed for the operation in a diversity system [140]. The encoder of a MDSQ sends information over each channel of the diversity system. The decoder reconstructs the video based on the information received from the channels that are currently working. The objective is to design a codec that minimizes the average distortion when both channels work. The structure of the enhanced two-stage multiple description codec is presented in Figure 8.2.

Figure 8. 2 The enhanced two-stage multiple description codec that is based on three-prediction-loop structure.

In Figure 8.2, r_0 is quantized using MDSQ-II to produce two outputs, $\tilde{r}_s$ and $\tilde{r}_0$. $\tilde{r}_0$ is coded into two descriptions, r_1 and r_2 . And $\tilde{r}_s$ is passed to the side prediction loops to generate the side information. If the quantizer in the side information loop is the same as the central prediction loop, $\tilde{r}_s$ will be identical to $\tilde{r}_0$. In this situation, $\tilde{r}_s$ cannot be utilized to improve the reconstruction quality of the residual signal.

If $\tilde{r}_s$ and $\tilde{r}_0$ are quantized by different coarse quantizers, a more finely

quantized residual signal can be obtained by combining them. Therefore, an additional MDSQ in the central prediction loop is utilized, before r_0 is coded into two descriptions. This forms a two-stage multiple description coding. When no error occurs, the decoder will receive $\tilde{r}_s$ and $\tilde{r}_0$. By combining $\tilde{r}_s$ and $\tilde{r}_0$, the reconstruction level of r_0 can be refined, and the frame can be reconstructed with a better quality.

8.3 A Multiple Description Coding Method Using Multiple Reference Frames

In this section, we present a new generation scheme of side information, which is based on the concept of multiple reference frames, for multiple description coding. There are two subsections. In the first subsection, we introduce the method to generate the side information based on multiple reference frames. In the second subsection, the macroblock selection criterion for side information is presented.

8.3.1 Side Information Based on Multiple Reference Frames

In Figure 8.3, a typical multiple description codec is shown. The encoding procedure of multiple description codec is similar to the general video encoding procedure, which includes motion compensated prediction, DCT and so on. The resulting symbol will be divided into various descriptions by a multiple descriptions coding method, which can

be one of the methods mentioned in Chapter 2. In the decoder side, if a description of the decoding frame is lost, the decoder can reconstruct the decoding frame with a degraded video quality. Unfortunately the reconstructions of the following frames may be affected, since the degraded frame may be used as reference frame in the decoding of the successive frames. To alleviate the drift effect of reference frame, we propose a two-prediction-loop multiple description coding structure, as shown in Figure 8.4. For an encoding frame F_n , it will be encoded by two separate loops, one loop is referred to as central prediction loop, and the other is denoted as side prediction loop. And each loop contains motion compensation, DCT and quantization. But the reference frames used in these two loops are different. The immediately previous frame F_{n-1} is used in central prediction loop, and the next previous frame F_{n-2} is used in side prediction loop. The framework of central prediction loop is a general structure of multiple description coding, as presented in Figure 8.3. The descriptions $D_1 \cdots D_n$ are generated by the central prediction loop.

Figure 8. 3 A typical multiple description coder.

The side prediction loop is used to encode the side information, which can contribute to the reduction of drift. The encoding procedure of side prediction loop is slightly different from the central prediction loop. There is no motion prediction step in side prediction loop. Each macroblock in F_n is compensated directly by the same location macroblock in F_{n-2} . The residuals are encoded as side information, and transmitted along with descriptions.

Figure 8. 4 The proposed coder structure for multiple description coding.

We do not include motion prediction in side prediction loop, because the side information is redundancy when all descriptions $D_1 \cdots D_n$ are correctly received. If both the motion vector and the compensated residual are encoded as side information, more bits will be used by side information. When the bandwidth of channel is limited, more bits are spent on side information means that fewer bits are spent on descriptions. The video quality may be impaired.

Figure 8. 5 The decoder of multiple description coding.

If any description of the reference frame F_{n-1} is lost during transmission, the decoder can use another reference frame F_{n-2} , which is used in side prediction loop, to reconstruct the decoding frame, as shown in Figure 8.5. The drift effect can be obviously alleviated by the structure with two reference frames. When we calculate the loss probability of each frame, we should consider the error propagation effect caused

by motion compensated prediction, which is utilized in most video coding standards. This technique can improve the compression efficiency by removing the temporal redundancy. By using this method, only the differences between neighboring frames are encoded. In the decoder side, the decoding frame is reconstructed by a previously decoded frame and the differences between these two frames. If one frame is corrupted during transmission, the error may affect the following frames. Since the multiple description coding usually adopts motion prediction and motion compensation in its central prediction loop, the drift in one frame will propagate to the following frames. A frame can be correctly reconstructed when both itself and its reference frame are properly received. We can assume that all transmission channels have the same packet loss probabilities p , and a frame is packed as one packet in each description. When the encoder only uses one reference frame in encoding, the probability of correctly receiving a pixel in the n^{th} frame can be calculated as Equation (8.1), where n is the frame index, and the index of first I frame is 1. $(1 - p)^{n-1}$ indicates the probability of correct receipt of all previous frames. Since the error can propagate from one frame to another frame, the correct receipt probabilities of pixels in all previous frames should be considered in the calculation. And $(1 - p)$ is the correct receipt probability of a pixel in the currently decoded frame.

$$P_n = (1 - p)^{n-1}(1 - p) = (1 - p)^n \quad (8.1)$$

If there are two different reference frames used during encoding, the probability of drift propagation can be efficiently reduced. For example, the correct receipt probability of pixel in the third frame can be computed as Equation (8.2).

$$\begin{aligned}\hat{P}_3 &= (1-p)^2(1-p) + (1-p)(1-p) - (1-p)^2(1-p)(1-p) \\ &= (1-p)^3 + (1-p)^2 - (1-p)^4\end{aligned}\quad (8.2)$$

When all macroblocks in the third frame are predicted by two reference frames and encoded as side information, the correct receipt probability of each pixel in the third frame can be improved by $(1-p)^2 - (1-p)^4$. As shown in Equation (8.3), P_{imp}^n is the improvement of the correct receipt probability. When the loss probability increases, P_{imp}^n also increases.

$$P_{imp}^3 = \hat{P}_3 - P_3 = (1-p)^2 - (1-p)^4 \quad (8.3)$$

For the n^{th} frame with $n > 3$, when all pixels are predicted by two reference frames, the correct receipt probability of pixel can be calculated by Equation (8.4).

$$\hat{P}_n = (\hat{P}_{n-1} + \hat{P}_{n-2})(1-p) - \hat{P}_{n-1}\hat{P}_{n-2}(1-p) \quad (8.4)$$

And the improvement of the correct receipt probability can be presented by Equation (8.5).

$$P_{imp}^n = \hat{P}_n - P_n \quad (8.5)$$

8.3.2 Selection of Macroblocks Contained in Side Information

In this subsection, we present the selection criterion of macroblocks that are contained in side information. If we encode all macroblocks of a frame in side information, the bitrate will be increased tremendously. Since the side information is absolute redundancy when all descriptions are correctly received, it is not a good idea to spend too many bits on side information. We propose that the side information only contains

partial macroblocks in each frame rather than all macroblocks to save the bits of side information, and the number of macroblocks that are contained in side information should be related to the loss probability of transmission channel.

When only partial macroblocks of every frame are encoded in side information, the percentage improvement of correct receipt probability $\tilde{P}_{imp}^n$ will be less than P_{imp}^n . For example, when there are M macroblocks predicted by two reference frames, for QCIF frame, $\tilde{P}_{imp}^n$ is equal to $\frac{M}{99} P_{imp}^n$. In other words, the more macroblocks encoded in side information, the better error resilient performance can be achieved, but more bits are cost by side information. We should find a tradeoff between the bit cost and error resilient performance of video. Hence, the number of macroblocks that are encoded in side information should be adjusted according to the channel condition and the requirement of decoded video quality. When the loss probability increases or the requirement of video quality rises, the number of macroblocks encoded in side information should be increased, and higher correct receipt probability can be achieved.

We define a proportion factor α that is used to determine the number of macroblocks in the n^{th} frame to be encoded in side information. For QCIF video, α is equal to $M / 99$, where M represents the number of macroblocks that are contained in side information. When the proportion factor α equals to 1, all macroblocks in the n^{th} frame will be encoded in side information. As we mentioned before, the proportion factor α is related to the loss probability of the transmission channel and the

improvement of the correct receipt probability. When we have a $\tilde{P}_{imp}^n$ value, α can be calculated as Equation (8.6).

$$\alpha = \frac{\tilde{P}_{imp}^n}{P_{imp}^n}, \quad \text{where } 0 \leq \tilde{P}_{imp}^n \leq P_{imp}^n \quad (8.6)$$

According to the definition of $\tilde{P}_{imp}^n$, the value of $\tilde{P}_{imp}^n$ should be adjusted according to the index of the encoding frame. That means each frame should be assigned a particular $\tilde{P}_{imp}^n$. For video encoding, it is impossible to assign $\tilde{P}_{imp}^n$ to each frame. On the other hand, the calculation of correct receipt probability will become very complicated, when some macroblocks are predicted by two reference frames and the rest macroblocks are predicted by single reference frame. In order to make the calculation of α practical, we provide a simplified equation to calculate α , and the result is represented as $\hat{\alpha}$.

$$\hat{\alpha} = \frac{wp}{(1-p)^2 - (1-p)^4} \quad (8.7)$$

In Equation (8.7), w is a weight factor that represents the expected improvement of the correct receipt probability. Therefore w has the same definition with $\tilde{P}_{imp}^n$. The better the correct receipt probability is needed, the larger w should be used in encoder. But unlike $\tilde{P}_{imp}^n$, w has a fixed value for all frames.

When the loss probability increases, larger P_{imp}^n can be achieved by using two reference frames. Larger $\tilde{P}_{imp}^n$ can be assigned to the encoding frame, and more macroblocks will be predicted by two reference frames and encoded as side information to maintain the video quality. But w is fixed, and it does not change

according to the loss probability. To solve this problem, we multiply the weight factor w with the loss probability p .

P_{imp}^n in Equation (8.6) is replaced by P_{imp}^3 , which is equal to $(1-p)^2 - (1-p)^4$.

The calculation of P_{imp}^3 represents a most simple situation, in which only the error propagation effects of two reference frames are considered. According to Equation (8.7), when the packet loss probability becomes larger, more macroblocks will be encoded in side information. If the requirement of video quality increases, a larger value should be assigned to w accordingly.

The location selection of macroblock is based on the macroblock difference between the currently encoding frame and the immediately previous frame. The macroblocks that have largest difference with their co-located macroblocks in the immediately previous frame are encoded in side information, since the corruptions of these macroblocks are difficult to be concealed by post-processing. This method can efficiently alleviate the drift propagation of multiple description coding. The simulation results prove that the video quality under different packet loss rates can be obviously improved by this method.

8.4 Simulation Results

We select the QCIF video sequence, Mobile, to conduct the experiment. In this experiment, we use JM6.0, which is a standard codec of H.264, to encode the video sequences, as the video coding standard H.264 adopts multiple reference frames in

codec. Since the proposed method is used to create side information, we use a simple multiple description coding method, spatial interleaving, to implement the simulation. Two different descriptions are generated by the encoder. Macroblocks are divided into different descriptions according to their locations, as shown in Figure 8.6. Each small square represents a macroblock. For the odd index frames, the information of macroblocks in odd lines is assigned to D_1 and S_1 , and the information of macroblocks in even lines is involved in D_2 and S_2 . For frames with even index, the locations of macroblocks contained in each description and side information are reversed as odd frames. If either of the descriptions is lost, the missing macroblocks will be replaced by the same location macroblocks in previous frame.

Figure 8. 6 Multiple description coding method used in simulation

We tested the error resilient performance of the proposed method under different packet loss rates. The simulation results are presented in Figure 8.7. The weight factor w is set to 0.4 and 0.6 respectively. The proportion factor is calculated by Equation (8.7). For the macroblocks encoded in side information, the reference frame used in side prediction loop can be used in reconstruction, when either of the descriptions of the reference frame used in central prediction loop is lost.

According to the simulation results, the side information that is generated by the proposed method increases the bitrate slightly. When w is equal to 0.6 and the macroblock loss rate is 15%, the proportion factor α will be 0.449, and 45 macroblocks in a QCIF frame will be encoded in side information. The bitrate will be increased about 4% to 5%, compared to the bitstream without side information. For smaller w or lower macroblock loss rate, the increased bitrate will be less than 2%.

Figure 8. 7 The PSNR of different situations. (a) Mobile: 5% macroblocks lost. (b) Mobile: 10% macroblocks lost. (c) Mobile: 15% macroblocks lost.

As the loss probability increases, the side information will contain more macroblocks. Compared with the bitstream without side information, the proposed method can improve the video quality efficiently among different packet loss rates. And increasing the weight factor can obtain better error resilient performance. As shown in Figure 8.7, when w changes from 0.4 to 0.6, the average PSNR can be improved by up to 1 dB over a wide range of bit rates for 15% macroblock loss rate.

8.5 Conclusion

To prevent the drift problem in multiple description coding, we propose a new side information generation scheme that is based on a two-prediction-loop with the concept of multiple reference frames to generate the side information, which is transmitted

along with each description. Besides the reference frame used in central prediction loop, another previously encoded frame will be used as reference frame to encode some macroblocks in the currently encoding frame, and the prediction error will be encoded as side information. Since multiple reference frames are used in the encoder, when either of the reference frames is corrupted during transmission, the macroblocks involved in side information can be reconstructed by the correctly received reference frame. The simulation results show that the proposed method can obviously improve the error resilient performance of video transmission.

Chapter 9

Conclusion and Future Work

9.1 Conclusion

This chapter summarizes the main developments and results of the thesis, and indicates possible directions for the further research as well.

This thesis is focused on the error control of video compressed by H.264, and several error control schemes, including error concealments and error resilience methods, are presented.

- **The fundamental of video compression is introduced.**

In the first chapter of this thesis, an introduction about the basis concept of video compression is presented. Before compression, a natural visual scene is sampled spatially and temporally, and represented by a set of data. Then the original continuous video becomes digital video. In order to reduce the volume of bits that are used to

represent the digital video, the temporal and spatial redundancies of digital video are removed. This procedure is referred to as compression. Many international compression standards have been developed. Those popular compression techniques are introduced in this chapter. Since this thesis is focused on the video compressed by H.264, the unique compression schemes used in this new compression standard are presented.

- **Review of the error control schemes is presented.**

In the compressed video, only the information of differences between neighboring frames and pixels remains. That makes the compressed video very sensitive to error. A single bit error can cause obvious deterioration of video quality.

Many error control schemes are proposed to alleviate the effect of error. In the second chapter, we review some efficient error control methods. The pre-processing methods, which are referred to as error resilience methods, are implemented in the encoder. They reduce the error effect by introducing additional redundancy to the encoded video. The post-processing techniques, also called as error concealments, are based on the correlation of neighboring frames and pixels. The corrupted or lost part of the video can be recovered by the area that is spatially or temporally neighboring.

- **Maximally smooth error concealment is presented.**

For natural video, the values of neighboring pixels are closed. And the neighboring blocks should be connected smoothly with each other. Based on this property, we can

assume that the lost block connects smoothly with its neighboring blocks. And a constraint function that describes the connection of lost block and its neighboring blocks is formulated. By using the transform matrix, we can convert this constraint function from pixel domain to frequency domain. The coefficients of the lost block can be recovered by the constraint function. Since the smoothness connection constraint is only imposed on the boundary of the lost block, the recovered inner pixels may not be accuracy. A linear interpolation function is utilized to refine the recovered inner pixels. The simulation results show that the quality of the corrupted video can be efficiently improved by this maximally smooth error concealment method.

- **Three motion vector recovery algorithms are developed.**

In Chapter 4 to Chapter 6, we introduce three different motion vector recovery algorithms. As the neighboring blocks within a small area likely belong to the same object, they often move in the same direction. Therefore, the motion vectors of neighboring blocks are correlated.

By making use of the motion vectors that are neighboring to the lost macroblocks, we can formulate three different models to describe the motion tendency within a small area. And the motion vectors of the lost block can be estimated by these models.

- **Reducing the error propagation by periodic macroblock.**

In most video compression standards, motion compensated prediction is used to remove the temporal redundancy. This method can efficiently improve the coding

efficiency. On the other hand, it also increases the dependences between neighboring frames, and makes the error in one frame propagate to the next frames.

In order to reduce the probability of error propagation, some macroblocks in every n^{th} frame use the frame that is n frames away as the reference frame in encoding, and these macroblocks are referred to as periodic macroblocks. We have proved in theory that the probability of error propagation can be reduced by the periodic macroblocks. We implement the periodic macroblock by H.264 codec, since this standard supports multiple reference frames, no additional reference frame buffer is introduced to encoder.

- **Drift reduction multiple description coding.**

Multiple description is a widely used error resilience method. The main problem of multiple description is the mismatch of reference frames used in the encoder and the decoder. The frame reconstructed by all descriptions is used as reference frame in the encoder. When error occurs, the reference frame used in the decoder will be different from the encoder. To reduce this mismatch, we construct a two-prediction-loop multiple description coding structure. Two different previously encoded frames are used as reference frame in the motion compensation of the encoding frame. The prediction results are encoded as descriptions and side information respectively. The decoder can select either reference frame to implement decoding. The simulation results show that this two-prediction-loop structure can efficiently improve the video quality when errors occur.

9.2 Future Research Suggestion

In recently years, some error control schemes, which combine error concealments and error resilience methods together, have been presented. Compared with using either of them alone, the combination of error concealment and error resilience method can achieve better recovery performance, when error rate is high.

Most error concealments and error resilience methods are aimed to improve the peak signal-to-noise ratio (PSNR) of the decoded video. But sometimes the PSNR cannot reflect the real visual quality of the decoded video. For example, if an error occurs in the background of video, people may not notice it, even the PSNR is obviously decreased by the error. Future error control schemes should aim at improving the visual quality rather than PSNR value.

As the multicast communication network is prevalence, a more flexible video coding standard is required. This standard should be able to support various applications. Now, a scalable video coding standard is forthcoming, which can provide scalability in spatial and temporal domains. In order to provide the scalability, some complicated motion prediction schemes are used. The motion prediction is implemented not only between frames and pixels, but also implemented between layers. The relationships of neighboring frames and pixels will be different in the new coding standard. Therefore most previous error control methods are not suitable for this new standard. The error control for the scalable video coding is becoming critical.

Author's Publication

Journal

1. Jinghong Zheng and Lap-Pui Chau, "Error Resilient Transmission of H.264 Video over the Internet", Accepted by, IEEE Transactions on Broadcasting, Vol 52, June 2006.
2. Jinghong Zheng and Lap-Pui Chau, "Efficient Motion Vector Recovery Algorithm for H.264 Based on Polynomial Model", IEEE Transactions on Multimedia, Vol. 7, pp. 507—513, June 2005.
3. Jinghong Zheng and Lap-Pui Chau, "Error Concealment Algorithm for H.26L Using First-Order Plane Estimation", IEEE Transactions on Multimedia, Vol. 6, pp. 801—805, Dec 2004.
4. Jinghong Zheng and Lap-Pui Chau, "A Motion Vector Recovery Algorithm for Digital Video Using Lagrange Interpolation", IEEE Transactions on Broadcasting, Vol. 49, pp.383—389, Dec 2003.

Conference

5. Jinghong Zheng and Lap-Pui Chau, "Multiple Description Coding Using Multiple

- Reference Frame for Robust Video Transmission”, Proceeding, The 2005 International Symposium on Circuits and Systems, ISCAS '05, Vol.4, pp. 4006—4009, Kobe, Japan, 23-26 May 2005.
6. Jinghong Zheng and Lap-Pui Chau, “Content-Based Periodic Macroblock for Error-Resilient Transmission of H.264 Video”, Proceeding, 2004 International Conference on Image Processing, ICIP '04, Vol.2, pp. 845—848, Singapore, 24-27 October 2004.
 7. Jinghong Zheng and Lap-Pui Chau, “A Temporal Error Concealment Algorithm for H.264 Using Lagrange Interpolation”, Proceeding, The 2004 International Symposium on Circuits and Systems, ISCAS '04, Vol. 2, pp. II-133-6, Vancouver, Canada, 23-26 May 2004.
 8. Jinghong Zheng and Lap-Pui Chau, “A Temporal Error Concealment Algorithm for H.264 Based on Plane Estimation”, Proceeding, The 2003 Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia, Vol. 1, pp. 253—257, Singapore, 15-18 December 2003.
 9. Jinghong Zheng and Lap-Pui Chau, “An Efficient Spatial Domain Error Concealment Method for H.264 Video”, Proceeding, The 2003 Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia, Vol. 1, pp. 26—30, Singapore, 15-18 December 2003.

Bibliography

1. F. Pereira and T. Ebranhimi, “The MPEG-4 Book”, Prentice Hall PTR, New Jersey NJ, 2002.
2. I. E. G. Richardson, “H.264 and MPEG-4 Video Compression”, John Wiley & Sons Ltd, Chichester England, 2003.
3. I. E. G. Richardson, “Video Codec Design”, John Wiley & Sons Ltd, Chichester England, 2002.
4. ISO/IEC MPEG-2: Generic coding of moving pictures and associated audio information, December 2000.
5. ITU-T H.261: Video codec for audiovisual services at $p \times 64$ kbit/s, March 1993.
6. ITU-T H.263: Video coding for low bit rate communication, February 1998.
7. Joint Video Team of ISO/IEC MPEG & ITU-T VCEG: Version 3 of H.264/AVC, July 2004.
8. H. Wang, and R. M. Mersereau, “Fast algorithm for the estimation of motion vectors”, IEEE Transactions on Image Processing, Vol. 8, pp. 435—438, March 1999.
9. Y. Keller and A. Averbuch, “Fast motion estimation using bidirectional gradient methods”, IEEE Transactions on Image Processing, Vol. 13, pp. 1042—1054, August 2004.
10. D. Wang, “Unsupervised video segmentation based on watersheds and

- temporal tracking”, *IEEE Transactions on Circuits and System for Video Technology*, Vol. 8, pp. 539—546, September 1998.
11. B. C. Song and K. W. Chun, “Multi-resolution block matching algorithm and its VLSI architecture for fast motion estimation in an MPEG-2 video encoder”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 14, pp. 1119—1137, September 2004.
 12. D. Yu, S. K. Jang and J. B. Ra, “Fast motion estimation for shape coding in MPEG-4”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, pp. 358—363, April 2003.
 13. J. Zan, M. O. Ahmad and M. N. S. Swamy, “Pyramidal motion estimation techniques exploiting intra-level motion correlation”, *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 50, pp. 83—93, February 2003.
 14. K. Sayood, H. H. Out and N. Demir, “Joint source/channel coding for variable length codes”, *IEEE Transactions on Communication*, Vol. 48, pp. 787—794, May 2000.
 15. H. Nicolas, S. Pateux and D. L. Guen, “Minimum description length criterion and segmentation map coding for region-based video compression”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 11, pp. 184—198, February 2001.
 16. C. Tu and T. D. Tran, “Context-based entropy coding of block transform coefficients for image compression,” *IEEE Transactions on Image Processing*, Vol. 11, pp. 1271—1283, November 2002.
 17. R. G. Gallager, “Variations on a theme by Huffman”, *IEEE Transactions on Information Theory*, Vol. IT-24, pp. 668—674, November 1978.
 18. W. Szpankowski, “Asymptotic average redundancy of Huffman (and other) block codes”, *IEEE Transactions on Information Theory*, Vol. 46, pp. 2434—2443, November 2000.

19. M. H. Ackroyd, "Huffman sequences with approximately uniform envelopes or cross-correlation functions", *IEEE Transactions on Information Theory*, Vol. IT-23, pp. 620—623, September 1977.
20. T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC video coding standard", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, pp. 560—576, July 2003.
21. T. Wiegand, H. Schwarz, A. Joch, F. Kossentini and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, pp. 688—703, July 2003.
22. H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in H.264", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, pp. 598—603, July 2003.
23. M. Wien, "Variable block-size transforms for H.264/AVC", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, pp. 604—613, July 2003.
24. T. Wiegand, X. Zhang and B. Girod, "Long-term memory motion-compensated prediction", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 9, pp. 70—84, February 1999.
25. Y. C. Chang, M. S. Beg and T. F. Tang, "Performance evaluation of MPEG-4 visual error resilient tools over a mobile channel", *IEEE Transactions on Consumer Electronics*, Vol. 49, pp. 6—13, February 2003.
26. F. Delicado, P. Cuenca, L. Orozco-Barbosa, A. Garrido and F. Quiles, "Performance evaluation of error-resilient mechanisms for MPEG-4 video communications over wireless ATM links", *2001 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, 2001. Vol. 2, pp. 26—28, August 2001.

27. W. Tan and A. Zakhor, "Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol", *IEEE Transactions on Multimedia*, Vol. 1, pp. 172—186, June 1999.
28. V. Parthasarathy, J. W. Modestino, and K. S. Vastola, "Reliable transmission of high-quality video over ATM networks", *IEEE Transactions on Image Processing*, Vol. 8, pp. 361—374, March 1999.
29. P. Frossard, "FEC performance in multimedia streaming", *IEEE Communications Letters*, Vol. 5, pp. 122—124, March 2001.
30. E. W. Biesack, "Performance evaluation of forward error correction in an ATM environment", *IEEE Journal on selected areas in communications*, Vol. 11, pp. 631—640, May 1993.
31. J. Faucher, M. B. Venditti and D. V. Plant, "Application of parallel forward-error correction in two-dimensional optical-data links", *Journal of Lightwave Technology*, Vol. 21, pp. 466—475, February 2003.
32. K. Stuhlmuller, N. Farber, M. Link, and B. Girod, "Analysis of Video Transmission over Lossy Channels", *IEEE Journal on Selected Areas in Communication*, Vol. 18, pp. 1012—1032, June 2000.
33. L. Yin, J. Lu, K. B. Letaief and Y. Wu, "Burst-error correcting algorithm for Reed-Solomon codes", *Electronics Letters*, Vol. 37, pp. 695—697, May 2001.
34. B. Fong, P. B. Rapajic and G. Y. Hong, "Forward error correction with Reed-Solomon codes for wearable computers", *IEEE Transactions on Consumer Electronics*, Vol. 48, pp. 917—921, Nov 2003.
35. P. Frossard and O. Verscheure, "Joint source/FEC rate selection for quality-optimal MPEG-2 video delivery", *IEEE Transactions on Image Processing*, Vol. 10, pp. 1815—1825, December 2001.
36. T. W. A. Lee, S. H. G. Chan, Q. Zhang, W. W. Zhu and Y. Zhang, "Allocation of layer bandwidths and FECs for video multicast over wired and wireless networks", *IEEE Transactions on Circuits and Systems for Video Technology*,

- Vol. 12, pp. 1059—1070, December 2002.
37. A. Alavi, R. Link and S. Kallel, “Adaptive unequal error protection for subband image coding”, *IEEE Transactions on Broadcasting*, Vol. 46, pp. 197—205, September 2000.
38. W. Tan and A. Zakhor, “Video multicast using layered FEC and scalable compression”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 11, pp.373—386, March 2001.
39. M. V. D. Schaar and H. Radha, “Unequal packet loss resilience for fine-granular-scalability video”, *IEEE Transactions on Multimedia*, Vol. 3, pp. 381—394, December 2001.
40. A. E. Mohr, E. A. Rishkin and R. E. Ladner, “Unequal loss protection: graceful degradation of image quality over packet erasure channels through forward error correction”, *IEEE Journal on Selected Areas in Communications*, Vol. 18, pp. 819—828, June 2000.
41. K. Brayer and S. Natarajan, “An investigation of ARQ and hybrid FEC-ARQ on an experimental high latitude meteor burst channel”, *IEEE Transactions on Communications*, Vol. 37, pp. 1239—1242, November 1989.
42. A. Shiozaki, K. Okuno, K. Suzuki and T. Segawa, “A hybrid ARQ scheme with adaptive forward error correction for satellite communication”, *IEEE Transactions on Communications*, Vol. 39, pp. 482—484, April 1991.
43. S. Kallel, “Efficient hybrid ARQ protocols with adaptive forward error correction”, *IEEE Transactions on Communications*, Vol. 42, pp. 281—289, April 1994.
44. N. Kashyap and D. L. Neuhoff, “Data synchronization with timing”, *IEEE Transactions on Information Theory*, Vol. 47, pp. 1444—1460, May 2001.
45. D. W. Redmill and N. G. Kingsbury, “The EREC: an error-resilient technique for coding variable-length blocks of data”, *IEEE Transactions on Image*

- Processing, Vol. 5, pp. 565—574, April 1996.
46. M. C. Davey and D. J. C. Mackay, “Reliable communication over channels with insertions, deletions, and substitutions”, *IEEE Transactions on Information Theory*, Vol. 46, pp. 687—698, February 2001.
 47. P. Frossard and O. Verscheure, “AMISP: a complete content-based MPEG-2 error-resilient scheme”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 11, pp. 989—998, September 2001.
 48. T. Lee, H. Hsu and P. Chang, “Restart marker regulation technique for progressive JPEG image coding in mobile communications”, *IEEE Communication Letters*, Vol. 4, pp. 411—413, December 2000.
 49. G. Cote, S. Shirani, and F. Kossentini, “Optimal mode selection and synchronization for robust video communications over error-prone networks”, *IEEE Journal on Selected Areas in Communications*, Vol. 18, pp. 952—965, June 2000.
 50. K. Yoo, “Adaptive resynchronisation marker positioning method for error resilient video transmission”, *Electronics Letters*, Vol. 34, pp. 2084—2085, October 1998.
 51. J. W. Suh and Y. S. Ho, “Error concealment based on directional interpolation”, *IEEE Transactions on Consumer Electronics*, Vol. 43, pp. 295—302, August 1997.
 52. G. Grevera and J. K. Udupa, “Shape-based interpolation of multidimensional grey-level images”, *IEEE Transactions on Medical Imaging*, Vol. 15, pp. 881—892, December 1996.
 53. W. E. Higgins, C. Morice and E. L. Ritman, “Shape-based interpolation of tree-like structures in three-dimensional images”, *IEEE Transactions on Medical Imaging*, Vol. 12, pp. 439—449, September 1993.
 54. S. Hsia, “An edge-oriented spatial interpolation for consecutive block error concealment”, *IEEE Signal Processing Letters*, Vol. 11, pp. 577—580, June

- 2004.
55. W. Kwok and H. Sun, "Multi-directional interpolation for spatial error concealment", *IEEE Transactions on Consumer Electronics*, Vol. 39, pp. 455—460, August 1993.
 56. X. Li and M. T. Orchard, "New edge-directed interpolation", *IEEE Transactions on Image Processing*, Vol. 10, pp. 1521—1527, October 2001.
 57. J. W. Park, J. W. Kim and S. U. Lee, "DCT coefficients recovery-based error concealment technique and its application to the MPEG-2 bit stream error", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, pp. 845—854, December 1997.
 58. Y. J. Chung, J. W. Kim and C. C. J. Kuo, "DCT based error concealment for RTSP video over a modem internet connection", *IEEE International Symposium on Circuits and Systems*, Vol. 6, pp. 518—521, 31 May-3 June 1998.
 59. Y. J. Chung, J. W. Kim and C. C. J. Kuo, "Real-time streaming video with adaptive bandwidth control and DCT-based error concealment", *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing*, Vol. 46, pp. 951—956, July 1999.
 60. J. Ridge, F. W. Ware and J. D. Gibson, "Permuted smoothed descriptions and refinement coding for images", *IEEE Journal on Selected Areas in Communications*, Vol. 18, pp. 915—926, June 2000.
 61. Y. Wang, Q. Zhu and L. Shaw, "Maximally smooth image recovery in transform coding", *IEEE Transactions on Communications*, Vol. 41, pp. 1544—1550, October 1993.
 62. W. Zhu, Y. Wang and Q. F. Zhu, "Second-order derivative-based smoothness measure for error concealment in DCT-based codecs", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 8, pp. 713—718, October 1998.

63. W. M. Lam, A. R. Reibman and B. Liu, "Recovery of lost or erroneously received motion vectors", International Conference on Acoustics, Speech, and Signal Processing, ICASSP, Vol. 5, pp. 417—420, March 1993.
64. C. S. Park, J. Ye and S. U. Lee, "Lost motion vector recovery algorithm", IEEE International Symposium on Circuits and Systems, ISCAS 94, Vol. 3, pp. 229—232, 30 May-2 June 1994.
65. J. Feng, K. Lo, H. Mehrpour and A. E. Karbowiak, "Loss recovery techniques for transmission of MPEG video over ATM networks", IEEE International Conference on Communication, 1996. ICC 96, Conference Record, Converging Technologies for Tomorrow's Applications, Vol. 3, pp. 1406—1410, 23-27 June 1996.
66. T. Chen, X. Zhang and Y. Q. Shi, "Error concealment using refined boundary matching algorithm", International Conference on Information Technology: Research and Education, pp. 55—59, 11-13 August 2003.
67. Y. H. Jung, Y. Kim and Y. Choe, "Robust error concealment algorithm using iterative weighted boundary matching criterion", International Conference on Image Processing, Vol. 3, pp. 384—387, 10-13 September 2000.
68. T. H. Tsai, Y. X. Lee and Y. F. Lin, "Video error concealment techniques using progressive interpolation and boundary matching algorithm" International Symposium on Circuits and Systems, ISCAS 2004, Vol. 5, pp. V-433--V-436, 23-26 May 2004.
69. Y. C. Lee, Y. Altunbasak and R. Mersereau, "A temporal error concealment method for MPEG coded video using a multi-frame boundary matching algorithm", International Conference on Image Processing, Vol. 1, pp. 990—993, 7-10 October 2001.
70. M. E. Al-Mualla and C. N. Canagarajah and D. R. Bull, "Multiple-reference temporal error concealment", IEEE International Symposium on Circuits and Systems, ISCAS 2001, Vol. 5, pp. 149—152, 6-9 May 2001.

71. M. J. Chen, C. S. Chen and M. C. Chi, "Recursive block-matching principle for error concealment algorithm", International Symposium on Circuits and Systems, ISCAS' 03, Vol. 2, pp. II-528-II-531, 25-28 May 2003.
72. S. Tsekeridou and I. Pitas, "MPEG-2 error concealment based on block-matching principles", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, pp. 646—658, June 2000.
73. R. Zhang, Y. Zhou and X. Huang, "Content-adaptive spatial error concealment for video communication", IEEE Transactions on Consumer Electronics, Vol. 50, pp. 335—340, February 2004.
74. S. H. Lee, D. H. Choi and C. S. Hwang, "Error concealment using affine transform for H.263 coded video transmissions", Electronics Letters, Vol. 37, pp. 218—220, February 2001.
75. S. Shirani, F. Kossentini and R. Ward, "A concealment method for video communications in an error-prone environment", IEEE Journal on Selected Areas in Communications, Vol. 18, pp. 1122—1128S, June 2000.
76. S. Shirani, B. Erol and F. Kossentini, "A concealment method for shape information in MPEG-4 coded video sequences", IEEE Transactions on Multimedia, Vol. 2, pp. 185—190, September 2000.
77. Y. Zhang and K. K. Ma, "Error concealment for video transmission with dual multiscale Markov random field modeling", IEEE Transactions on Image Processing, Vol. 12, pp. 236—242, February 2003.
78. S. Belfiore, M. Grangetto, E. Magli and G. Olmo, "An edge and texture preserving algorithm for video error concealment", 2002 IEEE Workshop on Multimedia Signal Processing, pp. 121—124, 9-11 December 2002.
79. P. Salama, N. B. Shroff and E. J. Delp, "Error concealment in MPEG video streams over ATM networks", IEEE Journal on Selected Areas in Communications, Vol. 18, pp. 1129—1144, June 2000.
80. Y. S. Lee, K. K. Ong and C. Y. Lee, "Error-resilient image coding (ERIC) with

- smart IDCT error concealment technique for wireless multimedia transmission”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, pp. 176—181, February 2003.
81. D. Zhang and Z. Wang, “Image information restoration based on long-range correlation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, pp. 331—341, May 2002.
82. D. S. Turaga and T. Chen, “Model-based error concealment for wireless video”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, pp. 483—495, June 2002.
83. G. M. Schuster, X. Li and A. K. Katsaggelos, “Shape error concealment using Hermite splines”, IEEE Transactions on Image Processing, Vol. 13, pp. 808—820, June 2004.
84. L. D. Soares and F. Pereira, “Spatial shape error concealment for object-based image and video coding”, IEEE Transactions on Image Processing, Vol. 13, pp. 586—599, April 2004.
85. S. C. Pei and Y. Z. Chou, “Novel error concealment method with adaptive prediction to the abrupt and gradual scene changes”, IEEE Transactions on Multimedia, Vol. 6, pp. 158—173, February 2004.
86. J. Pyun, J. S. Lee, J. W. Jeong, J. H. Jeong and S. J. Ko, “Robust error concealment for visual communications in burst-packet-loss networks”, IEEE Transactions on Consumer Electronics, Vol. 49, pp. 1013—1019, November 2003.
87. M. J. Chen, L. G. Chen and R. M. Weng, “Error concealment of lost motion vectors with overlapped motion compensation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 7, pp. 560—563, June 1997.
88. M. A. Mualla, N. Canagarajah and D. R. Bull, “Temporal error concealment using motion field interpolation”, Electronics Letters, Vol. 35, pp. 215—217, February 1999.

89. W. S. Lee, M. R. Frater, M. R. Pickering and J. F. Arnold, "Spatial temporal concealment of lost blocks in coded video", International Conference on Image Processing, Vol. 3, pp. 477—481, 4-7 October 1998.
90. P. J. Lee and M. J. Chen, "Robust error concealment algorithm for video decoder", IEEE Transactions on Consumer Electronics, Vol. 45, pp. 851—859, August 1999.
91. S. Tsekeridou, F. A. Cheikh, M. Gabbouj and I. Pitas, "Vector rational interpolation schemes for erroneous motion field estimation applied to MPEG-2 error concealment", IEEE Transactions on Multimedia, Vol. 6, pp. 876-- , December 2004.
92. Y. C. Lee, Y. Altunbasak and R. M. Mersereau, "Multiframe error concealment for MPEG-coded video delivery over error-prone networks", IEEE Transactions on Image Processing, Vol. 11, pp. 1314—1331, November 2002.
93. L. Tang and S. Yu, "Error concealment with error level tracking and backward frame update", Electronics Letters, Vol. 40, pp. 1049—1050, 19 August 2004.
94. J. W. Suh and Y. S. Ho, "Error concealment techniques for digital TV", IEEE Transactions on Broadcasting, Vol. 48, pp. 299—306, December 2002.
95. L. Wang, M. N. S. Swamy and M. O. Ahmad, "Multiple description image coding using pixel interleaving and wavelet transform," The 2002 45th Midwest Symposium on Circuits and Systems, Vol. 2, pp. II-235—II-238, August 2002.
96. C. S. Kim and S. U. Lee, "Multiple description coding of motion fields for robust video transmission", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, pp. 999—1010, September 2001.
97. Y. Wang, M. T. Orchard, V. Vaishampayan and A. R. Reibman, "Multiple description coding using pairwise correlating transforms", IEEE Transactions on Image Processing, Vol. 10, pp. 351—366, March 2001.

98. A. Reibman, H. Jafarkhani, Y. Wang and M. Orchard, "Multiple description video using rate-distortion splitting", International Conference on Image Processing, Vol. 1, pp. 978—981, October 2001.
99. R. Venkataramani, G. Kramer and V. K. Goyal, "Multiple description coding with many channels", IEEE Transactions on Information Theory, Vol. 49, pp. 2106—2114, September 2003.
100. J. Y. Liao and J. Villasenor, "Adaptive intra block update for robust transmission of H.263", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, pp. 30—35, February 2000.
101. S. Dogan, A. Cellatoglu, M. Uyguroglu, A. H. Sadka and A. M. Kondo, "Error-resilient video transcoding for robust Internet network communications using GPRS", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, pp. 453—464, June 2002.
102. H. Chiou, Y. Lee and C. Lin, "Error-resilient transcoding using adaptive intra refresh for video streaming", International Symposium on Circuits and Systems, ISCAS' 04, Vol. 3, pp. III-777-80, 23-26 May 2004.
103. C. Zhao, K. N. Ngan, J. Zhang, R. Mathew and X. Zhang, "Using inter-frame dependence history to select intra-refresh blocks", Electronics Letter, Vol. 38, pp. 1337—1338, October 2002.
104. P. Chang and T. Lee, "Precise and fast error tracking for error-resilient transmission of H.263 video", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, pp. 600—607, June 2000.
105. S. T. Worrall, A. H. Sadka, A. M. Kondo and P. Sweeney, "Motion adaptive intra refresh for MPEG-4", Electronics Letter, Vol. 36, pp. 1924—1925, November 2000.
106. Z. He, J. Cai, and C. W. Chen, "Joint source channel rate-distortion analysis for adaptive mode selection and rate control in wireless video coding", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, pp.

- 511—523, June 2002.
107. Z. Ma, S. Yu and Z. Zeng, "Error resilient intra refresh scheme for H.26L stream", *Electronics Letters*, Vol. 38, pp. 1153—1154, September 2002.
108. G. J. Sullivan, T. Wiegand and T. Stockhammer, "Using the draft H.26L video coding standard for mobile application," *International Conference on Image Processing*, 2001, Vol. 3, pp. 573-576, October 2001.
109. G. Bjontegaard (editor) "H.26L test model long term number 8 (TML-8) draft0," *ITU-Telecommunication Standardization Sector, VCEG*, June 2001.
110. G. S. YU, M. K. Liu, and M.W. Marcellin, "POCS-based error concealment for packet video using multiframe overlap information", *IEEE Transaction on Circuits and System for Video Technology*, Vol. 8, pp.422—433, August 1998.
111. H. Sun and W. Kwok, "Concealment of damaged block transform coded image using projections onto convex sets", *IEEE Transaction on Image Processing*, Vol. 4, pp. 470—477, April 1995.
112. G. Heising, D. Marpe, H. L. Cycon and A. P. Petukhov, "Wavelet-based very low bit-rate video coding using image warping and overlapped block motion compensation", *IEE Proceedings on Vision, Image and Signal Processing*, Vol. 148, Issue: 2, pp. 93—101, April 2001.
113. Y. Wang and Q. F. Zhu, "Error control and concealment for video communication: A review," *International Conference on Image Processing*, Vol. 2, pp. 974—997, May 1998.
114. L. Atzori, F. G. B. De Natale, and C. Perra, "A spatial-temporal concealment technique using boundary matching algorithm and mesh-based warping(BMA-MBW)," *IEEE Transactions on Multimedia*, Vol. 3, pp. 326—337, September 2001.
115. Z. Wang, Y. Y. and D. Zhang, "Best neighboring matching: an information loss restoration technique for block-based image coding systems", *IEEE Transactions on Image Processing*, Vol. 7, pp. 1056—1061, July 1998.

116. S. Valente, C. Dufour, F. Groliere and D. Snook, "An efficient error concealment implementation for MPEG-4 video streams," *IEEE Transactions on Consumer Electronics*, Vol. 47, pp. 568—578, August 2001.
117. J. W. Park and S. U. Lee, "Recovery of corrupted image data based on the NURBS interpolation," *IEEE Transactions on Circuit and Systems for Video Technology*, Vol. 9, pp. 1003—1008, October 1999.
118. T. Wiegand (editor), "Joint Final Committee Draft (JFCD) of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)," Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, July 2002.
119. S. M. Pizer, "Numerical computing and mathematical analysis", Science Research Associates, INC. 1975.
120. Y. Wang, M. M. Hannuksela, V. Varsa, A. Hourunranta and M. Gabbouj, "The Error Concealment Feature in the H.26L Test Model," *International Conference on Image Processing 2002, Proceedings*, Vol. 2, September 2002.
121. Q. F. Zhu, Y. Wang, and L. Shaw, "Coding and Cell-Loss Recovery in DCT-Based Packet Video," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 3, pp. 248—258, June 1993.
122. J. W. Suh and Y. S. Ho, "Directional Interpolation for Spatial Error Concealment," *International Conference on Consumer Electronic*, pp. 26—27, June 1997.
123. Y. Wang and S. Lin, "Error-resilient video coding using multiple description motion compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, pp. 438—452, June 2002.
124. I. K. Kim and N. I. Cho, "Error resilient video coding using optimal multiple description of DCT coefficients," *International Conference on Image Processing*, Vol. 2, September 2003.
125. S. S. Hemami, "Robust image transmission using resynchronizing variable-length codes and error concealment," *IEEE Journal on Selected Areas*

- in Communications, Vol. 18, pp. 927—939, June 2000.
126. J. Kim, R. M. Mersereau and Y. Altunbasak, “Error-resilience image and video transmission over the Internet using unequal error protection,” IEEE Transactions on Image Processing, Vol. 12, pp. 121—131, February 2003.
127. T. C. Wang, H. C. Fang and L. G. Chen, “Low-delay and error-robust wireless video transmission for video communications,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, pp. 1049—1058, December 2002.
128. T. Wiegand, N. Farber, K. Stuhlmuller and B. Girod, “Error –resilient video transmission using long-term memory motion-compensated prediction,” IEEE Journal on Selected Areas in Communications, Vol. 18, pp. 1050—1062, June 2000.
129. I. Rhee and S. R. Joshi, “Error recovery for interactive video transmission over the Internet,” IEEE Journal on Selected Areas in Communication, Vol. 18, pp. 1033—1049, June 2000.
130. M. Gallant and F. Kossentini, “Rate-distortion optimized layered coding with unequal error protection for robust Internet video,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, pp. 357—372, March 2001.
131. K. Zhang and J. Kittle, “Multiple-thread long-term memory scheme for efficient video coding”, Electronics Letters, Vol. 35, pp. 1452—1453, August 1999.
132. T. Wiegand, E. Steinbach and B. Girod, “Long-term memory prediction using affine motion compensation”, International Conference on Image Processing, Vol. 1, pp. 51—55, 1999.
133. M. Budagavi and J. D. Gibson, “Multiframe video coding for improved performance over wireless channels”, IEEE Transactions on Image Processing, Vol. 10, pp. 252—265, February 2001.
134. Reibman. A. R, Jafarkhani. H, Wang. Y, Orchard. M. T and Puri. R,

- “Multiple-description video coding using motion-compensated temporal prediction,” *IEEE Transactions on Circuits and System for Video Technology*, Vol. 12, pp. 193—204, March 2002.
135. Tang. X and Zakhor. A, “ Matching pursuits multiple description coding for wireless video,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, pp.566—575, June 2002.
 136. Y. C. Lee, Y. Altunbasak and R. M. Mersereau, “An enhanced two-stage multiple description video coder with drift reduction”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 14, pp. 122—127, January 2004.
 137. “Chapter 7 Least Squares Estimation”, [http:// mahi.ucsd.edu / cathy / Classes / SIO223 / Part1 / sio223.chap7.pdf](http://mahi.ucsd.edu/cathy/Classes/SIO223/Part1/sio223.chap7.pdf), 2004.
 138. U. Horn, K. Stuhlmuller, M. Link and B. Girod, “Robust Internet video transmission based on scalable coding and unequal error protection”, *Signal Processing: Image Communication*, Vol. 15, pp. 77—94, 1999.
 139. C. W. Yap, K. N. Ngan and R. Liyanapathirana, “A combined source-channel video coding scheme for mobile channels”, *Signal Processing: Image Communication*, Vol. 14, pp. 559—574, 1999.
 140. V. A. Vaishampayan, “Design of multiple description scalar quantizer”, *IEEE Transactions on Information Theory*, Vol. 39, pp. 821—834, May 1993.
 141. R. Anand, K. Ramchandran, and I. V. Kozintsev, “Continuous error detection (CED for reliable communication”, *IEEE Transactions on Communications*, Vol. 49, pp. 1540—1549, September 2001.
 142. C. W. Lin and J. L. Wu, “On error detection and error synchronization of reversible variable-length codes”, *IEEE Transactions on Communications*, Vol. 53, pp. 826—832, May 2005.
 143. A. S. Tom, C. L. Yeh and F. Chu, “Packet video for cell loss protection using deinterleaving and scrambling”, *International Conference on Acoustics*,

Speech and Signal Processing, Vol. 4, pp. 2857—2860, April 1991.