ROBUST TRANSMISSION OF 3D MODELS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MEHMET OĞUZ BİCİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

NOVEMBER 2010

Approval of the thesis:

ROBUST TRANSMISSION OF 3D MODELS

submitted by **MEHMET OĞUZ BİCİ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İsmet Erkmen
Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. Gözde Bozdağı Akar
Supervisor, **Electrical and Electronics Engineering Dept., METU** _____

Examining Committee Members:

Prof. Dr. Aydın Alatan
Electrical and Electronics Engineering Dept., METU _____

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering Dept., METU _____

Prof. Dr. A. Enis Çetin
Electrical and Electronics Engineering Dept., Bilkent University _____

Assoc. Prof. Dr. Uğur Güdükbay
Computer Engineering Dept., Bilkent University _____

Assist. Prof. İlkay Ulusoy
Electrical and Electronics Engineering Dept., METU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MEHMET OĞUZ BİCİ

Signature :

ABSTRACT

ROBUST TRANSMISSION OF 3D MODELS

Bici, Mehmet Oğuz
Ph.D., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. Gözde Bozdağı Akar

November 2010, 141 pages

In this thesis, robust transmission of 3D models represented by static or time consistent animated meshes is studied from the aspects of scalable coding, multiple description coding (MDC) and error resilient coding. First, three methods for MDC of static meshes are proposed which are based on multiple description scalar quantization, partitioning wavelet trees and optimal protection of scalable bitstream by forward error correction (FEC) respectively. For each method, optimizations and tools to decrease complexity are presented. The FEC based MDC method is also extended as a method for packet loss resilient transmission followed by in-depth analysis of performance comparison with state of the art techniques, which pointed significant improvement. Next, three methods for MDC of animated meshes are proposed which are based on layer duplication and partitioning of the set of vertices of a scalable coded animated mesh by spatial or temporal subsampling where each set is encoded separately to generate independently decodable bitstreams. The proposed MDC methods can achieve varying redundancy allocations by including a number of encoded spatial or temporal layers from the other description. The algorithms are evaluated with redundancy-rate-distortion curves and per-frame reconstruction analysis. Then for layered predictive compression of animated meshes, three novel prediction structures are proposed and integrated into

a state of the art layered predictive coder. The proposed structures are based on weighted spatial/temporal prediction and angular relations of triangles between current and previous frames. The experimental results show that compared to state of the art scalable predictive coder, up to 30% bitrate reductions can be achieved with the combination of proposed prediction schemes depending on the content and quantization level. Finally, optimal quality scalability support is proposed for the state of the art scalable predictive animated mesh coding structure, which only supports resolution scalability. Two methods based on arranging the bitplane order with respect to encoding or decoding order are proposed together with a novel trellis based optimization framework. Possible simplifications are provided to achieve tradeoff between compression performance and complexity. Experimental results show that the optimization framework achieves quality scalability with significantly better compression performance than state of the art without optimization.

Keywords: 3D mesh, multiple description coding, error resilient coding, predictive coding, scalable coding

ÖZ

3B MODELLERİN DAYANIKLI İLETİMİ

Bici, Mehmet Oğuz
Doktora, Elektrik ve Elektronik Mühendisliğ Bölümü
Tez Yöneticisi : Prof. Dr. Gözde Bozdağı Akar

Kasım 2010, 141 sayfa

Bu tezde statik veya zaman tutarlı hareketli tel örgüler ile temsil edilen 3B modellerin dayanıklı iletimi ölçeklenebilir kodlama, çoklu anlatım (ÇA) kodlama ve hataya dayanıklı kodlama yönlerinden incelenmiştir. İlk olarak, statik tel örgülerin ÇA kodlanması için çoklu anlatım skaler nicemleme, dalgacık ağaçların ayrılması ve ölçeklenebilir bitkatarının ileri hata koruma (İHK) ile en iyi korunmasına dayalı üç metot önerilmiştir. Her metot için en iyilemeler ve karmaşıklığı azaltıcı araçlar sunulmuştur. İHK kullanan ÇA kodlama metodu ayrıca paket kayıplarına dayanıklı iletim için bir metot olarak genişletilmiş ve derinlemesine analizın ardından en ileri teknoloji teknikler ile performansı karşılaştırılmış, önemli iyileşmeler gözlenmiştir. Sonra, hareketli tel örgülerin ÇA kodlanması için katman kopyalama ve ölçeklenebilir kodlanmış tel örgünün düğüm kümesinin uzamsal veya zamansal alt örneklenmesiyle bölüntülenmesine dayalı üç metot önerilmiştir. Her küme bağımsız olarak kodçözülebilir bitkatarı üretmek üzere ayrı olarak kodlanmıştır. Önerilen ÇA kodlama metotları diğer kümeden belli sayıda uzamsal ya da zamansal katmanları içererek değişken artıklık tahsis edebilmektedir. Önerilen metotlar artıklık-hız-bozulum eğrileri ve çerçeve başına geri çatılım analizi ile değerlendirilmiştir. Daha sonra, hareketli tel örgülerin katmanlı ve öngörücü sıkıştırılması için üç özgün öngörü yapısı önerilmiş ve son teknoloji katmanlı öngörücü bir

kodlayıcıya eklenmiştir. Önerilen yapılar ağırlıklandırılmış uzamsal/zamansal öngörü ve şu anki ve önceki çerçeve arasındaki üçgenlerin açısal ilişkilerine dayanmaktadır. Deneylerde son teknoloji ölçeklenebilir öngörücü kodlama ile karşılaştırıldığında, içerik ve nicemleme seviyesine bağlı olarak önerilen yapıların kombinasyonları ile 30%'a varan bit hızı kazancı elde edilebildiği görülmüştür. Son olarak, son teknoloji ölçeklenebilir öngörücü hareketli tel örgü kodlama yapısına en iyi kalite ölçeklenebilme desteği önerilmiştir; ki o sadece çözünürlük ölçeklenebilirliği desteklemektedir. Bit düzlemi sıralamasının kodlama veya kodçözme sırasına göre ayarlanmasına dayalı iki metot önerilmiştir. Metotlar özgün olarak kafese dayalı en iyileme çerçevesi kullanmaktadır. Sıkıştırma performansı ve karmaşıklık arası ödünleşim elde etmek için olası basitleştirmeler sunulmuştur. Deneysel sonuçlar en iyileme çerçevesinin en iyilemesiz son teknolojiden önemli boyutta daha iyi sıkıştırma performansı elde ettiğini göstermiştir.

Anahtar Kelimeler: 3B tel örgü, çoklu anlatım kodlama, hataya dayanıklı kodlama, öngörücü kodlama, ölçeklenebilir kodlama

To my wife

ACKNOWLEDGMENTS

I would like to express my sincere and deepest gratitude to my supervisor Gözde Bozdağı Akar for her supervision, guidance and encouragements. I deeply appreciate her support in every part of my PhD. I would also like to thank Prof. Aydın Alatan and Assoc. Prof. Uğur Güdükbay for their valuable comments and feedbacks during my thesis progress meetings.

I would like to acknowledge the following researchers for their cooperation and contributions to my thesis. I would like to thank Anıl Aksay who always shared his knowledge and experience with me. I would like to acknowledge Andrey Norkin for our efficient collaborative works. I would like to thank Nikolce Stefanoski for hosting me, for providing his dynamic mesh coder software and for the very helpful discussions. I would like to acknowledge Kıvanç Köse for our useful discussions about meshes.

I would like to acknowledge all my friends in Multimedia Research Group at METU for creating a very enjoyable lab environment in the past years. Thank you Çağdaş Bilen, Cevahir Çığla, Erman Okman, Anıl Aksay, Özgü Alay, Alper Koz, Özlem Pasin, Yusuf Bediz, Eren Gürses, Ahmet Saraçoğlu, Engin Tola, Birant Örten, Burak Özkalaycı, Evren İmre, Yusuf Bayraktaroğlu, Serdar Gedik, Murat Deniz Aykın, Bertan Günyel, Emrah Bala, Elif Vural, Yoldaş Ataseven, Oytun Akman, Cem Vedat Işık, Berkan Solmaz, Döne Buğdaycı, Murat Demirtaş and others.

I would like to dedicate this thesis to my dear wife, İpek Bici, who always supported me with her endless love in every part of my life. Finally, I would like thank all my family for their love and support.

This thesis is partially supported by European Commission within FP6 under Grant 511568 with the acronym 3DTV, within FP7 under Grant 216503 with the acronym MOBILE3DTV and The Scientific and Technological Research Council of Turkey (TÜBİTAK) under National Scholarship Programme for PhD Students.

The chicken character was created by Andrew Glassner, Tom McClure, Scott Benza, and

Mark Van Langeveld. This short sequence of connectivity and vertex position data is distributed solely for the purpose of comparison of geometry compression techniques.

TABLE OF CONTENTS

ABSTRACT iv

ÖZ vi

ACKNOWLEDGMENTS ix

TABLE OF CONTENTS xi

LIST OF TABLES xv

LIST OF FIGURES xvii

LIST OF ABBREVIATIONS xxii

CHAPTERS

1 INTRODUCTION 1

1.1 Scope and Outline of the Thesis 3

2 BACKGROUND 4

2.1 3D Mesh Data 4

2.2 Error Resilient Coding 5

2.2.1 Multiple Description Coding 6

2.2.1.1 Redundancy Allocation 8

2.2.1.2 Mismatch Control 8

2.2.2 Packet Loss Resilient Streaming 9

2.3 Scalable Coding 9

2.4 Forward Error Correction 10

3 3D MESH CODING 11

3.1 Literature Review on Static 3D Mesh Coding 11

3.1.1 Wavelet Based Scalable Static 3D Mesh Coding and PGC 12

3.1.2 Compressed Progressive Meshes (CPM) 13

3.2 Literature Review on Animated 3D Mesh Coding 14
3.2.1 Scalable Predictive Coding (SPC) - MPEG-4 FAMC . . . 18
3.3 3D Mesh Distortion Metrics 23
3.3.1 Static 3D Mesh Distortion Metrics 23
3.3.2 Animated 3D Mesh Distortion Metrics 24
4 MULTIPLE DESCRIPTION CODING OF STATIC 3D MESHES 25
4.1 Introduction - Literature Review 25
4.2 Proposed MDSQ 26
4.3 Proposed Tree Partitioning 31
4.4 Proposed FEC Based Approach for MDC 40
4.4.1 Problem Definition and Proposed Solution 40
4.4.2 MDC Experimental Results 42
4.5 Extension of MD-FEC to Packet Loss Resilient Coding 43
4.5.1 Packet Loss Resilient Coding Based on CPM 44
4.5.2 Proposed Modifications for CPM based Loss Resilient Coding 45
4.5.3 Distortion Metric and Simplifications in Calculations . . . 46
4.5.4 Channel Model 46
4.5.5 Experimental Results 47
4.5.5.1 Proposed kStep for CPM Based Methods . . . 48
4.5.5.2 Comparison of CPM Based Methods 49
4.5.5.3 Performance of D-R Curve Modeling for PGC Based Methods 50
4.5.5.4 Comparison of CPM and PGC Based Methods 50
4.5.5.5 Mismatch Scenario 53
4.5.5.6 Complexity Comparison 54
4.5.5.7 Visual Comparison of CPM and PGC Based Methods 55
4.6 Conclusions 55
5 MULTIPLE DESCRIPTION CODING OF ANIMATED MESHES 60
5.1 Introduction 60

5.2 Reference Dynamic Mesh Coder 61
5.3 Proposed MDC Methods 61
5.3.1 Vertex Partitioning Based MDC 62
5.3.1.1 MD Encoder 62
5.3.1.2 MD Side Decoder 64
5.3.1.3 MD Central Decoder 65
5.3.2 Temporal Subsampling Based MDC 66
5.3.2.1 MD Encoder 66
5.3.2.2 MD Side Decoder 67
5.3.2.3 MD Central Decoder 68
5.3.3 Layer Duplication Based MDC 68
5.3.3.1 MD Encoder 68
5.3.3.2 MD Side Decoder 70
5.3.3.3 MD Central Decoder 70
5.3.4 Comments on the Mismatch 70
5.3.5 Further Possible Improvements on Error Concealment . . 71
5.4 Results 71
5.4.1 Vertex Partitioning 73
5.4.2 Temporal Subsampling 75
5.4.3 Layer Duplication 75
5.4.4 Comparison 76
5.4.5 Per Frame Analysis 79
5.5 Conclusions and Future Work 85
6 IMPROVED PREDICTION METHODS FOR SCALABLE PREDICTIVE ANIMATED MESH COMPRESSION 86
6.1 Prediction Structures 86
6.1.1 Proposed Weighted Spatial Prediction 88
6.1.2 Proposed Weighted Temporal Prediction 89
6.1.3 Proposed Angle Based Predictor 90
6.2 Experimental Results 95
6.3 Conclusion 104

7 OPTIMAL QUALITY SCALABLE CODING OF ANIMATED MESHES . . 105
7.1 Introduction 105
7.2 Proposed Quality Scalable Coding: Decoding Order Based 106
7.2.1 Optimization 107
7.2.2 Rate, Distortion and Path Metrics 112
7.2.3 Algorithmic Simplifications 112
7.3 Proposed Quality Scalable Coding: Encoding Order Based 113
7.3.1 Rate, Distortion and Path Metrics 114
7.4 Experimental Results 116
7.4.1 Slope Based Optimization 116
7.4.2 Simple Encoding Parameters - Full Trellis 117
7.4.3 Effect of Rate Intervals 118
7.4.4 Comparison of DOB and EOB 120
7.4.5 Complexity Considerations 122
7.4.6 Visual Comparisons 123
7.5 Conclusion 127
8 CONCLUSION 128
REFERENCES 131
CURRICULUM VITAE 138

LIST OF TABLES

TABLES

Table 4.1 File Sizes for different R when $k = 1$ 28

Table 4.2 Relative L^2 errors for the case in Table 4.1 29

Table 4.3 File Sizes for different k when $R = 6$ 31

Table 4.4 Relative L^2 errors for the case in Table 4.3 31

Table 4.5 An example FEC assignment on an embedded bitstream. There are $N = 5$ packets each composed of $L = 4$ symbols. Therefore there are 4 source segments, S_i, $i = 1, 2, 3, 4$ each of which contains m_i data symbols and f_i FEC symbols where $m_i + f_i = N$. In this example $m_1 = 2$, $f_1 = 3, m_2 = 3$, $f_2 = 2, m_3 = 3$, $f_3 = 2, m_4 = 4$, $f_4 = 1$. Earlier parts of the bitstream are assigned more FEC symbols since they contribute more to overall quality. 41

Table 4.6 An example CPM output with $L_M + 1 = 3$ layers. Pi denotes Packet i generated horizontally while FEC is applied vertically. In this simple example $N = 6$, $k_0 = 2$, $k_1 = 3$ and $k_2 = 4$. 44

Table 4.7 Simulated distortion results of the first scenario for various P_{LR} values. The distortion metric is relative L^2 error in units of 10^{-4}. 51

Table 4.8 Optimization times of different methods. Each method is optimized for $P_{LR} = 4\%$ 55

Table 5.1 The test sequences 72

Table 6.1 The test sequences 95

Table 6.2 Bitrate reductions compared to SPC for cowheavy, chicken crossing and dance sequences 100

Table 6.3 Bitrate reductions compared to SPC for horse gallop, face and jump 101

Table 7.1 Average optimization time during encoding per GOM in minutes. 123

LIST OF FIGURES

FIGURES

Figure 2.1 Single description and multiple description coding. R_0: central or single description bitrate. D_0: central distortion. R^1,R^2: side bitrates. D^1,D^2: side distortions. 7

Figure 3.1 Classification of 3D mesh compression methods 12

Figure 3.2 Generation of embedded bitstream from PGC coder. The bitstream starts with compressed coarsest level connectivity (*C*) as it is the most important part on which the whole mesh connectivity depends. The next part of the bitstream is a predetermined number of bit-planes (5 in the figure) of the coarsest level geometry (*G1G2G3G4G5*) since wavelet coefficients would have no use without coarsest level geometry. Remaining part of the bitstream consists of the output bitstream of SPIHT for different quantization levels (*S1S2S3..*) and after each quantization level, refinement bit-planes of coarsest level geometry (*G6G7..*) are inserted for improved progressivity. 13

Figure 3.3 Hierarchical temporal prediction structure 20

Figure 3.4 Prediction of a vertex in SPC. The vertex to be predicted is denoted by $\mathbf{v}_c^c$. . 21

Figure 4.1 Encoder block diagram 27

Figure 4.2 Decoder block diagram 28

Figure 4.3 Test data (a) *Bunny* model; (b) *Venus head* model. 29

Figure 4.4 (a) File Sizes (b) Relative L^2 errors for different R when $k = 1$ 30

Figure 4.5 (a) File Sizes (b) Relative L^2 errors for different k when $R = 6$ 30

Figure 4.6 Reconstructed models for ML(6,1): (a)-(b) Reconstruction from one description (*Side1* and *Side2*) (c) Reconstruction from both of the descriptions (*Central*). 30

Figure 4.7 TM-MDC encoder scheme. 32

Figure 4.8 Reconstruction of model *Bunny* from one description among four descriptions for different types of tree grouping. (a) Spatially disperse grouping; PSNR= 50.73 dB. (b) Spatially close grouping, group size is 10; PSNR= 48.51 dB. 33

Figure 4.9 Comparison between the Weibull model (10 points) and operational D-R curve (L^2) for *Bunny* model. (a) Relative L^2 error; (b) PSNR. 35

Figure 4.10 Reconstruction of *Bunny* model from different number of received descriptions. The results are given for bit allocations for different packet loss rates (PLR). The redundancy ρ is given in brackets. (a) Relative L^2 error; (b) PSNR. 36

Figure 4.11 Model *Bunny* encoded into 8 descriptions at total 25944 Bytes. Reconstructed from different number of descriptions. Compared to unprotected SPIHT. . 36

Figure 4.12 The comparison of network performance of the proposed TM-MDC with a simple MDC scheme and unprotected SPIHT. (a) Relative L^2 error; (b) PSNR. . . 37

Figure 4.13 Reconstruction of the *Bunny* model from (a) one description (48.36 dB), (b) two descriptions (63.60 dB), (c) three descriptions (71.44 dB), (d) four descriptions (74.33 dB). 38

Figure 4.14 Reconstruction of *Venus head* model from (a) one description (53.97 dB), (b) two descriptions (65.18 dB), (c) three descriptions (72.51 dB), (d) four descriptions (77.08 dB). 39

Figure 4.15 Reconstruction from different number of descriptions (PSNR) for *Bunny* model. (a) Relative L^2 error; (b) PSNR. 42

Figure 4.16 The comparison of the MD-FEC with TM-MDC coder from [1]. (a) Relative L^2 error; (b) PSNR. 43

Figure 4.17 Two state Markov channel model. 46

Figure 4.18 Effect of k step size on quality: Simulated PSNR vs k step size for *Bunny* model optimized for P_{LR} = 2%, 4% and 10% and coded at 3.5 bpv. 48

Figure 4.19 Effect of k step size on complexity: Optimization time vs k step size for *Bunny* model optimized for P_{LR} = 4% and coded at 3.5 bpv. 49

Figure 4.20 Comparison of CPM based methods for Bunny model in terms of simulated distortions for various P_{LR}'s. 49

Figure 4.21 Comparison of using original D-R curve and using modeled D-R curve during optimization for *Bunny* model in terms of simulated distortions for various P_{LR}'s. 51

Figure 4.22 P_{LR} vs Simulated distortion in PSNR scale for *Bunny* model coded at 3.5 bpv. 52

Figure 4.23 P_{LR} vs Simulated distortion in PSNR scale for *Bunny* model coded at 1.2 bpv. 52

Figure 4.24 P_{LR} vs Simulated distortion in PSNR scale for *Venus head* model coded at 4 bpv. 53

Figure 4.25 *Bunny* model is coded at 3.5 bpv and FEC assignment is optimized with respect to three different P_{LR}'s for *PGCStankovic* and *AlRegib kStep = 5* methods. Performance of the three different assignments for various P_{LR}'s in terms of simulated distortion in PSNR scale. 54

Figure 4.26 Expected reconstructions of the *Bunny* model, left column by *PGCStankovic* method and right column by *AlRegib kStep=5* method. (a)-(b): P_{LR} = 2%, (c)-(d):P_{LR} = 10%, (e)-(f):P_{LR} = 20%. PSNR values: (a) 72.18 dB (b) 62.44 dB (c) 70.52 dB (d) 61.57 dB (e) 69.09 dB (f) 58.61 dB 56

Figure 4.27 Expected reconstructions of the *Venus head* model, left column by *PGCStankovic* method and right column by *AlRegib kStep=5* method. (a)-(b): P_{LR} = 2%, (c)-(d):P_{LR} = 10%, (e)-(f):P_{LR} = 20%. PSNR values: (a) 76.95 dB (b) 67.79 dB (c) 75.04 dB (d) 65.19 dB (e) 74.24 dB (f) 65.19 dB 57

Figure 5.1 Temporal subsampling based MDC. The example sequence is decomposed into three temporal layers and three spatial layers. One spatial layer from the other description is duplicated. 66

Figure 5.2 Layer duplication based MDC: three layer ordering examples 69

Figure 5.3 (a)Vertex Partitioning based MDC: RRD curves for varying number of duplicated layers (b) All achievable RRD points and their convex hull 74

Figure 5.4 Comparison of mismatch-free and mismatch-allowed vertex partitioning based MDC 74

Figure 5.5 Temporal Subsampling based MDC: (a) RRD curves for different prediction methods (b) Zoomed view 75

Figure 5.6 Layer Duplication based MDC: All redundancy-side error pairs and the convex hull 76

Figure 5.7 RRD performance comparison of the MDC methods 78

Figure 5.8 PSNR values of side reconstruction of each frame for horse gallop sequence MD coded at (a) %25 redundancy, (b) %45 redundancy 79

Figure 5.9 Central and side reconstructions of frames 1 and 2 of horse gallop sequence MD coded at 25% redundancy 81

Figure 5.10 Central and side reconstructions of frames 1 and 2 of horse gallop sequence MD coded at 45% redundancy 82

Figure 5.11 Central and side reconstructions of frames 259 and 260 of chicken sequence MD coded at 44% redundancy 83

Figure 5.12 Visualization of errors between central and side reconstructions for horse-gallop and chicken crossing MD coded at 25% and 44% redundancy, respectively. Deviation from red indicates increasing error. 84

Figure 6.1 One incident triangle in previously encoded and current frame. Note that $\mathbf{v}_0^{p,c}$, $\mathbf{v}_1^{p,c}$, $\mathbf{v}_s^{p,c}$ and $\mathbf{v}_{cr}^{p,c}$ lie on the same plane $\mathbf{P_s^{p,c}}$. 91

Figure 6.2 Angle based prediction (a) Angles same (b) Angle differences same (c) Local displacement between $\mathbf{v}_{cr}$ and $\mathbf{v}_s$ same 94

Figure 6.3 The compression performance of angle based prediction methods as a function of multiplicative constant of standard deviation in outlier removal process as given in the legend. Q=8 bits: 96

Figure 6.4 The compression performance of angle based prediction methods as a function of multiplicative constant of standard deviation in outlier removal process as given in the legend. Q=12 bits 96

Figure 6.5 Change in prediction error compared to SPC per frame 97

Figure 6.6 Percentage bitrate reduction compared to SPC per frame 98

Figure 6.7 RD comparison of the SPC and best performing proposed method for each quantization level. 103

Figure 7.1 Illustration of different fixed bitplane orderings on an example case 107

Figure 7.2 Rate-distortion curves of fixed bitplane ordering methods. 108

Figure 7.3 Illustration of first branches of the trellis structure used in optimization (a) Initial state (b) First branches (c) Some of the next possible branches (d) Elimination of paths when two paths end up at the same state (same RD point) 110

Figure 7.4 Comparison of slope based optimizations with fixed ordering policies . . . 117

Figure 7.5 Comparison of full trellis and slope based optimization for simple coding parameters 118

Figure 7.6 Comparison of different RInt and MaxP parameters for the rate interval based optimization method. 119

Figure 7.7 Comparison of best fixed ordering policies, slope based and rate interval based optimizations. 119

Figure 7.8 Comparison of proposed DOB and EOB methods for cowheavy model. . . 120

Figure 7.9 Comparison of proposed DOB and EOB methods for chicken crossing model. 121

Figure 7.10 Comparison of proposed DOB and EOB methods for face model. 121

Figure 7.11 Comparison of visual reconstructions of frames 4,6,7,8 for several methods decoded at 8 bpvf. 125

Figure 7.12 Comparison of visual reconstructions of frames 4,6,7,8 for several methods decoded at 12 bpvf. 126

LIST OF ABBREVIATIONS

3D	3 Dimensional
3B	3 Boyutlu
BFOS	Breiman, Friedman, Olshen, and Stone
BP	Bit Plane
bpvf	bits per vertex per frame
CABAC	Context-Adaptive Binary Adaptive Coding
CPM	Compressed Progressive Meshes
DCT	Discrete Cosine Transform
DOB	Decoding Order Based
D-R	Distortion-Rate
EOB	Encoding Order Based
FAMC	Frame-based Animated Mesh Compression
FEC	Forward Error Correction
GOM	Group Of Meshes
KG	Karni and Gotsman
LD	Layer Duplication
LOD	Level of Detail
LPC	Linear Prediction Coding
LS	Least Squares
MDC	Multiple description coding
MDSQ	Multiple Description Scalar Quantization
MPEG	Moving Picture Experts Group
MPT	Multi Path Transmission
MSE	Mean Squared Error
PCA	Principal Component Analysis
PGC	Progressive Geometry Compression
PLR	Packet Loss Rate
PSNR	Peak Signal to Noise Ratio
RD	Rate Distortion
RDMC	Reference Dynamic Mesh Coder
RIC	Rotation Invariant Coordinate
RRD	Redundancy Rate Distortion
RS	Reed Solomon
SDC	Single Description Coding
SL	Spatial Layer
SMC	Skinning based Motion Compensation
SPC	Scalable Predictive Coding
SPIHT	Set Partitioning In Hierarchical Trees
SVD	Singular Value Decomposition
TG	Touma and Gotsman
TL	Temporal Layer
TS	Temporal Subsampling
TM-MDC	Tree-based Mesh MDC
VP_{MA}	Vertex Partitioning Mismatch Allowed
VP_{MF}	Vertex Partitioning Mismatch Free

CHAPTER 1

INTRODUCTION

With an increasing demand for visualizing and simulating three dimensional (3D) objects in applications such as video gaming, engineering design, architectural walk-through, virtual reality, e-commerce, scientific visualization and 3DTV, it is very important to represent the 3D data efficiently. The most common representations for the 3D data are volumetric data, parametric surfaces and 3D meshes. Among the representations, the triangular 3D meshes which model the surface of the 3D objects by combination of triangles are very effective and widely used. Consequently, the main focus of the thesis is the 3D mesh structure and throughout the text, 3D model and 3D mesh are used interchangeably.

Typically, 3D mesh data consist of geometry and connectivity data. While the geometry data specifies 3D coordinates of vertices, connectivity data describes the adjacency information between vertices. A single 3D mesh whose geometry does not change with time is also called a *static mesh* whereas a series of static meshes is called an *animated 3D mesh* (or dynamic 3D mesh/3D mesh sequence). More information about the 3D mesh data structure is provided in Section 2.1.

To maintain a convincing level of realism, many applications require highly detailed complex models represented by 3D meshes consisting of huge number of triangles. This requirement results in several challenges for storage and transmission of the models. Due to storage space and transmission bandwidth limitations, it is needed to efficiently compress the mesh data. The aim of compression is to reduce the number of bits required to represent the data at a certain quality level. An important subclass of compression is scalable coding where the data is compressed such that predefined subsets of the compressed bitstream can be used to reconstruct model with reduced resolution and/or quality. Another important topic is the

transmission of 3D meshes over error-prone channels where packets may be lost or delayed because of congestion, buffer overflow, uncorrectable bit errors or misrouting.

This thesis is about robust transmission of both static and animated meshes and related issues. In particular, the investigated issues are scalable coding, multiple description coding and error resilient coding. Major contributions of this thesis to the existing body of knowledge can be summarized as follows:

Multiple Description Coding of Static Meshes [2, 1, 3, 4, 5, 6] Three methods for multiple description coding of static meshes are introduced together with optimization and complexity reduction tools. An optimal loss resilient transmission system based on forward error correction is developed. In-depth analysis of performance comparison with the state of the art is performend and significant improvement is reported.

Multiple Description Coding of Animated Meshes [7, 8, 9] Three methods for MDC of animated meshes are proposed, which are the first works in the literature. The methods are deeply analyzed and compared with respect to performance in varying redundancy conditions and flexibility in redundancy allocation.

Improved Prediction Methods for Scalable Predictive Animated Mesh Compression [10, 11] For the animated mesh coding structures that are scalable and predictively coded, several improvements are proposed in the prediction part. Experimental results indicate that up to 30% percent bitrate reduction can be achieved.

Optimal Quality Scalable Coding of Animated Meshes [12] Two methods for extending the state of the art scalable predictive animated mesh coding structure, which only support resolution scalability, to support quality scalability are proposed. An optimization framework is introduced with possible simplifications to trade off between compression performance and complexity. Experimental results show that optimization framework achieves quality scalability with significantly better compression performance than state of the art without optimization.

1.1 Scope and Outline of the Thesis

In Chapter 2, the necessary background information about the concepts related to the thesis is provided. Initially, the mesh data on which the proposed methods are based is explained. Then the error resilient coding concept is introduced as two chapters focus on this subject. In particular, MDC and packet loss resilient streaming approaches are explained as the error resilient coding means. The following concept is scalable coding, which is also an important subject in most of the proposed works. Finally, forward error correction mechanism is introduced as it is an important tool in error resilient coding.

In Chapter 3, mesh coding for both static and animated meshes is introduced. The chapter begins with a literature review followed by explaining two particular coding methods in more detail as these methods are closely related to the proposed algorithms. The chapter ends with the presentation of error metrics used for static and animated meshes during the experiments.

In Chapter 4, first the details of three propose methods for MDC of static meshes are provided. Then one of the MDC methods, MD-FEC, is proposed to be used for packet loss resilient streaming purposes. Experimental simulations are provided for MDC results and packet loss resilient streaming results separately.

In Chapter 5, the details of three proposed methods for MDC of animated meshes are presented. Then the experimental results including objective results and visual reconstructions are provided.

In Chapter 6, the proposed prediction enhancement modules for animated mesh compression are presented followed by experimental results which contain percentage bitrate reduction for each combination of modules.

In Chapter 7, the details of how quality scalability can achieved from the state of the art scalable predictive coder in two ways and the optimization framework are introduced. Experiments are conducted to compare the performance of two proposed schemes and non-optimized scalable coding.

Finally, we conclude in Chapter 8.

CHAPTER 2

BACKGROUND

2.1 3D Mesh Data

A mesh is a graphics object composed of, typically, triangles or quadrilaterals that share vertexes and edges, and thus can be transmitted in a compact format to a graphics accelerator. The basic elements in a mesh and related definitions are as follows:

Vertex: Single point in the mesh.

Edge: Line segment whose end points are vertices. Degree of a vertex is defined as the number of edges connected to it.

Face: Convex polygon that live in 3D, bounded by edges. Among different polygons, triangle faces are the most popular due to

- Simplicity in storage
- Possibility of fanning convex polygons into triangles
- Wide usage of triangles by the 3D graphics APIs such as OpenGL and Direct3D.

Polygonal mesh (or polymesh): A finite collection of vertices, edges, and faces satisfying following conditions:

- Each vertex must be shared by at least one edge. (No isolated vertices are allowed.)
- Each edge must be shared by at least one face. (No isolated edges or polylines allowed.)

- If two faces intersect, the vertex or edge of intersection must be a component in the mesh. (No interpenetration of faces is allowed.)

Triangular mesh: Polygonal mesh whose faces are triangles.

There are three types of information in a mesh:

Geometry: Concerning with the embedding in a metric space, e.g. Vertex, normal coordinates.

Connectivity or Topology: Providing the connecting structure of the mesh, the adjacency information between vertices

Pictoric information: Providing additional information useful for visualizing the model (e.g. color, textures, or scalar field values).

In this thesis, we are concerned with the coding and transmission of geometry (vertex coordinates in particular) and connectivity information of 3D triangle meshes. The pictoric information is usually embedded with each vertex location and treated in the same way with the geometry information. Therefore, we do not explicitly deal with the pictoric information.

We further classify the 3D meshes into two subcategories: Static meshes and animated meshes. A single 3D mesh whose geometry does not change with time is also called a *static mesh* whereas a series of static meshes is called an *animated 3D mesh* (or dynamic 3D mesh/3D mesh sequence). Each static mesh in the sequence is called a mesh frame or simply frame which corresponds to a time instant. An important subclass of animated meshes, which is also subject of this work, is the time consistent animated meshes where each mesh frame shares the same connectivity.

2.2 Error Resilient Coding

When transmitting multimedia data, usually the transmitting medium or the channel (e.g. wireless/wired internet, broadcast service, multicast network, peer-to-peer networks, wireless transmission) is lossy, i.e. the data transmitted and received are not the same. Although there are inherent error correction mechanisms in most of the transmission protocols, the losses

may still occur due to higher error rate than error correction capability, network congestion or other reasons. Some systems (e.g. TCP/IP) employ feedback channel to retransmit the lost parts of the data. However, this approach has the disadvantage that delays in reception may occur, which is usually undesirable for the multimedia data where real time reception is an important issue.

Feedback/retransmission based systems can be considered as post-processing based error resilient schemes. Another approach, which is also used in this thesis, is pre-processing based approaches. In these approaches, the data is processed before the transmission, usually resulting in an increase in the bitrate, so that the resultant bitstream is more resilient to losses. In error resilient coding, a pre-processing based error resiliency approach, the resiliency is achieved during the compression stage by joint compression and resiliency operations. In this sense, the error resilient coding schemes are often called as joint source channel coding.

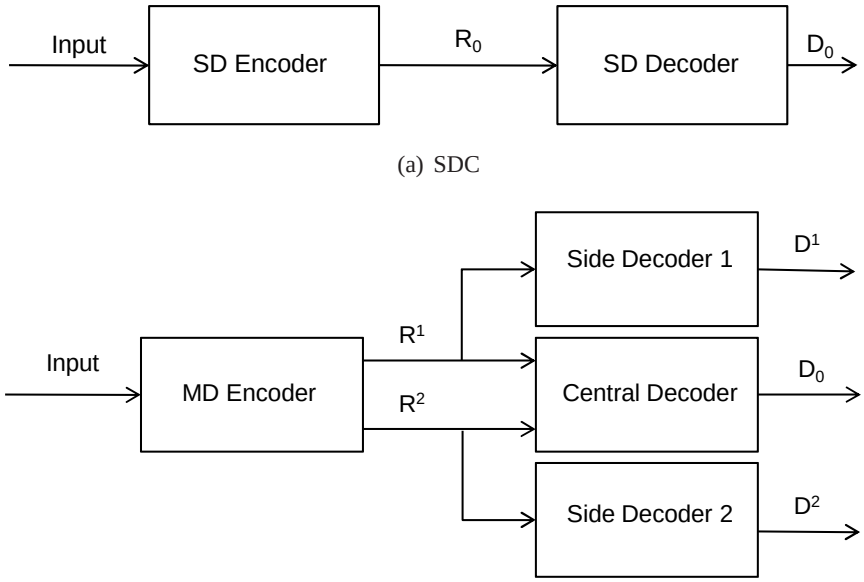
In this thesis, we subdivide the general error resilient coding paradigm into parts as *Multiple Description Coding* (MDC) and *Packet Loss Resilient Streaming* and treat the two problems separately, as they are the main focuses in our works. The details of the two concepts used in this thesis are as follows.

2.2.1 Multiple Description Coding

MDC has emerged as an efficient method for error resilient coding of multimedia data. The idea of MDC is coding the source into multiple independent bitstreams or so-called descriptions instead of a single bitstream/description. The independency implies that each description can be decoded on its own without the need of any other descriptions. This property gives power to MDC in lossy scenarios.

Figure 2.1 illustrates the Single Description Coding (SDC) and MDC cases. In SDC, the input is encoded at one target bitrate (R_0), resulting in a distortion of D_0. In the most common MDC setting, the MDC encoder generates two descriptions having equal bitrates (R^1 and R^2) and importance. The descriptions are packetized independently and sent over either same or separate channels. As long as the two descriptions are not lost simultaneously, the MDC decoder can make a reconstruction. If only one of the descriptions is received, the MDC decoder decodes the received description using the *side decoder* and reconstructs the data with

a low but acceptable quality. The resulting distortion of the data is called the *side distortion* (D^1 or D^2). If all of the descriptions are received successfully, the MDC decoder decodes the descriptions together using the *central decoder* with a higher quality. The resulting distortion is called the *central distortion* (D_0). In the more general settings, there can be more than two descriptions not necessarily having identical bitrates. In this work, we deal with MDC scenarios with the descriptions having equal bitrates.



(a) SDC

(b) MDC

Figure 2.1: Single description and multiple description coding. R_0: central or single description bitrate. D_0: central distortion. R^1,R^2: side bitrates. D^1,D^2: side distortions.

In-depth analysis of MDC with different applications can be found in [13, 14]. Here we provide several important applications of MDC. An important application of MDC is multimedia transmission over lossy links. Providing adequate quality without the need of retransmission of packets, MDC can be very useful in real-time applications and simplifies the network design. It is also useful for Multi Path Transmission (MPT) scenarios where the data is sent over multiple independent paths instead of a single path. In this way, traffic dispersion and load balancing can be achieved in the network, which can effectively relieve congestion at hotspots and increase overall network utilization.

Another application where MDC is suitable is distributed storage. Distributed storage is common in the use of edge servers for popular content in databases of encoded media data. If identical data is stored at the servers, reception of multiple copies does not bring any advan-

tage. However, if the distributed storage is performed with multiple descriptions, then a user would have fast access to the local image copies and in order to achieve higher quality, more remote copies could be retrieved and combined with the local copy.

MDC can also be utilized in P2P networks where the users help each other to download/stream multimedia files. If the files are cut into pieces blindly, then in case of partial reception of the pieces (e.g. due to unfinished download, missing pieces in the network or downling/uplink capacity mismatch during live streaming), it would not be possible to achieve a useful playout. However, if the pieces are generated with multiple descriptions, then it is still possible to obtain a playout at a reduced quality in case of partial reception.

2.2.1.1 Redundancy Allocation

All the mentioned useful properties of MDC come at a price: Extra/redundant bits need to be spent compared to conventional single description coding. Therefore, the performance of an MD coder depends on how efficient the redundancy is allocated.

One of the most common ways to measure the performance of an MDC scheme is the Redundancy-Rate-Distortion (RRD) curve [15]. The RRD curve shows the effects of redundant bits on the average side distortion for a given central distortion. Mathematically speaking, for the two descriptions case without loss of generality, R_0 and D_0 denote the bitrate and distortion (central bitrate and distortion) that result when the data is coded with single description, R^1 and R^2 denote the bitrates of descriptions 1 and 2 and $D^{1(2)}$ denotes the distortion when only description 1(2) is received (side distortion) as depicted in Figure 2.1. Also note that, receiving both of the descriptions result in the same or very similar central distortion D_0 as the single description case. Then the redundancy, ρ, as a percentage of single description bitrate can be expressed as $\rho = (R^1 + R^2 - R_0)/R_0$ and the average side distortion, D_1 can be calculated as $D_1 = (D^1 + D^2)/2$. As a result, the RRD curve shows the variation of D_1 with respect to different ρ values for a given D_0 value.

2.2.1.2 Mismatch Control

Efficient coders make use of predictive coding extensively. The mismatch condition occurs when the encoder uses a signal for prediction that is unavailable in decoder due to loss of

descriptions. For the time-varying multimedia data like video or dynamic meshes, it is common to exploit inter-frame temporal redundancies. For example, let frame F_i be encoded by predicting from frame F_{i-1} and assume that an error occurs in F_{i-1}. The error affects both F_{i-1} and F_i. Similarly let frame F_{i+1} be encoded by predicting from F_i. Since F_i is not reconstructed perfectly in the decoder, the reconstruction of F_{i+1} is affected as well. As a result, the mismatch also causes the propagation of error throughout the time. Therefore, the mismatch needs to be controlled efficiently in an MDC scheme. Because of the aforementioned temporal dependency, the mismatch occurs in time-varying data (like video, dynamic meshes) more frequently than static data (like image, static meshes).

2.2.2 Packet Loss Resilient Streaming

In a packet loss resilient streaming scheme, the source is encoded and controlled redundancy is added to the source bitstream. Packets are generated from this bitstream and typically the packet sizes are much less than a size of description in an MDC scheme. Moreover, the packets in this scenario are not necessarily independent, i.e losing some of the packets may cause remaining packets to be useless. But the main aim is to optimize the redundancy and packetization so that expected distortion in the receiver is minimized.

2.3 Scalable Coding

An important class of multimedia compression techniques is scalable coding. In scalable coding, the data is compressed such that, decoding a subset of resultant bitstream allows reconstruction of the data at a reduced fidelity. The most common scalability types are spatial/temporal scalability where a subset of the bitstream results in a data with reduced resolution and quality scalability where a subset of the bitstream results in an increased distortion. In an ideal quality scalable coder, it is desired for every bitrate point obtained by a subset of the bitstream to achieve the same distortion with the non-scalable coding at that bitrate. Several applications benefitting from scalable coding are error resilient coding, rate control and transmission with heterogenous clients.

2.4 Forward Error Correction

The Forward Error Correction (FEC) is used very commonly for error resilient coding purposes. In our works, we also make use of FEC as we will see in Chapter 4. In particular, Reed Solomon (RS) codes are perfectly suited for error protection against packet losses because they are non-trivial maximum distance separable codes, i.e., there exist no other codes that can reconstruct erased symbols from a smaller fraction of received code symbols [16].

An $RS(n,k)$ code of length n and dimension k encodes k information symbols containing m bits each into a codeword of n such symbols. The codeword length n is restricted by $n \leq 2^m - 1$. In our works m is chosen as 8 so that the symbols are bytes and n is chosen as 100.

Among n sent packets, error-free reception of any subset of k packets are enough to recover original information by erasure decoding since the packets are numbered and the locations of lost packets are known.

CHAPTER 3

3D MESH CODING

In this chapter, we present a literature review on both static and animated 3D mesh coding, followed by the error metrics to measure fidelity of a mesh. In the following chapters where we describe the proposed works, this chapter is referenced for the mesh coding related concepts.

3.1 Literature Review on Static 3D Mesh Coding

The field of static 3D mesh compression is very mature and numerous works about compression of both geometry and connectivity of 3D mesh data exist in the literature. The reader is referred to [17] and [18] for detailed surveys. Since we do not propose any improvement on static 3D mesh compression, we provide a brief classification and give details of specific methods which are important for error resilient coding.

Performance of loss resilient coding techniques is highly correlated with the compression techniques on which they are based. 3D mesh compression techniques can be classified into two categories: Single-rate compression and Progressive compression. In single-rate compression, the aim is to compress the mesh as much as possible. The single-rate compressed mesh can only be decompressed if the whole compressed bitstream is available, i.e. no intermediate reconstruction is possible with fewer bits. Progressive compression is more suitable for transmission purposes in which some parts of the compressed bitstream can be lost or erroneous. By progressive compression, the mesh is represented by different levels of detail (LODs) having different sizes. Progressive compression techniques can further be classified into two categories: connectivity driven and geometry driven techniques. In connectivity driven progressive mesh compression schemes, the compact representation of connectivity

data is given a priority and geometry coding is driven by connectivity coding. On the other hand, in geometry driven compression, data is compressed with little reference to connectivity data, for example even the mesh connectivity can be changed in favor of a better compression of geometry data. It is shown in [18] that better compression ratios can be obtained by geometry driven progressive compression methods. Figure 3.1 summarizes the classification of 3D mesh coding techniques.

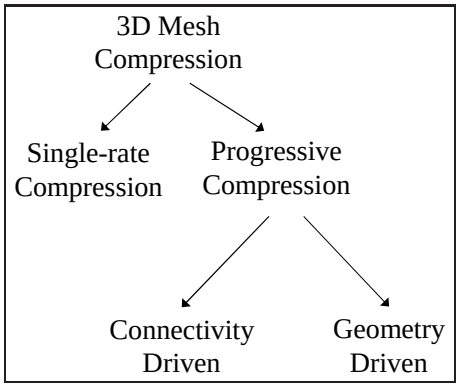


Figure 3.1: Classification of 3D mesh compression methods

Next, we present two representative progressive static mesh compression algorithms each of which belongs to a progressive compression category described above. The progressive compression algorithms and the corresponding categories are as follows:

- Geometry driven - geometry encoded progressively: Wavelet based scalable coding, in particular *Progressive Geometry Compression* (PGC) scheme [19].
- Connectivity driven - connectivity encoded progressively: *Compressed Progressive Meshes* (CPM) scheme [20].

3.1.1 Wavelet Based Scalable Static 3D Mesh Coding and PGC

Wavelet based mesh coding techniques belong to the geometry driven progressive mesh coding category. In the literature, there exist a number of efficient wavelet based compression schemes such as [19], [21], [22], [23], [24], [25] and the wavelet subdivision surfaces tool of MPEG-4's Animation Framework eXtension (AFX) [26],[27].

PGC is a progressive compression scheme for arbitrary topology, highly detailed and densely

sampled meshes arising from geometry scanning. The method is based on smooth semi-regular meshes, i.e., meshes built by successive triangle quadrisection starting from a coarse irregular mesh. Therefore the original model in PGC should be remeshed [28] to have a semi-regular structure so that a subdivision based wavelet transform can be applied. After the remeshing, the resultant model with full resolution may consist of 3-4 times of the original number of vertices. However, despite this increase, better compression performance is achieved. Resulting semi-regular mesh undergoes a Loop-based [29] or butterfly-based [30] wavelet decomposition to produce a coarsest level mesh and wavelet coefficients [19]. Since the coarsest level connectivity is irregular, it is coded by Touma and Gotsman's (TG)[31] single-rate coder, which is one of the very efficient single-rate coders reported in the literature. Wavelet coefficients are coded with (SPIHT) algorithm [32]. For improved progressivity, a predetermined number of bit-planes of the coarsest level geometry can be transmitted initially with the coarsest level connectivity. The remaining refinement bit-planes can be transmitted as the SPIHT coder descends a given bit-plane of wavelet coefficients [19]. As a result, an embedded bitstream is generated as illustrated in Fig. 3.2.

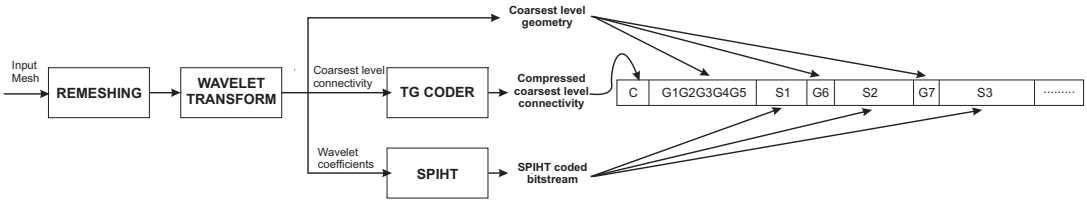


Figure 3.2: Generation of embedded bitstream from PGC coder. The bitstream starts with compressed coarsest level connectivity (*C*) as it is the most important part on which the whole mesh connectivity depends. The next part of the bitstream is a predetermined number of bit-planes (5 in the figure) of the coarsest level geometry (*G1G2G3G4G5*) since wavelet coefficients would have no use without coarsest level geometry. Remaining part of the bitstream consists of the output bitstream of SPIHT for different quantization levels (*S1S2S3*..) and after each quantization level, refinement bit-planes of coarsest level geometry (*G6G7*..) are inserted for improved progressivity.

3.1.2 Compressed Progressive Meshes (CPM)

The CPM method is a connectivity driven progressive mesh compression scheme. Therefore if the whole bitstream is received successfully, then the original connectivity of the model can be reconstructed. The encoder starts with the original mesh and generates meshes at different LODs iteratively. During each iteration a simplified and coarser LOD of the model

is generated from the present LOD of the model.

The basic operation for coarsening the present LOD is the edge collapse operation. This operation combines two vertices of an edge into one vertex by collapsing the edge. This results in a decrease in the number of triangles by two. The destroyed triangles form the cut-edges that are incident on the newly generated vertex. In each iteration, a certain subset of edges are chosen to be collapsed. The encoder decides to stop generating coarser LODs at a point and ends up with the simplest base mesh and M LODs.

The decoder performs just in the reverse direction of encoder. It starts with the base mesh and constructs finer LODs in each iteration. The basic operation for this construction is the vertex split operation. This operation produces two new vertices from the vertex that was generated by collapsing an edge in the encoder. The locations of new vertices are predicted and displacement errors are corrected. The details of the levels increase in each iteration as new triangles are generated from the cut-edges.

All the operations needed for decoder to decode a finer level from the present level in an iteration is coded as a batch in the encoder. The encoded batch bitstream is composed of 1) *Collapse Status*, one bit to specify whether a vertex is to be splitted or not 2) *Cut Edges*, the indices of cut-edges for the vertices to be splitted and 3) *Position Error*, quantized and entropy coded difference in geometric coordinates between the collapsed vertex and the predicted vertex locations. For quantizing the prediction error, a predetermined number of bits is used. Compressing the base mesh with a single-rate coder, the final bitstream of the CPM algorithm is generated by the concatenation of compressed bitstream of base mesh (base layer) of size $R^{(0)}$ and the M batches (M enhancement layers) of size $R^{(i)}$, $i = 1, ..., M$. As it can be noticed, each batch contains information regarding to both connectivity and geometry. Therefore, the connectivity is encoded progressively in this scheme.

3.2 Literature Review on Animated 3D Mesh Coding

Recently, compression of animated 3D meshes (or dynamic 3D meshes/3D mesh sequences) represented as series of static meshes with same connectivity has attracted great attention. Each static mesh in the sequence is called a mesh frame or simply frame which corresponds to a time instant. In the literature, the reported works about the compression of animated

meshes can be broadly grouped into *segmentation based methods*, *wavelet based methods*, *Principal Component Analysis (PCA) based methods* and *prediction based methods*.

Segmentation based methods: Lengyel [33] presented the pioneering work for mesh sequence compression. In this work, the input mesh sequence is initially segmented and the motion of each segment is approximated by an affine transform. After the transform, the approximation residuals and the transform parameters are encoded. Another segmentation based compression method was proposed by Ahn et al. [34] where the approximation residuals are encoded using Discrete Cosine Transform (DCT). Zhang and Owen [35, 36] proposed to represent mesh sequences with a reduced set of motion vectors generated for each frame by analyzing the motion between consecutive frames where the motion is represented with an octree. Motion for vertices within a cell is approximated using tri-linear interpolation of the motion vectors. Mueller et al. [37, 38] presented another octree-based approach and introduced RD optimization which includes different prediction modes, namely mean replacement, trilinear interpolation, and direct encoding as well as an RD cost computation that controls the mode selection across all possible spatial partitions to find the clustering structure together with the associated prediction modes. Mamou et al. [39] introduced an approach based on a skinning animation technique with improved clustering and motion compensation. After segmenting into patches, corresponding affine transforms approximating the frame-wise motion of each patch are obtained. Then frame-wise motion of each vertex is represented by weighting previous affine transforms. Subsequently, motion compensation is applied followed by DCT of residual errors.

Wavelet based methods: Guskov and Khodakovsky [40] proposed the first wavelet-based coding approach in which the input mesh sequence is transformed with an anisotropic wavelet transform running on top of a progressive mesh hierarchy. The difference of wavelet coefficients between adjacent frames are progressively encoded. Payan and Antonini [41, 42] used a temporal wavelet filtering to exploit temporal coherence. The resulting wavelet coefficients are quantized optimally by a bit allocation process. Cho et al. [43] proposed a similar wavelet based coding algorithm which also supports lossless compression. Boulfani-Cuisinaud and Antonini [44] proposed a coder based on the clustering of the input mesh geometry into groups of vertices following the same affine motion and employing a scan-based temporal wavelet filtering geometrically compensated.

Briceno et al. [45] presented a new data structure called the geometry video which is based on the geometry image mesh representation [46]. In this approach, every frame of the original mesh sequence is resampled into a geometric image. In this way, the new data structure provides a way to treat an animated mesh as a video sequence. The first frame is intra-coded while subsequent frames are predictively coded with an affine motion compensation. This approach was later enhanced by Mamou et al. [47].

PCA based methods: Alexa and Muller [48] proposed the first scheme that represents 3D animation sequences based on the principal component analysis (PCA). In this scheme, a matrix containing the data of all frames of the animation is formed initially. Then singular value decomposition (SVD) is applied to obtain a basis set consisting of so called eigen-frames and a value for each eigen-frame indicating its importance on the reconstruction quality. In this way, the mesh sequence can also be represented by the set of eigen-frames and the projected values of each frame onto each eigen-frame so called PCA coefficients. The idea behind the compression is to represent each frame with a subset of the eigen-frames that have the highest contribution to the reconstructed mesh quality and only encode the corresponding PCA coefficients. Later, Karni and Gotsman [49] proposed applying second-order linear prediction coding (LPC) to the PCA coefficients in order to further reduce the code size by exploiting the temporal coherence present in the sequence. Sattler et al. [50] introduced clustering for PCA based compression. Instead of analyzing the set of vertices for each frame, the vertex trajectories are analyzed which lead to segmentation of the mesh into meaningful clusters. Then the clustered parts are compressed separately using PCA. Amjoun et al. [51, 52] suggested using trajectory based analysis along with expressing each trajectory in a local coordinate frame defined for each cluster. Additionally, a bit allocation procedure is applied, assigning more bits to cluster where more PCA coefficients are needed to achieve desired precision. For the basis compression, simple direct encoding without prediction and with uniform quantization of the basis matrices is suggested.

Heu et al. [53] proposed a SNR and temporal scalable PCA based coding algorithm using SVD. The basis vectors obtained by SVD to represent a mesh sequence are encoded with a bit plane coder. After analytically deriving the contribution of each bit plane to the reconstruction quality, the bit planes are transmitted in the decreasing order of their amounts of contribution.

Váša and Skala also proposed several compression schemes based on the trajectory space

PCA, suggesting a combination of the PCA step with an EdgeBreaker-like [54] predictor. In their first work named the Coddyac algorithm [55], PCA coefficients are predicted using the parallelogram local predictor and better performance than the clustering-based approaches is achieved. Following the Coddyac, vertex decimation was introduced as a part of the compression [56]. In this work, the encoder can partially steer the decimation process according to the accuracy of the predictors, making the approach well suited for interchanging predictors. The authors also proposed efficient compression of the PCA basis in [57] and significant improvements were reported. Finally, the authors reported further improvements in [58] by proposing two geometric predictors suitable for PCA based compression schemes. Knowledge about the geometrical meaning of the data is exploited by the predictors allowing a more accurate prediction.

Prediction based methods: Yang et al. [59] proposed a mesh sequence coder based on the traversal of frames in the same breadth-first order and two-stage vertex-wise motion vector prediction. In the first stage, motion vector of the vertex is predicted by using the neighborhood. In order to exploit the redundancy in prediction errors, in the second stage, the error vectors are predicted spatially or temporally by using a rate-distortion optimization technique. Ibarria and Rossignac [60] proposed to obtain a vertex encoding order by a deterministic region-growing algorithm. Using the order, each vertex is predicted using three of its neighbors in current and previous frames and prediction residuals are encoded. Two extrapolating space-time predictors are introduced, namely the ELP extension of the Lorenzo predictor, and the Replica predictor. In [61], Amjound and Strasser proposed to use predictive and spatial DCT coding instead of PCA used in their previous works [52], which makes it suited for real-time compression. In [62], the authors introduced a new connectivity-guided predictive scheme for single-rate compression for animated meshes based on a region growing encoding order, and encoding prediction errors in a local coordinate system, which splits into two tangential and one normal components.

Stefanoski and Ostermann [63] presented a predictive compression approach using an angle-preserving predictor. This predictor is based on the assumption that the dihedral angle between neighboring triangles remains invariant from frame to frame. Then, Stefanoski et al. introduced spatial scalability support for predictive coding with a linear predictor in [64] and a non-linear predictor based on local coordinate frames. They proposed a patch-based mesh-simplification algorithm to derive a decomposition of connectivity in spatial layers. Later,

Stefanoski and Ostermann added temporal scalability in [65]. Finally, the authors introduced their codec named SPC (Scalable Predictive Codec) in [66]. In this work, the prediction is performed in the space of rotation-invariant coordinates compensating local rigid motion. The quantized prediction errors are entropy coded by fast binary arithmetic coding approach proposed by Marpe et al. [67]. In addition to the support of spatial and temporal scalability, SPC enables efficient compression together with fast encoding/decoding and low memory requirements.

Apart from the aforementioned works, a standardization process was initiated by the Moving Picture Experts Group (MPEG) for compression of animated meshes and a new standard referred to as Frame-based Animated Mesh Compression (MPEG-4 FAMC) [68, 65] has been adopted. MPEG-4 FAMC is based on the skinning-based approach of Mamou et al. [39] and the spatially scalable predictive approach of Stefanoski et al. [64, 65] where Context-Adaptive Binary Arithmetic Coding (CABAC) [69] is employed as the entropy coder. Several modes are supported in MPEG-4 FAMC, such as download-and-play mode where the only aim is efficient compression and several types of scalable modes.

Among these works, the most notable ones are Váša's PCA based works, MPEG-4 FAMC and Stefanoski's SPC. Váša's works and download mode of MPEG-4 FAMC show very high compression performance. However, neither of the approaches allow scalable frame-wise decoding. They are suited for download the whole sequence and play scenarios. On the other hand, SPC provides these features at a comparable compression performance and a better performance than scalable and streaming modes of MPEG-4 FAMC [66]. Moreover, SPC is a one-pass (no need to examine the whole sequence initially) coder with low complexity and memory requirements.

3.2.1 Scalable Predictive Coding (SPC) - MPEG-4 FAMC

Since we heavily make use of the layered decomposition and prediction structure present in both the SPC algorithm and MPEG-4 FAMC in the proposed works, we present the necessary details of the SPC algorithm in this section. Since the scalable mode of MPEG-4 FAMC and SPC employ the same layered structure and SPC achieves better compression performance [66], we use SPC as the reference scalable coder in the rest of the text. SPC is a layered predictive method which efficiently compresses by creating embedded scalable bit streams

that allow layer-wise decoding and successive reconstruction.

The SPC takes four encoding parameters: 1) S: number of spatial layers, 2) T: number of temporal layers, 3) Q: quantization bin size and 4) temporal prediction mode: Whether short term or long term temporal prediction used. The S parameter is used in spatial layered decomposition and affects vertex encoding order and spatial prediction structures. The T parameter is used in temporal layered decomposition and affects the frame encoding order and inter-frame prediction structure. The Q parameter affects the precision of reconstructed geometry locations.

Before the encoding, the mesh sequence first undergoes a spatial and temporal decomposition generating S spatial layers for each frame and T temporal layers. Spatial layered decomposition is applied on the whole set of vertices. The decomposition does not take into account the vertex locations, instead only the vertex connectivity is used. Since the connectivity is constant throughout the sequence, spatial layered decomposition is performed once at the beginning and all frames use the same decomposition. Therefore, this is a deterministic decomposition for a given 3D mesh.

The aim of the spatial layered decomposition is to generate S disjoint sets of vertices where each set is called a spatial layer. Let $SL_i, i = 0, 1, ..., S-1$ denote i-th spatial layer. Then the union of all layers ($SL_0 \cup SL_1 \cup ... \cup SL_{S-1}$) is equal to the set of all vertices. The 0th spatial layer is also called *spatial base layer* and it is self-reconstructable, i.e. it does not need any other spatial layers to be available. On the other hand, for an arbitrary spatial layer, SL_j, to be reconstructed, all the previous layers, $SL_{j-1}, SL_{j-2}, ..., SL_0$, are required to be available.

The decomposition process in summary is as follows: The input mesh undergoes an iterative mesh simplification process $S-1$ times. At each iteration, a set of vertices is removed with a patch-based vertex removal algorithm. The triangle patches are non-overlapping and the middle vertex of the patch is removed followed by a re-triangulation. Then next iteration chooses new triangle patches for this simplified mesh and removes new vertices. This process continues for a total of $S-1$ times and finally the simplest remaining mesh is our spatial base layer SL_0. Consequently, the vertices removed at each iteration are assigned to a spatial layer in the reverse order. As a result of the decomposition, other than the base layer, every vertex can be predicted by the surrounding neighbor vertices belonging to previous spatial layers.

Temporal layered decomposition is applied on the whole set of frames and after the decomposition, each frame is placed in a temporal layer set out of T temporal layers. Let $TL_i, i = 0, 1, ..., T-1$ represent ith temporal layer. Each temporal layer corresponds to a frame rate level. Therefore TL_0 corresponds to a sequence with frame rate equal to $1/2^{T-1}$ of original frame rate. If we continue with the remaining temporal layers, the frame rate increases by a factor of 2 with each new temporal layer. In this way, a hierarchical temporal structure is obtained.

Having obtained the temporal decomposition, the prediction direction of each frame is determined. Similar to the video coding, I frames make no temporal prediction, P frames predict from only past frames and B frames predict bidirectionally from both past and future frames. Temporal prediction directions and frame encoding order of an example sequence consisting of 9 frames and decomposed into 3 temporal layers is shown in Figure 3.3.

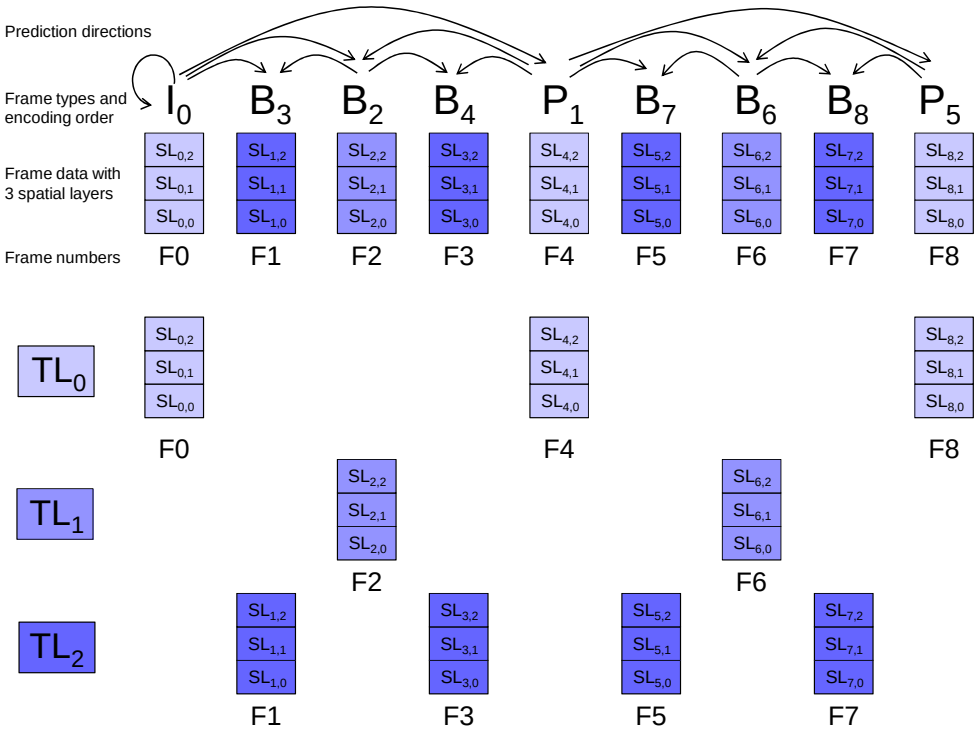


Figure 3.3: Hierarchical temporal prediction structure

After the spatial and temporal layering structures are generated, the encoding process starts. The general idea of the encoder is to process each frame and each vertex in a frame with

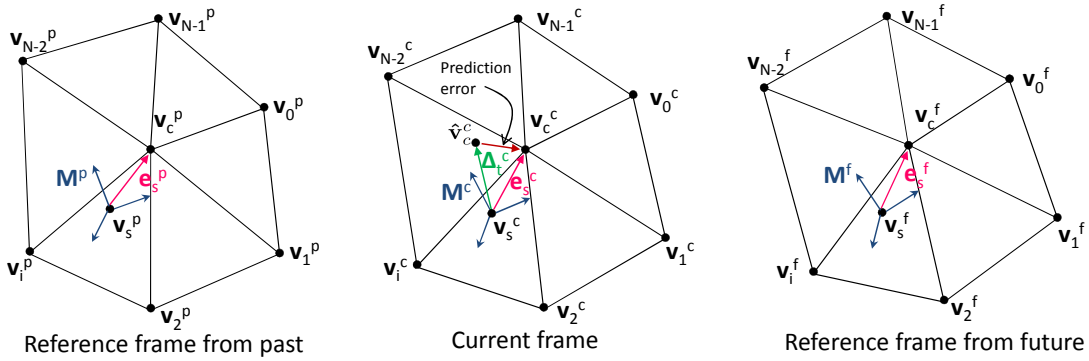


Figure 3.4: Prediction of a vertex in SPC. The vertex to be predicted is denoted by $\mathbf{v}_c^c$.

the order defined by the spatial and temporal decompositions, predict the location of each vertex using previously processed vertices, quantize the prediction residuals and entropy encode. The vertex traversal order during encoding is obtained in spatial layered decomposition process and the frame encoding order is obtained in temporal layered decomposition.

Spatial and temporal neighboring information of an example vertex ($\mathbf{v}_c^c$) to be predicted/encoded is illustrated in Figure 3.4. In the figure and rest of the text, superscripts p, c and f are used for past, current and future frames respectively. $\mathbf{v}_c^c$ is the vertex to be predicted in the current frame and $\mathbf{v}_c^p$ and $\mathbf{v}_c^f$ denote the vertices at the same connectivity location with $\mathbf{v}_c^c$ in the past and future frames respectively. $\mathbf{v}_i^{p,c,f}, i = 0, 1, ..., N-1$ denote the topological neighbors of $\mathbf{v}_c^{p,c,f}$. In this example, $\mathbf{v}_c^c$ belongs to a B frame and makes prediction from one past and one future frame. Note that, both past and future frames are already encoded before the current frame. Future is used in the sense of frame display order. The prediction of $\mathbf{v}_c^c$ consists of spatial prediction followed by a temporal prediction.

The motivation behind the prediction of a vertex is as follows: First the vertex is predicted spatially using its topological neighbor vertices. This prediction only makes use of spatial information and then temporal information is taken into account to increase the accuracy of the spatial prediction to generate the final prediction. Note that for the vertex at the same topological position with the current vertex in the previously encoded frame(s), the same topological neighboring structure also exists. Since the spatial prediction and the original locations of these vertices are already available, the spatial prediction errors in the previous frames are used to refine the current spatial prediction, which significantly increases the prediction accuracy. The details of the prediction process is given in the following paragraphs.

In the SPC, the first step during the prediction in the encoder is calculating a spatial prediction. The spatial prediction of the vertex $\mathbf{v}_c^{p,c,f}$ denoted by $\mathbf{v}_s^{p,c,f}$ is calculated as average of the topological neighbors ($\mathbf{v}_i^{p,c,f}, i = 0, 1, ..., N-1$):

$$\mathbf{v}_s^{p,c,f} = \frac{1}{N} \sum_{i=0}^{N-1} \mathbf{v}_i^{p,c,f} \tag{3.1}$$

where N is the number of topological neighbor vertices. After the spatial prediction, the spatial prediction error of $\mathbf{v}_c^{p,c,f}$ denoted by $\mathbf{e}_s^{p,c,f} = \mathbf{v}_c^{p,c,f} - \mathbf{v}_s^{p,c,f}$ is obtained as illustrated in Figure 3.4.

The spatial prediction errors are not directly used for encoding except for the I frames. For P and B frames, the spatial prediction is followed by a temporal prediction procedure which aims to refine the spatial prediction error ($\mathbf{e}_s^c$) in order to obtain the final prediction of $\mathbf{v}_c^c$ denoted by $\hat{\mathbf{v}}_c^c$. $\hat{\mathbf{v}}_c^c$ is calculated as:

$$\hat{\mathbf{v}}_c^c = \mathbf{v}_s^c + \mathbf{\Delta}_t^c \tag{3.2}$$

where $\mathbf{\Delta}_t^c$ can be regarded as the temporal prediction or a spatial prediction correction/refinement term coming from previously encoded frames. $\mathbf{\Delta}_t^c$ is actually a prediction of $\mathbf{e}_s^c$ and calculated as

$$\mathbf{\Delta}_t^c = \frac{\tilde{\mathbf{e}}_s^p + \tilde{\mathbf{e}}_s^f}{2} \tag{3.3}$$

where $\tilde{\mathbf{e}}_s^p$ and $\tilde{\mathbf{e}}_s^f$ are the spatial prediction correction terms corresponding to past and future frames respectively and calculated as

$$\tilde{\mathbf{e}}_s^{p,f} = (M^c)^{-1} M^{p,f} \mathbf{e}_s^{p,f} \tag{3.4}$$

where $M^{p,c,f}$ denotes the local coordinate frame for $\mathbf{v}_c^{p,c,f}$ which is a Rotation-Invariant Coordinate (RIC) system defined at $\mathbf{v}_s^{p,c,f}$ [66]. In other words, $M^{p,f}\mathbf{e}_s^{p,f}$ is the spatial prediction error transformed into local coordinate frames defined in past/future frames. $\tilde{\mathbf{e}}_s^{p,f}$ is obtained by transforming back to global coordinates using $(M^c)^{-1}$. Note that, setting M^c and $M^{p,f}$ equal to identity matrix results in a linear and rotation-varying prediction.

The temporal prediction procedure explained is for a $\mathbf{v}_c^c$ in a B frame. No temporal prediction is employed for I frames. For P frames, the only difference is that $\mathbf{\Delta}_t^c$ is calculated as $\tilde{\mathbf{e}}_s^p$ in Equation 6.3.

After predicting each vertex, the prediction residuals are uniformly quantized with Q parameter. Then the quantized residuals in each spatial layer are entropy coded separately using CABAC [67]. The decoder performs the inverse operations of the encoder. First, the prediction residuals are entropy decoded followed by dequantization. Then the frames and the vertices are traversed in the same order of the encoder and the same predictions are applied. Finally, the prediction errors are corrected by the dequantized residuals and the vertex locations are reconstructed.

3.3 3D Mesh Distortion Metrics

In order to measure the performance of a method which processes a given mesh, it is necessary to measure the quality or fidelity of the processed mesh compared to original mesh. For example, the processed mesh may be obtained by compressing the original mesh or simulating a received mesh after lossy transmission. For this purpose, we present the error metrics used in measuring the quality of meshes in the following parts:

3.3.1 Static 3D Mesh Distortion Metrics

In order to describe the quality of a processed/reconstructed 3D model, either objective or subjective quality measures should be defined. There is no immediate objective distortion metric in 3D meshes like mean-square error in images. One distortion metric used in the literature is the Hausdorff distance, $d_H(X,Y)$, between two surfaces X and Y which is defined by

$$d_H(X,Y) = \max\{\max_{x\in X} d(x,Y),\ \max_{y\in Y} d(y,X)\}, \tag{3.5}$$

where $d(x,Y)$ is the Euclidean distance from a point x on X to the closest point on Y. Another distortion metric is the L^2 distance, $d_L(X,Y)$, between two surfaces X and Y and is defined by

$$d_L(X,Y) = \max\{d(X,Y),\ d(Y,X)\}, \tag{3.6}$$

where

$$d(X,Y) = \left(\frac{1}{area(X)}\int_{x\in X} d(x,Y)^2 dx\right)^{1/2}. \tag{3.7}$$

Hausdorff distance takes the maximum of Euclidean distances whereas L^2 distance takes root mean square of the distances. Therefore L^2 distance reflects the average distortion of a 3D model while the Hausdorff distance reflects the maximum error. For this reason we use L^2 distance as the objective distortion metric in the experiments. To compute this distance, we use Metro tool [70].

3.3.2 Animated 3D Mesh Distortion Metrics

Although there is not a consensus on the best distortion metric for animated meshes in the literature, the most widely used metric is the error metric which is defined in Karni and Gotsman's work [49]. In our works, we also use this metric in order to be able to compare our results with the literature. We denote this error by *KG Error* and it is calculated as:

$$KGError = 100 \sqrt{\frac{\sum_{v=0}^{V-1} \sum_{f=0}^{F-1} \left\| \mathbf{G}(v,f) - \bar{\mathbf{G}}(v,f) \right\|^2}{\sum_{v=0}^{V-1} \sum_{f=0}^{F-1} \left\| \mathbf{G}(v,f) - \tilde{\mathbf{G}}(v,f) \right\|^2}}, \tag{3.8}$$

where $\mathbf{G}(v,f)$ is the original mesh data, $\bar{\mathbf{G}}(v,f)$ is the reconstructed mesh data and $\tilde{\mathbf{G}}(v,f)$ is per frame average of the original mesh data with V vertices and F frames.

CHAPTER 4

MULTIPLE DESCRIPTION CODING OF STATIC 3D MESHES

4.1 Introduction - Literature Review

The pioneering work in error resilient transmission of 3D models is that of Bajaj et al. [71] where compressed VRML streaming problem is addressed. In this method, the encoded bit-stream is classified into independent layers according to the depth-first order of the vertices. In this way, a layer can be decoded regardless of whether other layers are received or not. In [72], error resilience is achieved by segmenting the mesh and transmitting each segment independently. At the decoder, these segments are stitched using the joint-boundary information which is considered the most important. The drawback of these algorithms is that they are not scalable with respect to the channel packet loss rate, P_{LR}, and they do not provide a coarse-to-fine representation of the model.

In [73], the 3D mesh is first converted to a *Geometry Image* using the algorithm in [46] and coded with JPEG 2000. The resulting coded image is streamed using JPIP (JPEG 2000 Internet Protocol) [74]. In [75], a generic 3D middleware between the 3D application layer and the transport layer is proposed. However this study is mostly concerned with network issues which is not within the scope of this thesis.

Multiple Description Coding (MDC) is used to achieve error resiliency in [76]. where the multiple descriptions are generated by splitting the mesh geometry into submeshes and including the whole connectivity information in each description. However, in this scheme, descriptions are created with heuristic methods and no optimal solutions are proposed for varying network conditions.

In [77], [78], [79], [80] error resilient techniques that are scalable with respect to both channel

bandwidth and P_{LR} are proposed. The methods in [77], [78], [79], [80] try to achieve error resilience by assigning optimal *Forward Error Correction* (FEC) codes to layers of the progressively coded 3D mesh. The progressive scheme employed in these works is Compressed Progressive Meshes (CPM) [20]. While the ideas used in these works are similar, a more general optimization problem is tackled in [78], which maximizes the expected decoded quality for a given model, total bit budget and P_{LR}. Another important property of these methods is that the 3D model can be reconstructed at a resolution between coarse and fine representation with respect to varying packet loss rates.

Even though CPM based error resilient transmission techniques have been studied in the literature, no study exists on wavelet based methods even though wavelet based compression techniques have superior distortion-rate performance compared to CPM. Only in [77], it is stated that the given algorithm can be applied to any progressively coded 3D date with minor modifications. However, since the algorithms are not designed for granular scalability, it is not efficient to apply it to an embedded bitstream produced by a wavelet based codec.

4.2 Proposed MDSQ

In this MDC approach [2], we apply multiple description scalar quantization (MDSQ) [81] to wavelet transformed geometry data. Wavelet transformation is obtained by the PGC method described in Section 3.1.1 In this way, two independently quantized sets of wavelet coefficients are obtained. Each description is obtained by combining a coded quantized wavelet coefficients set and the compressed bitstream of connectivity data.

MDSQ was first proposed by Vaishampayan [81] as a practical solution to MDC and its analysis is thoroughly addressed in [82]. An important property of MDSQ is that it provides an asymptotic performance close to the rate-distortion bound.

Design of MDSQs can be viewed as creating two coarse side quantizers with acceptable distortions when used alone and one finer central quantizer which is obtained by combining two coarse side quantizers. Actually MDSQ implementation consists of a central quantization with regular joint cells followed by an index assignment operation to create side quantizers. The index assignment is often represented by an index assignment matrix, whose elements are central quantization indices and the column and row indices are the side quantizers' indices.

Therefore cells of side quantizers consist of union of corresponding central quantizer cells and depending on the matrix these side quantizer cells are usually union of disjoint intervals.

There are two parameters, R and k, for adjusting bitrate and distortion of central and side descriptions. R is the bits per source symbol for side decoders as 2R is the column or row length of index assignment matrix. 2k is the number of diagonals closest to main diagonal in index assignment matrix which adjusts distortions by introducing different amounts of redundancy. As an extreme case, when k=0, index assignment matrix consists of only main diagonal and both of the side descriptions are same as central description which causes maximum redundancy but minimum side distortions for a given R. As k increases, redundancy decreases causing more side distortion but central distortion decreases as it is quantized with more source symbols.

Detailed block diagrams of encoder and decoder of the proposed algorithm are given in Figures 4.1 and 4.2. After applying PGC and MDSQ, two sets of wavelet coefficients are obtained as if they are two distinct coarsely quantized wavelet coefficients. Then, each set of wavelet coefficients is coded by SPIHT. Descriptions are obtained by adding TG-coded coarsest level irregular connectivity data and coarsest level geometry data which is uniformly quantized with a chosen number of bits giving acceptable distortion (14 bits is a good suggestion) to each set. The reason why coarsest level geometry vertices are not quantized to two descriptions by MDSQ is that even small errors in this level cause significant visual distortion. Since the size of this level is small compared to remaining levels, the redundancy introduced by including it in all descriptions does not affect overall performance considerably.

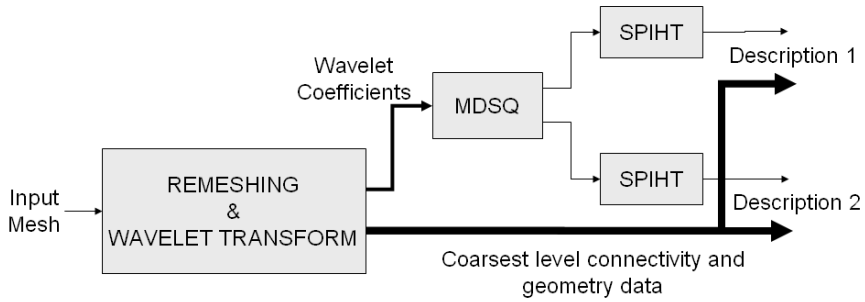


Figure 4.1: Encoder block diagram

Experiments are performed with the *Bunny* model which is composed of 34835 vertices and 69472 triangles. The original uncompressed model is shown in Figure 4.3. Effects of two

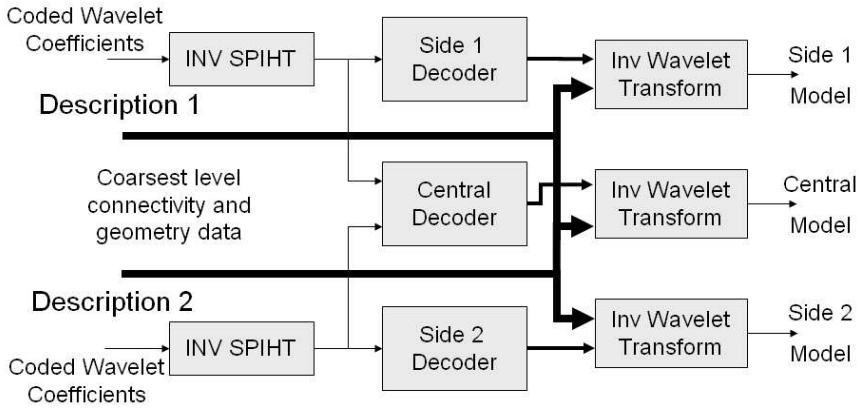


Figure 4.2: Decoder block diagram

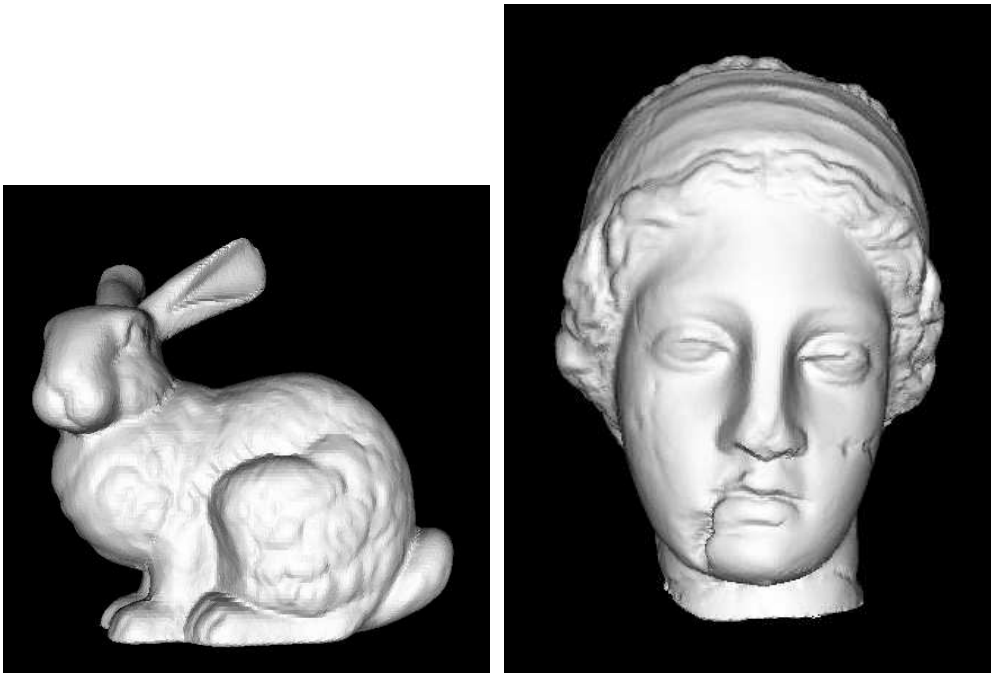
parameters in index assignment matrix, R (bits per source symbol for side decoders) and k (half of number of diagonals closest to main diagonal) are investigated. Tables 4.1, 4.2 and Figure 4.4 show the MDC performance for different R when $k = 1$ and Tables 4.3, 4.4 and Figure 4.5 show the performance for different k when $R = 6$. Results are given for compressed file sizes in bytes and relative L^2 distance as objective distortion metric. Relative L^2 distance is obtained by dividing distance by bounding box diagonal. All the L^2 errors in this section are given in units of 10^{-4}. Visual illustrations are also shown in Figure 4.6.

Reconstructed models with one description are labeled as *Side1* and *Side2* and the one with both of the descriptions is labeled as *Central*. In addition, error values for single description coded model having average file sizes of side descriptions are given and labeled as *Single*.

Table 4.1: File Sizes for different R when $k = 1$

Model	ML 4,1	ML 5,1	ML 6,1	ML 7,1	ML 8,1	ML 9,1
Side 1	1757	2402	3548	5152	7775	11941
Side 2	1870	2420	3546	5217	7862	11931
Central	2227	3180	4715	6981	10614	16360

As shown by the results, increasing bits per source symbol of side wavelet coefficients decrease all distortions and difference between side and central distortions are much larger at low rates. Increasing number of diagonals closest to main diagonal in index assignment matrix decreases redundancy by adding more index values to central quantization resulting increase



(a) (b)

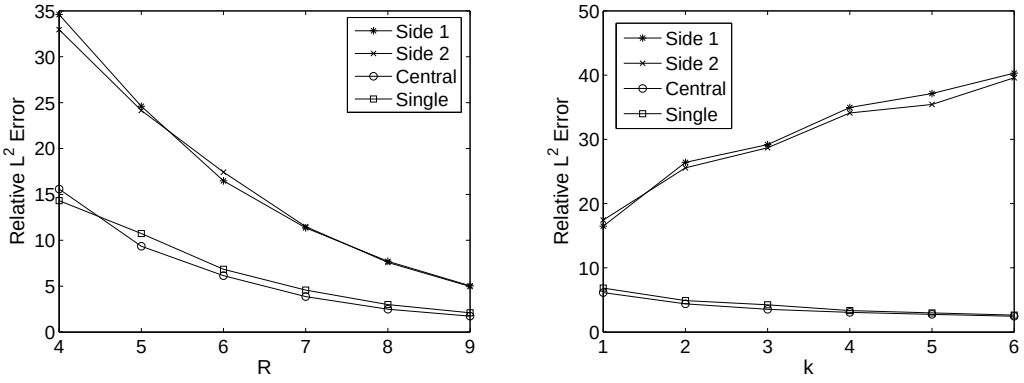
Figure 4.3: Test data (a) *Bunny* model; (b) *Venus head* model.

Table 4.2: Relative L^2 errors for the case in Table 4.1

Model	ML 4,1	ML 5,1	ML 6,1	ML 7,1	ML 8,1	ML 9,1
Side 1	34,59	24,61	16,48	11,35	7,71	5,04
Side 2	32,95	24,15	17,43	11,49	7,59	4,96
Central	15,58	9,35	6,15	3,87	2,49	1,74
Single	14,32	10,74	6,84	4,57	3,00	2,10

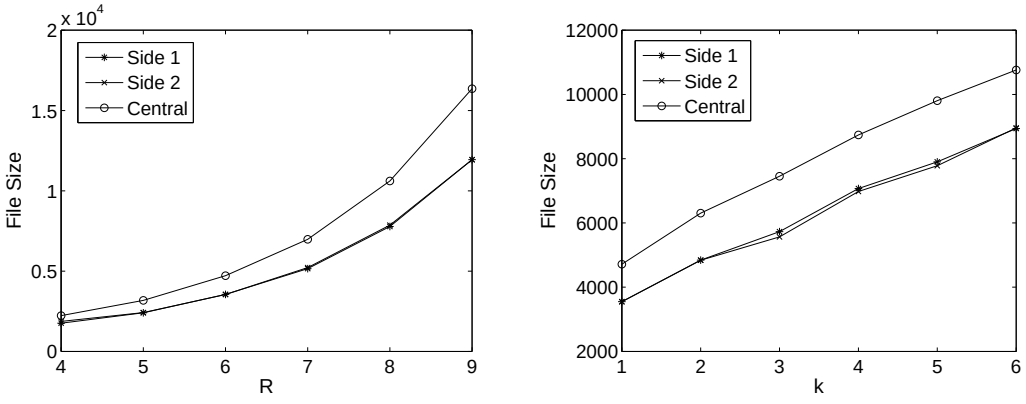
in side distortions and decrease in central distortion. However, it is observed that increase in side distortions are much larger than decrease in central distortion.

Final remark for the proposed MDSQ based method is that this work was performed at the early stages of the thesis and the results are not favorable compared to the proposed methods in the following sections. Also, this method has the disadvantage that increasing the number of descriptions is not efficient. Therefore, we do not provide performance comparison of following proposed methods with this method.



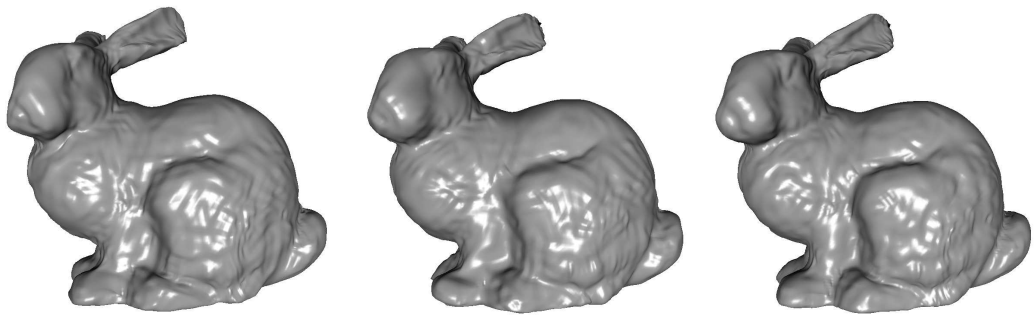
(a) (b)

Figure 4.4: (a) File Sizes (b) Relative L^2 errors for different R when $k = 1$



(a) (b)

Figure 4.5: (a) File Sizes (b) Relative L^2 errors for different k when $R = 6$



(a) (b) (c)

Figure 4.6: Reconstructed models for ML(6,1): (a)-(b) Reconstruction from one description (*Side1* and *Side2*) (c) Reconstruction from both of the descriptions (*Central*).

Table 4.3: File Sizes for different k when $R = 6$

Model	ML 6,1	ML 6,2	ML 6,3	ML 6.4	ML 6,5	ML 6,6
Side 1	3548	4842	5728	7071	7902	8939
Side 2	3546	4840	5564	6980	7782	8958
Central	4715	6305	7452	8739	9805	10762

Table 4.4: Relative L^2 errors for the case in Table 4.3

Model	ML 6,1	ML 6,2	ML 6,3	ML 6.4	ML 6,5	ML 6,6
Side 1	16,48	26,41	29,18	34,95	37,13	40,30
Side 2	17,43	25,57	28,69	34,11	35,43	39,59
Central	6,15	4,39	3,54	3,06	2,76	2,47
Single	6,84	4,90	4,24	3,34	2,98	2,64

4.3 Proposed Tree Partitioning

This MDC approach [1] is also based on Progressive Geometry Compression (PGC) scheme while it can be adapted to any mesh coding scheme employing wavelet transform and zero-tree coding. In order to obtain multiple descriptions, wavelet coefficient trees are grouped into several sets which are to be independently coded. These sets are packetized into multiple descriptions in such a way that each description contains one tree set which is coded with higher rate and several redundant tree sets which are coded with lower rates.

The general scheme is shown in (Figure 4.7). Wavelet coefficient trees are split into several sets W_i, $i = 1 \ldots N$ and coded by SPIHT algorithm with high bitrate. Each description contains M copies of different tree sets ($M \leq N$). Namely, *Description i* contains one set W_i coded at rate $R_{i,0}$ and $M-1$ sets of redundant trees W_j, $j \neq i$. These $M-1$ tree sets represent coding redundancy and are coded at lower rates than $R_{i,0}$. The redundancy included in each description is obtained as a result of the optimization algorithm described in following paragraphs. As the most important information in the embedded stream is located at the beginning of the bit-stream, the redundant copies would be used if the descriptions with corresponding high-rate coded tree subsets are lost. If some descriptions are lost, the most important parts of the original trees in those descriptions will be recovered because their copies at lower rates will be present in the received descriptions. The compressed coarsest mesh representation C with rate R_C is included in every description to facilitate the inverse wavelet transform even if only one description is received. Duplicating coarsest mesh C also increases coding redundancy.

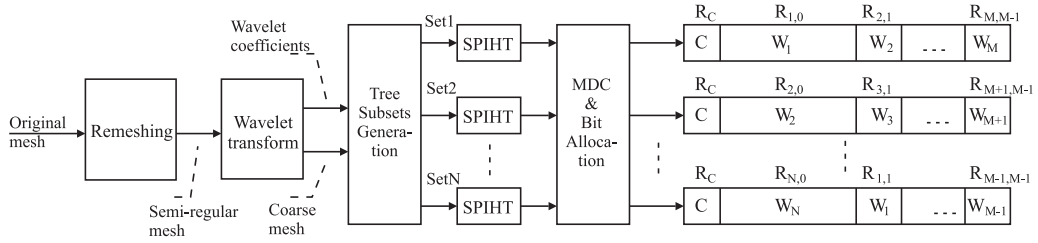
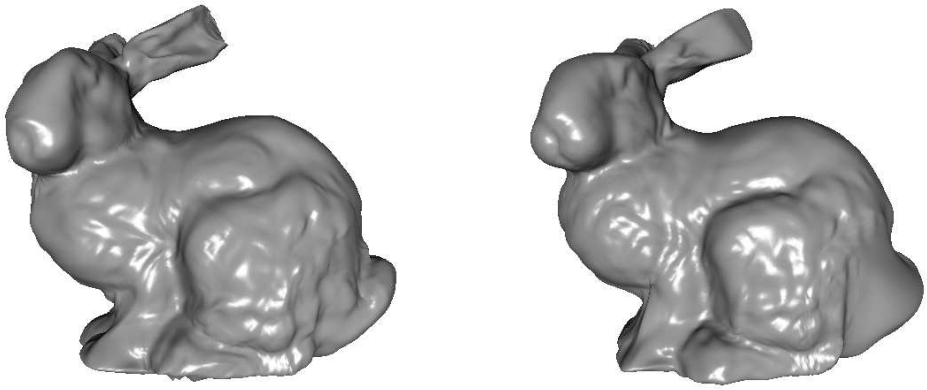


Figure 4.7: TM-MDC encoder scheme.

The way of grouping the coefficient trees into sets is particularly important, since different sets are reconstructed with different quality in case of description loss. Therefore, 3D mesh locations corresponding to different tree sets will have different quality. To perform grouping of trees into sets, ordering of the coarsest mesh vertices is performed, as proposed in [83, 84]. It provides ordering of the vertices that has good locality and continuity properties. Then, the desired type of wavelet trees grouping is obtained by sampling the one-dimensional array [83, 84]. Two types of tree set splitting have been compared: first - grouping the closely located trees together and second - grouping spatially disperse trees. The spatially close grouping is obtained by assigning successive vertices from the array to the same group. The disperse grouping is obtained by sampling the array in a round-robin fashion. It has been observed the latter case yields annoying artifacts in case when only one description is received and that the former case given better visual quality in general. This is illustrated in Figure 4.8 where model *Bunny* is encoded into four descriptions and optimized for PLR = 15%. One can see that although grouping disperse disperse trees achieves lower objective distortion than grouping close trees, it produces annoying visual artifacts. Therefore remaining results of this work are obtained by spatially close grouping method.

Redundancy of the proposed algorithm is determined by the number of redundant tree copies, their rates and the coarsest mesh size. Bit allocation problem has to minimize expected distortion at the decoder subject to probability of packet loss P and the target bit budget. A simple channel model where probability of packet loss P for each packet is assumed to be the same and independent of previous channel events is used. Another assumption is that one packet corresponds to one description. If the description has to be fragmented into different packets, probability of the description loss P can be found from PLR.

Suppose that N descriptions are generated. Then, coefficient trees are split into N tree sets,



(a) PSNR= 50.73 dB (b) PSNR= 48.51 dB

Figure 4.8: Reconstruction of model *Bunny* from one description among four descriptions for different types of tree grouping. (a) Spatially disperse grouping; PSNR= 50.73 dB. (b) Spatially close grouping, group size is 10; PSNR= 48.51 dB.

and M copies of tree sets are included in one description ($M \leq N$). Given P, it is easy to determine for each copy of the tree set the probabilities P_i, $i = 0, \ldots M$ that this copy is used for reconstruction where P_0 is the probability of using the full-rate copy of the tree set, and P_M is the probability of not receiving any copy of the tree set. Probabilities P_i can easily be found from P and the packetization strategy. Thus, we have to minimize the expected distortion

$$E[D] = \sum_{i=1}^{N} \sum_{j=0}^{M} P_j D_{ij}(R_{ij}), \tag{4.1}$$

where D_{ij} is the distortion incurred by using j-th copy of a tree set i and R_{ij} represents bits spent for j-th copy of i-th tree set. Optimization is performed under following bitrate constraints

$$\sum_{i=1}^{N} \sum_{j=0}^{M} R_{ij} + NR_C \leq R, \tag{4.2}$$

where R is the target bitrate and NR_C is the rate of the coarsest mesh. The rate of the coarsest mesh is chosen constant with geometry information quantized to 14 bitplanes.

Optimization of bit allocation requires computation of $D(R)$ function for every allocation step. Calculation of $D(R)$ is a computationally expensive operation. However, each tree set contributes to total distortion D. Since each tree set corresponds to some separate location on the mesh surface (defined by the root edge) in grouping spatially close trees, the distortions corresponding to separate tree sets can be considered additive. Therefore, distortion-rate (D-R) curve $D_i(R_i)$ for each coefficient tree set is obtained in advance. Calculations of $D_i(R_i)$ are

performed only once before the optimization algorithm is used for the first time. Then D-R curves are saved and can be used every time in bit allocation algorithm for new values of R and P.

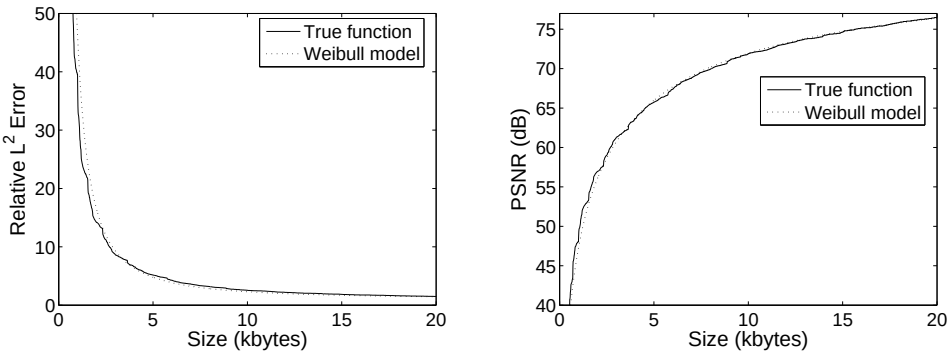
Optimization is performed with generalized Breiman, Friedman, Olshen, and Stone (BFOS) algorithm [85]. BFOS algorithm first allocates high rate for each copy of the tree set. Then, the algorithm consequently deallocates bits from the sets where $D(R)$ curves shows the lowest decay at allocated bitrate. This process stops when bit budget constraints are satisfied. In case optimization brings zero rates for some redundant trees copies, these copies are not included in the descriptions.

The embedded bitstream can be stopped literally at any point. Thus, calculation of the whole D-R curve requires considerable time. Therefore we employ the Weibull modeling of D-R curve presented in [86] for coding of images. It has been shown in [1] that the output of PGC coder can also be approximated with this model. The model is described by

$$D(R) = a - be^{-cR^d}, \tag{4.3}$$

where real numbers a, b, c, and d are the parameters which depend on the D-R characteristics of the compressed bitstream. As there are four parameters in this model, $D(R)$ curve can be found by using at minimum four points. This model can approximate both L^2 and PSNR curves. To fit this model to D-R samples, we use nonlinear least-squares regression. Figure 4.9 shows the comparison of true operational D-R curves of *Bunny* model and their Weibull models. One can see that the model closely approximates the real data. Moreover, the model has a nice feature of convexity, which is desirable for bit allocation algorithm.

Experiments are performed for models *Bunny* and *Venus head*. The original uncompressed models are shown in Figure 4.3. In the experiments, model *Bunny* is coded into four descriptions at total 22972 Bytes (5743 Bytes per description) and eight descriptions at total 25944 Bytes (3243 Bytes per description). Model *Venus head* is coded into four descriptions at 24404 Bytes (6101 Bytes per each description). The reconstruction distortion metric is the same metric as in Section 4.2 which is relative L^2 distance. Also the same numbers are provided in PSNR scale where PSNR $= 20\log_{10} peak/d$, *peak*, peak is the bounding box diagonal, and d is the L^2 error.



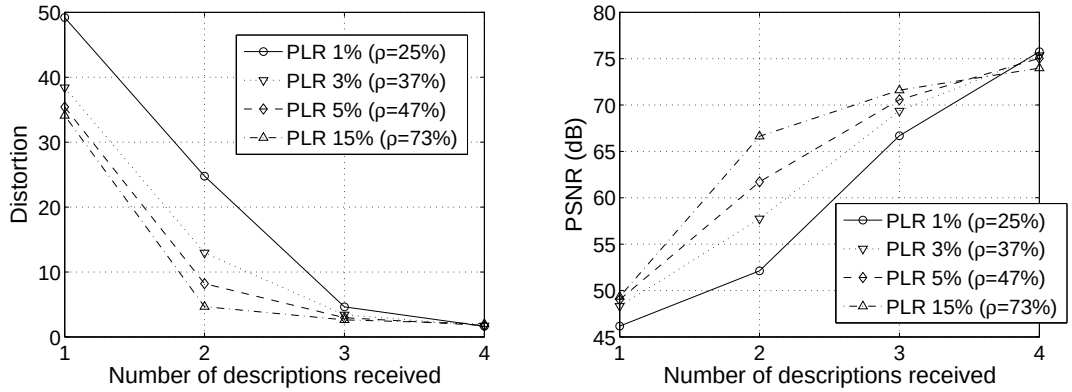
(a) (b)

Figure 4.9: Comparison between the Weibull model (10 points) and operational D-R curve (L^2) for *Bunny* model. (a) Relative L^2 error; (b) PSNR.

In the experiments, three coders are compared. The first coder is the one proposed in the algorithm, which is named Tree-based Mesh MDC (TM-MDC). The second coder is a simple MDC coder in which each description contains the coarsest mesh and one set of wavelet coefficient trees. The sets of coefficient trees in both coders are formed from spatially close groups of trees of size 10. This coder is the same as TM-MDC optimized for $P = 0$ (for $P = 0$, no redundant trees are included in the descriptions). The third coder is unprotected SPIHT. The packetization for unprotected SPIHT is performed in the following way. The output bitstream of PGC coder is divided into N parts of equal size, where N is the number of descriptions in the MD coder that unprotected SPIHT is compared to. PGC produces the embedded bitstream. Thus, the received part can be used for reconstruction if all the packets containing earlier parts of bitstream have been received. For example, if parts one, two, and four are received, only parts one and two are used for reconstruction. If part one is lost, no reconstruction is available.

Figures 4.10 and 4.11 show the average distortions for reconstruction from different number of received descriptions for model *Bunny* coded into four and eight descriptions respectively. The curves are generated for TM-MDC with bit allocations optimized for different P. From the figures, it is observed that when all of the descriptions are not received, the coder optimized for higher P has the best PSNR value. However this does not mean that higher loss rates lead to better performance because the coder is optimized to minimize expected distortion driven by the packet loss rate. Therefore since the coder optimized for a high loss rate expects more description losses, it tries to achieve better distortion in cases when small num-

ber of descriptions are received. On the other hand, the coder optimized for a low loss rate is expected to receive more descriptions and it tries to achieve better distortion in cases when most of the descriptions are received. This can be verified from the figures by the observing that the coder optimized for $P = 1\%$ shows the best performance in case all descriptions are received.



(a) (b)

Figure 4.10: Reconstruction of *Bunny* model from different number of received descriptions. The results are given for bit allocations for different packet loss rates (PLR). The redundancy ρ is given in brackets. (a) Relative L^2 error; (b) PSNR.

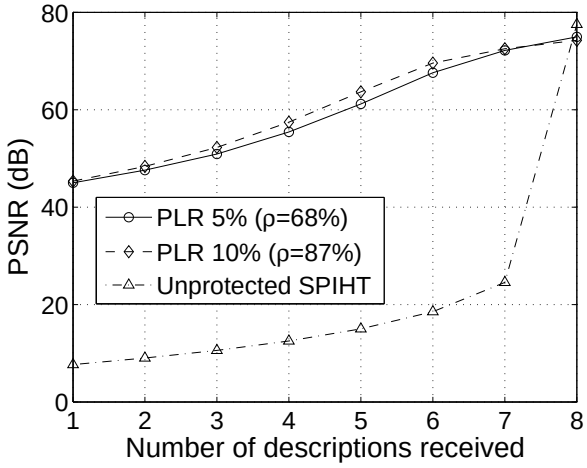
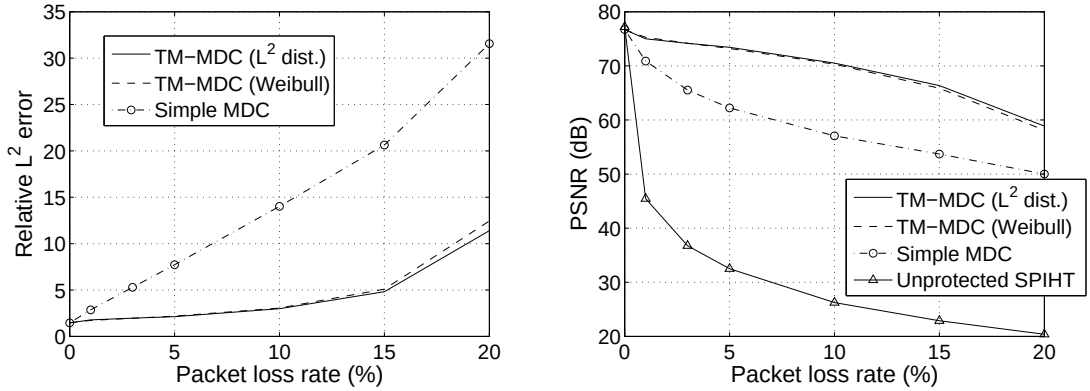


Figure 4.11: Model *Bunny* encoded into 8 descriptions at total 25944 Bytes. Reconstructed from different number of descriptions. Compared to unprotected SPIHT.

Figure 4.16 compares the performance of the proposed TM-MDC, the simple MD coder, and unprotected SPIHT for model *Bunny*. The results are calculated for $P = 0, 1, 3, 5, 10, 15, 20\%$.



(a) (b)

Figure 4.12: The comparison of network performance of the proposed TM-MDC with a simple MDC scheme and unprotected SPIHT. (a) Relative L^2 error; (b) PSNR.

In TM-MDC coder, bit allocation is optimized for each P. For simple MDC coder, redundancy is always fixed $\rho = 10\%$ which occurs due to including coarsest level data in each description. For each P, the average distortion is calculated by averaging results of 100000 experiments (simulations of packet losses). For $P = 0$, coders show the same performance because the optimized for $P = 0$ TM-MDC coder and simple MDC coder are in fact the same coder. However, for higher packet loss rates, the performance of simple MDC coder dramatically decreases while the reconstruction quality of TM-MDC shows only mild degradation. For $P = 20\%$, the optimized TM-MDC coder is 15 dB higher than the simple MD coder.

In the figure, TM-MDC method results are shown for two different labels, namely TM-MDC (L^2 dist) and TM-MDC (Weibull). The former corresponds to the results obtained by using original D-R curves in optimization while the latter corresponds to results obtained by modeling D-R curves by Weibull model [86] to decrease complexity.

Figure 4.13 shows visual reconstruction results for the model *Bunny* which is encoded with redundancy $\rho = 63\%$ and Figure 4.14 shows visual reconstruction results for the model *Venus head* which is encoded with redundancy $\rho = 53\%$. The reconstructed visual models correspond to reconstructions from one, two, three, and four description. One could see that even the reconstruction from one description provides acceptable visual quality.

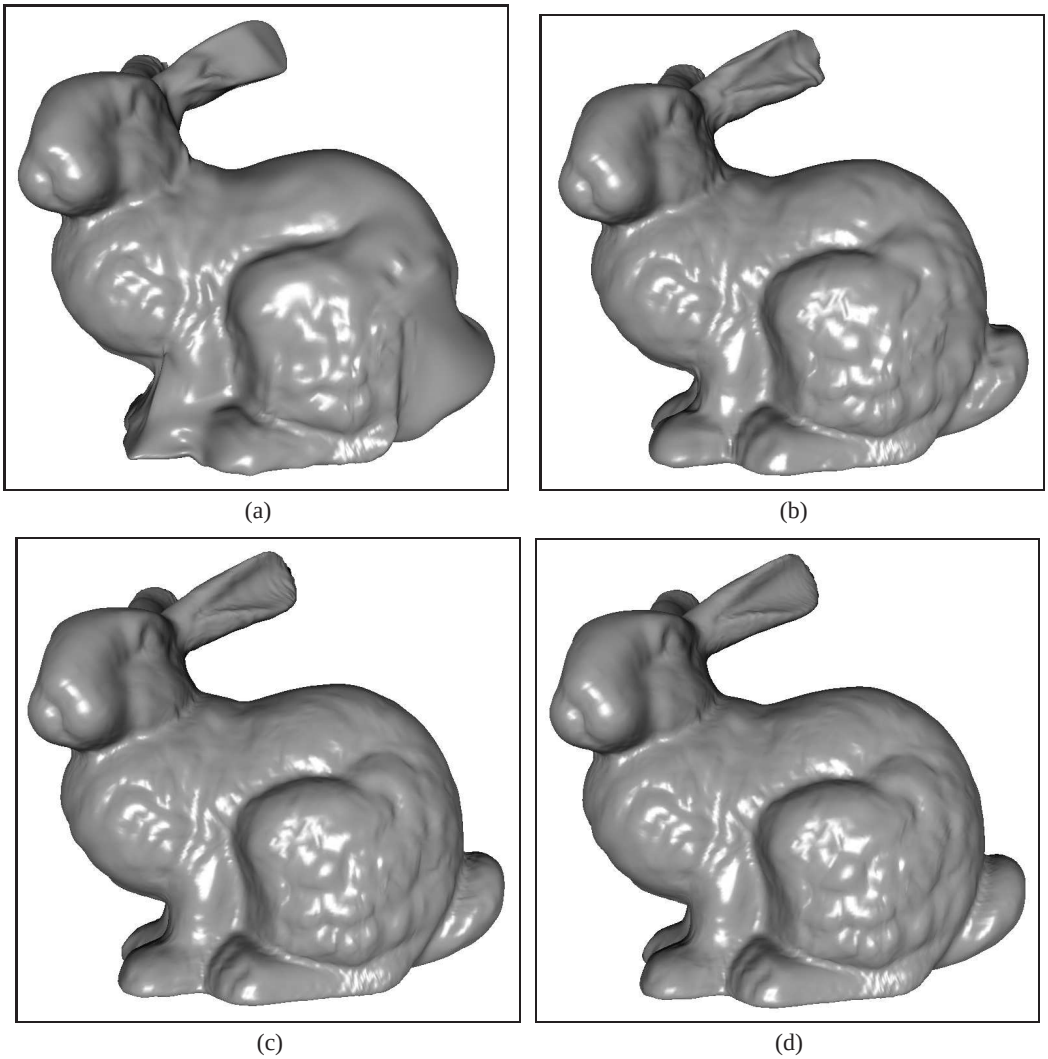


Figure 4.13: Reconstruction of the *Bunny* model from (a) one description (48.36 dB), (b) two descriptions (63.60 dB), (c) three descriptions (71.44 dB), (d) four descriptions (74.33 dB).

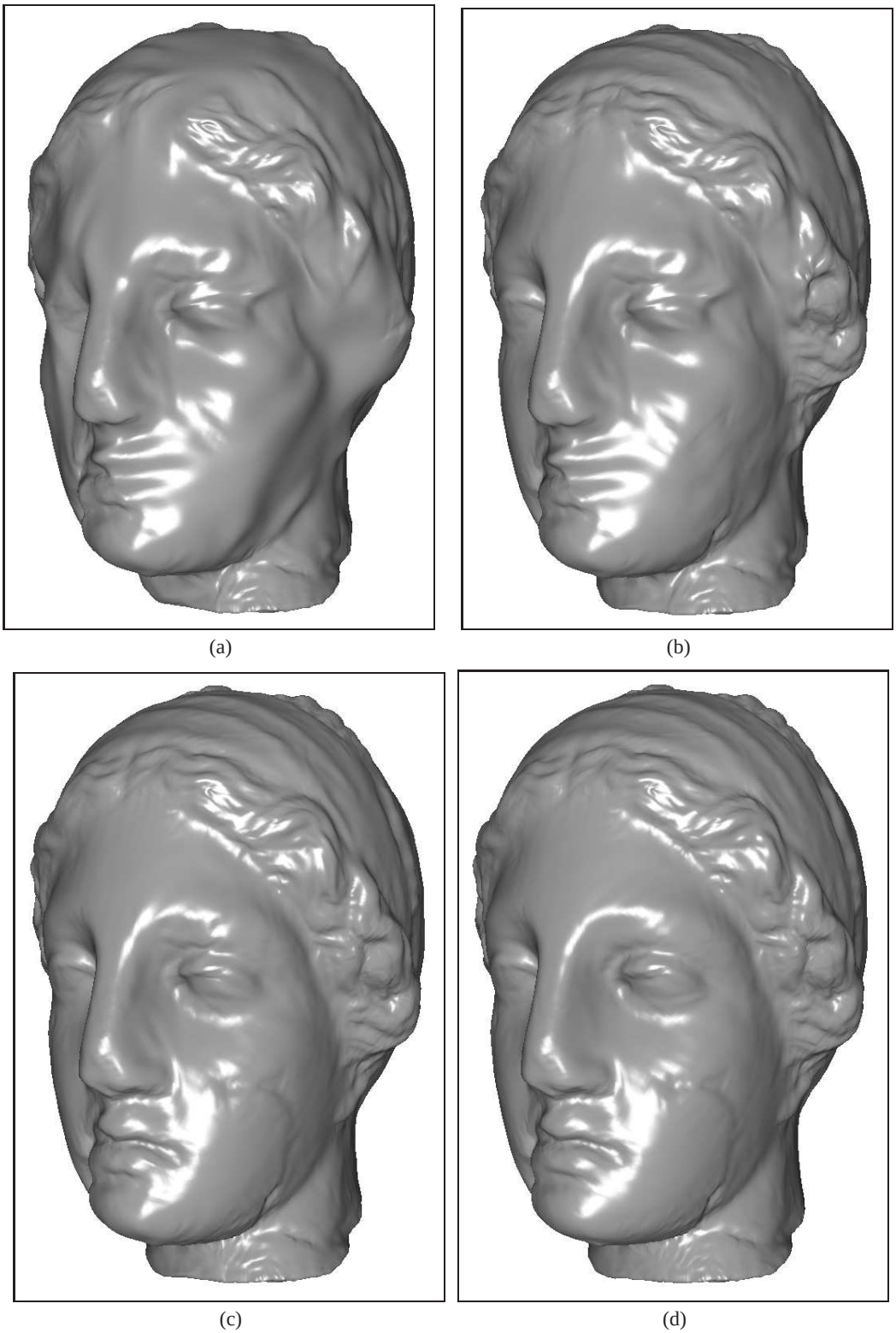


Figure 4.14: Reconstruction of *Venus head* model from (a) one description (53.97 dB), (b) two descriptions (65.18 dB), (c) three descriptions (72.51 dB), (d) four descriptions (77.08 dB).

4.4 Proposed FEC Based Approach for MDC

This approach can be used for both MDC [4] and packet loss resilient streaming [5]. The packet loss resilient streaming analysis is provided in Section 4.5. Our aim is to achieve best reconstruction quality with respect to channel bandwidth and packet loss rate (P_{LR}). The proposed algorithm adapts the FEC based packet loss resilient image coding schemes [87]. The algorithm first generates an embedded bitstream by compressing the mesh with PGC algorithm. The bitstream is protected by optimal FEC assignment.

4.4.1 Problem Definition and Proposed Solution

In this work, we try to minimize the expected reconstruction distortion of the 3D model transmitted over an erasure channel for a given target bit-rate, P_{LR} and channel model. In order to achieve this, first the 3D model is compressed with PGC as described in Section 3.1.1 and an embedded bitstream is produced. After the embedded bitstream is produced, the problem of optimum loss protection is stated as follows: The embedded bitstream is protected with RS codes and transmitted over an erasure channel in N packets each with the length of L symbols (bytes in this work). The protection system builds L source segments S_i, $i = 1, ..., L$ which have the lengths $m_i \in \{1, ..., N\}$ and protects each segment with an $RS(N, m_i)$ code. For each $i = 1, ..., L$, let $f_i = N - m_i$ denote the number of RS redundancy symbols protecting the segment S_i. An example of this FEC assignment is illustrated in Table 4.5. If n out of N packets are lost, then the RS codes ensure that all the segments that contain at most $N - n$ source symbols can be recovered. Adding the constraint $f_1 \geq f_2 \geq ... \geq f_L$, one can be sure that if f_i packets are lost, then the receiver can decode at least the first i segments. Let $\mathcal{F}$ denote the set of L-tuples $(f_1, ..., f_L)$ such that $f_i \in \{0, ..., N-1\}$ for $i = 1, ..., L$ and $f_1 \geq f_2 \geq ... \geq f_L$. Let $p(m, N)$ denote the probability of losing exactly m packets out of N and let

$$c_N(k) = \sum_{m=0}^{k} p(m, N) \text{ for } k = 0, ..., N \tag{4.4}$$

Then $c_N(f_i)$ is the probability that the segment S_i can be decoded successfully. Let $D(R)$ denote the distortion-rate (D-R) function of the source coder. In order to achieve an optimum packet loss protection, we need to find $F = (f_1, ..., f_L) \in \mathcal{F}$ such that the expected distortion

$$E_D = c_N(N)D(r_0) + \sum_{i=1}^{L} c_N(f_i)(D(r_i) - D(r_{i-1})) \tag{4.5}$$

is minimized where

$$r_i = \begin{cases} 0, & \text{for } i = 0 \\ \sum_{k=1}^{i} m_k = iN - \sum_{k=1}^{i} f_k, & \text{for } i = 1, ..., L \end{cases} \tag{4.6}$$

Note that the D-R curve modeling introduced in Section 4.3 is also applicable during the optimization of this problem.

The next step is to determine the optimum FEC assignments by minimizing E_D in Equation 4.5. In the literature, there are several efficient methods for similar optimization problems used for scalable image coders [87], [88], [89], [90], [91], [92]. In [91], it is shown that the method in [88] performs very well in terms of expected distortion and the method in [91] has the lowest computational complexity with slightly worse expected distortion performance.

In [88], given $p = LN$ points on the operational D-R curve of the source coder, the algorithm first computes the h vertices of their convex hull. Then, the solution is found in $O(hN \log N)$ time. This solution is optimal under the assumption of the convexity of the D-R function and of fractional bit allocation assignment. In [91], a local search algorithm with $O(NL)$ complexity is presented that starts from a solution that maximizes the expected number of received source bits and iteratively improves this solution. The reader is referred to [88], [91] for the details of the algorithms. Since we also use a scalable bitstream produced by PGC coder, we employ in our experiments the optimization methods from [88] and [91].

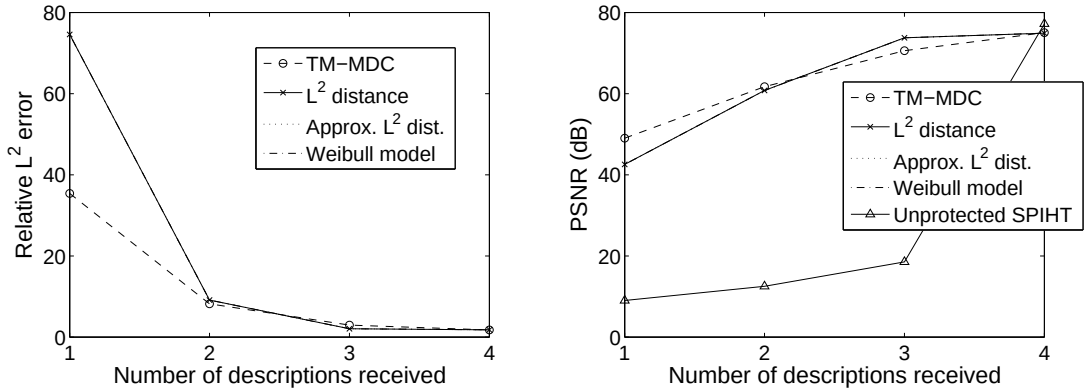
Table 4.5: An example FEC assignment on an embedded bitstream. There are $N = 5$ packets each composed of $L = 4$ symbols. Therefore there are 4 source segments, S_i, $i = 1, 2, 3, 4$ each of which contains m_i data symbols and f_i FEC symbols where $m_i + f_i = N$. In this example $m_1 = 2$, $f_1 = 3, m_2 = 3$, $f_2 = 2, m_3 = 3$, $f_3 = 2, m_4 = 4$, $f_4 = 1$. Earlier parts of the bitstream are assigned more FEC symbols since they contribute more to overall quality.

	P1	P2	P3	P4	P5
Segment 1	1	2	FEC	FEC	FEC
Segment 2	3	4	5	FEC	FEC
Segment 3	6	7	8	FEC	FEC
Segment 4	9	10	11	12	FEC

4.4.2 MDC Experimental Results

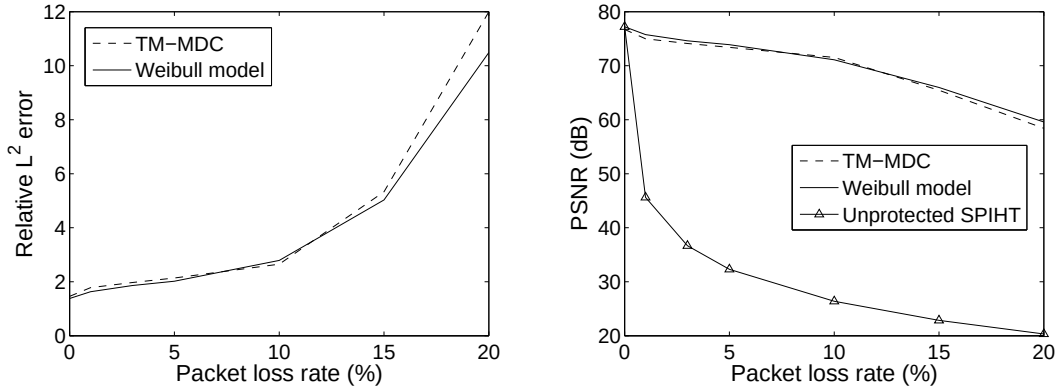
In this part of the experiments, we try to obtain the MDC performance of the proposed method. We compare the proposed coder with the coder TM-MDC of previous section [1]. We have performed the experiments on model *Bunny*. In the experiments, model *Bunny* is coded at 22972 Bytes (5743 Bytes per each description). The reconstruction distortion is the relative L^2 error and PSNR values are calculated as in previous sections.

In the experiments, Figure 4.15 shows reconstruction from different number of descriptions. In the figure, label L^2 *distance* method corresponds to using L^2 distance obtained by metro tool, *approximate* L^2 distance method corresponds to using approximate L^2 distance value obtained by disabling face and edge samplings in metro tool and label *Weibull model* method corresponds to using $D(R)$ curve obtained by modeling original $D(R)$ curve with 10 values of L^2 distances during optimization procedures. Both MDC coders are optimized for PLR = 5%. As one can see, both MD coders outperform unprotected SPIHT except for the case, when all the descriptions are received. The TM-MDC achieves higher PSNR for reconstruction from one description, but lower PSNR for reconstruction from three descriptions. We think that this can be strongly connected with the fact that each description in TM-MDC method includes whole coarsest level geometry while descriptions in our method does not contain all bitplanes of coarsest level geometry. Another observation is that results of L^2 *distance*, *approximate* L^2 and *Weibull model* methods are indistinguishable in the figure which proves the success of modeling.



(a) (b)

Figure 4.15: Reconstruction from different number of descriptions (PSNR) for *Bunny* model. (a) Relative L^2 error; (b) PSNR.



(a) (b)

Figure 4.16: The comparison of the MD-FEC with TM-MDC coder from [1]. (a) Relative L^2 error; (b) PSNR.

Figure 4.16 compares the average performance in the lossy environment of the proposed coder using L^2 distance, approximate L^2 distance, Weibull model with the performance of unprotected SPIHT and TM-MDC coder from [1]. As seen in the figure, the proposed approach shows competitive results compared to TM-MDC and considerably outperforms unprotected SPIHT.

4.5 Extension of MD-FEC to Packet Loss Resilient Coding

In this section, we extend the MD-FEC approach described in the previous section to be used in packet loss resilient streaming scenarios by simply increasing the number of descriptions to achieve smaller and meaningful packet sizes for common network protocols. This would not be possible with the TM-MDC method since increasing the number of descriptions bring extra redundancy.

We compare our results with the state of the art CPM based methods [78],[80] in terms of expected distortion and flexibility in packetization. Therefore, we first provide an overview of CPM based loss resilient coding approaches. In the literature, CPM based methods were tested only with 3D models with small number of triangles. In this work by decreasing the complexity, we manage to do the performance evaluation for 3D models with high number of triangles. The experimental results show that, higher quality with more flexible packetization can be achieved by the proposed algorithm.

4.5.1 Packet Loss Resilient Coding Based on CPM

The compression algorithm of CPM was described in Section 3.1.2. For loss resilient transmission of the data generated by the CPM algorithm, optimal error correcting codes (in particular Reed-Solomon - RS codes) can be assigned to the layers of the progressively coded mesh. Let $RS(N, k_0)$ denote the RS code applied to base layer (level-0) and $RS(N, k_j)$ denote the RS code applied to j-th enhancement layer (level-j), where $j = 1, ..., L_M$. Here L_M denotes the number of enhancement layers transmitted out of M enhancement layers according to the bitrate of the channel. RS codes are applied vertically and packetization is performed horizontally. Therefore receiving any k_j out of N packets of level-j allows successful decoding of level-j. A simple protected CPM output bitstream is illustrated in Table 4.6. A drawback of this packetization is that since N and $R^{(i)}$ are constant, packet sizes of different layers vary for different assignments of k_j values, which is not the case in wavelet based loss resilient techniques. Padding the shorter packets with zeros would cause an increase in the bandwidth requirements.

Table 4.6: An example CPM output with $L_M + 1 = 3$ layers. *Pi* denotes Packet *i* generated horizontally while FEC is applied vertically. In this simple example $N = 6$, $k_0 = 2$, $k_1 = 3$ and $k_2 = 4$.

Base Mesh			Batch 1			Batch 2			
P1	1	2	P7	5	6	P13	11	12	13
P2	3	4	P8	7	8	P14	14	15	16
P3	FEC	FEC	P9	9	10	P15	17	18	19
P4	FEC	FEC	P10	FEC	FEC	P16	20	21	22
P5	FEC	FEC	P11	FEC	FEC	P17	FEC	FEC	FEC
P6	FEC	FEC	P12	FEC	FEC	P18	FEC	FEC	FEC

The problem definition can be formulated as follows: Given a 3D model and a total bit budget B, the aim is to determine an optimal combination of the following parameters to minimize the expected decoded model distortion ($E_D(L_M)$): 1) l, number of bits used in quantizing the position error; 2) L_M, the number of transmitted batches; 3) C, the total number of channel coding bits; 4) $C_L = [C^{(0)}, C^{(1)}, ..., C^{(L_M)}]$, $C^{(i)}$ denoting the number of channel coding bits applied to level i (or $[k_0, k_1, ..., k_{L_M}]$ since k_i is a function of $C^{(i)}$, $R^{(i)}$ and N).

To quantify the expected distortion $E_D(L_M)$, first let P_j denote the probability of terminating the decoding operation at level-j and it can be calculated as

$$P_j = \sum_{m=N-k_j+1}^{N} p(m, N) \tag{4.7}$$

where $p(m, N)$ is the probability of losing m packets within a block of N packets. Then expected distortion $E_D(L_M)$ can be calculated as

$$E_D(L_M) = P_0 D_{NR} + \sum_{j=1}^{L_M} P_j D_{j-1} \prod_{i=0}^{j-1}(1 - P_i) + D_{L_M} \prod_{j=0}^{L_M}(1 - P_j) \tag{4.8}$$

where D_j is the distortion of level-j, D_{NR} is the distortion when no reconstruction of the model is possible (i.e. if the base mesh is lost), D_{L_M} is the distortion if all the levels are successfully received.

4.5.2 Proposed Modifications for CPM based Loss Resilient Coding

AlRegib et al. [78] propose an algorithm to find the optimal solution for this problem. According to the distortion-rate (D-R) curves, the algorithm selects the best (l, L_M) pair by varying C values using a step size Q. For each selected l and L_M, the lowest expected distortion and C_L are found using a local search algorithm. Final output of the algorithm is the C_L corresponding to lowest distortion among all steps.

A drawback of this algorithm is that it contains many repeated operations since the results are iterated by varying C using the step size Q. As there are finite choices of l, L_M and k_j values for a C, it is very likely to encounter same l, L_M and k_j values for several C values during the local search. In [77], this computational redundancy is removed by iterating only the finite k_j values and putting the constraint that $k_0 \leq k_1 \leq ... \leq k_L$. Although in [77] the problem definition also assumes that C is given and L_M is fixed, we can generalize the algorithm by removing this assumption. In our experimental results, we combine the two methods such that, for given possible finite sets of l's and L_M's:

- The combined algorithm computes expected distortion for every k_j values satisfying the bit budget requirement and the $k_0 \leq k_1 \leq ... \leq k_L$ condition.
- k_j values corresponding to the least expected distortion is chosen as the optimum FEC assignment.

AlRegib et al. [79] tackle with a similar problem in which l is also assumed to be given in addition to our general problem definition. An exhaustive search of $[C^{(0)}, C^{(1)}, ..., C^{(L)}]$ is

proposed to find the optimal solution. In [80], Ahmad et al. propose improvements on [79] in terms of complexity and packetization flexibility inspired from the work in [92]. In our experimental results, in order to obtain Ahmad et al.'s results, we first find the local optimum parameters for all possible l values. Then the parameters corresponding to the lowest expected distortion is chosen as the global optimum parameters.

4.5.3 Distortion Metric and Simplifications in Calculations

In order to use Equations (4.8) and (4.5), the distortions D_j and $D(R)$ are chosen as L^2 distance as in previous sections. To reduce complexity of L^2 distance computations, modeling D-R curve with Weibull Model is employed as in Section 4.3. For CPM based methods, this complexity is decreased by using quadric error metric as proposed in [77].

4.5.4 Channel Model

In order to take into account the packet loss behavior to minimize the expected reconstruction distortion, the channel is needed to be modeled appropriately. In this work, we use a two-state Markov model which is shown to be very effective in modeling packet losses [93], [94].

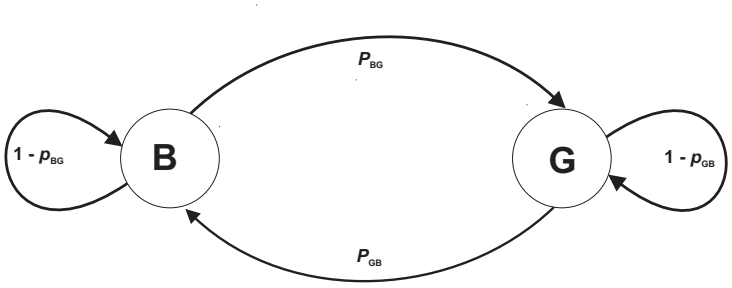


Figure 4.17: Two state Markov channel model.

The Markov model described in [93] and [94] is a renewal model, i.e., the event of a loss resets the memory of the loss process. Such models are determined by the distribution of error free intervals (gaps). Let a gap of length v be the event that after a lost packet, $v-1$ packets are received and then again a packet is lost. The gap density function $g(v)$ gives the probability of a gap length v, i.e., $g(v) = Pr(0^{v-1}1|1)$, where '1' denotes a lost packet and '0^{v-1}' denotes $v-1$ consecutively received packets. The gap distribution function $G(v)$ gives the probability

of a gap length greater than $v-1$, i.e., $G(v) = Pr(0^{v-1}|1)$. In our model, in state B all packets are lost ('1'), while in state G all packets are received ('0'), yielding

$$g(v) = \begin{cases} 1 - p_{BG}, & \text{if } v = 1 \\ p_{BG}(1 - p_{GB})^{v-2} p_{GB}, & \text{if } v > 1 \end{cases} \tag{4.9}$$

$$G(v) = \begin{cases} 1, & \text{if } v = 1 \\ p_{BG}(1 - p_{GB})^{v-2}, & \text{if } v > 1 \end{cases} \tag{4.10}$$

Let $R(m, N)$ be the probability of $m-1$ packet losses within the next $N-1$ packets following a lost packet. It can be calculated using the recurrence.

$$R(m, N) = \begin{cases} G(N), & m = 1 \\ \sum_{v=1}^{N-m+1} g(v) R(m-1, N-v), & 2 \leq m \leq N \end{cases} \tag{4.11}$$

Then the probability of m lost packets within a block of N packets given in Equations 4.4 and 4.7 is

$$p(m, N) = \sum_{v=1}^{N-m+1} P_B G(v) R(m, N-v+1), \text{if } 1 \leq m \leq N \tag{4.12}$$

where P_B is the average loss probability.

4.5.5 Experimental Results

In this part of the experiments, we try to obtain the packet loss resilient streaming performance of the proposed method where the number of packets are much higher compared to MDC case. We compare the proposed method with CPM based methods. We have used the test models *Bunny* and *Venus head*. The packets are sent over the two-state Markov modeled packet erasure channel with the average burst length of 5. All the experiments are performed using an Intel Pentium 4 2.2GHz 1Gb RAM Windows XP installed computer. Although it is not possible to estimate exact complexity with these settings (e.g. due to possible inefficient implemented parts of algorithms or multitasking of OS), we provide the results to mention the order of complexity of the algorithms compared to each other.

In the following experimental results, if the 3D model name and the coding bitrate are not explicitly specified, then these results correspond to the *Bunny* model coded at 3.5 bpv and packetized with $N = 100$.

In the rest of this section, we categorize the experiments and present in different subsections. We start with the results of CPM based methods which include the proposed $kStep$ parameter and comparison of CPM based method simulation distortions. Then we provide the results for D-R curve modeling for PGC based methods followed by the comparison of CPM and PGC based methods in terms of simulation distortions. Then we examine the mismatch scenario which occurs when the real loss rate and the one used in optimization differ. We continue the results with complexity comparisons for the optimization methods and finally we provide visual results for subjective evaluation.

4.5.5.1 Proposed kStep for CPM Based Methods

For AlRegib's CPM based methods, since iterating all possible $RS(N, k_j)$ pairs for each layer is not feasible due to significant complexity requirements, we propose a new parameter $kStep$. With $kStep$ parameter, instead of iterating k_j's in $RS(N, k_j)$ pairs one by one, we increment and decrement k_j values by an amount of $kStep$ in the iterations. Figure 4.18 shows the simulated PSNR values and Figure 4.19 shows the optimization times for different $kStep$ values.

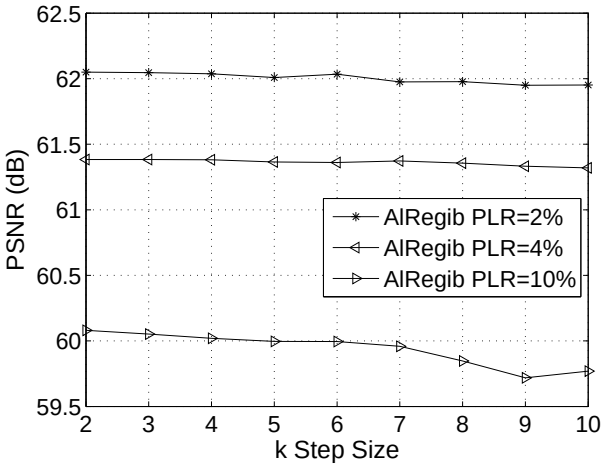


Figure 4.18: Effect of k step size on quality: Simulated PSNR vs k step size for *Bunny* model optimized for P_{LR} = 2%, 4% and 10% and coded at 3.5 bpv.

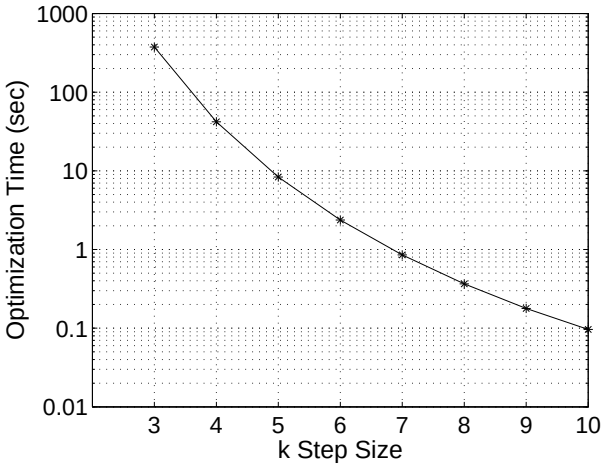


Figure 4.19: Effect of k step size on complexity: Optimization time vs k step size for *Bunny* model optimized for P_{LR} = 4% and coded at 3.5 bpv.

From the figures, it can observed that it is possible to save great amount of time during optimization by increasing the $kStep$ value. In addition, while increasing the $kStep$ value, the decrease in the simulated PSNR value is not significant for different packet loss rates.

4.5.5.2 Comparison of CPM Based Methods

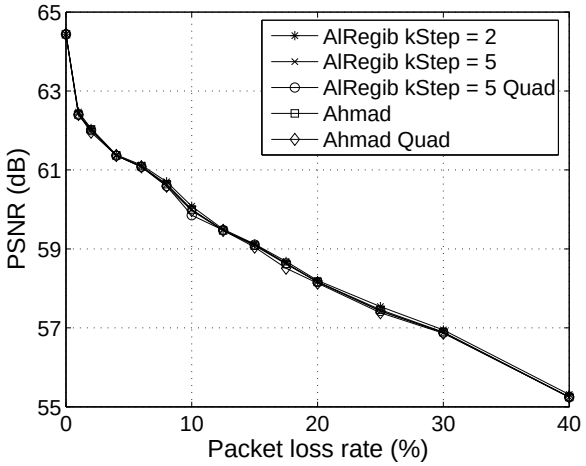


Figure 4.20: Comparison of CPM based methods for Bunny model in terms of simulated distortions for various P_{LR}'s.

After defining the $kStep$ parameter, we proceed to comparison of all aforementioned CPM based methods. Figure 4.20 summarizes all of the mentioned CPM based error resilient meth-

ods. In the figure, simulated PSNR vs P_{LR} values are presented for several configurations. The curves in the figure are AlRegib's method with a fine $kStep$ value of 2 (*AlRegib kStep = 2*), AlRegib's method with a coarser $kStep$ value of 5 (*AlRegib kStep = 5*), AlRegib's method with $kStep = 5$ and using quadric error metric during optimization (*AlRegib kStep = 5 Quad*), generalized Ahmad's method (*Ahmad*) and generalized Ahmad's method with using quadric error metric during optimization (*Ahmad Quad*).

From the figure, it can be deduced that all the CPM based methods similar performance, where none of the methods achieve significantly higher simulated PSNR. Nonetheless, best PSNR is achieved by *AlRegib kStep = 2* as expected.

4.5.5.3 Performance of D-R Curve Modeling for PGC Based Methods

After examining the CPM based methods, we start the analysis of PGC based methods. For the PGC based methods, FEC assignments are optimized with the algorithms of Mohr et al. [88] and Stankovic et al. [91] and labeled as *PGCMohr* and *PGCStankovic* in the rest of the chapter.

We initially examine the performance of the proposed D-R curve modeling described in previous sections. Figure 4.21 shows simulated distortions corresponding to various P_{LR}'s for *PGCMohr* employing the original D-R curve and modeled D-R curve during optimization. It is observed that quite acceptable results can be achieved by D-R curve modeling. Therefore we present the remaining PGC based results with modeled D-R curves which significantly reduce optimization times.

4.5.5.4 Comparison of CPM and PGC Based Methods

In this part, we present comparison of the CPM and PGC based methods in terms of simulation distortions for three cases: 1) Bunny Model Coded at 3.5 bpv, 2) Bunny Model Coded at 1.2 bpv and 3) Venus Model Coded at 4 bpv.

Bunny Model Coded at 3.5 bpv

For this case, a comprehensive summary of the results for all methods is presented in Table 4.7. Also comparison of CPM based and PGC based methods in terms of simulated PSNR can

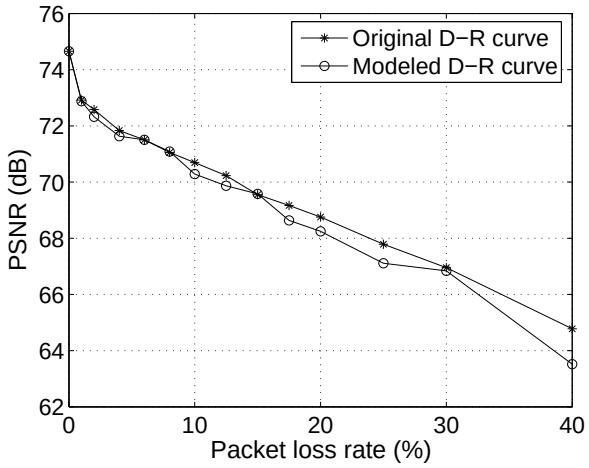


Figure 4.21: Comparison of using original D-R curve and using modeled D-R curve during optimization for *Bunny* model in terms of simulated distortions for various P_{LR}'s.

be seen in Figure 4.22. From the results, one can notice that PGC based method significantly outperform the CPM based methods.

Table 4.7: Simulated distortion results of the first scenario for various P_{LR} values. The distortion metric is relative L^2 error in units of 10^{-4}.

	Simulated distortion for different P_{LR}								
PLR values	0%	2%	4%	6%	10%	15%	20%	30%	40%
PGCStankovic	1.89	2.52	2.61	2.78	3.09	3.48	3.78	4.7	5.89
PGCmohr	1.85	2.42	2.62	2.66	3.06	3.32	3.87	4.55	6.67
AlRegib kStep=2	6	7.90	8.53	8.78	9.91	11.06	12.30	14.21	17.16
AlRegib kStep=5	6	7.94	8.55	8.83	10.00	11.08	12.36	14.32	17.29
AlRegib kStep=5 Quad	6	7.94	8.55	8.83	10.17	11.08	12.36	14.32	17.29
Ahmad	6	7.92	8.54	8.82	10.01	11.11	12.35	14.30	17.28
Ahmad Quad	6	7.98	8.55	8.83	10.04	11.16	12.40	14.35	17.28

Bunny Model Coded at 1.2 bpv

In this case, we decrease the bitrate and code Bunny model at 1.2 bpv to observe low bitrate error resilient characteristics. Comparison of PGC and CPM based methods is provided in Figure 4.23. We observe that the results do not change by decreasing the coding bitrate.

Venus Model Coded at 4 bpv

Finally in this case, we repeat the experiments performed on *Bunny* model with *Venus head*

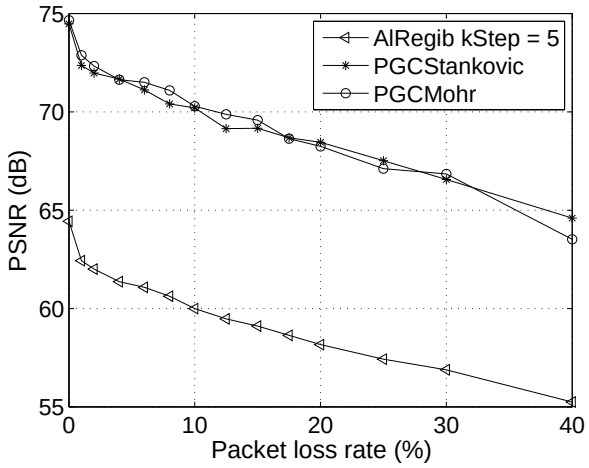


Figure 4.22: P_{LR} vs Simulated distortion in PSNR scale for *Bunny* model coded at 3.5 bpv.

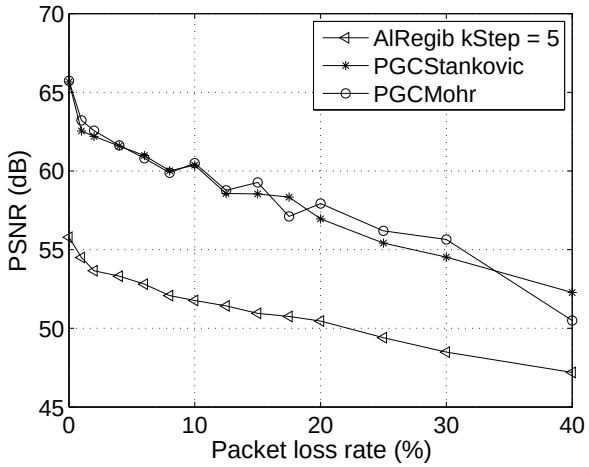


Figure 4.23: P_{LR} vs Simulated distortion in PSNR scale for *Bunny* model coded at 1.2 bpv.

model which is coded at 4 bpv and packetized with $N = 100$. Simulated PSNR comparison of PGC based and CPM based methods can be seen in Figure 4.24. Similar to the results with the *Bunny* model, we observe that PGC based methods again significantly outperform the CPM based methods.

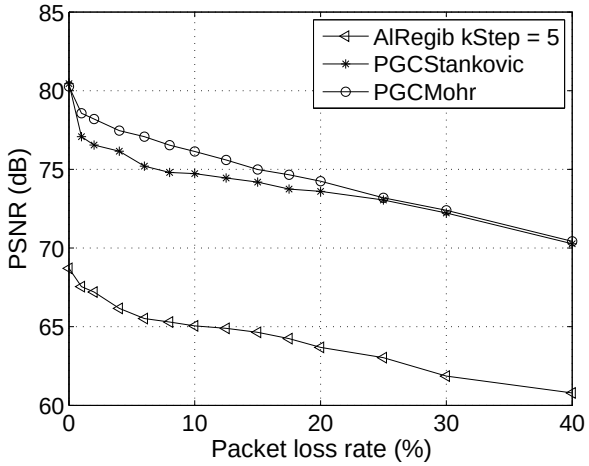


Figure 4.24: P_{LR} vs Simulated distortion in PSNR scale for *Venus head* model coded at 4 bpv.

4.5.5.5 Mismatch Scenario

In the experiments presented so far, the assumption is that during the optimization we know the channel loss rate and optimize the protection parameters accordingly. However, in real cases, the channel packet loss rate used during the optimization and the actual loss rate encountered may differ. Therefore in this part, we investigate what happens when a model optimized for a loss rate is transmitted over a channel with a different lost rate. The results of this experiment are shown in Figure 4.25.

The first observation in the figure is that when the transmission packetization is optimized for a low loss rate and a channel with a higher loss rate is encountered, the performance degradation can be severe. On the other hand, when the model encounters a channel with a lower loss rate, the performance loss is not significant. Another observation is that both CPM and PGC based methods behave similarly in mismatch scenario and the performance gap between the methods does not vary.

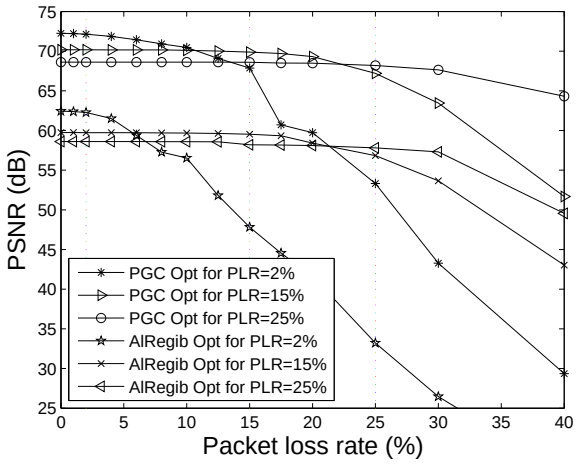


Figure 4.25: *Bunny* model is coded at 3.5 bpv and FEC assignment is optimized with respect to three different P_{LR}'s for *PGCStankovic* and *AlRegib kStep = 5* methods. Performance of the three different assignments for various P_{LR}'s in terms of simulated distortion in PSNR scale.

4.5.5.6 Complexity Comparison

In the packet loss resilient 3D mesh transmission frameworks presented in this chapter, it should be noted that the methods are not suitable for real time transmission. The reason is that the algorithms need to compress the model first and obtain D-R curves, which require significant amount of time. However, if we are allowed to compress models and store D-R curves offline, then real time transmission may be possible unless the time spent during the optimization, so called optimization time, is high. In this case, the optimization time of an algorithm becomes an important measure for real time transmission.

In order to compare the complexities of the optimization parts of the methods, we measured the optimization time for each algorithm as described at the beginning of Section 4.5.5. Table 4.8 summarizes time requirements for different optimization schemes mentioned in this chapter.

Table 4.8 shows that *PGCStankovic* has the smallest complexity. However, as seen in previous results, the complexity is smaller than that of *PGCMohr* at the cost of occasional slightly worse simulated PSNR. Examining the CPM based methods, we see that *Ahmad* and *AlRegib kStep=5* show significantly higher complexity. However, from $kStep = 8$, the complexity values are comparable with those of PGC based methods.

Table 4.8: Optimization times of different methods. Each method is optimized for P_{LR} = 4%

Optimization Times		
Methods	Time (sec)	PSNR
PGCMohr	0.250	71.63
PGCStankovic	0.004	71.68
Ahmad	11.110	61.37
AlRegib kStep=5	8.350	61.36
AlRegib kStep=8	0.365	61.35

4.5.5.7 Visual Comparison of CPM and PGC Based Methods

Apart from the objective distortion metric results, we also present results of *Bunny model coded at 3.5 bpv* and *Venus head model coded at 4 bpv* in terms of visual reconstructions. Figures 4.26 and 4.27 show visual reconstructions for *Bunny* and *Venus head* model for various P_{LR} values, respectively.

4.6 Conclusions

In this chapter, we have presented various MDC and packet loss resilient coding techniques for static 3D meshes. In the literature, there was only one work related to MDC of 3D static meshes [76]. We have proposed three MDC methods, namely *Multiple Description Scalar Quantization Based Approach* [2], *Partitioning Wavelet Coefficient Trees Based Approach* [1] and *Forward Error Correction Based Approach* [5] [4]. To compare the techniques, all of them except the one in [76] are based on wavelet coding. Hence better compression ratios and bitrates for descriptions are obtained in our wavelet based schemes. Also neither the work in [76] nor *MDSQ* based scheme in Section 4.2 does not employ any optimization with respect to varying bandwidth and loss rate of the channel while the *TM-MDC* method in Section 4.3 and *FEC* based method in Section 4.4 make use of optimization to significantly improve expected distortion performance. Moreover, number of descriptions and description sizes can be adjusted more flexibly in *TM-MDC* and *FEC* based methods.

To compare *TM-MDC* and *FEC* based methods, although the methods show similar performance in terms of expected distortion, *FEC* based method has several advantages. While the *FEC* based method generates one compressed bitstream to optimize FEC assignment for

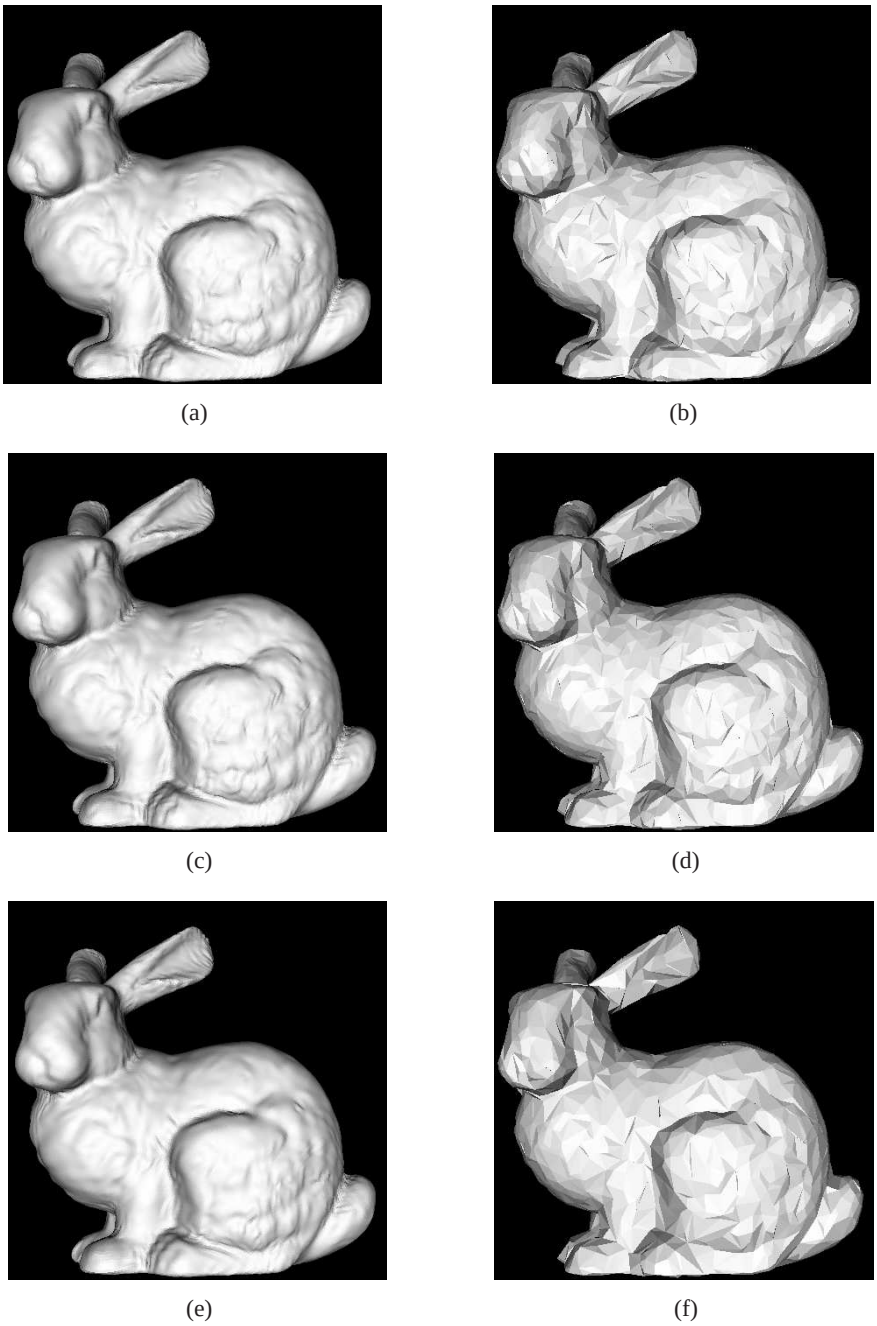


Figure 4.26: Expected reconstructions of the *Bunny* model, left column by *PGCStankovic* method and right column by *AlRegib kStep=5* method. (a)-(b): P_{LR} = 2%, (c)-(d):P_{LR} = 10%, (e)-(f):P_{LR} = 20%. PSNR values: (a) 72.18 dB (b) 62.44 dB (c) 70.52 dB (d) 61.57 dB (e) 69.09 dB (f) 58.61 dB

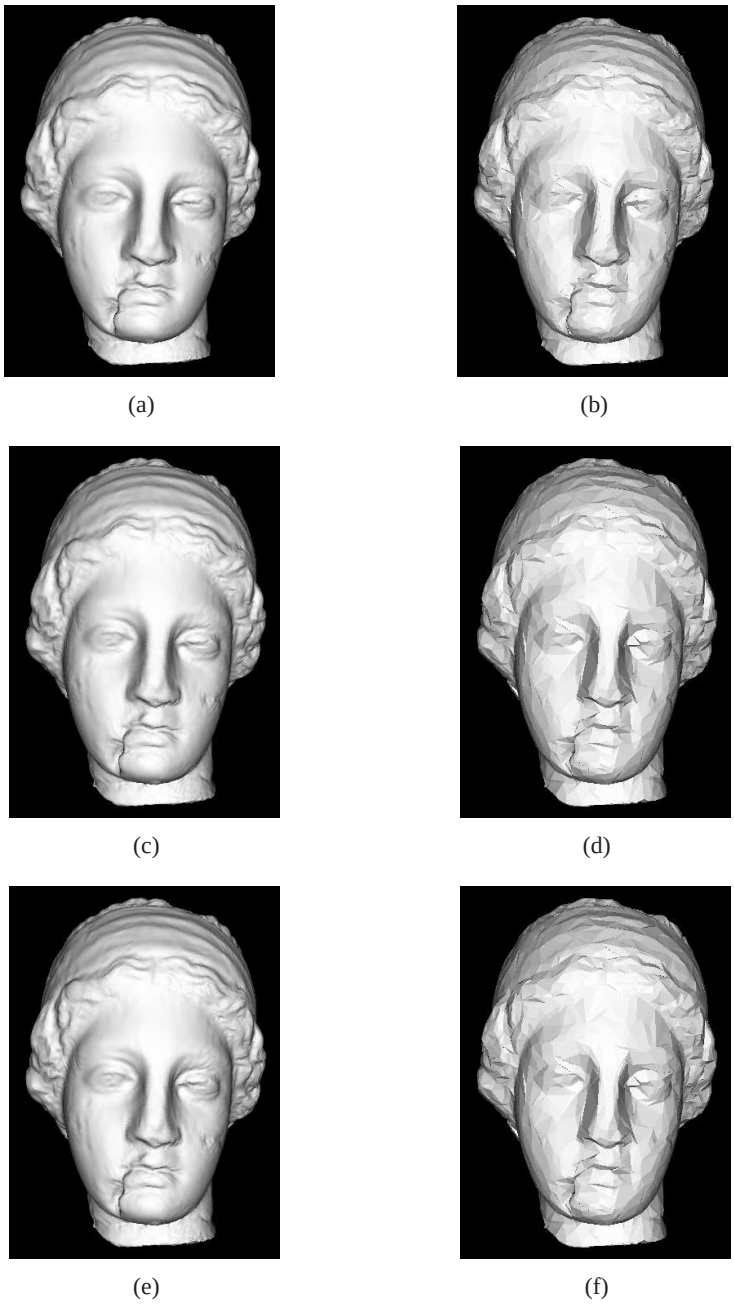


Figure 4.27: Expected reconstructions of the *Venus head* model, left column by *PGC-Stankovic* method and right column by *AlRegib kStep=5* method. (a)-(b): P_{LR} = 2%, (c)-(d):P_{LR} = 10%, (e)-(f):P_{LR} = 20%. PSNR values: (a) 76.95 dB (b) 67.79 dB (c) 75.04 dB (d) 65.19 dB (e) 74.24 dB (f) 65.19 dB

different total number of descriptions, *TM-MDC* needs to generate different compressed bitstream to optimize for different total number of descriptions. The reason is that the set of wavelet coefficient trees to be assigned to descriptions differs with respect to different total number of descriptions. Therefore *FEC* based method can produce any number descriptions using the same compressed bitstream whereas *TM-MDC* method needs to have the knowledge of total number of descriptions to be able to generate bitstreams for optimization of bit allocation for each description. Another advantage of *FEC* based method is that *TM-MDC* needs to include whole coarsest level geometry in each description whereas *FEC* based method spreads the bitplanes of coarsest level geometry according to their importances in compressed bitstream. In this way more important compressed wavelet coefficients are assigned more FECs than lower bitplanes of coarsest level geometry and these bitplanes are not included in each description unless they are assigned repetition codes by optimization algorithm.

As mentioned before, our proposed *FEC* based method is also used for packet loss resilient streaming purposes. Regarding to this part, we have presented an extensive analysis of loss resilient coding methods for 3D models. The methods are based on optimally protecting compressed bitstreams with FEC codes with respect to given channel bandwidth and packet loss rate constraints.

We first examined the CPM based methods reported in the literature and came up with a general problem definition and solution. We introduced a $kStep$ parameter to iterate protection rates with different steps and showed that increasing $kStep$ considerably decreases optimization times at the expense of very small PSNR degradation.

Then we compared CPM and PGC based methods and experimental results show that PGC methods achieve approximately 10db better PSNR for all loss rates. It was already reported in [19] that, compression performance of PGC method is 10dB better than CPM method. In our results, we show that the 10dB compression performance gap between the methods is preserved in packet loss resilient transmission. For the same reason, expected reconstructed models of PGC method at the decoder have a better subjective quality than the ones of CPM method. We also note that the performance difference may depend on the coarseness of the model. Apart from the PSNR performance, PGC based methods have an advantage of flexible packetization. Since the bitstream of PGC method is embedded, the bitstream is generated only once. Given the channel and bandwidth conditions, the bitstream can be truncated to

desired bitrate precisely and FEC assignment is performed easily. The CPM based methods need to generate different bitstreams for different quantization values and number of bitrate values that can be achieved is limited.

Finally we simulated performance of optimization methods in a mismatch scenario i.e. the model is protected with FEC codes optimized for a given loss rate but transmitted through a channel with a different loss rate. We observed that when the model is optimized for a low loss rate and encounters a channel with a higher loss rate, the performance degradation can be severe. On the other hand, when the model encounters a channel with a lower loss rate, the loss in the performance is not significant. Therefore we conclude that when the channel conditions are uncertain or time varying, it is more robust to optimize loss protection with respect to a higher loss rate.

CHAPTER 5

MULTIPLE DESCRIPTION CODING OF ANIMATED MESHES

5.1 Introduction

As presented in Section 3.2, recently many works have been proposed for compression of animated meshes in the literature. On the other hand, there is not much research on transmission and error resilient coding of 3D dynamic meshes. There is only one previous work [95], where the mesh is spatially partitioned into segments and each segment is encoded and protected by Forward Error Protection (FEC). The disadvantage of the method is that the loss of some parts of the bitstream means complete loss of corresponding spatial areas. In this chapter, we focus on MDC of animated meshes as the error resilient tool. Although there are many works related to MDC of video [13, 15, 96, 14, 97, 98] and MDC of static 3D meshes [76, 2, 1, 4, 99], MDC of animated mesh data is at a very early stage.

In this work, we propose three MDC schemes for animated 3D meshes. The main difference between the MDC of static and animated meshes is that the extra dimension of time in animated meshes brings additional challenges. The first proposed scheme is based on partitioning vertices spatially to be encoded independently. This scheme resembles the static mesh methods in [76] and [1] (TM-MDC) in the sense that spatial partitioning of the vertices is common in all the methods. However, the proposed method has the additional challenge of handling reduced compression performance due to inferior temporal prediction. The second proposed scheme is based on subsampling frames temporally by exploiting the hierarchical B frame structure, which makes it a completely a different approach compared to static meshes. The final proposed scheme makes use of the layered scalable structure and is based on du-

plicating layers in the descriptions. From the bitstream protection point of view, this scheme shows similarity to the static mesh approach in [4] (MD-FEC) with two descriptions.

The rest of the chapter is organized as follows. In Section 5.2, we mention the reference dynamic mesh coder on which the proposed MDC methods depend. In Section 5.3, we describe the proposed MDC methods. In Section 5.4, we present the experimental results and finally we conclude in Section 5.5.

5.2 Reference Dynamic Mesh Coder

The single description (SD) dynamic mesh coder on which our MDC methods is based is the scalable predictive dynamic mesh coder presented in [66]. The coder can also be viewed as the layered prediction structure of MPEG-4 FAMC [65] omitting any other modules like skinning-based motion compensation (SMC). We have chosen this coder as our reference coder because of both efficient compression performance and the layered structure which gives flexibility to the design of MD structures. Additional tools like SMC that improve the SD compression performance may also be incorporated but this may not necessarily affect the RRD performance for MDC, i.e. if a tool improving the SD compression improves the compression of multiple descriptions at the same level, then there is not much RRD performance difference. Since the primary focus of this work is to investigate and compare MDC performances of the proposed methods, we address the possible effects of adding tools that improve the SD compression performance on the MDC performance as a future work. The details of the coder, which is referred as *Reference Dynamic Mesh Coder* (RDMC) in the rest of the text, are provided in Section 3.2.1.

5.3 Proposed MDC Methods

In this section, we describe the proposed MDC methods for dynamic 3D meshes. We propose three methods which are named after the strategies they are based on:

***Vertex partitioning*:** Based on spatially partitioning the set of vertices into two sets for each frame.

***Temporal subsampling*:** Based on encoding odd and even frames separately.

***Layer duplication*:** Based on duplicating the bitstream corresponding to a subset of layers and splitting the remaining bitstream for each description.

A common property in all of the methods is that the compressed mesh connectivity is included in all descriptions. Since we deal with time consistent dynamic meshes whose connectivity does not change with time, including this data in each description causes very small and negligible overhead.

5.3.1 Vertex Partitioning Based MDC

5.3.1.1 MD Encoder

In this approach, the first step is to partition the set of all vertices into two disjoint sets where each set will correspond to a description. The partitioning strategy affects the MD performance since the missing vertices should be concealed from the received ones when only one of the descriptions is received. For example, an option is cutting the model into halves. In this case, the received vertices (the ones in the received half) would be compressed and decoded very efficiently. On the other hand, it would be very difficult to conceal missing vertices (the ones in the unavailable half) since most of the neighbor vertices would be missing. For this reason, we try to partition the vertices in both descriptions as dispersed as possible, which increases the chance of predicting the location of a missing vertex more accurately.

The problem of optimal vertex partitioning to achieve best MDC performance is a difficult problem. The exhaustive solution is unpractical as it requires too many combinations. Therefore we focus on a good enough solution which tries to achieve dispersed set of vertices rather than the optimal one. In the literature, the authors in [83, 84] propose an algorithm which provides 1D ordering of vertices, which has good locality and continuity properties. As a result, sampling the 1D array with odd and even samples generates two spatially dispersed sets of vertices. We tested the performance of this partitioning strategy and achieved quite acceptable MDC performance. Therefore, we adopt this strategy for the vertex partitioning part of the proposed MDC scheme. We also note that trying other heuristic ad-hoc partitioning methods resulted in slightly worse performance.

Note that the spatial layered decomposition in the RDMC and the vertex partitioning pre-

sented for this MDC method are two independent processes. The vertex partitioning occurs after the spatial layered decomposition and does not depend on the decomposition. The vertex partitioning is applied on the whole mesh vertices, not spatial layer by spatial layer. In our future work, we also plan to investigate a better vertex partitioning strategy which also takes into account the spatial layered decomposition.

Having obtained the two sets of vertices, in order to obtain the two descriptions, we encode the vertices using the RDMC either in a mismatch-free or mismatch-allowed manner, as described below. Note that, for these methods, we refer to the mismatch during decoding of the received set. Otherwise, mismatch and error propagation during decoding of the missing set is unavoidable.

Mismatch-free In order to avoid mismatch, we modify the reference dynamic mesh encoder structure so that when making a prediction for a vertex, only the neighbor vertices from the same set are allowed. In other words, no vertex from the other set is used for the prediction of the vertex. This strategy results in poorer predictions and consequently more bits are spent to encode the two sets separately compared to encoding the vertices as one set (or equivalently single description coding).

Mismatch-allowed In this case, we keep the existing prediction and reconstruction structure in the RDMC same. However, at the entropy coding stage, we entropy encode only the quantized residuals of corresponding vertices for each description. This method causes mismatch when the bitstream of only one description is to be decoded since both sets are used during predictions, which results in poorer side distortion. On the other hand, the redundancy due to poorer prediction accuracy is eliminated. However, this time the redundancy occurs due to poorer entropy coding as smaller amount of residuals are entropy encoded together. As it will be seen in experimental results, the resulting overall redundancy decreases compared to mismatch-free based method.

Rather than independent coding of disjoint vertex sets, in order to trade-off between redundancy and side distortion, we can encode a subset of spatial and/or temporal layers without partitioning using the RDMC and duplicate the resultant bitstream in both descriptions. In this way, redundancy is introduced to increase the accuracy of vertex location predictions both for encoding and concealment. In some cases, even the resulting redundancy may decrease if

the bitrate saved by better compression due to higher accuracy predictions exceeds the extra bitrate spent for duplicating the layers. For example, we always duplicated the encoded spatial base layer in both descriptions because the experimental tests showed unacceptably poor results when the spatial base layer was partitioned into two sets too.The points on the RRD curves of this algorithm presented in Section 5.4 correspond to various number of spatial and temporal layers duplicated and the resulting average side distortions.

5.3.1.2 MD Side Decoder

When only one of the descriptions is received, the available information is compressed prediction residuals of the corresponding vertex set and the duplicated spatial/temporal layers. The general idea in side decoding is to decode received vertices and estimate/conceal the lost vertices by making use of available vertices.

As the concealment algorithm, we make use of the prediction structures which are already available from the reference dynamic mesh decoder as it is used to minimize the prediction error for compression purpose. In the decoding process of an arbitrary frame, vertices in the frame are decoded in the order of spatial base layer to finest level spatial layer. After the spatial base layer is decoded, the vertices in the following spatial layers are predicted using the vertices from already decoded spatial layers and prediction errors are corrected by using the information from compressed bitstream. However, in the MD side decoder, only a subset of the prediction errors are available since the rest of them are lost with the lost description. Therefore, we assume that prediction errors were encoded as zero and estimate the locations of missing vertices as the predicted locations in the decoder. Note that during the concealment, it is not required to use the same prediction method used in the encoder since mismatch is unavoidable. Therefore either RIC or No-RIC based prediction can be employed.

We can summarize the side decoding process of mismatch-free MDC for each frame in four steps:

1. Use the RDMC decoder for the duplicated spatial/temporal layers.

2. Use the RDMC decoder for the vertices from the received description. Note that only the restricted neighbors from the same description are used during spatial prediction.

3. Set prediction error residuals of unavailable vertices to zero.

4. Use the RDMC decoder for the unavailable vertices. Note that in this case, there is no restriction on the neighbors used during spatial prediction since it is already impossible to match the prediction in the encoder due to losses. We observed that using all the neighbors during prediction performs the best in the experiments. Also either RIC or No-RIC based prediction can be employed.

Similarly, we can summarize the side decoding process of mismatch-allowed MDC for each frame in four steps:

1. Use the RDMC decoder for the duplicated spatial/temporal layers.

2. Entropy decode the received description and obtain dequantized prediction error residuals.

3. Set prediction error residuals of unavailable vertices to zero.

4. For all the vertices, run the RDMC decoder procedures that follow the entropy decoding.for all vertices. Either RIC or No-RIC based prediction can be employed for the unavailable vertices while for the received vertices, same predictions as in the encoder are used.

5.3.1.3 MD Central Decoder

The central decoding is performed when both descriptions are available. In the mismatch-free version, first the bitstreams of the descriptions are decoded separately and then the separately reconstructed vertices are combined. In the mismatch-allowed version, first the bitstreams are entropy decoded separately and prediction error residuals are obtained after dequantizing. Then the prediction error residuals from the two descriptions are combined and the vertices are reconstructed using the reference dynamic mesh decoder.

5.3.2 Temporal Subsampling Based MDC

5.3.2.1 MD Encoder

In this approach, the mesh frames are split into two sets where each set corresponds to a description: The first set contains even indexed frames (Frames 0, 2, 4 ... assuming that the indexing starts from 0) and the second set contains the odd indexed frames (Frames 1, 3, 5 ...). In this way, we obtain two new dynamic mesh sequences with a frame rate of half of the original sequence. Then each sequence is compressed using the RDMC and each description contains one of the compressed bitstreams. The total bitrate of the descriptions exceed the bitrate of single description coding because in the MD case, since the frames cannot predict from the frames in the other set, the interframe prediction efficiency decreases.

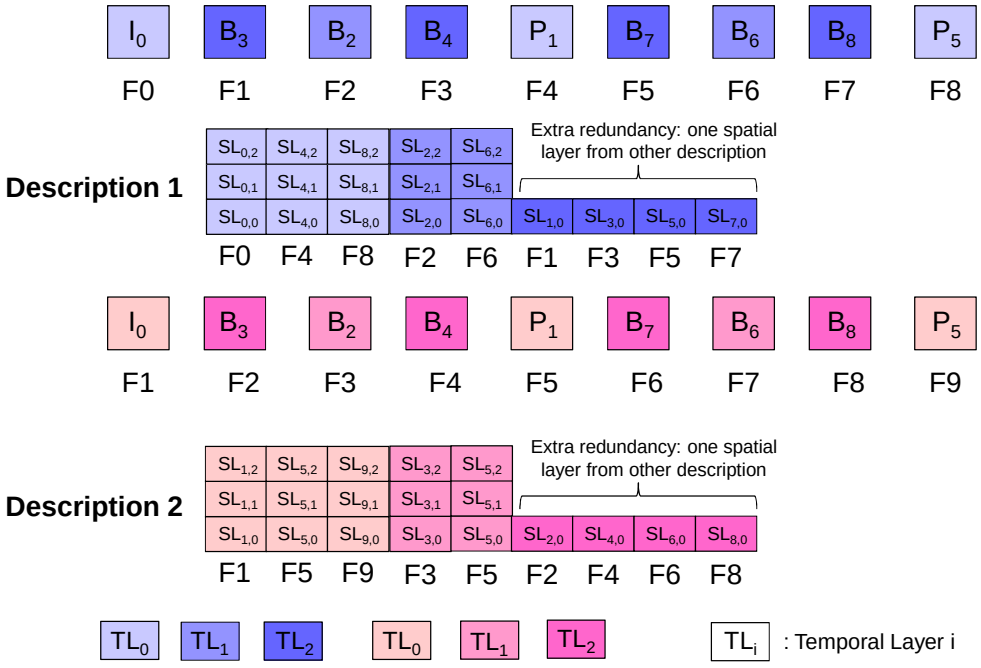


Figure 5.1: Temporal subsampling based MDC. The example sequence is decomposed into three temporal layers and three spatial layers. One spatial layer from the other description is duplicated.

The MD encoding strategy described above results in the minimum possible redundancy and maximum side distortion for this method. It is also possible to add more redundancy (increase in side bitrate) to decrease side distortions. This can be achieved by adding a subset of compressed spatial layers from the other set of frames. The extra number of bits for this procedure

is not very high because actually, the frames of the other set are the frames to be encoded in the last temporal layer in single description case (TL_2 in Figure 3.3).

One can notice the relation between the temporal subsampling based MDC and the temporal layered decomposition in the RDMC. To explain the relation with an example, assume that our sequence consists of 10 frames ($F_0, F_1, ..., F_8$ for description 1 and $F_1, F_2, ..., F_9$ for description 2) decomposed into three temporal layers and each frame is decomposed into three spatial layers as illustrated in Figure 5.1. Let TL_i denote the ith temporal layer. For the first description, initially, F_0, F_4 and F_8 in TL_0 are encoded and then F_2 and F_6 in TL_1 are encoded. At this point, one can notice that, we have encoded frames F_0, F_2, F_4, F_6 and F_8 or in other words, we have encoded even indexed frames without predicting from odd indexed frames. This situation corresponds to the lowest possible redundancy case. If we wish to add more redundancy, we can continue encoding spatial layers from the remaining TL_2 which consists of frames F_1, F_3, F_5 and F_7. The amount of redundancy is adjusted by the number of encoded spatial layers in TL_2 frames. For example, if we encode all of the spatial layers in TL_2 frames, then we have a redundancy of 100 % which is equivalent to repetition of the descriptions. In the example case illustrated in Figure 5.1, one spatial layer from the last temporal layer is included in both descriptions. The points on the RRD curves of this algorithm presented in Section 5.4 correspond to how many spatial layers of the last temporal layer are included in each description and the resulting average side distortion.

For the second description which corresponds to the odd indexed frames, the same procedures are applied with the exception that F_1 is considered as the starting frame of the original sequence as depicted in Figure 5.1. F_0 and F_9 (first and the last frames) are exceptional frames for description 1 and 2 respectively and can either be neglected or included in the encoding process by allowing to predict from the nearest frame.

5.3.2.2 MD Side Decoder

When only one of the descriptions is received, the received bitstream is fed into the RDMC decoder. First, all the temporal layers except the last one (which contains frames of the other set) are decoded in the usual way. Then, if any spatial layers of the last temporal layer (the extra spatial layers from the frames of the other description) exist in the received description, the vertices in these spatial layers are decoded in the usual way as well. Finally, for the missing

spatial layers, the decoder assumes that the prediction errors for the vertices in these spatial layers were found to be zero during encoding process. Therefore, the decoder predicts the vertex locations for the missing spatial layers using either RIC or No-RIC and the predicted locations are used as the concealed reconstructed locations.

Another possibility is using linear interpolation rather than using the prediction structure in the decoder. In this approach, the missing vertex locations are estimated as the average of corresponding vertex locations from the previous and next frames. In the experimental results, we compare the performances of using linear interpolation and the prediction structures of the decoder.

5.3.2.3 MD Central Decoder

When both of the descriptions are available, each description is fed into the reference dynamic mesh decoder separately. During the decoding, the decoding of the last temporal layer is discarded. In this way, decoding one of the descriptions generate only even indexed frames and the other generates the odd indexed frames. Finally, the even and odd indexed frames are combined to generate the mesh sequence at full frame rate.

5.3.3 Layer Duplication Based MDC

5.3.3.1 MD Encoder

The idea of the method is as follows: Initially, the input mesh sequence is compressed with the RDMC such that the output bitstream consists of two meaningful sub-bitstreams appended head to tail. The first sub-bitstream is self decodable and decoding results in a lower quality reconstruction which corresponds to a lower spatial or temporal resolution. The second sub-bitstream is not self decodable but when it is decoded together with the first sub-bitstream, it improves the quality of the first sub-bitstream decoding. Therefore the second sub-bitstream depends on the first sub-bitstream to be decoded. After the sub-bitstreams are generated, the first sub-bitstream is duplicated in both of the descriptions and the second sub-bitstream is cut into half so that each half bitstream is placed into a distinct description as illustrated in Figure 5.2.

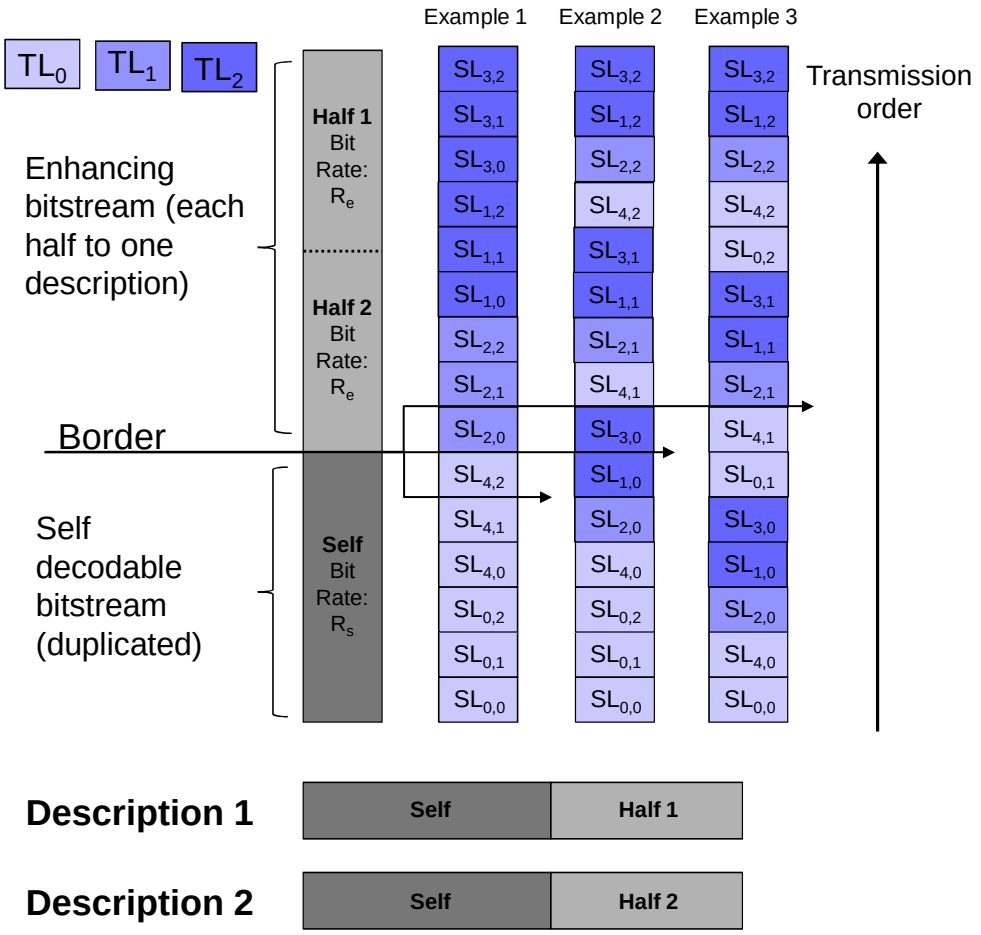


Figure 5.2: Layer duplication based MDC: three layer ordering examples

The amount of redundancy in this method is equal to the size of first sub-bitstream as it is duplicated in both descriptions. Therefore, by playing with the size of first sub-bitstream, the trade-off between redundancy and side distortion can be adjusted. Due to the layered structure (both temporal and spatial) of the RDMC, there are many possible ways to generate the first sub-bitstream. Figure 5.2 shows three examples for a sequence with 5 frames decomposed into 3 temporal and spatial layers. In general, assume that the RDMC performs T temporal and S spatial layer decompositions during encoding. The ordinary order of the output bitstream is concatenation of T encoded temporal layers where each encoded bitstream of a frame in a temporal layer is concatenation of S encoded spatial layers from spatial base layer to finest level layer. Let $TL_i(s)$ represent the bitstream of ith temporal layer in which each frame is encoded up to sth spatial layer out of S spatial layers. Then the ordinary bitstream

order is $[TL_0(s=S)]$ $[TL_1(s=S)]$ ... $[TL_{T-1}(s=S)]$. However, this ordering is not unique. In the more general case, it is possible to encode S_i spatial layers of temporal layer TL_i for $i = 0, 1, .., T-1$ initially and define the resulting bitstream as first sub-bitstream (or self decodable sub-bitstream). Afterwards, we can encode remaining $S - S_i$ spatial layers of each temporal layer TL_i for $i = 0, 1, .., T-1$ and define the resulting bitstream as second sub-bitstream (or enhancing sub-bitstream) which is appended after the first sub-bitstream. Then the general ordering of the bitstream becomes: $[TL_0(s=S_0)]$ $[TL_1(s=S_1)]$ ... $[TL_{T-1}(s=S_{T-1})]$ (sub-bitstream border) $[TL_0(s=S)-TL_0(s=S_0)]$ $[TL_1(s=S)-TL_1(s=S_1)]$ ... $[TL_{T-1}(s=S)-TL_{T-1}(s=S_{T-1})]$ where $[TL_i(s=S)-TL_i(s=S_i)]$ denotes the remaining spatial layers from first sub-bitstream in ith temporal layer (spatial layers from S_{i+1} to S). It is important to note the constraint $S_0 \geq S_1 \geq ... \geq S_{T-1}$ because a vertex at a spatial layer needs to predict from a vertex at the same spatial layer which was encoded in the previous temporal layer.

5.3.3.2 MD Side Decoder

The MD side decoder receives only one of the descriptions. As mentioned earlier, a description contains a self decodable sub-bitstream and half of the sub-bitstream which enhances the quality of the first sub-bitstream. Since the whole enhancing sub-bitstream is unavailable, the received half of the enhancing sub-bitstream is discarded. The self decodable sub-bitstream is decoded by the usual reference dynamic mesh decoder and the resulting reconstruction is the side reconstruction of this MD system.

5.3.3.3 MD Central Decoder

The MD central decoder discards the duplicated self decodable sub-bitstream, combines the halves of the enhancing sub-bitstream and as a result obtains the whole single description coded bitstream. Using the reference dynamic mesh decoder, the central reconstruction is obtained.

5.3.4 Comments on the Mismatch

All the proposed methods can be considered mismatch-free except the mismatch-allowed version of vertex partitioning based MDC. In the mismatch-free version, the vertices in a set

are disallowed to predict from vertices in the other set. In the temporal subsampling method, the description of even indexed frames do not predict from the odd indexed frames and vice versa. In the layer duplication algorithm method, since half of the enhancing sub-bitstream is discarded and the self decodable bitstream is duplicated, there exists no mismatch.

5.3.5 Further Possible Improvements on Error Concealment

For the missing vertices, the side decoder may choose either using or not using the rotation-invariant coordinates during the prediction for error concealment. However, for different frames, the performance of the predictions differ. Therefore, we add a property in encoder so that, the encoder calculates the resultant error of each frame for all prediction methods and writes the index of better prediction method for each frame in the bitstream. In this way, the decoder uses this information to decide on which prediction method to use for concealment. This process slightly increases encoder complexity and results in a negligible bitrate overhead.

We note that the inherent prediction structure in the RDMC for the error concealment in the side decoder may not be the optimal solution. Any other sophisticated technique like [100] which exploits the already available connectivity data can be employed to estimate missing vertices. In some cases, a lower distortion may be achieved compared to current error concealment predictions. Furthermore, as described in the previous paragraph, the encoder may test several other concealment techniques and signal to the side decoder which technique brings the lowest distortion. On the other hand, using only the inherent prediction structures of the RDMC simplifies the codec design by omitting implementation of new modules for sophisticated estimators and brings significant gain in complexity. In our works, we use only the available prediction structures in the RDMC and address the incorporation of sophisticated estimators for the error concealment as future work.

5.4 Results

In this section, we evaluate and compare the performances of the proposed MDC methods. Since we aim to analyze MDC performance, we do not provide a comparison with the work in [95] where the MDC usage is not considered. In addition, the compression methods employed in [95] are less efficient than the one we employed. Therefore in a transmission scenario, it

is very likely that a comparison would favor the method employing a better compressor since the bandwidth can be used more efficiently. However, this comparison would be unfair since the performance difference would be mostly due to the compression efficiency difference.

RRD curves presented in the previous sections are used to evaluate MDC performances of the proposed methods. In the RRD curves, the redundancy (the additional bitrate introduced by MD coding) is given as the percentage of single description bitrate. The bitrate values are expressed in bits per vertex per frame (bpvf). To measure the side and central distortions of the reconstructed mesh sequences, we use the error metric which is defined in Karni and Gotsman's work [49] as it is a widely used metric in the literature. We denote this error by *KG Error* and it is calculated as shown in Equation 3.8.

We perform the experiments on the following test dynamic mesh sequences: *Cowheavy*, *Chicken crossing*, *Dance*, *Horse Gallop*, *Face* and *Jump*. The properties of the sequences are shown in Table 6.1. We used the following coding parameters for all models: *number of temporal layer decompositions* (nTL)= 4, *number of spatial layer decompositions*(nSL)= 8, *quantization parameter*(Q)= 12 and using rotation invariant coordinates in the encoder. Q = 12 is usually regarded as the visually lossless quantization parameter. These settings result in the following central KG error (D_0) - central bitrate (R_0) pairs: 0.0349% at 10.41 bpvf for *cowheavy*, 0.0432% at 4.79 bpvf for *chicken crossing*, 0.0422% at 4.24 bpvf for *dance*, 0.0435% at 6.91 bpvf for *horse gallop*, 0.0537% at 9.323 bpvf for *face* and 0.0426% at 4.95 bpvf for *jump*.

Table 5.1: The test sequences

Name	# Vertices	# Triangles	Frames used
Cowheavy	2904	5804	1-204
Chicken crossing	3030	5664	1-400
Dance	7061	14118	1-201
Horse Gallop	8431	16843	1-48
Face	539	1042	1-200
Jump	15830	31660	431-652

In the following subsections, we first introduce the results of each algorithm individually using the *cowheavy* sequence as we obtained very similar individual results with the other sequences. Then we compare each MDC method for all the test sequences. In the results and figures, we use the following abbreviations for the related MDC parameters:

nTLDup Number of temporal layers duplicated in both descriptions to adjust redundancy and side distortion.

nSLDup Number of spatial layers duplicated in both descriptions to adjust redundancy and side distortion.

Ric Rotation-invariant coordinates based prediction of the coder is used for concealing the missing vertices in the side decoder.

No Ric Rotation-invariant coordinates are not used during the prediction of missing vertices in the side decoder.

Best Both Ric and No Ric are tested in the encoder for side distortion and the one with lower error is used during concealment.

VP_{MF} Vertex partitioning based MDC method with mismatch-free version.

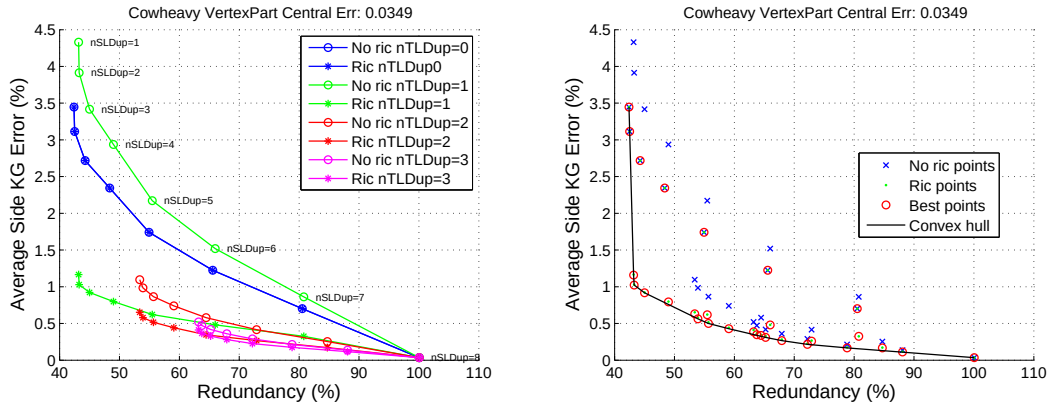
VP_{MA} Vertex partitioning based MDC method with mismatch-allowed version.

TS Temporal subsampling based MDC method.

LD Layer duplication based MDC method.

5.4.1 Vertex Partitioning

As stated in Section 5.3.1, there are two ways to vary redundancy in VP_{MF} and VP_{MA}. First one is duplicating a number of spatial layer and the other one is duplicating a number of temporal layers in each description. Figure 5.3(a) shows the effect of varying number of duplicated layers. We observe that when no temporal layer is duplicated, Ric and No Ric perform the same since the temporal prediction cannot be made use of. However, with increasing the number of duplicated temporal layers, we observe significant improvement with Ric. Ric results in lower error than No Ric since especially for frames having large separation, same global spatial error correction assumption is weak but it is more valid for local errors. As seen in the figure, many RRD points can be obtained by duplicating different number of layers. In order to compare the method with other MDC methods, we find the convex hull of the possible points as shown in Figure 5.3(b). In this figure, we also show the Best points and it can be observed that the Best points are almost always Ric points.



(a) (b)

Figure 5.3: (a)Vertex Partitioning based MDC: RRD curves for varying number of duplicated layers (b) All achievable RRD points and their convex hull

In Figure 5.4, the RRD comparison of mismatch-free and mismatch-allowed based methods are shown. It can be observed that mismatch-allowed curve is a shifted version of the mismatch-free curve with a higher side distortion but lower redundancy. However the effect of lower redundancy is much more significant and allows for low redundancy allocations.

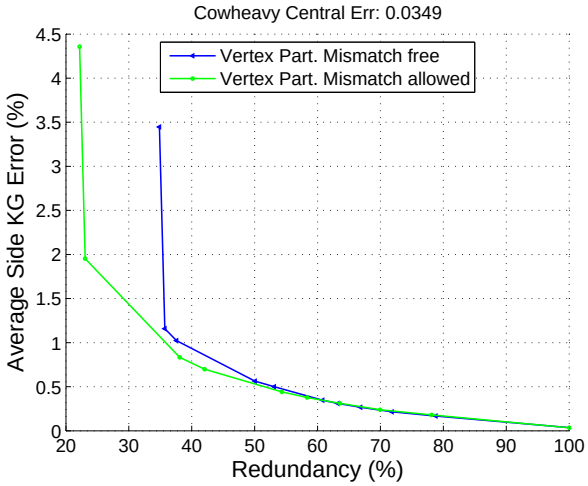
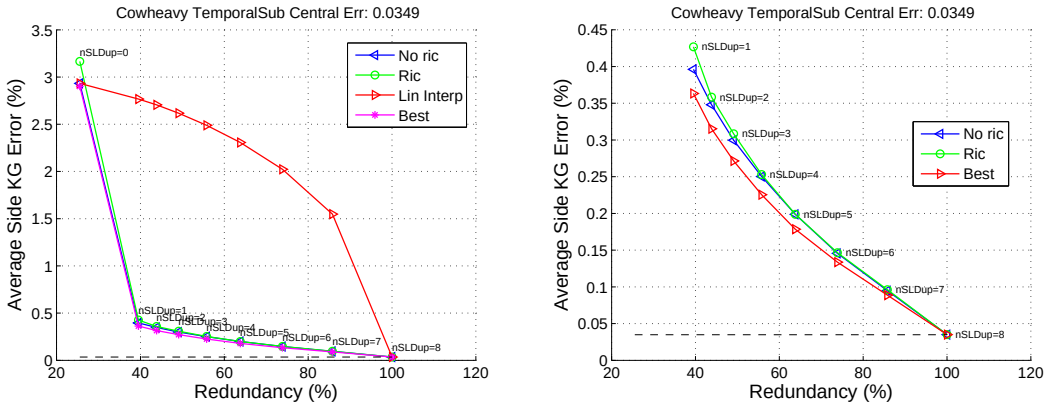


Figure 5.4: Comparison of mismatch-free and mismatch-allowed vertex partitioning based MDC

5.4.2 Temporal Subsampling

Figure 5.5(a) shows the RRD curves of different prediction methods including simple linear interpolation. The first observation is that using the prediction structure of the coder is significantly better than linear interpolation. Another observation is that duplicating the spatial base layer (transition from $nSLDup = 0$ to $nSLDup = 1$) results in the most remarkable side distortion improvement. Further but smaller side distortion improvements can be achieved by duplicating more spatial layers at the expense of increasing redundancy. Figure 5.5(b) shows the zoomed view of the Figure 5.5(a). It is observed that, there is no significant difference between Ric and No Ric and Best slightly improves the performance.



(a) (b)

Figure 5.5: Temporal Subsampling based MDC: (a) RRD curves for different prediction methods (b) Zoomed view

5.4.3 Layer Duplication

As shown in Figures 5.2 and 5.6, there are numerous redundancy-side error pairs that can be achieved by the *LD*. The red and green points on the Figure 5.6 correspond to all sub-bitstream partitioning possibilities obtained by different layer orderings using No Ric and Ric respectively. Most of the points on the figure are useless since a lower side distortion can be achieved with the same redundancy ratio. For this reason, we find the convex hull of the points and use it as the RRD curve of this method during the comparison. In this way, we obtain the best achievable performance of the layer duplication based MDC. From the complexity point of view, a faster algorithm to compute the convex hull without calculating all possibilities

is required. Nevertheless, since we concentrate on the achievable MDC performance in this work, we address the issue as a future work.

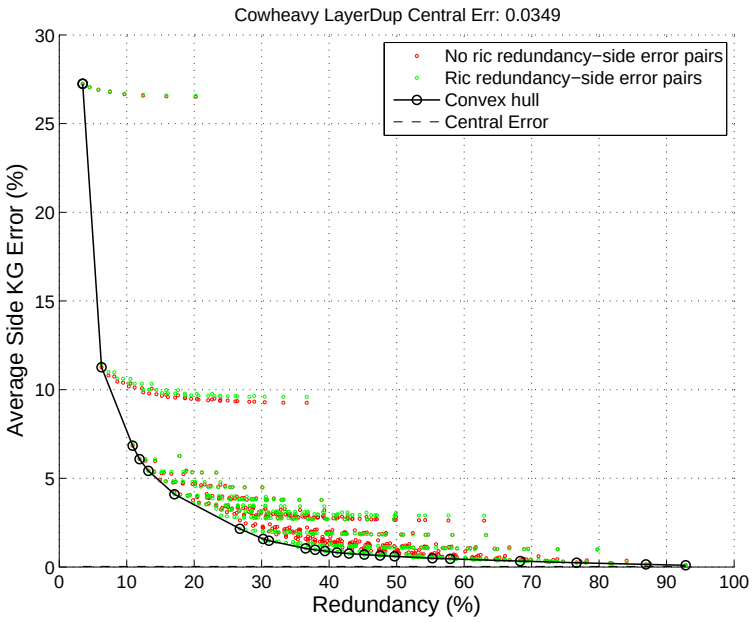


Figure 5.6: Layer Duplication based MDC: All redundancy-side error pairs and the convex hull

5.4.4 Comparison

Figure 5.7 shows the comparison of each proposed MDC method for the test mesh sequences. Each method uses the Best prediction during concealment.

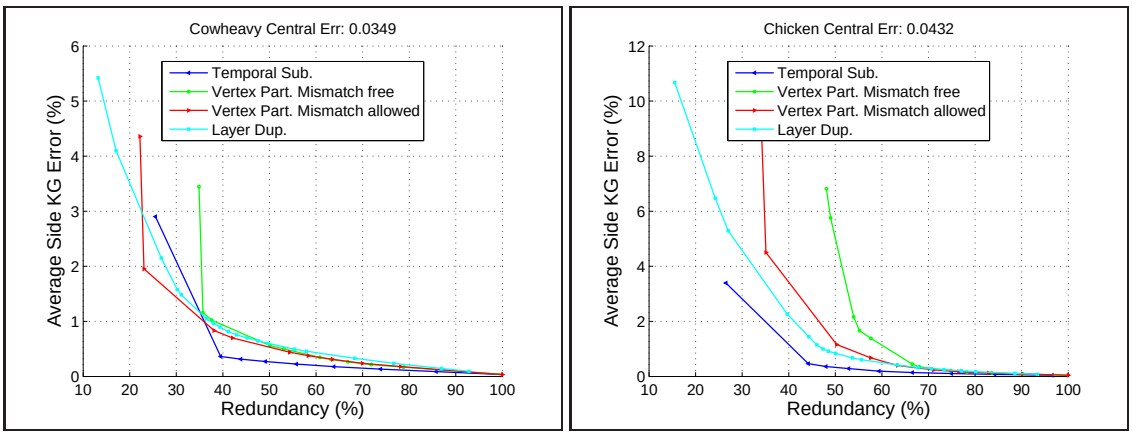
VP_{MF} always performs the worst, which means allowing mismatch for better predictions during encoding is a better approach than restricting the vertex neighbors during prediction to avoid mismatch.

LD can achieve very low redundancies other methods cannot achieve. However, with increasing redundancy, the method is outperformed by TS and VF_{MA}. On the other hand, due to numerous possible layer orderings, the desired redundancy can be achieved with a better accuracy. The other methods cannot produce as many RRD points as the *LD* produces.

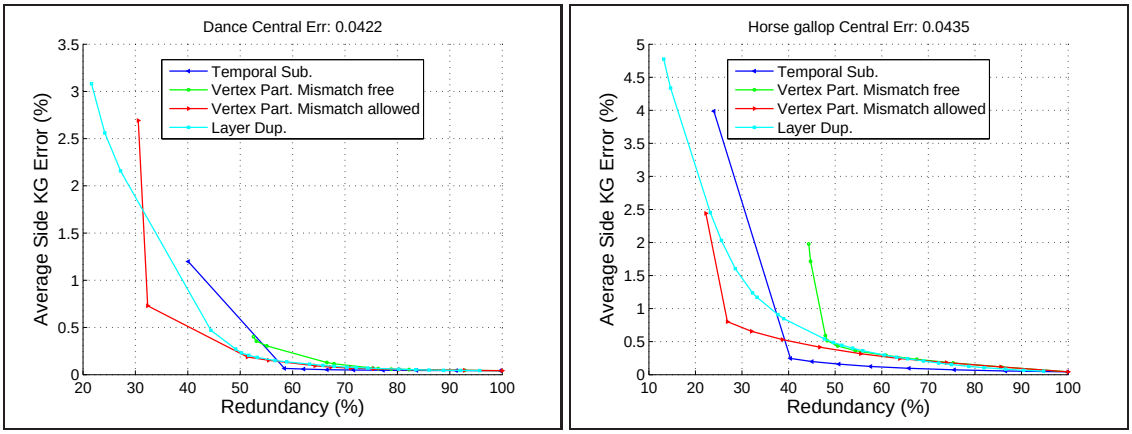
VP_{MA} performs better at low redundancies except for the models chicken crossing and face. The reason is that performance of the VP_{MA} decreases with spatially coarser models. Considering the lower spatial prediction accuracy in VP_{MA} due to partitioned vertices, the spatial

prediction is affected substantially when the model is spatially coarse. For example, face is a very coarse model with only 539 vertices per frame. On the other hand, although cowheavy and chicken crossing models have similar number of vertices, vertex part. mismatch allowed performs well with cowheavy but worse with chicken crossing. The reason is that chicken crossing model is composed of many coarse tiny parts like the wing, causing sharp and non-smooth regions and reduced spatial prediction accuracy which is also shown in Figure 5.11.

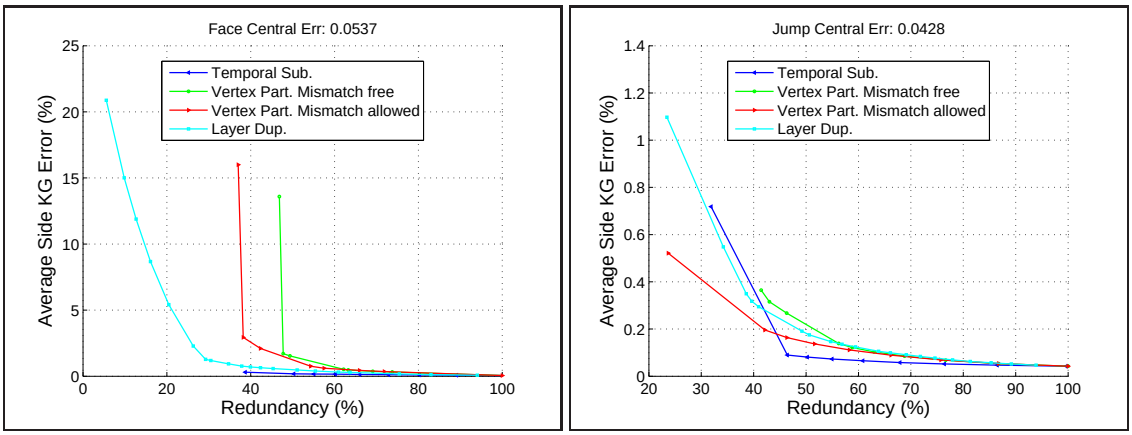
Increasing the redundancy from low to moderate regions, we observe that TS almost always performs the best. Note that these regions correspond to the cases where the spatial base layer is already included in both descriptions for the TS method. For high redundancy values, the MDC methods perform very similarly.



(a) Cowheavy (b) Chicken crossing



(c) Dance (d) Horse gallop

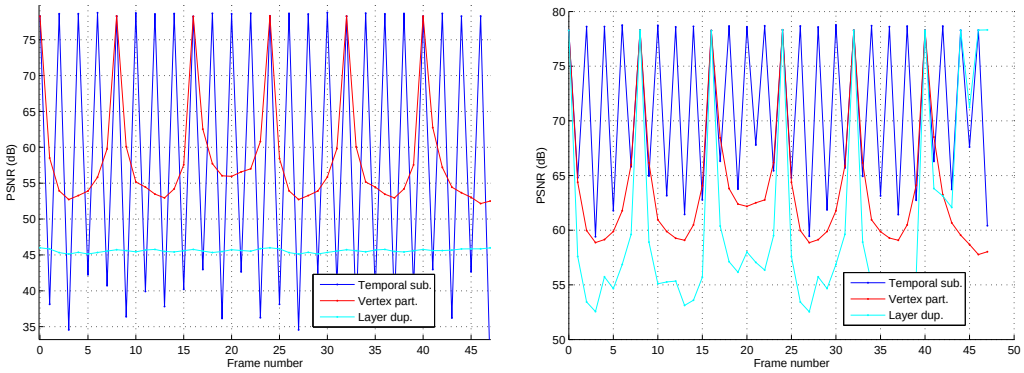


(e) Face (f) Jump

Figure 5.7: RRD performance comparison of the MDC methods

5.4.5 Per Frame Analysis

In the previous part, we evaluated the methods using the overall sequence whereas there may be disturbing effects for individual frames with respect to the MDC methods. In this part, we examine the per-frame performance of the MDC methods. Figures 5.8(a) and 5.8(b) show the quality of each reconstructed frame of horse gallop sequence MD coded at 25% and 45% redundancy respectively. In the figures, we express the quality in Peak Signal-to-Noise Ratio (PSNR) scale for better visualization. We calculate the PSNR value as $PSNR = 10log_{10}(MSE/bboxdiag^2)$ where MSE is the mean squared error between the reconstructed and original vertex locations and *bboxdiag* is the bounding box diagonal of the mesh sequence. For the same sequence and redundancy values, Figures 5.9 and 5.10 show visually the reconstructed frames 1 and 2 obtained by central decoding and side decoding of the first description. Figure 5.11 shows visually the reconstructed frames 259 and 260 of the chicken crossing sequence MD coded at 44%. Finally, Figure 5.12 shows the visualization of error between central and side reconstructions for horse gallop and chicken crossing sequences MD coded at 25% and 44% redundancy, respectively.



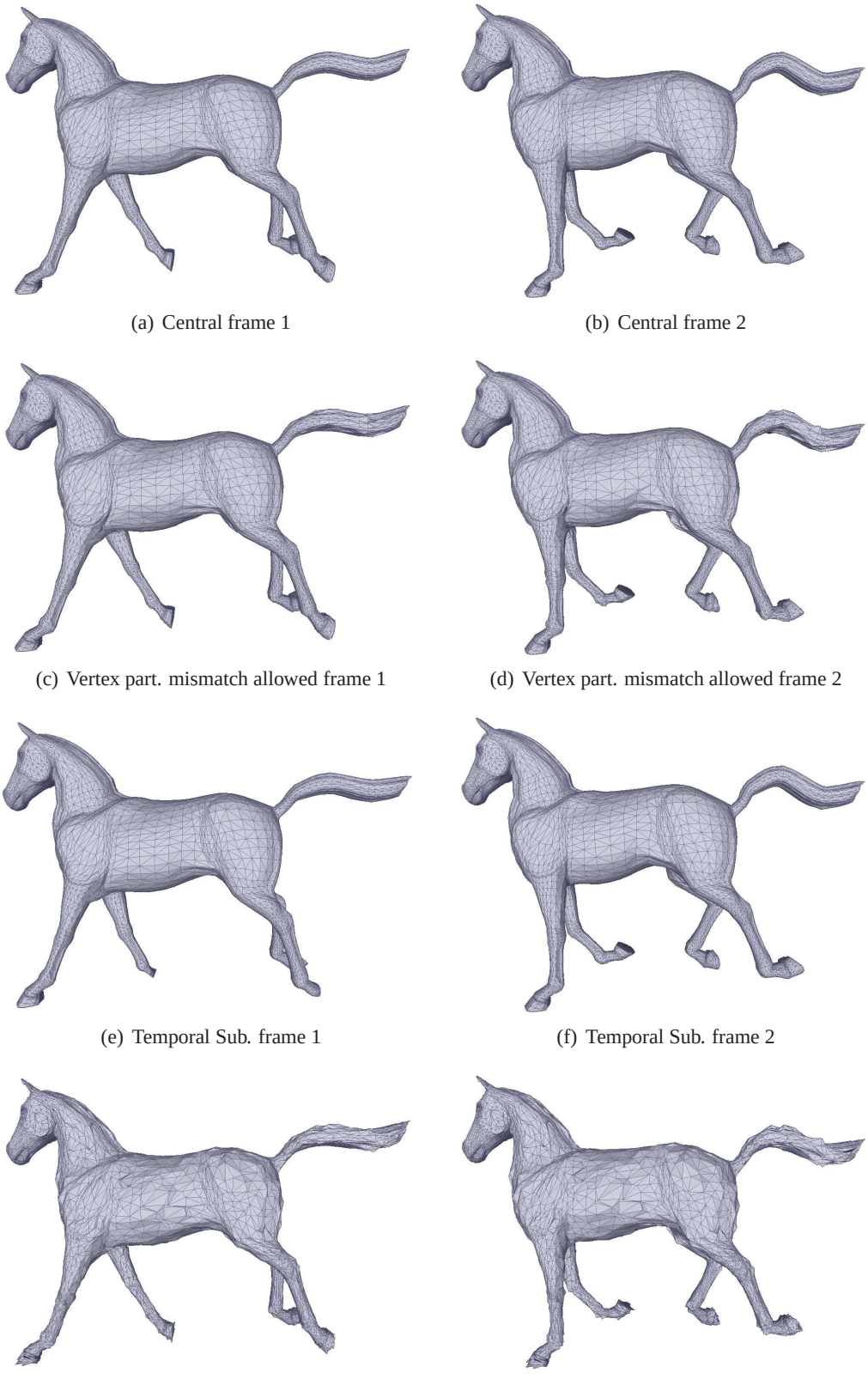
(a) (b)

Figure 5.8: PSNR values of side reconstruction of each frame for horse gallop sequence MD coded at (a) %25 redundancy, (b) %45 redundancy

Examining the figures, for the side reconstruction of the TS, even frames achieve a high PSNR (at central decoding quality) and odd frames achieve a lower PSNR as shown in Figure 5.8. This situation of quality difference between consecutive frames may cause significant problems depending on the content when watching the sequence. For the low redundancy case where the whole spatial base layer is not included in each description, comparing the TS with VP_{MA} and LD, it is observed that even frames have higher or equal PSNR values whereas

odd frames have lower PSNR values. However, VP_{MA} and *LD* achieve much smaller quality difference between consecutive frames. Considering both the reconstruction PSNRs and the inter-frame quality difference, VP_{MA} performs the best for the low redundancy case. On the other hand, when the redundancy is increased at least as much as including the whole spatial base layer for the TS, the PSNR values of TS frames increase significantly and always exceed VP_{MA} and *LD*. Note that although the frames of the TS achieve higher PSNR, the average quality difference between consecutive frames is still the highest. Final observation is that although the *LD* performs the worst PSNR performance for all cases, it has the capability to achieve minimum quality difference between consecutive frames. For example in Figure 5.8(a), the *LD* curve achieving very low difference can be seen. In Figure 5.8(b), a higher difference is observed. The reason is that, among many redundancy allocation possibilities for the *LD* method, the layer ordering with the minimum KG error is shown in the figure. However, it is also possible to choose a layer ordering with the same redundancy but lower inter-frame quality difference at the cost of slightly higher KG error.

The observations can also be seen in the visual Figures 5.9, 5.10, 5.11 and 5.12 showing reconstruction of two consecutive frames for the aforementioned cases. Several noticeable cases are as follows: The poor reconstruction quality of the *LD* with respect to the others can be seen in Figures 5.9(g) and 5.9(h). For the TS, the problems with the leg and foot area of the horse due to high motion can be seen in Figures 5.9(e) and 5.12(c). For the VP_{MA}, problems with the spatial regions can be observed in Figures 5.12(e) and 5.12(f).



(a) Central frame 1

(b) Central frame 2

(c) Vertex part. mismatch allowed frame 1

(d) Vertex part. mismatch allowed frame 2

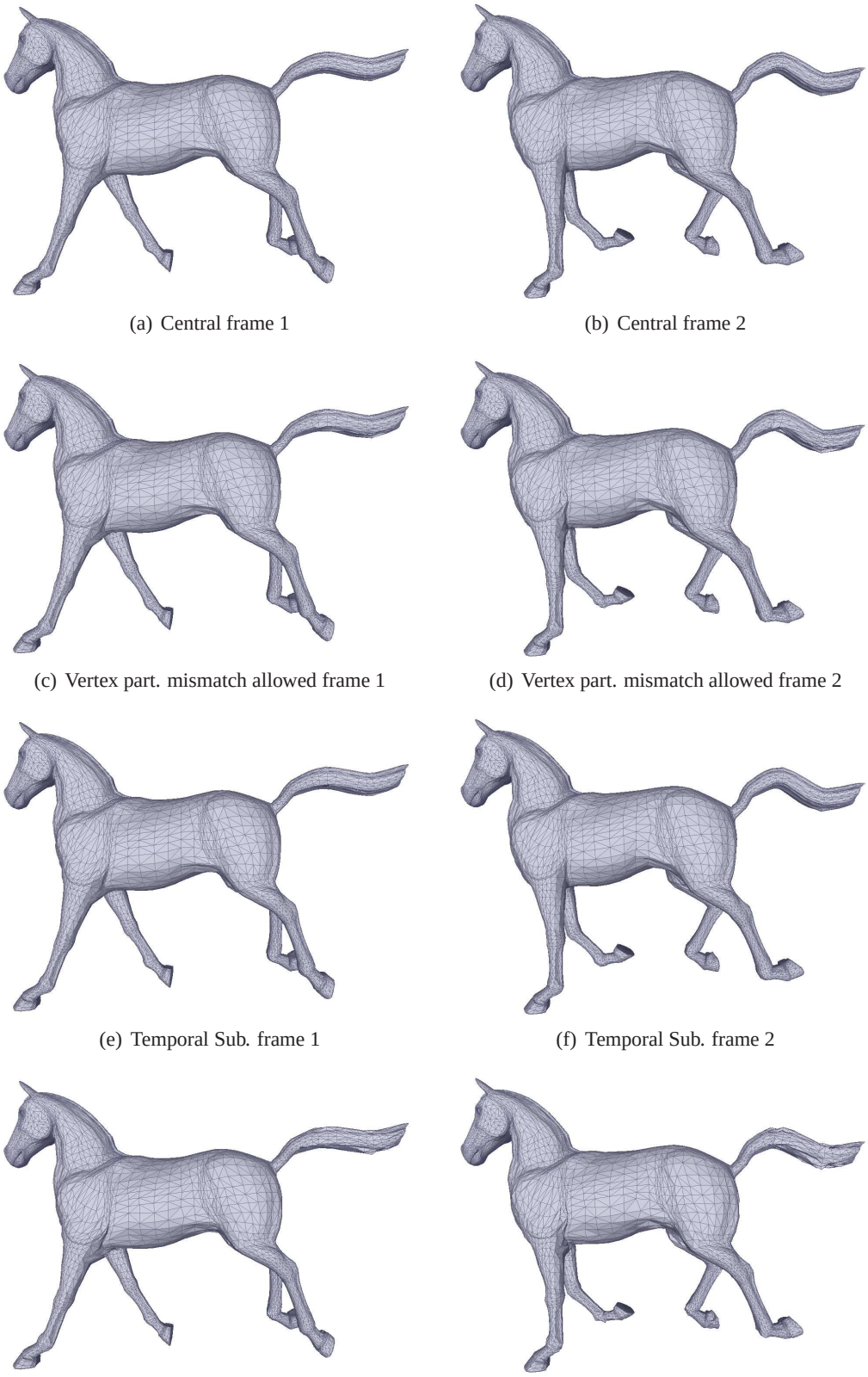
(e) Temporal Sub. frame 1

(f) Temporal Sub. frame 2

(g) Layer Dup. frame 1

(h) Layer Dup. frame 2

Figure 5.9: Central and side reconstructions of frames 1 and 2 of horse gallop sequence MD coded at 25% redundancy



(a) Central frame 1

(b) Central frame 2

(c) Vertex part. mismatch allowed frame 1

(d) Vertex part. mismatch allowed frame 2

(e) Temporal Sub. frame 1

(f) Temporal Sub. frame 2

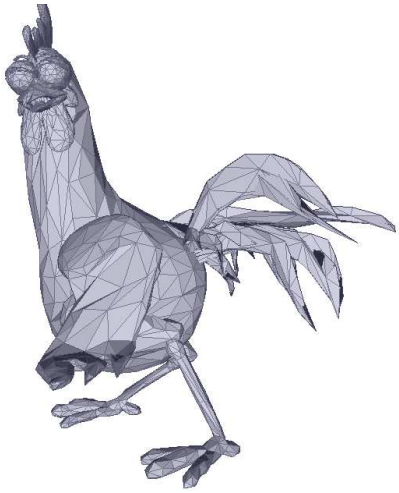
(g) Layer Dup. frame 1

(h) Layer Dup. frame 2

Figure 5.10: Central and side reconstructions of frames 1 and 2 of horse gallop sequence MD coded at 45% redundancy

(a) Central frame 259

(b) Central frame 260



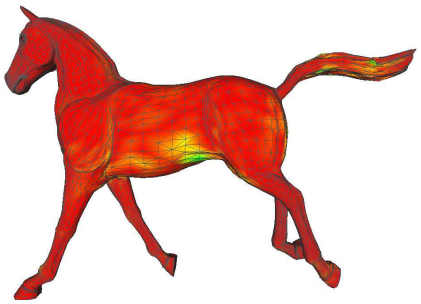
(c) Vertex part. mismatch allowed frame 259

(d) Vertex part. mismatch allowed frame 260

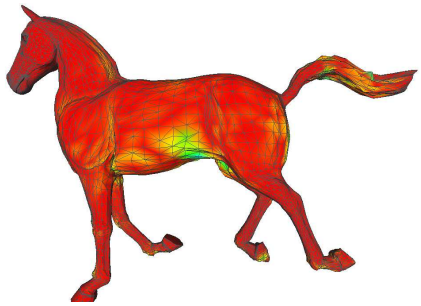
(e) Temporal Sub. frame 259

(f) Temporal Sub. frame 260

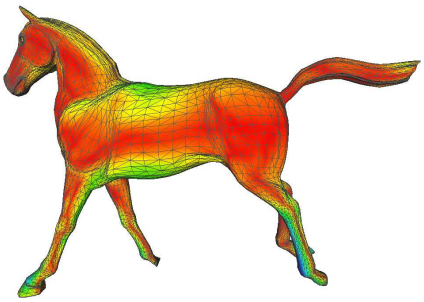
Figure 5.11: Central and side reconstructions of frames 259 and 260 of chicken sequence MD coded at 44% redundancy



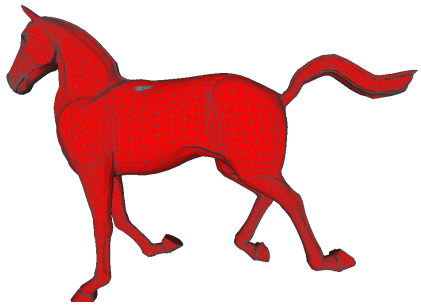
(a) Vertex part. mismatch allowed frame 1



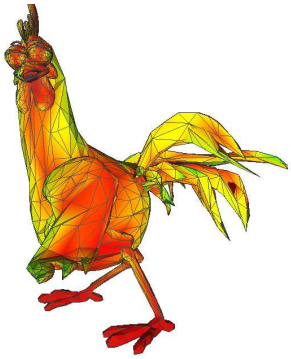
(b) Vertex part. mismatch allowed frame 2



(c) Temporal Sub. frame 1



(d) Temporal Sub. frame 2



(e) Vertex part. mismatch allowed frame 259

(f) Vertex part. mismatch allowed frame 260

(g) Temporal Sub. frame 259

(h) Temporal Sub. frame 260

Figure 5.12: Visualization of errors between central and side reconstructions for horse-gallop and chicken crossing MD coded at 25% and 44% redundancy, respectively. Deviation from red indicates increasing error.

5.5 Conclusions and Future Work

In this chapter, we have proposed and evaluated three novel Multiple Description Coding (MDC) methods for reliable transmission of compressed animated meshes. The methods make use of an efficient 3D dynamic mesh coder based on temporal/spatial layer decompositions. We have also proposed necessary modifications to adjust the amount of redundancy to gain resiliency to losses.

We have presented the experimental results with redundancy-rate-distortion curves and visual reconstructions. The experimental results show that *Vertex partitioning* performs better at low redundancies for especially spatially dense models. *Temporal subsampling* performs better at moderate redundancies (corresponding to including at least the spatial base layer in both descriptions) as well as low redundancies for spatially coarse models. Layer duplication based MDC can achieve the lowest redundancies with flexible redundancy allocation capability and can be designed to achieve the smallest variance of reconstruction quality between consecutive frames.

Possible future works include increasing the number of descriptions, searching for the optimized MDC method and parameters for a given model according to the spatial and temporal properties and decreasing the complexity of finding the optimal layer ordering in *Layer duplication*.

CHAPTER 6

IMPROVED PREDICTION METHODS FOR SCALABLE PREDICTIVE ANIMATED MESH COMPRESSION

In this work, our aim is to obtain efficient predictive animated mesh compression with spatial-temporal scalability support and suitable for low-delay streaming scenarios. Therefore we integrated our proposed methods into the spatial and temporal layered decomposition structure proposed in the SPC. The first contribution of our work is the introduction of a weighted spatial prediction scheme. The second contribution is a weighted temporal prediction scheme. Even though the integration is done by SPC, the proposed weighting based prediction structures can be used in any predictive coder which makes use of spatial and temporal layered structure. Finally, we propose a novel angle based predictor. In experimental results, we show that significant improvements can be achieved for both prediction errors and compression rate.

The rest of the chapter is organized as follows: We provide the details of the proposed prediction schemes, namely weighted spatial prediction in Section 6.1.1, weighted temporal prediction in Section 6.1.2 and angle based prediction in Section 6.1.3. We present the experimental results in Section 6.2 and finally, we conclude in Section 6.3.

6.1 Prediction Structures

The prediction of vertices is the most crucial part of predictive animated mesh coding. Although the SPC algorithm was presented in Section 3.2.1, we first revisit the prediction structure in the SPC algorithm in this section. Then we provide the details of the proposed prediction structures. All the prediction structures depend on the previously mentioned spa-

tial/temporal layered mesh sequence decomposition procedure, which is a part of the SPC method and MPEG-4 FAMC standard.

Spatial and temporal neighboring information of an example vertex ($\mathbf{v}_c^c$) to be predicted/encoded is illustrated in Figure 3.4. In the figure and rest of the text, superscripts p, c and f are used for past, current and future frames respectively. $\mathbf{v}_c^c$ is the vertex to be predicted in the current frame and $\mathbf{v}_c^p$ and $\mathbf{v}_c^f$ denote the vertices at the same connectivity location with $\mathbf{v}_c^c$ in the past and future frames, respectively. $\mathbf{v}_i^{p,c,f}, i = 0, 1, ..., N-1$ denote the topological neighbors of $\mathbf{v}_c^{p,c,f}$. In this example, $\mathbf{v}_c^c$ belongs to a B frame and makes prediction from one past and one future frame. Note that, both past and future frames are already encoded before the current frame. Future is used in the sense of frame display order. The prediction of $\mathbf{v}_c^c$ consists of spatial prediction followed by a temporal prediction.

In the SPC, the first step during the prediction in the encoder is calculating a spatial prediction. The spatial prediction of the vertex $\mathbf{v}_c^{p,c,f}$ denoted by $\mathbf{v}_s^{p,c,f}$ is calculated as average of the topological neighbors ($\mathbf{v}_i^{p,c,f}, i = 0, 1, ..., N-1$):

$$\mathbf{v}_s^{p,c,f} = \frac{1}{N} \sum_{i=0}^{N-1} \mathbf{v}_i^{p,c,f}, \tag{6.1}$$

where N is the number of topological neighbor vertices. After the spatial prediction, the spatial prediction error of $\mathbf{v}_c^{p,c,f}$ denoted by $\mathbf{e}_s^{p,c,f} = \mathbf{v}_c^{p,c,f} - \mathbf{v}_s^{p,c,f}$ is obtained as illustrated in Figure 3.4.

The spatial prediction errors are not directly used for encoding except for the I frames. For P and B frames, the spatial prediction is followed by a temporal prediction procedure which aims to refine the spatial prediction error ($\mathbf{e}_s^c$) in order to obtain the final prediction of $\mathbf{v}_c^c$ denoted by $\hat{\mathbf{v}}_c^c$. $\hat{\mathbf{v}}_c^c$ is calculated as:

$$\hat{\mathbf{v}}_c^c = \mathbf{v}_s^c + \mathbf{\Delta}_t^c, \tag{6.2}$$

where $\mathbf{\Delta}_t^c$ can be regarded as the temporal prediction or a spatial prediction correction/refinement term coming from previously encoded frames. $\mathbf{\Delta}_t^c$ is actually a prediction of $\mathbf{e}_s^c$ and calculated as

$$\mathbf{\Delta}_t^c = \frac{\tilde{\mathbf{e}}_s^p + \tilde{\mathbf{e}}_s^f}{2}, \tag{6.3}$$

where $\tilde{\mathbf{e}}_s^p$ and $\tilde{\mathbf{e}}_s^f$ are the spatial prediction correction terms corresponding to past and future

frames, respectively, and calculated as

$$\tilde{\mathbf{e}}_s^{p,f} = (M^c)^{-1} M^{p,f} \mathbf{e}_s^{p,f}, \tag{6.4}$$

where $M^{p,c,f}$ denotes the local coordinate frame for $\mathbf{v}_c^{p,c,f}$ which is a Rotation-Invariant Coordinate (RIC) system defined at $\mathbf{v}_s^{p,c,f}$ [66]. In other words, $M^{p,f}\mathbf{e}_s^{p,f}$ is the spatial prediction error transformed into local coordinate frames defined in past/future frames. $\tilde{\mathbf{e}}_s^{p,f}$ is obtained by transforming back to global coordinates using $(M^c)^{-1}$. Note that, setting M^c and $M^{p,f}$ equal to identity matrix results in a linear and rotation-varying prediction.

The temporal prediction procedure explained is for a $\mathbf{v}_c^c$ in a B frame. No temporal prediction is employed for I frames. For P frames, the only difference is that $\mathbf{\Delta}_t^c$ is calculated as $\tilde{\mathbf{e}}_s^p$ in Equation 6.3.

6.1.1 Proposed Weighted Spatial Prediction

The calculation of spatial prediction in layered predictive animated mesh coding is not unique. For example in SPC, equal weighting of neighbors with $1/N$ is used to calculate $\mathbf{v}_s$ as shown in Equation 6.1. However, we can generalize the calculation of $\mathbf{v}_s$ with the possibility of assigning different weights for each neighbor as shown in Equation 6.5.

$$\mathbf{v}_s^{p,c,f} = \sum_{i=0}^{N-1} a_i^{p,c,f} \mathbf{v}_i^{p,c,f} \tag{6.5}$$

If we search for the optimal a_i^c values for every $\mathbf{v}_c^c$ and encode them, this would increase the bitrate significantly. However, if we can obtain a_i^c values by using only the previously encoded vertices, then we do not have to encode the weight values since both the encoder and the decoder have access to same values of previously encoded vertices. With this motivation, we propose to use the relation between $\mathbf{v}_c^{p,f}$ and $\mathbf{v}_i^{p,f}$ to calculate the weight of each $\mathbf{v}_i^c, i = 0, 1, ..., N-1$ used in spatial prediction of $\mathbf{v}_c^c$.

For the vertices in the I frames, since there is no information from previous frames, we keep the equal weights for the neighbors. For a $\mathbf{v}_c^c$ in a P frame, we use $\mathbf{v}_c^p$ and calculate a_i^p values according to inverse proportions between the euclidian distances between $\mathbf{v}_c^p$ and its neighbors

$\mathbf{v}_i^p, i = 0, 1, ..., N-1$ as shown in Equation 6.6.

$$a_i^{p,f} = \frac{\frac{1}{\left\|\mathbf{v}_i^{p,f} - \mathbf{v}_c^{p,f}\right\|}}{\sum_{j=0}^{N-1} \frac{1}{\left\|\mathbf{v}_j^{p,f} - \mathbf{v}_c^{p,f}\right\|}} \tag{6.6}$$

Using the a_i^p values in Equation 6.6 to obtain $\mathbf{v}_s^p$ results in a more accurate spatial prediction of $\mathbf{v}_c^p$ than using equal weights of a_i^p values. We assume that, for the current frame, same a_i^p coefficients also yield similarly better spatial prediction of $\mathbf{v}_c^c$. Therefore we set $a_i^c = a_i^p$. For a B frame, a vertex obtains two sets of a_i values from previous and future frames (a_i^p and a_i^f). The a_i^f values are obtained in the same way as a_i^p values are obtained as shown in Equation 6.6. Here the effect of two frames can be weighted with c and $1 - c$ as shown in Equation 6.7.

$$a_i^c = ca_i^p + (1-c)a_i^f \tag{6.7}$$

In our experiments, we always used $c = 0.5$ which consistently produced good results. As a future work, the effect of c value on the compression performance can also be investigated.

Even though we integrated the proposed weighted spatial prediction into SPC, it can be used in any prediction method that makes use of a spatial prediction using neighboring vertices. We will see in Section 6.1.3 that the proposed angle based prediction makes use of the weighted spatial prediction as well.

6.1.2 Proposed Weighted Temporal Prediction

In the SPC, we observe in Equation 6.3 that the spatial prediction error terms from previous and future frames ($\tilde{\mathbf{e}}_s^p$ and $\tilde{\mathbf{e}}_s^f$) are equally weighted. We propose to use unequal weighting for better prediction as shown in Equation 6.8:

$$\Delta_t^c = k_c^c \tilde{\mathbf{e}}_s^p + (1 - k_c^c)\tilde{\mathbf{e}}_s^f \tag{6.8}$$

Therefore our aim is to approximate the optimal k_c^c values for each vertex using only the previously encoded vertices. Since the encoder and the decoder reconstruct the same vertices identically, there is no need to encode the k_c^c values. The main idea behind our proposals is as follows: During the encoding of $\mathbf{v}_c^c$, all the spatial neighbors of $\mathbf{v}_c^{p,c,f}$, i.e., $\mathbf{v}_i^{p,c,f}, i = 0, 1, ..., N-1$ are encoded before. Here we introduce two new error terms related to encoding process of $\mathbf{v}_i^c$: $\tilde{\mathbf{e}}_{s,i}^{p,f}$ denotes the spatial prediction correction term corresponding to past/future

frame and $\mathbf{e}^c_{s,i}$ denotes the actual spatial prediction error for $\mathbf{v}^c_i$. $\mathbf{e}^c_{s,i}$ is obtained by

$$\mathbf{e}^c_{s,i} = \mathbf{v}^c_i - \mathbf{v}^c_{s,i}, \tag{6.9}$$

where $\mathbf{v}^c_{s,i}$ denotes the spatial prediction of $\mathbf{v}^c_i$. We assume that the relation between $\tilde{\mathbf{e}}^{p,f}_{s,i}$ and $\mathbf{e}^c_{s,i}$ for $i = 0, 1, .., N-1$ is similar to the relation between $\tilde{\mathbf{e}}^{p,f}_s$ and the current spatial prediction error term ($\mathbf{e}^c_s$) we are trying to predict. Therefore we try to approximate k^c_c such that weighting $\tilde{\mathbf{e}}^p_{s,i}$ and $\tilde{\mathbf{e}}^f_{s,i}$ with k^c_c for $i = 0, 1, .., N-1$ produces an approximation of $\mathbf{e}^c_{s,i}$ better than by equally weighting.

In order to approximate the k^c_c value, we propose two methods. The first method is based on inverse proportionality of the distance between $\tilde{\mathbf{e}}^p_{s,i}$ and $\mathbf{e}^c_{s,i}$ and the distance between $\mathbf{e}^c_{s,i}$ and $\tilde{\mathbf{e}}^p_{s,i}$. In this method, k^c_c approximation is obtained by

$$k^c_c = \frac{1}{N}\sum_{i=0}^{N-1} k^c_{c,i}, \tag{6.10}$$

where $k^c_{c,i}, i = 0, 1, .., N-1$ is the k^c_c approximation obtained by using $\tilde{\mathbf{e}}^p_{s,i}$, $\mathbf{e}^c_{s,i}$ and $\tilde{\mathbf{e}}^f_{s,i}$ as shown in Equation 6.11.

$$k^c_{c,i} = \frac{\|\mathbf{e}^c_{s,i} - \tilde{\mathbf{e}}^f_{s,i}\|}{\|\mathbf{e}^c_{s,i} - \tilde{\mathbf{e}}^f_{s,i}\| + \|\mathbf{e}^c_{s,i} - \tilde{\mathbf{e}}^p_{s,i}\|} \tag{6.11}$$

In the second proposed method, we calculate the k^c_c with Least Squares (LS). We construct an overdetermined system of linear equations using all the neighboring vertices as shown in Equations 6.12 and 6.13.

$$k^c_c\tilde{\mathbf{e}}^p_{s,i} + (1 - k^c_c)\tilde{\mathbf{e}}^f_{s,i} = \mathbf{e}^c_{s,i} \text{ for } i = 0, ..., N-1 \tag{6.12}$$

$$(\tilde{\mathbf{e}}^p_{s,i} - \tilde{\mathbf{e}}^f_{s,i})k^c_c = \mathbf{e}^c_{s,i} - \tilde{\mathbf{e}}^f_{s,i} \text{ for } i = 0, ..., N-1 \tag{6.13}$$

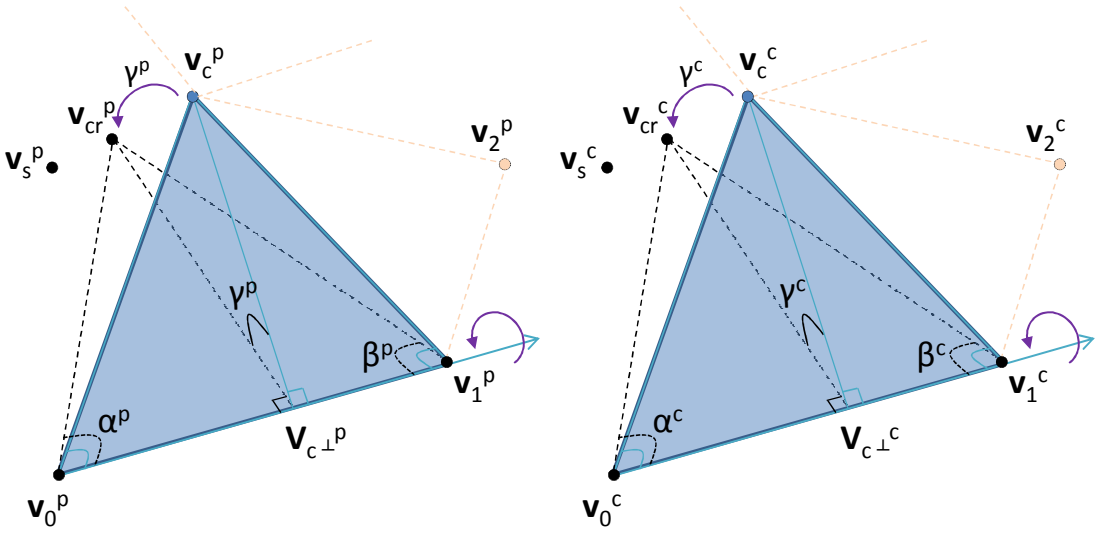
Then writing the equations in matrix form and taking the pseudoinverse gives the LS solution of k^c_c, calculated as in Equation 6.14.

$$k^c_c \approx \frac{\sum_{i=0}^{N-1}(\tilde{\mathbf{e}}^p_{s,i} - \tilde{\mathbf{e}}^f_{s,i})^T(\mathbf{e}^c_{s,i} - \tilde{\mathbf{e}}^f_{s,i})}{\sum_{i=0}^{N-1}\|\tilde{\mathbf{e}}^p_{s,i} - \tilde{\mathbf{e}}^f_{s,i}\|^2} \tag{6.14}$$

6.1.3 Proposed Angle Based Predictor

In this section, we present the details of the proposed angle based predictor which is based on the same spatial/temporal layered structure of SPC. The predictor is used for either P or B

frames as it predicts the location of a vertex in the current frame using its previously encoded neighbor vertices and corresponding vertices in the previously encoded frames. The algorithm initially makes predictions for the current vertex to be encoded ($\mathbf{v}_c^c$ in Figure 3.4) using each incident triangle in each previously encoded frame. As a result, it obtains predictions as many as the number of incident triangles for a P frame and twice the number of incident triangles for a B frame. Then it makes a final decision using all the predictions. Since the whole neighbor triangle information is not usually available in the spatial base layer, the algorithm can be applied to layers above spatial base layer. In this case, the vertices in the spatial base layer are encoded with the SPC algorithm.



(a) Previously encoded frame (b) Current frame

Figure 6.1: One incident triangle in previously encoded and current frame. Note that $\mathbf{v}_0^{p,c}$, $\mathbf{v}_1^{p,c}$, $\mathbf{v}_s^{p,c}$ and $\mathbf{v}_{cr}^{p,c}$ lie on the same plane $\mathbf{P_s^{p,c}}$.

An incident triangle for previously encoded and current frame is illustrated in Figure 6.1. In the figure, superscript c denotes the current frame and superscript p denotes the previously encoded frame. Let $\mathbf{v}_c^c$ denote the vertex to be encoded in the current frame and $\mathbf{v}_c^p$ denote the vertex at the same connectivity location in the previous frame. The vertex pairs $\mathbf{v}_0^p$,$\mathbf{v}_1^p$ and $\mathbf{v}_0^c$, $\mathbf{v}_1^c$ denote the remaining vertices of the incident triangle with corresponding angle pairs α^p, β^p and α^c, β^c in previous and current frames respectively. $\mathbf{v}_s^p$ and $\mathbf{v}_s^c$ denote the spatial predictions of $\mathbf{v}_c^p$ and $\mathbf{v}_c^c$ respectively, obtained by using the neighbor vertices. Let $\mathbf{P_s^p}$ denote the plane defined by the points $\mathbf{v}_0^p$, $\mathbf{v}_1^p$ and $\mathbf{v}_s^p$. Then the angle γ^p in the figure is the angle to rotate the triangle around its edge defined by $\mathbf{v}_0^p$ and $\mathbf{v}_1^p$ so that $\mathbf{v}_c^p$ lies on the plane $\mathbf{P_s^p}$. After

the rotation, $\mathbf{v}_c^p$ comes to the point denoted by $\mathbf{v}_{cr}^p$ as shown in Figure 6.1(a). Same relations also exist for $\mathbf{v}_0^c$, $\mathbf{v}_1^c$, $\mathbf{v}_s^c$, $\mathbf{P_s^c}$, $\mathbf{v}_c^c$ and $\mathbf{v}_{cr}^c$ as shown in Figure 6.1(b).

The basic idea behind the predictor is that the angles γ^p and γ^c are assumed to be similar, i.e., these angles do not vary significantly through the time. In Figure 6.1, all the vertices except $\mathbf{v}_c^c$ are previously encoded. Therefore, we do not have the values of α^c,β^c,γ^c during the encoding process of $\mathbf{v}_c^c$. However, in order to make use of the similarity assumption between γ^p and γ^c, we try to estimate $\mathbf{v}_{cr}^c$ using previously encoded vertices. Then we rotate back the triangle defined by $\mathbf{v}_0^c$, $\mathbf{v}_1^c$ and the estimation of $\mathbf{v}_{cr}^c$ with an angle of γ^p to predict $\mathbf{v}_c^c$. The detailed steps of the algorithm are as follows:

1. Calculate α^p and β^p: $\alpha^p = \arccos \frac{(\mathbf{v}_c^p - \mathbf{v}_0^p)^T(\mathbf{v}_1^p - \mathbf{v}_0^p)}{\|\mathbf{v}_c^p - \mathbf{v}_0^p\|\|\mathbf{v}_1^p - \mathbf{v}_0^p\|}$ and $\beta^p = \arccos \frac{(\mathbf{v}_c^p - \mathbf{v}_1^p)^T(\mathbf{v}_0^p - \mathbf{v}_1^p)}{\|\mathbf{v}_c^p - \mathbf{v}_1^p\|\|\mathbf{v}_0^p - \mathbf{v}_1^p\|}$

2. Calculate $\mathbf{v}_{c\perp}^p$, the point defined by the intersection of the edge $\mathbf{v}_0^p$-$\mathbf{v}_1^p$ and the perpendicular line segment from $\mathbf{v}_c^p$ to the edge $\mathbf{v}_0^p$-$\mathbf{v}_1^p$.

3. Calculate $\mathbf{v}_s^p$ using either simple averaging of the neighbors of $\mathbf{v}_c^p$ ($\mathbf{v}_0^p$, $\mathbf{v}_1^p$,...,$\mathbf{v}_{N-1}^p$) or the proposed weighted spatial prediction described in Section 6.1.1. Having calculated $\mathbf{v}_s^p$, the plane $\mathbf{P_s^p}$ defined by $\mathbf{v}_0^p$, $\mathbf{v}_1^p$ and $\mathbf{v}_s^p$ is obtained.

4. Calculate $\mathbf{v}_{cr}^p$ by redrawing the triangle ($\mathbf{v}_0^p$, $\mathbf{v}_1^p$, $\mathbf{v}_c^p$) and preserving the angles α^p and β^p so that the triangle lies on the plane $\mathbf{P_s^p}$. This operation is equivalent to rotating the triangle with an angle of γ^p. As mentioned before, after this operation, the positions of $\mathbf{v}_0^p$ and $\mathbf{v}_1^p$ do not change and the new position of $\mathbf{v}_c^p$ is equal to $\mathbf{v}_{cr}^p$.

5. Calculate γ^p, using the line segments $\mathbf{v}_{c\perp}^p$-$\mathbf{v}_c^p$ and $\mathbf{v}_{c\perp}^p$-$\mathbf{v}_{cr}^p$ ($\widehat{\mathbf{v}_c^p \mathbf{v}_{c\perp}^p \mathbf{v}_{cr}^p}$) as shown in Equation 6.15.

$$\gamma^p = \arccos \frac{(\mathbf{v}_c^p - \mathbf{v}_{c\perp}^p)^T(\mathbf{v}_{cr}^p - \mathbf{v}_{c\perp}^p)}{\|\mathbf{v}_c^p - \mathbf{v}_{c\perp}^p\|\|\mathbf{v}_{cr}^p - \mathbf{v}_{c\perp}^p\|} \tag{6.15}$$

   This step completes all the necessary information from previously encoded frame.

   Steps 6 through 9 give the procedure to find $\hat{\mathbf{v}}_c^c$, the estimation of $\mathbf{v}_c^c$, using the information obtained in steps 1-5. We will use Figure 6.2 to illustrate the following steps.

6. Calculate $\mathbf{v}_s^c$ again using a spatial prediction algorithm and obtain the plane $\mathbf{P_s^c}$ defined by $\mathbf{v}_0^c$, $\mathbf{v}_1^c$ and $\mathbf{v}_s^c$.

7. Calculate $\hat{\mathbf{v}}_{cr}^c$, the estimation of $\mathbf{v}_{cr}^c$.

   We propose three methods to estimate $\mathbf{v}_{cr}^c$:

(a) Angles same: As the simplest solution, α^c,β^c are approximated as α^p,β^p as illustrated in Figure 6.2(a) where the approximated angles are denoted by $\hat{\alpha}^c$ and $\hat{\beta}^c$. In this case, $\hat{\mathbf{v}}_{cr}^c$ is calculated such that the triangle ($\hat{\mathbf{v}}_{cr}^c$, $\mathbf{v}_0^c$, $\mathbf{v}_1^c$) lies on the plane $\mathbf{P_s^c}$ and the angles $\widehat{\hat{\mathbf{v}}_{cr}^c \mathbf{v}_0^p \mathbf{v}_1^p}$ and $\widehat{\hat{\mathbf{v}}_{cr}^c \mathbf{v}_1^p \mathbf{v}_0^p}$ are equal to $\hat{\alpha}^c$ and $\hat{\beta}^c$ respectively.

(b) Angle differences same: In this method, we first calculate the angles of the triangles ($\mathbf{v}_0^p$, $\mathbf{v}_1^p$, $\mathbf{v}_s^p$) and ($\mathbf{v}_0^c$, $\mathbf{v}_1^c$, $\mathbf{v}_s^c$). Let $\alpha_s^{p,c} \triangleq \mathbf{v}_s^{p,c}\widehat{\mathbf{v}_0^{p,c}\mathbf{v}_1^{p,c}}$ and $\beta_s^{p,c} \triangleq \mathbf{v}_s^{p,c}\widehat{\mathbf{v}_1^{p,c}\mathbf{v}_0^{p,c}}$ as depicted in Figure 6.2(b). The assumption in this method is that the difference between α^p, β^p and α_s^p, β_s^p is similar to the difference between α^c, β^c and α_s^c, β_s^c i.e. the angle differences between the triangles do not vary significantly with time. In summary, α^c and β^c are approximated as $\hat{\alpha}^c = \alpha_s^c + \alpha^p - \alpha_s^p$ and $\hat{\beta}^c = \beta_s^c + \beta^p - \beta_s^p$ as illustrated in Figure 6.2(b). After approximating the angles, rest of the procedure to calculate $\hat{\mathbf{v}}_{cr}^c$ is same as in *Angles same* method.

(c) Local displacement between $\mathbf{v}_{cr}$ and $\mathbf{v}_s$ same: The motivation behind this approach is based on the assumption that the local displacement between $\mathbf{v}_{cr}^{p,c}$ and $\mathbf{v}_s^{p,c}$ is same for the current and the previous frames as illustrated in Figure 6.2(c). The local coordinate frame is defined at $\mathbf{v}_s^{p,c}$ by the unit vectors $\mathbf{u}_0^{p,c}$ and $\mathbf{u}_1^{p,c}$ lying on the plane $\mathbf{P_s^{p,c}}$ with directions $\mathbf{v}_1^{p,c}$-$\mathbf{v}_0^{p,c}$ and $\mathbf{v}_{cr}^{p,c}$-$\mathbf{v}_{c\perp}^{p,c}$. Note that for the calculation of $\mathbf{u}_1^c$ during encoding, the points $\mathbf{v}_{cr}^{p,c}$ and $\mathbf{v}_{c\perp}^{p,c}$ are unavailable. However, since $\mathbf{u}_0^c$ and $\mathbf{u}_1^c$ lie on the same plane, we can calculate $\mathbf{u}_1^c$ as the vector orthogonal to $\mathbf{u}_0^c$. Then $\hat{\mathbf{v}}_{cr}^c$ is calculated as:

$$\hat{\mathbf{v}}_{cr}^c = \mathbf{v}_s^c + \left\langle \mathbf{v}_{scr}^p, \mathbf{u}_0^p \right\rangle \mathbf{u}_0^c + \left\langle \mathbf{v}_{scr}^p, \mathbf{u}_1^p \right\rangle \mathbf{u}_1^c \tag{6.16}$$

where $\langle , \rangle$ denotes the dot product operation and $\mathbf{v}_{scr}^p$ is calculated as:

$$\mathbf{v}_{scr}^p = \mathbf{v}_{cr}^p - \mathbf{v}_s^p \tag{6.17}$$

8. Calculate $\hat{\mathbf{v}}_{c\perp}^c$, the estimation of $\mathbf{v}_{c\perp}^c$. $\hat{\mathbf{v}}_{c\perp}^c$ is obtained by drawing a perpendicular line segment from $\hat{\mathbf{v}}_{cr}^c$ to the edge $\mathbf{v}_0^c$-$\mathbf{v}_1^c$.

9. Finally, calculate $\hat{\mathbf{v}}_c^c$. In this step, γ^c is estimated as γ^p and the line segment $\hat{\mathbf{v}}_{cr}^c$-$\hat{\mathbf{v}}_{c\perp}^c$ is rotated around the edge $\mathbf{v}_0^c$-$\mathbf{v}_1^c$ by an angle of γ^p. $\hat{\mathbf{v}}_c^c$ is the position where $\hat{\mathbf{v}}_{cr}^c$ resides after the rotation.

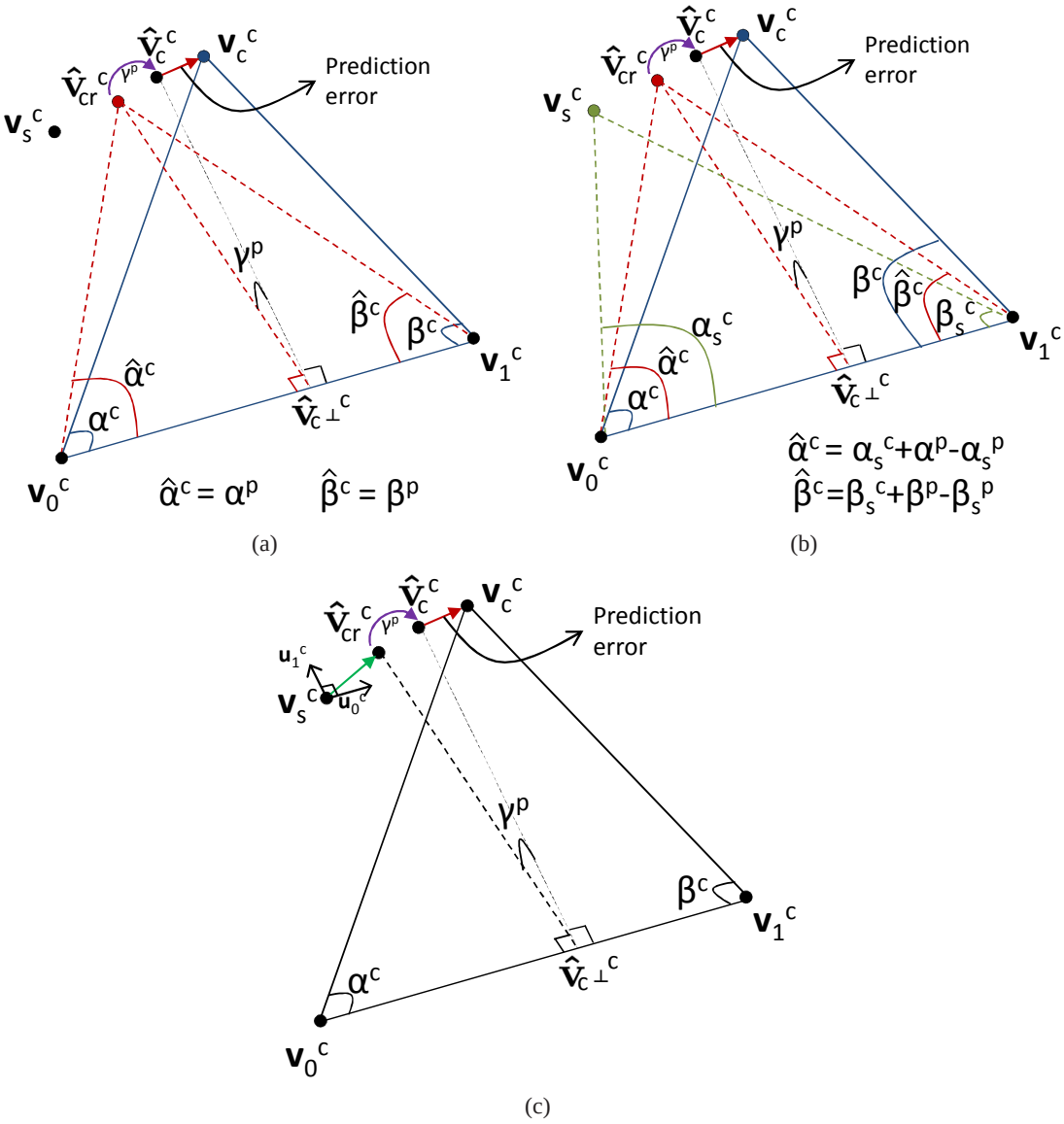


Figure 6.2: Angle based prediction (a) Angles same (b) Angle differences same (c) Local displacement between $\mathbf{v}_{cr}$ and $\mathbf{v}_s$ same

In this way, for a vertex to be encoded, *number of triangle neighbors* x *number of references* predictions are obtained. Final step is to make a decision for the prediction using all the predictions. The straightforward way is to average all the predictions. However, some of the predictions may result in very poorly compared others and become outliers. In order to cope with this, we use an outlier removal process as follows: We calculate the mean and the standard deviation of all predictions and reject the predictions that are a constant times standard deviation away from the mean.

6.2 Experimental Results

We evaluate the performance of the prediction schemes using the *Cowheavy*, *Chicken crossing*, *Dance*, *Horse Gallop*, *Face* and *Jump* test sequences. The properties of the sequences are shown in Table 6.1. We express the bitrate as bits per vertex per frame (bpvf). In order to measure the overall sequence distortion, we use the error metric which is defined in Karni and Gotsman's work [49] as it is a widely used metric in the literature. We denote this error by *KG Error* and it is calculated as shown in Equation 3.8. For all the experiments, we decomposed the mesh with 8 spatial layers and 4 temporal layers.

Table 6.1: The test sequences

Name	# Vertices	# Triangles	Frames used
Cowheavy	2904	5804	1-204
Chicken crossing	3030	5664	1-400
Dance	7061	14118	1-201
Horse Gallop	8431	16843	1-48
Face	539	1042	1-10001
Jump	15830	31660	431-652

Throughout the results, we label the prediction schemes to be tested as follows:

SPC Prediction structure of the SPC which is based on rotation-invariant coordinates [66].

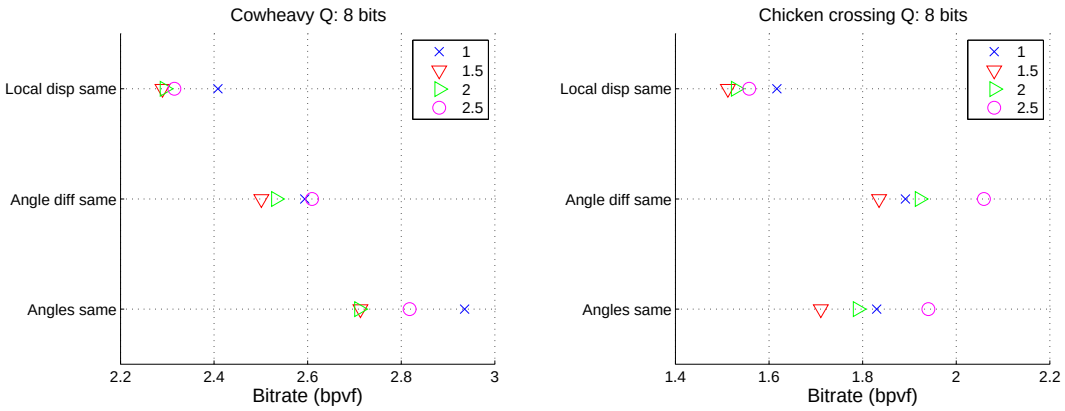
Angle Proposed angle based prediction.

+ Wsp Using the proposed weighted spatial prediction for either SPC or Angle

+ Wtp-LS Using the proposed least square based weighted temporal prediction for either SPC or Angle.

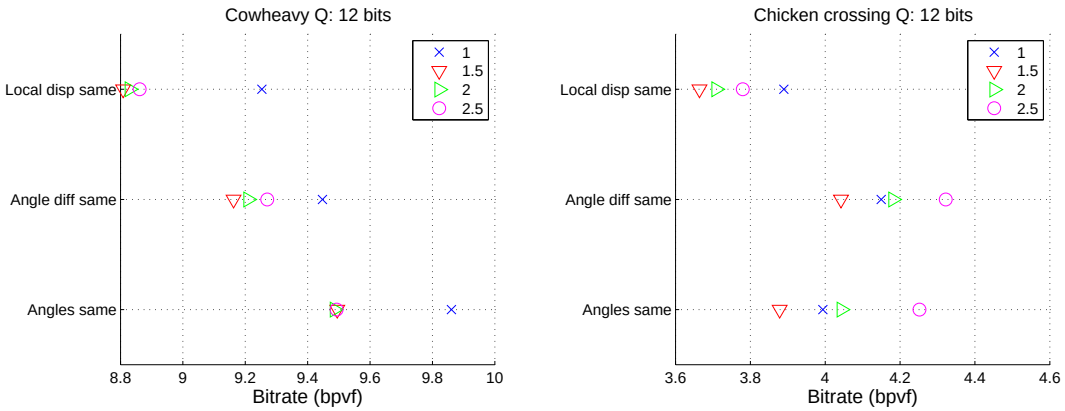
+ Wtp-Inv prop Using the proposed inverse proportionality based weighted temporal prediction for either SPC or Angle.

Note that weighted temporal prediction is not used in angle based prediction. However, in the experiments, *Angle + Wtp-LS* or *Wtp-Inv prop* means that weighted temporal prediction is used for the vertices in the spatial base layer as explained in section 6.1.3.



(a) Cowheavy (b) Chicken crossing

Figure 6.3: The compression performance of angle based prediction methods as a function of multiplicative constant of standard deviation in outlier removal process as given in the legend. Q=8 bits:



(a) Cowheavy (b) Chicken crossing

Figure 6.4: The compression performance of angle based prediction methods as a function of multiplicative constant of standard deviation in outlier removal process as given in the legend. Q=12 bits

We start with the angle based predictor and observe the effects of the three proposed methods to calculate $\hat{\mathbf{v}}_{cr}^{c}$ and the multiplicative constant of standard deviation in outlier removal process in Figures 6.3 and 6.4 which correspond to using quantization parameter of $Q = 8$ and $Q = 12$ respectively. The angle based methods are labeled as *Angles same*, *Angle diff same* and *Local disp same* in the figures. The bitrate values in the figures correspond to the resultant bitrates achieved after compression. We observe that, *Local disp same* performs the best and multiplying standard deviation with a value around 1.5 produces consistently good results.

Therefore, in the rest of the results, we continue with these parameters for the angle based prediction.

Figure 6.5: Change in prediction error compared to SPC per frame

In Figure 6.5, we show the prediction error improvements of the *Angle* and *Wsp* compared to *SPC* for the first 100 frames. The results are obtained for cowheavy sequence coded with $Q = 12$ quantization parameter. In this figure, we express the frame errors in Peak Signal-to-Noise Ratio (PSNR) scale for better visualization purpose. We calculate the PSNR value as $PSNR = 10log_{10}(MSE/bboxdiag^2)$ where MSE is the mean squared error between the predicted and original vertex locations and *bboxdiag* is the bounding box diagonal of the mesh sequence. The figure shows that, using *Wsp* consistently improves prediction error of SPC with around 1dB. Using only angle based prediction results in better or worse prediction error depending on the frames. However, adding *Wsp* to the angle based prediction results in significant improvement and better PSNR than *SPC+Wsp* for many frames.

In Figure 6.6, we show the percentage bitrate reduction compared to *SPC* for each frame at the same conditions with Figure 6.5. In the figure, we observe that except for a few frames

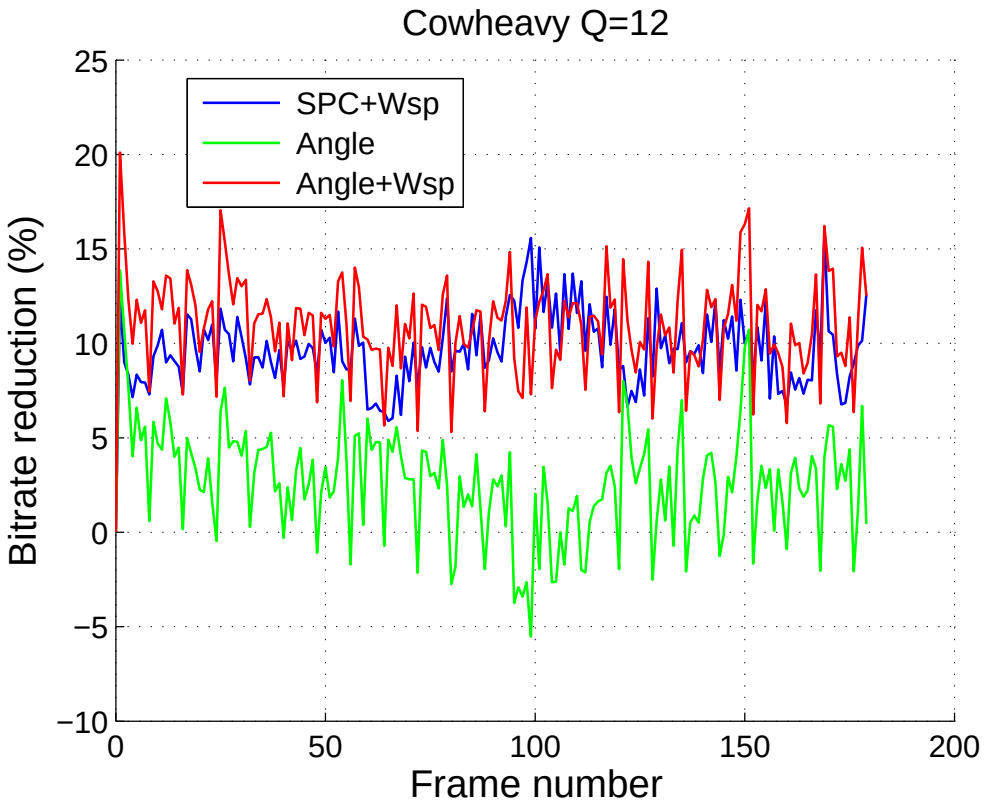


Figure 6.6: Percentage bitrate reduction compared to SPC per frame

for *Angle*, all proposed prediction schemes achieve bitrate reduction. The bitrate reduction of Angle in spite of poorer prediction shows that, the angle based prediction residuals are better compressible.

Next, we present the results of extensive compression experiments regarding to whole sequence bitrate and distortion. Tables 6.2 and 6.3 show the percentage bitrate reductions with respect to *SPC* achieved by the proposed prediction schemes. In the tables, each column corresponds to a quantization level obtained by varying the Q parameter. Note that SPC and the proposed methods achieve very similar distortions for the same Q value. The value with a * corresponds to maximal bitrate reduction achieved value. From the tables, it can be observed that *Wtp-Inv prop* performs consistently better than *Wtp-LS*. The reason is that in the *Wtp-LS* method, outliers in the neighbors may have a large effect in the calculation of k_c^c value. Although better predictions are achieved by *Wtp-LS* for some cases, very bad predictions due to very large magnitudes of k_c^c can be achieved for some other cases as well. This affects the compressibility of prediction residuals considerably. On the hand, *Wtp-Inv prop* method

restricts the magnitude of k_c^c to be between 0 and 1. Although a k_c^c value in this interval may not be the optimal value for some cases, it avoids very large prediction errors and provides a more robust solution.

Continuing with the tables, it is observed that usage of *Wsp* has a significant effect on the bitrate reduction both for *SPC* and *Angle*. *Angle* based predictions usually perform better at coarser quantization levels whereas *SPC* combined with proposed weighted predictions tend to perform better at finer quantization levels. Another observation is that the bitrate reductions achieved decrease with increasing *Q* values since the prediction accuracy of SPC increases with increasing precision. Examining all the results, around 6-30% bitrate reductions are observed to be achieved depending on the content and the quantization level.

Table 6.2: Bitrate reductions compared to SPC for cowheavy, chicken crossing and dance sequences

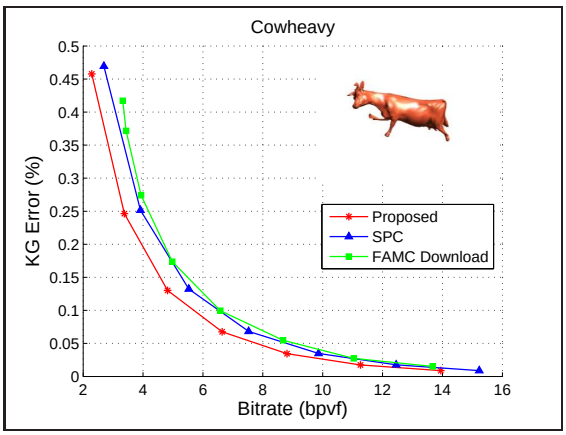
Methods	Q=8	Q=9	Q=10	Q=11	Q=12	Q=13	Q=14
Cowheavy							
SPC+Wtp-LS	-13,27	-8,07	-4,45	-2,21	-1,02	-0,43	-0,15
SPC+Wtp-Inv prop	1,77	2,47	3,2	3,46	3,39	3,2	2,91
SPC+Wsp	12,87	12,07	11,22	10,13	9,08	7,97	6,97
SPC+Wsp+Wtp-LS	-1,8	2,67	5,34	6,5	6,69	6,31	5,73
SPC+Wsp+Wtp-Inv prop	12,42	12,6	12,33	11,7	10,68*	9,57*	8,44*
Angle	4,12	2,81	2,49	2,38	2,34	2,14	1,92
Angle+Wtp-LS	4,74	3,61	3,29	3,18	2,98	2,66	2,36
Angle+Wtp-Inv prop	4,32	3,37	2,94	2,92	2,77	2,47	2,23
Angle+Wsp	14,8	13,28	12,28	11,27	10,2	9,03	7,95
Angle+Wsp+Wtp-LS	14,95*	13,67*	12,79*	11,76*	10,67	9,44	8,28
Angle+Wsp+Wtp-Inv prop	14,84	13,58	12,53	11,55	10,48	9,31	8,2
Chicken crossing							
SPC+Wtp-LS	-15,96	-16	-16,16	-15,6	-14,29	-12,81	-9,49
SPC+Wtp-Inv prop	0,61	1,81	2,08	2,38	2,73	2,99	3,52
SPC+Wsp	10,21	9,36	8,38	7,72	7,21	6,22	5,07
SPC+Wsp+Wtp-LS	-4,57	-6,62	-7,34	-7,82	-8,01	-7,81	-5,77
SPC+Wsp+Wtp-Inv prop	9,45	9,68	9,02	8,23	7,56	6,63	5,82
Angle	13,03	10,05	7,64	5,61	3,71	2,23	1,02
Angle+Wtp-LS	12,42	9,63	7,74	5,97	4,4	3	1,93
Angle+Wtp-Inv prop	13,16	10,41	8,37	6,34	4,74	3,22	2,13
Angle+Wsp	19,3	16,51	14	11,81	9,59	7,48	5,42
Angle+Wsp+Wtp-LS	18,07	15,71	13,72	11,72	9,88	7,89	5,95
Angle+Wsp+Wtp-Inv prop	19,37*	16,59*	14,25*	12,42*	10,28*	8,13*	6,10*
Dance							
SPC+Wtp-LS	-27,03	-23,48	-17,67	-11,25	-6,44	-2,85	-0,6
SPC+Wtp-Inv prop	-1,57	-0,21	1,26	2,73	3,91	4,83	5,47
SPC+Wsp	11,87	12,87	13,13*	12,89	12,38	11,8	11,09
SPC+Wsp+Wtp-LS	-13,02	-9,66	-4,53	0,49	4,32	6,81	8,22
SPC+Wsp+Wtp-Inv prop	9,41	11,59	12,88	13,84*	14,02*	14,00*	13,74*
Angle	8,33	1,82	-2,17	-3,63	-3,7	-2,86	-1,8
Angle+Wtp-LS	6,32	1,77	-0,96	-1,98	-2,08	-1,57	-0,72
Angle+Wtp-Inv prop	8,32	2,59	-1,07	-2,23	-2,51	-1,81	-0,89
Angle+Wsp	17,74*	13,66	10,96	9,33	8,63	8,59	8,73
Angle+Wsp+Wtp-LS	14,44	12,63	11,3	10,38	9,79	9,67	9,64
Angle+Wsp+Wtp-Inv prop	17,18	13,70*	11,66	10,37	9,65	9,51	9,52

Table 6.3: Bitrate reductions compared to SPC for horse gallop, face and jump

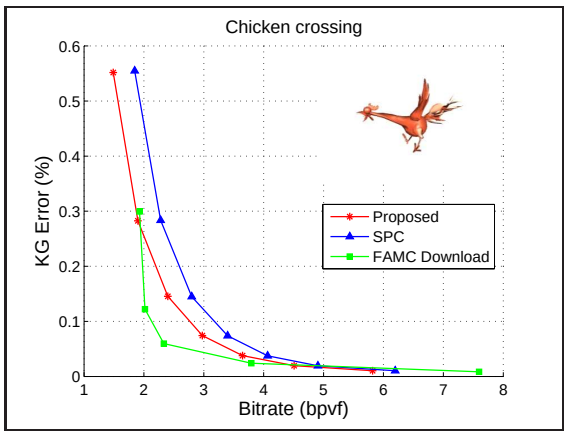
Methods	Q=8	Q=9	Q=10	Q=11	Q=12	Q=13	Q=14
Horse gallop							
SPC+Wtp-LS	-7,27	-4,17	-0,45	2,87	4,97	6,38	6,49
SPC+Wtp-Inv prop	0,2	2,26	4,39	6,53	7,4	7,98	7,55
SPC+Wsp	11,9	13,43	14,02	13,68	12,74	11,75	10,26
SPC+Wsp+Wtp-LS	3,48	7,22	9,76	11,48	12,11	11,97	10,81
SPC+Wsp+Wtp-Inv prop	9,65	12,96	14,48	15,4	15,19*	14,43*	12,97*
Angle	5,37	3,37	2,91	2,97	3	3,02	2,67
Angle+Wtp-LS	6,3	5,18	4,47	4,56	4,57	4,65	4,09
Angle+Wtp-Inv prop	6,51	5,16	4,28	3,97	4,13	4,17	3,71
Angle+Wsp	14,7	14,76	14,98	14,51	13,74	12,75	11,14
Angle+Wsp+Wtp-LS	14,82	15,53	16,03*	15,60*	14,84	13,87	12,25
Angle+Wsp+Wtp-Inv prop	15,49*	15,84*	15,89	15,5	14,58	13,56	11,96
Face							
SPC+Wtp-LS	-4,99	-2,65	0,3	2,88	4,52	5,18	5,01
SPC+Wtp-Inv prop	0,53	2,33	3,86	5,26	5,9	5,96	5,54
SPC+Wsp	5,34	5,04	4,89	4,72	4,61	4,27	3,77
SPC+Wsp+Wtp-LS	-1,27	0,95	3,35	5,4	6,82	7,29	6,95
SPC+Wsp+Wtp-Inv prop	4,54	5,77	6,8	7,72	8,26	8,23*	7,58*
Angle	12,65	10,36	8,23	6,46	5,19	4,24	3,58
Angle+Wtp-LS	11,81	9,81	8,2	7	5,91	5,05	4,32
Angle+Wtp-Inv prop	12,64	10,44	8,76	7,17	5,95	4,95	4,24
Angle+Wsp	15,85*	13,59*	11,73*	10,12	8,86	7,73	6,73
Angle+Wsp+Wtp-LS	14,04	12,24	10,94	9,89	8,98	8,05	7,07
Angle+Wsp+Wtp-Inv prop	15,19	13,42	11,68	10,30*	9,11*	8,07	7,06
Jump							
SPC+Wtp-LS	-9,78	-6,53	-3,42	-1,11	0,38	1,1	1,22
SPC+Wtp-Inv prop	2,4	3,57	3,66	3,52	3,14	2,81	2,39
SPC+Wsp	10,31	11	10,14	9,06	7,89	6,73	5,55
SPC+Wsp+Wtp-LS	0,22	3,97	5,92	6,86	6,95	6,5	5,64
SPC+Wsp+Wtp-Inv prop	11,04	13,05	12,67	11,42	9,86	8,44	6,98
Angle	23,13	15,1	9,13	5,79	4	3,19	2,61
Angle+Wtp-LS	17,81	12,4	8,16	5,65	4,28	3,46	2,83
Angle+Wtp-Inv prop	23,14	15,35	9,47	6,1	4,39	3,48	2,83
Angle+Wsp	30,36*	24,63	18,72	14,34	11,41	9,39	7,68
Angle+Wsp+Wtp-LS	24,74	21,18	17,06	13,88	11,41	9,48	7,76
Angle+Wsp+Wtp-Inv prop	30,2	24,64*	18,75*	14,50*	11,65*	9,58*	7,85*

Finally, we select the best performing proposed scheme for each Q value and compare the resultant rate-distortion curve with the state of the art coders, namely SPC and MPEG-4 FAMC download-and-play mode in Figure 6.7. Once again we note that the proposed methods are intended for scalable, streamable animated mesh compression allowing frame-wise decoding. Therefore, it is not a major concern to outperform a codec that requires the bitstream of whole

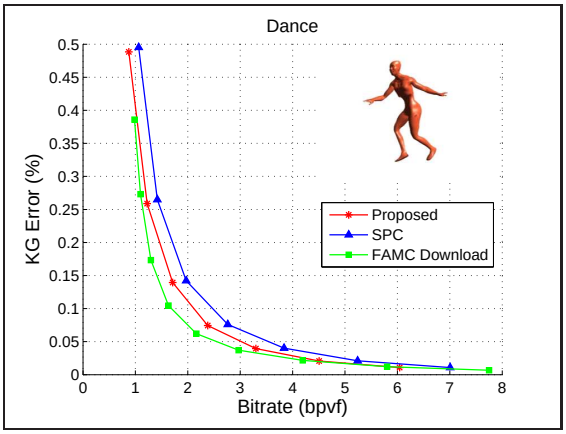
sequence information to decode like MPEG-4 FAMC download-and-play mode or efficient PCA based coders [58]. As observed from the figure, the proposed work always outperforms SPC and shows a competitive performance compared to MPEG-4 FAMC download-and-play mode.



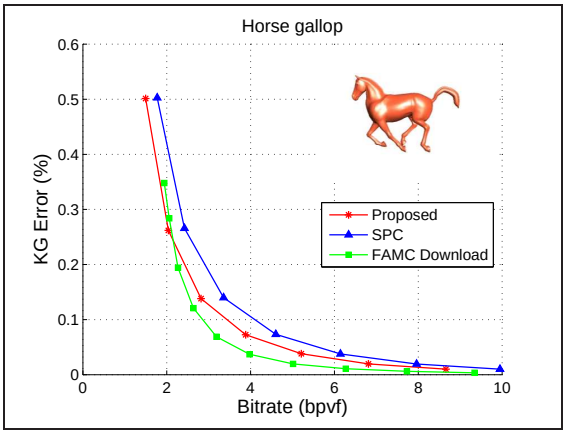
(a) Cowheavy



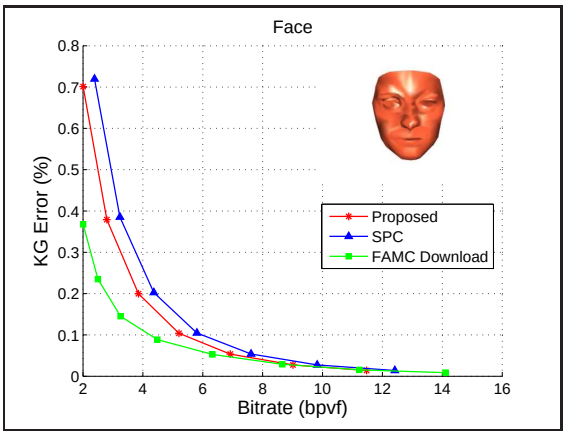
(b) Chicken crossing



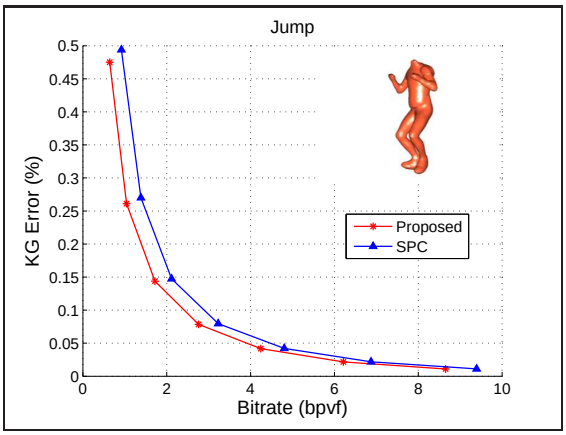
(c) Dance



(d) Horse gallop



(e) Face



(f) Jump

Figure 6.7: RD comparison of the SPC and best performing proposed method for each quantization level.

6.3 Conclusion

In this chapter, we have proposed three prediction improvements for scalable predictive animated mesh coding. The first improvement, weighted spatial prediction, is based on applying unequal weighting of topological neighboring vertices during spatial prediction using the previously encoded frame information. The second improvement, weighted temporal prediction, is based on unequal weighting of temporal predictions from past and future frames using the previously encoded neighboring vertex information. The final improvement, angle based prediction, makes use of the assumption of similarity between rotating angles of incident triangles to the plane obtained by spatial prediction in previous and current frames. The proposed methods depend on the spatial-temporal layer decomposition structure presented in the SPC, which is also a part of the MPEG-4 FAMC standard.

The experimental results show that up to 30% bitrate reductions compared to SPC can be achieved with the combination of proposed prediction schemes depending on the content and quantization level. The combination of proposed methods also achieve very competitive performance compared to non-scalable MPEG-4 FAMC coder.

CHAPTER 7

OPTIMAL QUALITY SCALABLE CODING OF ANIMATED MESHES

7.1 Introduction

In the literature, scalable coding has been thoroughly investigated for images, video and static 3D meshes. For the scalable animated mesh coding, the PCA based methods usually support scalability by decoding with a subset of the eigen-vectors. However, an important shortcoming of these methods is that scalability is provided for the entire mesh sequence, not a group of meshes and frame-wise decoding is not possible, which makes the methods unsuitable for streaming/transmission applications. Apart from PCA based methods, MPEG-4 FAMC and SPC support spatial and temporal scalability with desired streaming features by employing the same spatial and temporal layered decomposition. The SPC provide a slightly better scalable coding performance due to improvements in the prediction [66]. The quality scalability can also be achieved with these methods by treating lower resolution points as lower quality points by interpolating the missing vertices. But in this situation, the number of reconstruction points obtained is limited and the optimal order of layer decoding is unknown.

In this work, we propose two algorithms for optimized quality scalable coding of animated meshes. The first algorithm is based on optimal bitplane decoding order and the second one is based on optimal encoding order of bitplanes. The proposed algorithms make use of the spatial/temporal layered decomposition and prediction structure in SPC. However, by proposing bitplane extraction and optimal ordering, the proposed methods achieve more reconstruction points and better quality scalability performance than SPC.

The rest of the chapter is organized as follows: In Sections 7.2 and 7.3, we present the details

of the proposed quality scalability algorithms. In Section 7.4, we provide the experimental results and finally, we conclude in Section 7.5.

7.2 Proposed Quality Scalable Coding: Decoding Order Based

In this method, we adjust the decoding order of bitplanes by CABAC corresponding to each spatial layer in each frame. We name the method as Decoding Order Based (DOB). Note that we process frames GOM by GOM, i.e. we generate quality scalable bitstreams for each GOM.

The first contribution in this method is making use of CABAC bitplane decoding to achieve quality scalability, which is not available in MPEG-4 FAMC. We also make use of the properties that bitplanes of a compressed layer can be incrementally decoded individually and in mixed layer orders. In order to realize these properties, we embed the bitplane decoding order information as a header at the beginning of each GOM bitstream, which brings a negligible bitrate and complexity overhead. In this way, the decoder knows in what order it should use the GOM bitstream to decode the bitplanes.

Note that if the decoding order of the bitplanes is chosen such that decoding of a new spatial layer is started only after finishing the decoding of bitplanes of previous spatial layer completely, then what we obtain is simply the intermediate points of layer-wise scalable coding of SPC. However, it is not required to finish decoding of all bitplanes of a SL before proceeding to another SL. Figure 7.1 illustrates six example fixed bitplane orderings, which are ordering for each TL/SL/bitplane for each SL/bitplane/TL for each bitplane/TL/SL (denoted by TL/SL/BP->SL/BP/TL->BP/TL/SL in the figure). In this simple example, one GOM is encoded with three temporal layers, three spatial layers and two quantization levels. $BP_{i,j,k}$ denotes the bitstream of the quantization level k, SL j and frame i. Note that other than these six fixed orderings, any other bitplane ordering is also possible.

As different bitplane decoding orderings may result in better rate-distortion performance, we propose our second contribution: optimization of bitplane decoding order. To demonstrate the effect of different bitplane decoding orders, we illustrate a simple case where the first 9 frames of cow sequence is encoded with 8 spatial layers and 4 temporal layers. We examine the same six fixed bitplane ordering cases of Figure 7.1. The rate-distortion points are shown

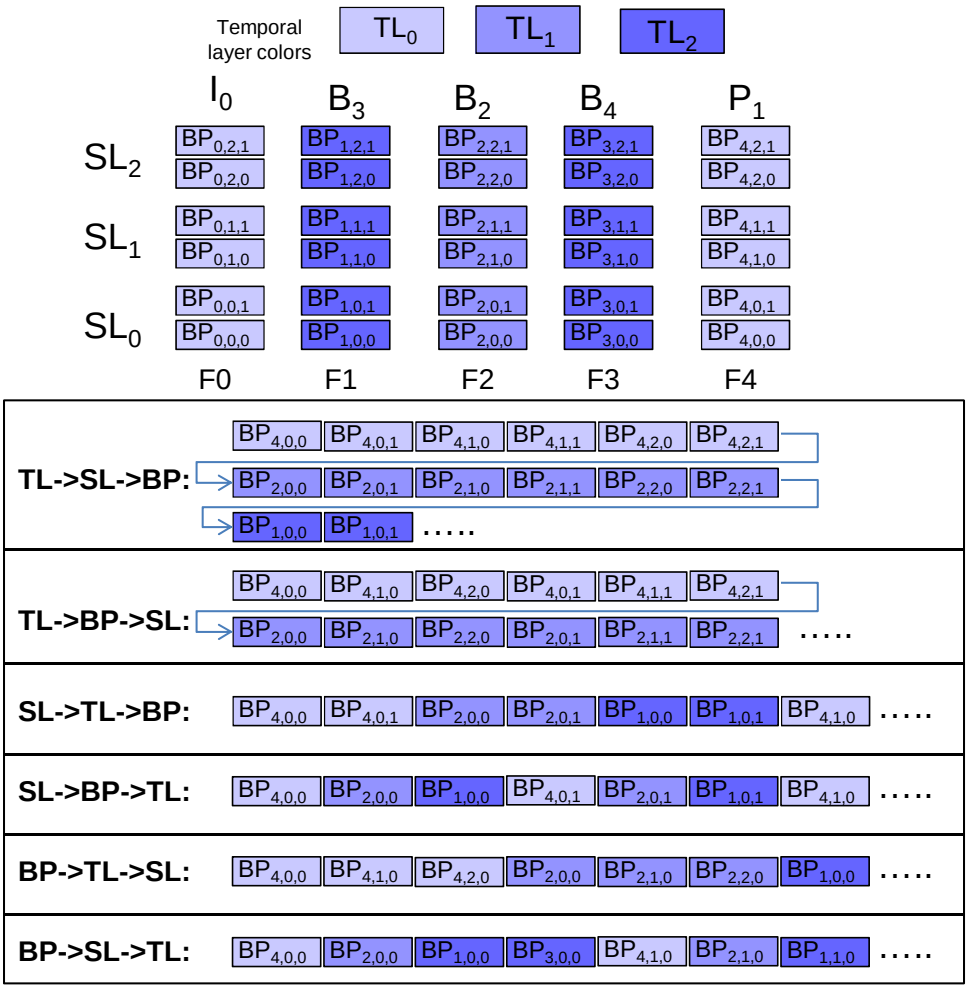


Figure 7.1: Illustration of different fixed bitplane orderings on an example case

in Figure 7.2. In the figure, the TL->SL->BP and SL->TL->BP cases consist of intermediate points of ordering TLs first and SLs first in SPC respectively. In all other cases, since predictions were performed using full bitplane precisions, whenever a prediction is performed using a lower bitplane precision, mismatch occurs. For this reason, for example, BL->SL->TL initially performs worse but then performs better at higher bitrates.

7.2.1 Optimization

As mentioned earlier, an optimization is needed to optimally order the decoding order of the bitplanes for each frame in a GOM and for each SL in a frame. Here, the optimization should be defined to minimize a cost function. In this work, as the optimal quality scalability

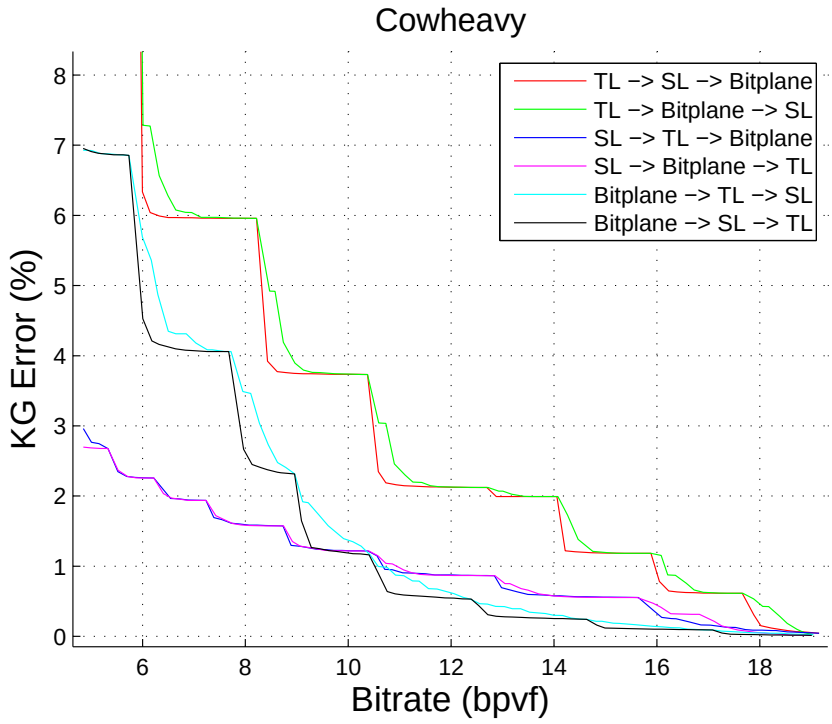


Figure 7.2: Rate-distortion curves of fixed bitplane ordering methods.

measure, we try to minimize the area under the RD curve to achieve good RD performance in the overall range of bitrates. Otherwise, it is also possible arrange the cost function so that better performance (lower distortion) is achieved for some bitrate intervals at the cost of poor performance in the remaining bitplanes. Other measures can also be incorporated depending on the application and we will see in the following sections that the proposed optimization framework can also support other optimization definitions.

Considering all the combinations of bitplane orderings, there exist too many possibilities to try. For this reason, we first impose reasonable restrictions on the decoding order of bitplanes such that for a bitplane in the decoding order, the following bitplanes should have been decoded before:

- All the previous bitplanes in the same SL and same frame
- All the previous SLs up to same bitplane level at the same frame
- All the reference frames up to same bitplane level at the same SL

After the restrictions, we can view the optimization procedure as an incremental bitplane selection process. Let $BP(f, s, q)$ denote the bitplane bitstream of compressed vertices in frame f of the GOM, spatial layer s, $0 \leq s \leq S-1$ in frame f and the quantization level q, $0 \leq q \leq Q-1$ such that

- f denotes the usual non-scalable encoding/decoding order of frames in a GOM obtained during hierarchical decomposition of temporal layers. Therefore f is between 0 and $2^{TL-1}-1$ and $f = 0$ means the first frame to encode/decode in a GOM and so on.
- $s = 0$ corresponds to the base spatial layer. Therefore, the usual encoding/decoding order is from $s = 0$ to $s = S-1$.
- $q = 0$ corresponds to the most significant bit case. As a result, the allowed bitplane decoding order is from $q = 0$ to $q = Q-1$

In this incremental procedure, initially, all the bitplanes are considered not decoded yet. We start with decoding $BP(0,0,0)$. Then, according to the previously mentioned decoding ordering restriction, next possible bitplanes are $BP(0,0,1)$ (next quantization level), $BP(0,1,0)$ (next spatial layer) and $BP(1,0,0)$ (next frame). After $BP(0,0,0)$, if $BP(0,0,1)$ is chosen, then the next possible bitplanes are $BP(0,0,2)$, $BP(0,1,0)$ and $BP(1,0,0)$. Similarly, if $BP(0,1,0)$ is chosen after $BP(0,0,0)$, then the next possible bitplanes are $BP(0,0,1)$, $BP(0,2,0)$ and $BP(1,0,0)$. Similarly, choosing $BP(1,0,0)$ results in the next possible bitplanes $BP(0,0,1)$, $BP(0,1,0)$, $BP(2,0,0)$ and $BP(3,0,0)$ (assuming GOM contains more than 2 frames).

In this way, many bitplane orderings are obtained incrementally. We formulate this ordering problem as a trellis structure. The initial branches of the trellis are illustrated in Figures 7.3(a), 7.3(b) and 7.3(c). In this trellis structure, each possible bitplane ordering is called a path. Each subset of bitplanes which can have an allowed decoding order is considered as a state. Therefore, each path can be viewed as a sequence of transitions between states where in each transition, the new state contains one bitplane more than the previous state. Each transition is called a branch. As a result, each path starts with the state which only consists of the single bitplane $BP(0,0,0)$, branches through different states as many as the total number of bitplanes in a GOM and finally ends with the state which consists of all the bitplanes.

Modeling the problem with the trellis structure, the optimization problem becomes finding

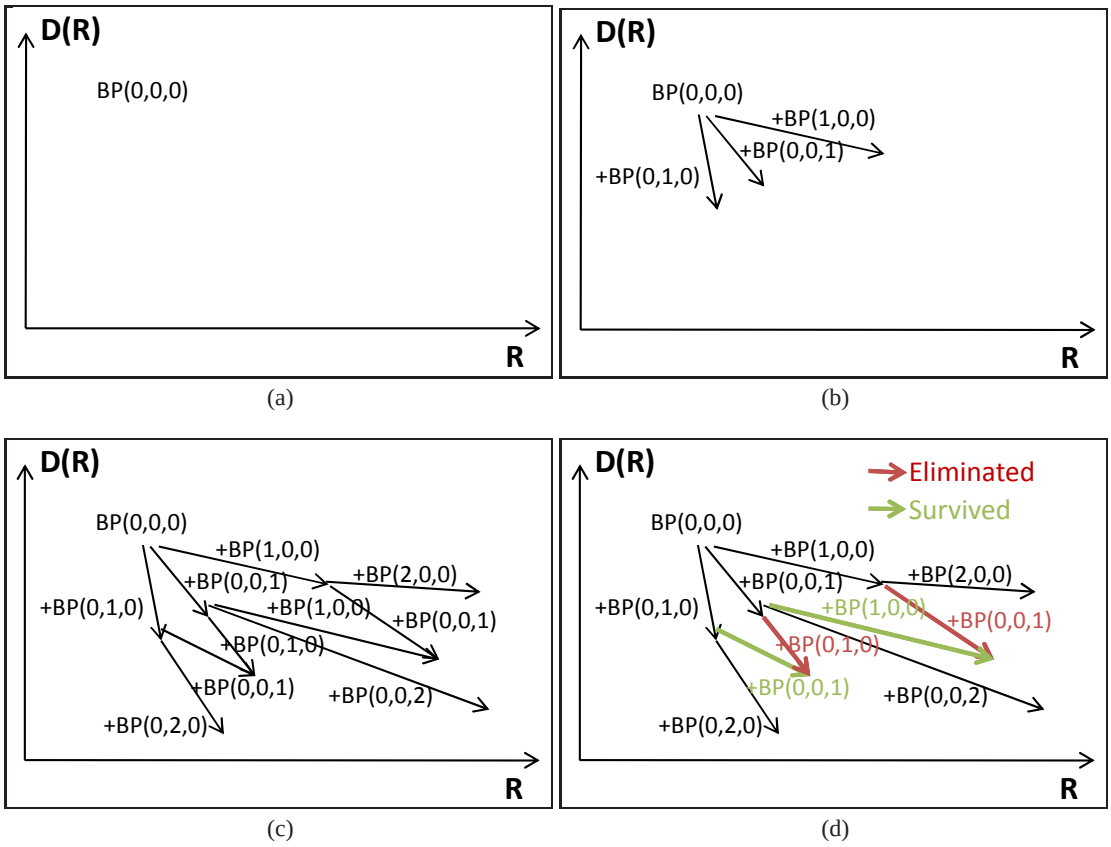


Figure 7.3: Illustration of first branches of the trellis structure used in optimization (a) Initial state (b) First branches (c) Some of the next possible branches (d) Elimination of paths when two paths end up at the same state (same RD point)

the optimal path in the trellis structure. Actually, this structure is very similar to the viterbi algorithm [101, 102] and we make use of it for the solution. In order to do this, we need to define a path metric which can be incrementally updated during each branch. Note that each branch corresponds to displacement in the RD curve. Since our aim is to minimize the area under the RD curve, we define the path metric to add during each branch as the area under the line defined by the previous RD point and the RD point achieved by the current branching.

In this way, among all possible paths, the one with the minimum path metric results in the RD curve with the minimum area under the RD curve. However, trying all the possible paths would be too complex. Again similar to the viterbi algorithm, this complexity is reduced as follows: After each branch, we check whether two paths reach the same state. If this happens, we compare the path metrics accumulated until that state and eliminate the path with higher path metric. This elimination is possible since the two paths start from the same RD point, reach the same RD point at that moment and they will continue with same branches since the

next possible bitplanes are the same and the same bitplanes are decoded until that point. This elimination process is illustrated in Figure 7.3(d).

In order to provide algorithmic description of the optimization procedure, we introduce the *path list* structure where the paths can be stored/removed during optimization. Before the optimization starts, the *path list* is initialized with $BP(0,0,0)$ and the corresponding RD point is calculated and stored. Then the description of the optimization algorithm is as follows:

Algorithm 1 Quality Scalability Optimization with DOB

Initialize the *path list* with $BP(0,0,0)$
while *path list* is not empty **do**
  for all Paths in the *path list* **do**
    Compute and store rate, distortion and path metric.
  end for
  for all Pairs of paths in the *path list* **do**
    if Two paths reach the same state **then**
      Remove the one with larger path metric from the list
    end if
  end for
  Eliminate a subset of the paths according to a rule, if exists.
  if All the paths are at the final state **then**
    break the loop.
  end if
  for all Paths remaining in the *path list* **do**
    Generate the next possible paths.
    if There exist next possible paths **then**
      Remove the current path from the list and update the list with the newly generated paths.
    end if
  end for
end while
The survived path in the *path list* is the optimal solution.

7.2.2 Rate, Distortion and Path Metrics

As seen in the optimization algorithm, rate, distortion and path metrics need to be defined. Rate metrics are straightforward to obtain since the number of compressed bits for each bit-plane is already available. For the distortion metric, the aim is to obtain the distortion of the GOM in the sense of distortion during optimization. In this work, we measure the distortion of a sequence with the KG error which is actually a scaled mean squared error (MSE). Therefore, the most accurate but also complex way to measure the distortion of a path is to decode the bitstream and compute the MSE between the decoded GOM and the original GOM. In our simulations we calculated the distortion metric in this way. We address the other possibilities to decrease complexity by approximating the distortion metric by using quantization values or incremental approximation of errors as future work. Once the rate and distortion metrics are obtained, the path metric is calculated as the sum of path metric in the previous state (the area under the RD curve until the previous state) and the resultant area under the RD curve with current branching.

7.2.3 Algorithmic Simplifications

As mentioned earlier, during the optimization if two paths reach the same state, one of them is eliminated by comparing the path metrics. The experiments showed that even with this simplification the optimization algorithm is still too complex. We note that any trellis based algorithm in the literature can be employed in our scenario. In order to decrease the complexity, we propose two algorithmic simplifications for path elimination in order to reduce complexity at the cost of worse optimization performance. The proposed simplified path elimination methods take place in the *Eliminate a subset of the paths according to a rule, if exists* step of the algorithm chart and described as follows:

- Slope based elimination: Path elimination occurs in every branch. During the optimization, after calculating RD points of all paths, the slopes of displacement in the RD curves are compared and all the paths other than the one with smallest slope (largest magnitude but smallest slope since slopes are negative) are eliminated.
- Rate interval based elimination: In this method, the motivation is that if a path is not the optimal solution until a rate value, then it is unlikely to be the optimal solution at

the end of the optimization. A rate interval is given as input parameter and the rate axis in the RD curve is divided into the given intervals. Then, instead of running the optimization and deciding in the end, a decision of path elimination is performed when all the paths just reach and possible pass the next rate interval. When a path reaches the rate interval, it pauses and waits for all the other paths to reach. When all the paths reach, the corresponding path metrics are compared and the smallest one survives. Choosing a larger rate interval makes the algorithm closer to the optimal solution but the complexity increases. Choosing a smaller rate interval converges to the slope based elimination method as the interval becomes very small. Therefore, this parameter gives a compromise between scalability performance and complexity.

- For the rate interval based elimination, we introduce another parameter denoted by *maxP*, which forces to make a path decision if the next rate value is not reached by the paths when the number of paths reached *maxP*. The aim of this parameter is to put an upper bound for the calculations since without the parameter and for some rate intervals, too many paths may be produced causing significant increase in the complexity.

7.3 Proposed Quality Scalable Coding: Encoding Order Based

The major problem in the DOB is that when decoding a SL, if the reference SL(s) are not decoded with full bitplane precision, then mismatch occurs and error propagates to following predicting SLs. In order to cope with this problem, we propose a mismatch free scalable coding method called Encoding Order Based (EOB).

In the EOB, unlike in DOB where the whole GOM is encoded initially and only the bitplane decoding order is optimized, the bitplane encoding procedure is also included during the optimization. The same trellis structure used in the DOB is employed with the following modifications: Before the optimization starts, no SLs are encoded. During the optimization, when a path branches to a BP which is the initial bitplane of a SL, the SL is fully encoded first and the reconstruction points and compressed bits spent corresponding to all the bitplanes are stored. During this encoding, reference SLs are used with the quantization precisions at that moment rather than the reconstruction values with full quantization (all bitplanes decoded case). In this way, mismatch is avoided since bitplane encoding order is same as the decoding

order. On the other hand, memory requirement is increased.

The optimization trellis again starts with $BP(0,0,0)$. Since initially no SL is encoded, the SL 0 in GOM frame 0 is encoded first and information of all bitplanes are stored. Again next possible bitplanes are $BP(0,0,1)$, $BP(0,1,0)$ and $BP(1,0,0)$. if $BP(0,0,1)$ is chosen, since the corresponding SL is encoded before, the stored information for this bitplane is used. If $BP(0,1,0)$ or $BP(1,0,0)$ is chosen, since these SLs are not encoded before, first the SL is encoded and information of all the bitplanes are stored. Note that during the encoding, these SLs use $SL(0,0)$ as reference. However, since the reconstruction of $SL(0,0)$ with only $BP(0,0,0)$ is used, the mismatch is avoided.

The disadvantage of EOB is that since reference SLs may be used with lower quantization precisions, the prediction accuracy decreases leading to loss in compression efficiency. On the other hand, increase in the distortion is much smaller compared to DOB which may face significant increase in distortion due to the mismatch. Another disadvantage is that during the optimization, according to different bitplane orderings, encoding of SLs need to be performed many times. On the other hand, the encoding in DOB is performed only once at the beginning and the compressed bit values are available during the optimization process.

In summary, the description of the optimization algorithm is as follows:

7.3.1 Rate, Distortion and Path Metrics

In order to achieve the optimal solution, the number of compressed bits for each bitplane should be used as the rate metric. Since this data change with respect to the quantization precisions of reference SLs, the complexity increases compared to DOB due to numerous encodings. For the distortion metric, the MSE between the decoded GOM and original GOM brings the optimal solution as in DOB. Again we address approximations of rate and distortion metrics to decrease complexity as future work.

Having obtained the rate and distortion metrics, the path metric is calculated as the sum of path metric in the previous state (the area under the RD curve until the previous state) and the resultant area under the RD curve with current branching as in DOB. However, this time when two paths reach the same state, although they achieve a similar distortion, the rate values may be different. The reason is that, during the branches and encoding of SLs, the reference SLs

Algorithm 2 Quality Scalability Optimization with EOB

```
1: Initialize the path list with BP(0, 0, 0)
2: while path list is not empty do
3:     for all Paths in the path list do
4:         Compute and store rate, distortion and path metric.
5:     end for
6:     for all Pairs of paths in the path list do
7:         if Two paths reach the same state then
8:             Remove the one with larger path metric from the list
9:         end if
10:    end for
11:    Eliminate a subset of the paths according to a rule, if exists.
12:    if All the paths are at the final state then
13:        break the loop.
14:    end if
15:    for all Paths remaining in the path list do
16:        Generate the next possible paths.
17:        if There exist next possible paths then
18:            Remove the current path from the list and update the list with the newly generated
               paths.
19:        end if
20:    end for
21: end while
22: The survived path in the path list is the optimal solution.
```

have different quantization levels causing different predictions. As a result, it is not sensible to compare the path metrics (or areas under RD curves) when two paths reach the same state since they do not end up at the same RD point. As a solution, we propose to compare the areas under the RD curves until the smaller rate point. For algorithmic simplifications, similar to the DOB, slope based elimination and rate interval based elimination are also employed in EOB.

7.4 Experimental Results

In this section, we evaluate the performances of different methods for scalable coding of animated meshes. The distortion of the reconstructed model is calculated with the KG error and the bitrate is expressed in bits per vertex per frame (bpvf). The results are presented with operational Rate-Distortion (RD) curves. For the scalable methods to be evaluated, a full scalable bitstream is obtained for the quantization level $Q = 14$ and the points on the RD curves correspond to possible reconstructions when the full bitstream is truncated until that bitrate value. The RD performance of reference encoder obtained by encoding the model with quantization levels between $Q = 8$ and $Q = 14$ are also provided as the ideal performance desired by the scalable methods. We denote the reference coder by non-scalable in the figures in the sense that each RD point on the curve correspond to a different bitstream, not a subset of the bitstream corresponding to the previous Q value. This should not be confused with the fact the reference encoder is also a scalable coder and we will also provide comparison with its scalable performance.

7.4.1 Slope Based Optimization

We start the results by first observing the performance of slope based optimization using the common encoding parameters $T = 4$ and $S = 8$. In Figure 7.4, we present comparison of slope based DOB and EOB with the best performing fixed bitstream ordering policies (SL->TL->BP, SL->BP->TL, BP->SL->TL). It can be observed that the proposed DOB and EOB perform significantly better than fixed policies in low and mid bitrates due to the fact that slope based methods start optimizing from lowest bitrate and chooses the next bitplane with sharpest slope. In this way initially, a good performance is achieved but as seen from

the figure, this does not guarantee a good performance in the high bitrates. However, on the average, the proposed slope based methods perform significantly better than fixed bitplane ordering policies.

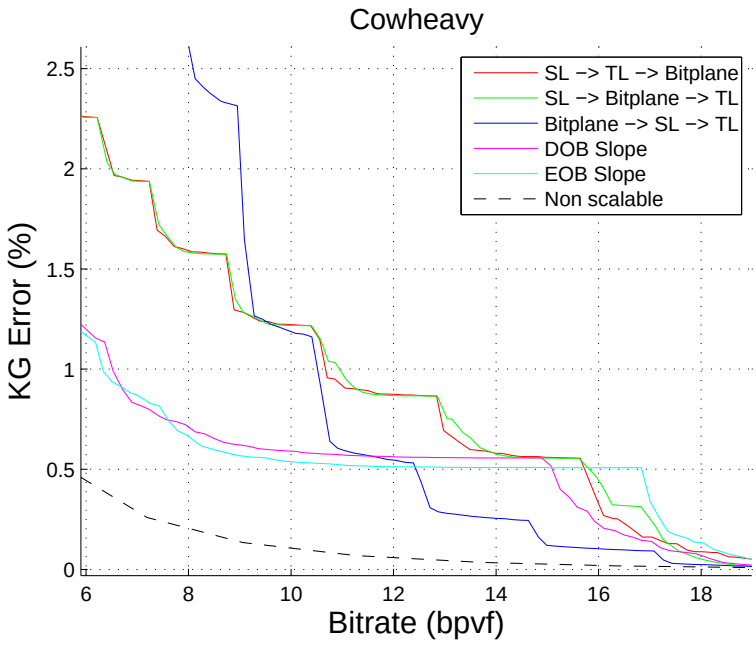


Figure 7.4: Comparison of slope based optimizations with fixed ordering policies

7.4.2 Simple Encoding Parameters - Full Trellis

Next, we continue with simple encoding parameters to evaluate the performance of full trellis structure. The simple encoding parameters consist of number of temporal layers (T) = 2, number of spatial layers (S) = 2. When larger encoding parameters are used, the simulation of full trellis takes unmeasurably long time. In Figure 7.5, comparison of full trellis and slope based approximation is provided for both DOB and EOB. It is observed that all the methods perform similarly for this specific simple parameters. The slope based methods perform usually slightly worse and sometimes moderately worse than full trellis based methods. In the rest of the experimental results, we will use the larger encoding parameters ($T = 4$ and $S = 8$) and will not present full trellis results due to considerably high complexity. However, the results in Figure 7.5 show that it may be possible to achieve results close to the full trellis.

As seen from the previous figures, the slope based methods may have problems due to memo-

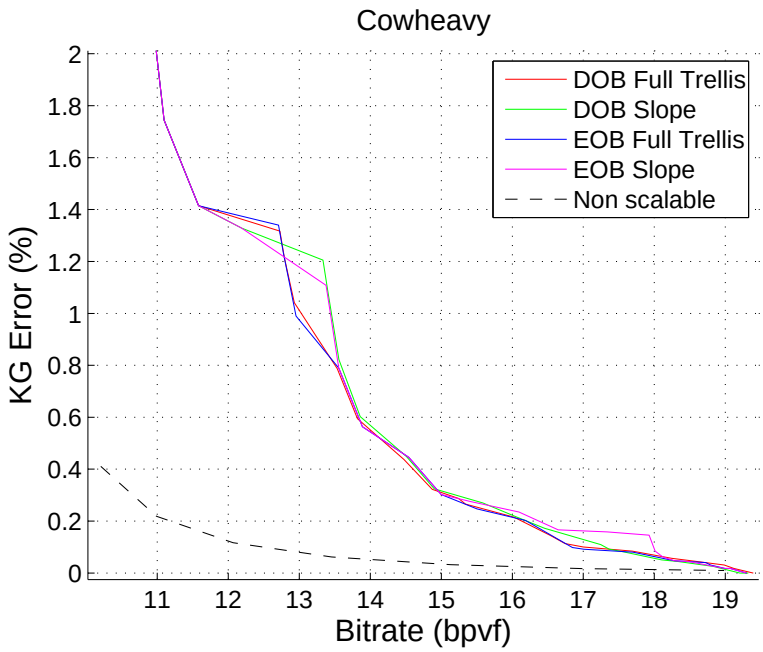


Figure 7.5: Comparison of full trellis and slope based optimization for simple coding parameters

ryless nature, i.e. the best decision at a branch may not be the best when more branches ahead are considered together. On the other hand, the full trellis structure which considers all feasible branching combinations takes unpractically long time. Therefore, we proposed the rate interval based optimization as a compromise solution which applies full trellis optimization for rate intervals given as a parameter. In the following results, we present the performance of rate interval based optimization by considering the rate interval, minP and maxP.

7.4.3 Effect of Rate Intervals

Figure 7.6 shows the comparison of choosing RInt=0.025, 0.050 and 0.100 bpvf for maxP=100 and 700. In the figure, one GOM of the cowheavy model is scalably encoded with DOB optimization. The results indicate that for this set of RInt values, increasing the RInt value brings minor improvement at the expense significant increase in the optimization time. Another observation is that by increasing the RInt value, the resultant RD curves tend to be more close to being convex, which is a desired property. Note that these results correspond to only one GOM case whereas results obtained by more GOMs provide better performance accuracy as will be presented in the following results.

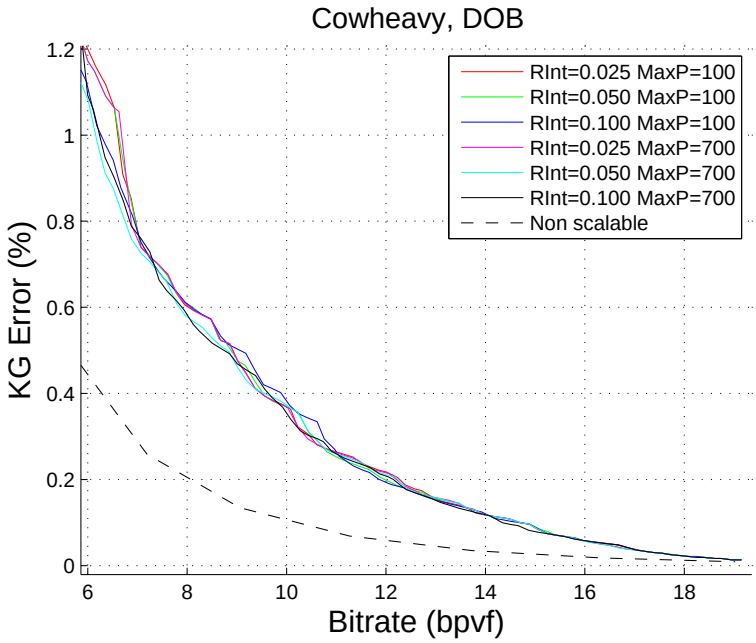


Figure 7.6: Comparison of different RInt and MaxP parameters for the rate interval based optimization method.

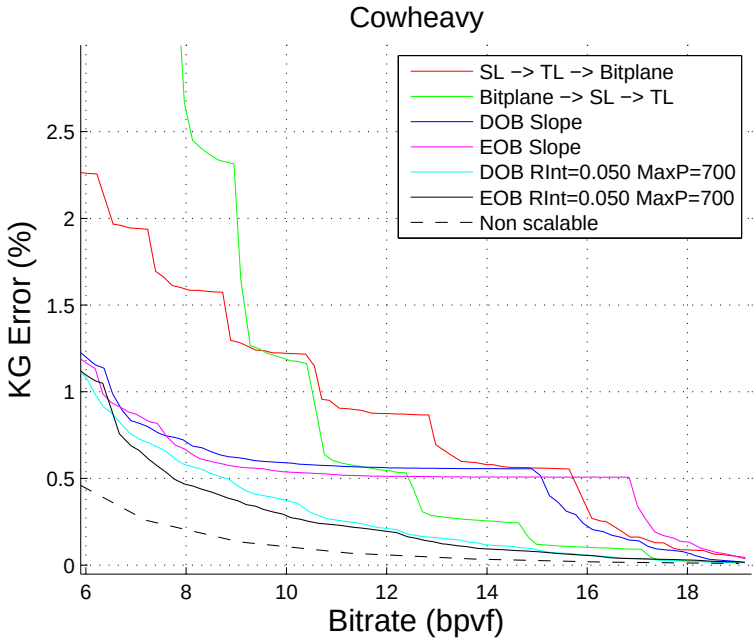


Figure 7.7: Comparison of best fixed ordering policies, slope based and rate interval based optimizations.

In Figure 7.7, we see a comparison of rate interval based optimization from previous figure with fixed ordering policies and slope based optimization. The figure shows that rate interval based methods outperform the other methods significantly. During comparison of slope based optimization with fixed ordering policies in Figure 7.4, we had observed that fixed policies performed better at high bitrates. Now we observe that using memory during the optimization by rate interval based optimization, it is possible to achieve better performance at all bitrates.

7.4.4 Comparison of DOB and EOB

In this part, we compare the operational rate distortion performances of the proposed methods DOB and EOB. The simulations are performed using the first 12 GOMs of the test models *cowheavy*, *chicken crossing* and *face*. For the DOB and EOB methods, slope based optimization and rate interval based optimization with various RInt and MaxP combinations are simulated.

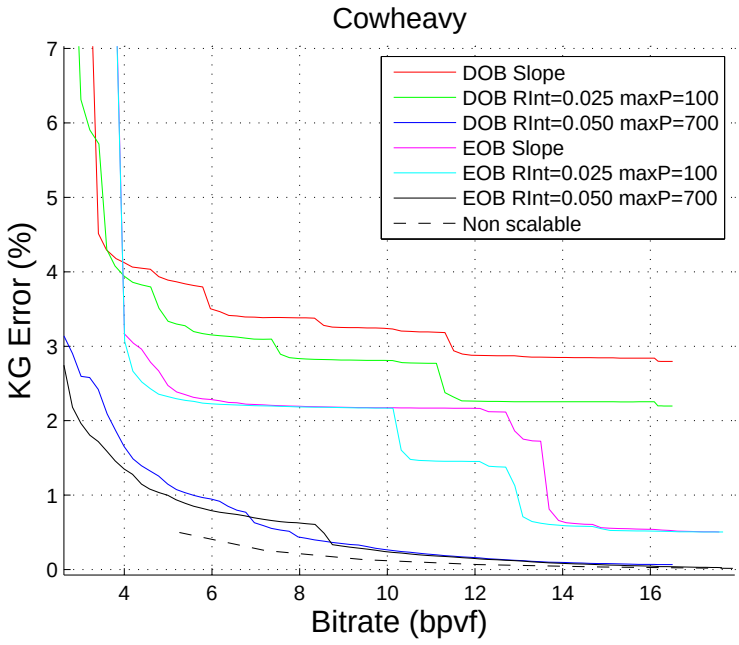


Figure 7.8: Comparison of proposed DOB and EOB methods for cowheavy model.

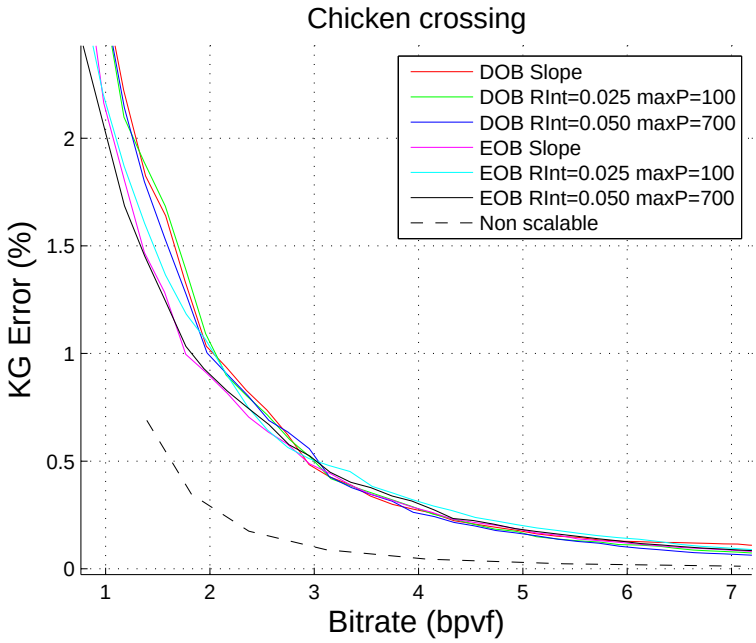


Figure 7.9: Comparison of proposed DOB and EOB methods for chicken crossing model.

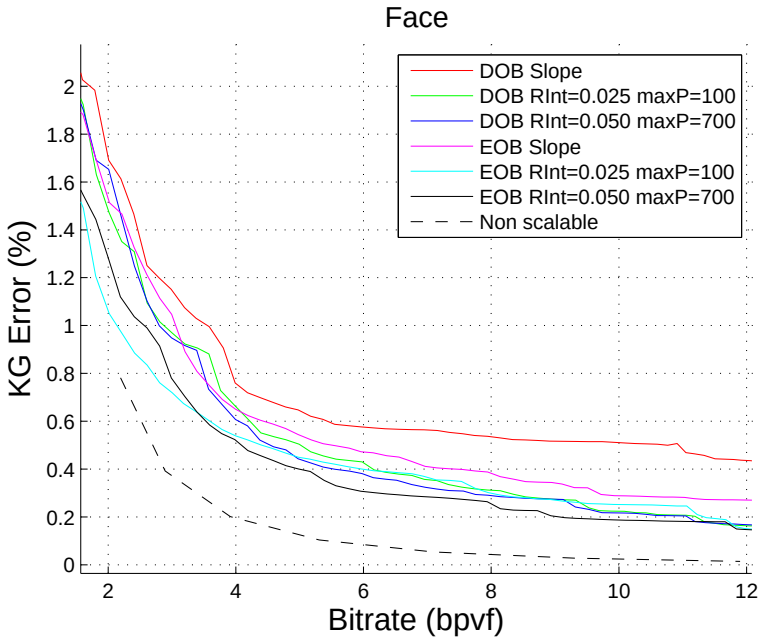


Figure 7.10: Comparison of proposed DOB and EOB methods for face model.

The resultant RD curves of the aforementioned combinations for the models cowheavy, chicken

crossing and face are presented in Figures 7.8, 7.9 and 7.10 respectively. From the figures, the following are observed:

- Except for the chicken crossing sequence, the rate interval based optimization methods outperform slope based methods significantly. For the chicken crossing sequence, the performances are quite similar.

- Comparing the results of one GOM and 12 GOMs for the cowheavy sequence, the performance difference between slope based optimization and rate interval based optimization is more significant for the 12 GOMs case. This is explained by the fact that the methods may behave differently for different GOMs.

- It can be observed from the figures that adding more memory into the system by increasing the RInt and maxP parameters improve the RD performance by getting closer to the non-scalable ideal curve and making the curve more convex.

- Comparing the proposed methods DOB and EOB, EOB performs better especially at lower bitrates due to the fact the mismatch problem of DOB is more severe in low bitrates.

7.4.5 Complexity Considerations

As mentioned earlier, running the full trellis optimization takes unmeasurably long time and requires huge memory usage. However, computation of the many paths during the optimization by full trellis is redundant since these paths have a very low chance of being the optimal solution. Therefore, we proposed several approaches to decrease the computation requirement and rate-distortion results showed that quite acceptable results can be achieved.

In this part, we provide the optimization times of the proposed approached in the encoder. Note that the optimization is performed in the encoder and the optimization result is embedded in the encoded bitstream. Since the decoder just reads this information and arranges the bitplane decoding order, there is negligibly small increase in the decoding time which is approximately same for all the proposed methods.

The optimization times of the proposed methods are given in Table 7.1. The resultant areas

Table 7.1: Average optimization time during encoding per GOM in minutes.

Method	Opt. time (m)	Area Under RD Curve
DOB Slope	1.28	52.30
EOB Slope	1.95	38.22
DOB RInt=0.025 maxP=100	2.67	45.95
EOB RInt=0.025 maxP=100	3.25	35.12
DOB RInt=0.050 maxP=700	67.94	9.97
EOB RInt=0.050 maxP=700	89.61	8.64

under the RD curves are also presented since the algorithms aim optimize this value. The values are calculated for number of minutes passed during the optimization process per GOM. From the table, it can be observed that DOB takes around 30-50% less time than EOB. As expected, slope based methods take the smallest time. Another observation is that the optimization time can significantly change with respect to *RInt* and *MaxP* parameters.

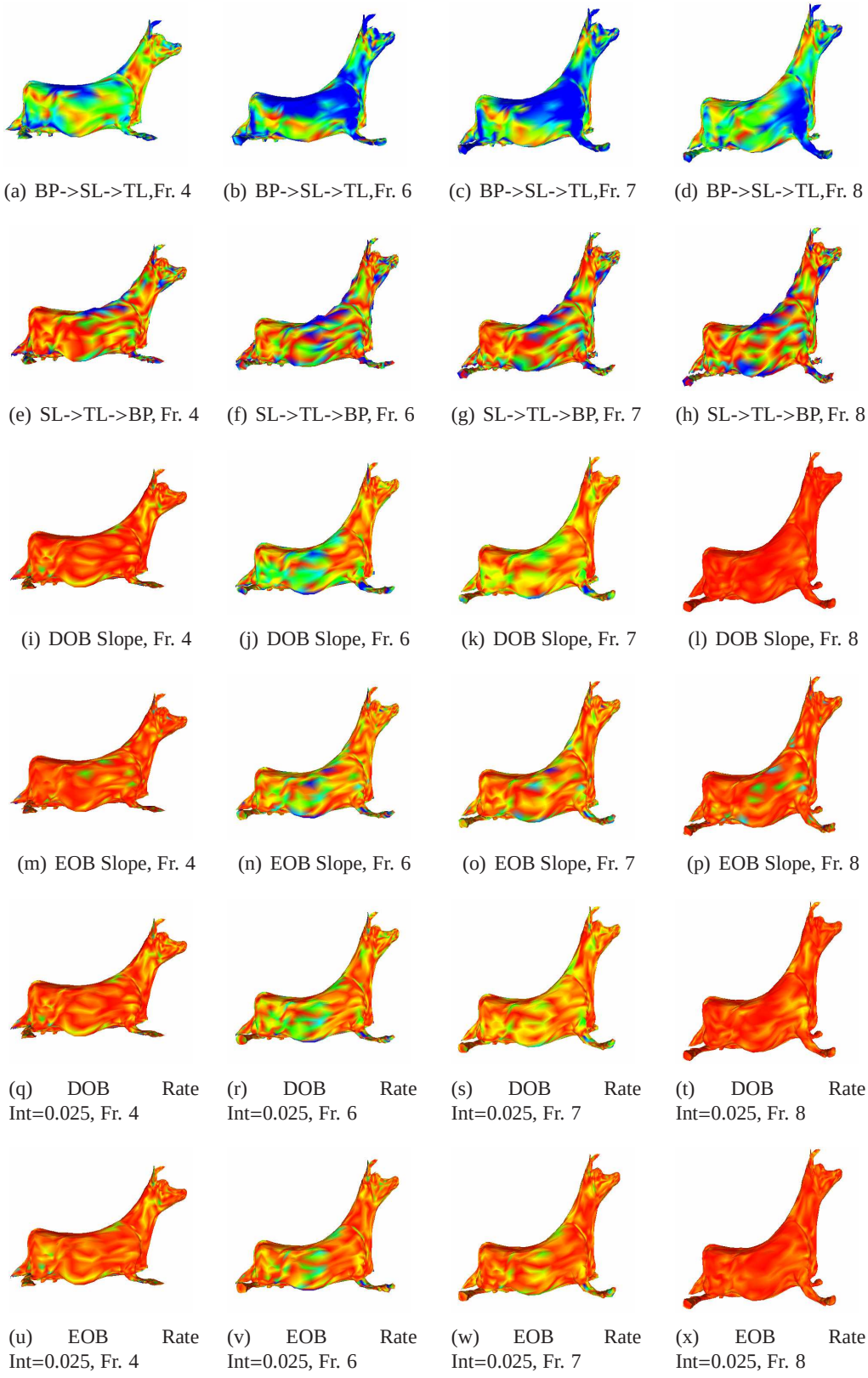
According to the current implementation and processor speeds, in order to achieve significant rate-distortion improvements with these parameters, optimization times at the order of hours are required, which may seem very high for some applications like the ones requiring real time compression. However, in many applications, the encoding may be performed offline and the decoder is required to be fast. For these applications, the proposed methods are suitable. In addition, the presented optimization time results are for the current implementation, where more efficient and optimized implementations may be possible. For example, parallel processing with multi-core processors or GPUs may be efficiently utilized in the trellis structure.

7.4.6 Visual Comparisons

In this part, apart from the objective metrics, we also show visual comparison of proposed methods and best performing fixed ordering policies. The frames are 4, 6, 7 and 8 of the cowheavy sequence are reconstructed by adjusting the compressed bitplane to achieve 8 bpvf and 12 bpvf. The reason behind choosing these frames is that every frame corresponds to a different temporal layer. Frame 8 is in TL0, frame 4 is TL1, frame 6 is in TL2 and frame 7 is in TL3. The reconstructed frames are illustrated in Figures 7.11 and 7.12 for 8 and 12 bpvf respectively. Note that in the figures, the models are colored according to the errors in the surface such that red means low error and deviation from red to other colors mean increasing

error.

Examining the figures, first of all, the figure for the 12 bpvf case has more red colors compared to 8 bpvf expectedly. When the methods are compared, the results of the objective comparisons are confirmed. The proposed methods have much smaller errors than fixed ordering policies. In addition, the distribution of error among frames are more uniform. In fixed ordering policies, it is possible to have some frames at very high quality and some of the frames at very low quality. The rate interval based methods perform better than slope based optimization. Comparing DOB and EOB, EOB looks to perform better slightly for some cases and moderately for some other cases.



(a) BP->SL->TL,Fr. 4 (b) BP->SL->TL,Fr. 6 (c) BP->SL->TL,Fr. 7 (d) BP->SL->TL,Fr. 8

(e) SL->TL->BP, Fr. 4 (f) SL->TL->BP, Fr. 6 (g) SL->TL->BP, Fr. 7 (h) SL->TL->BP, Fr. 8

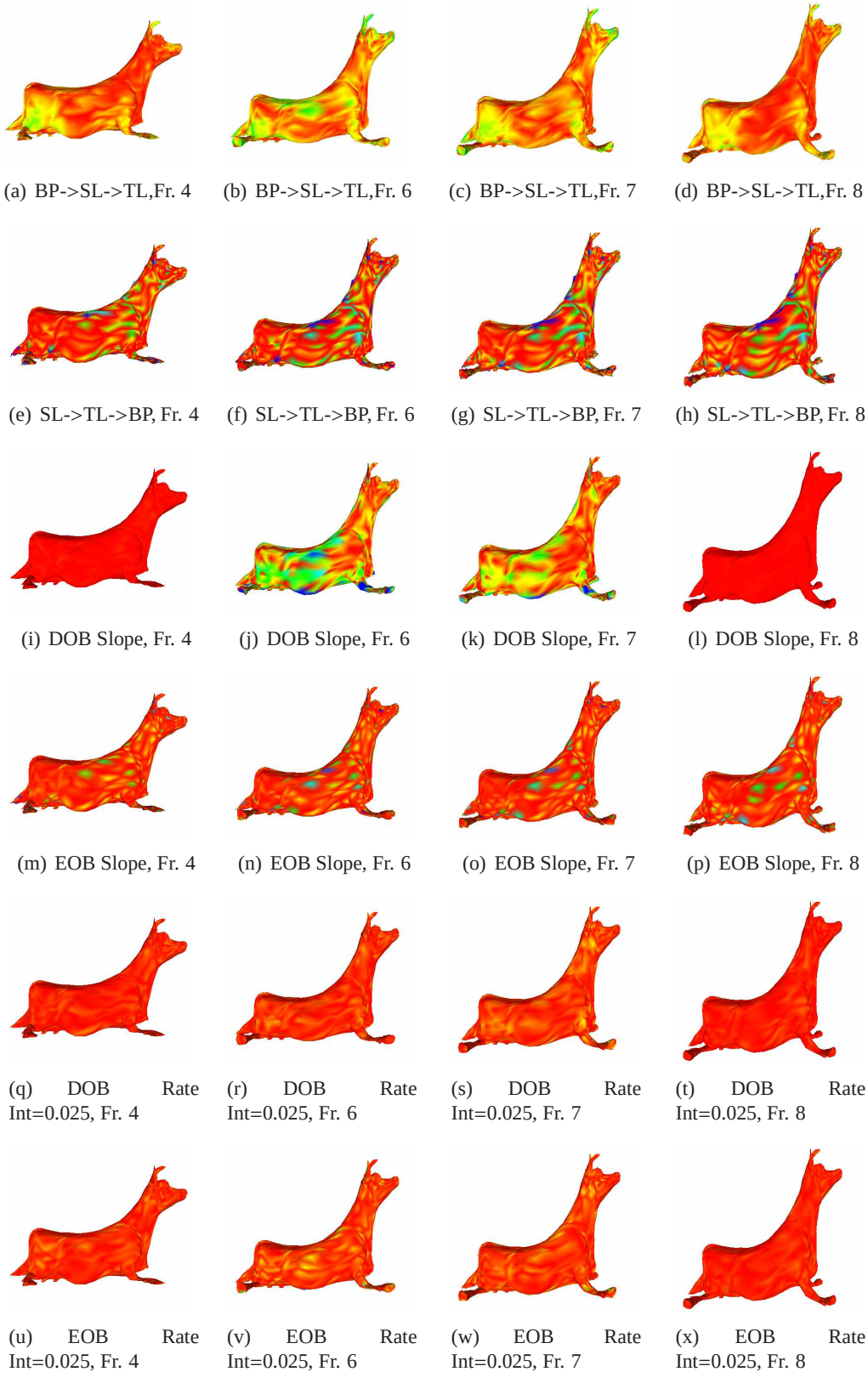
(i) DOB Slope, Fr. 4 (j) DOB Slope, Fr. 6 (k) DOB Slope, Fr. 7 (l) DOB Slope, Fr. 8

(m) EOB Slope, Fr. 4 (n) EOB Slope, Fr. 6 (o) EOB Slope, Fr. 7 (p) EOB Slope, Fr. 8

(q) DOB Rate Int=0.025, Fr. 4 (r) DOB Rate Int=0.025, Fr. 6 (s) DOB Rate Int=0.025, Fr. 7 (t) DOB Rate Int=0.025, Fr. 8

(u) EOB Rate Int=0.025, Fr. 4 (v) EOB Rate Int=0.025, Fr. 6 (w) EOB Rate Int=0.025, Fr. 7 (x) EOB Rate Int=0.025, Fr. 8

Figure 7.11: Comparison of visual reconstructions of frames 4,6,7,8 for several methods decoded at 8 bpvf.



(a) BP->SL->TL,Fr. 4 (b) BP->SL->TL,Fr. 6 (c) BP->SL->TL,Fr. 7 (d) BP->SL->TL,Fr. 8

(e) SL->TL->BP, Fr. 4 (f) SL->TL->BP, Fr. 6 (g) SL->TL->BP, Fr. 7 (h) SL->TL->BP, Fr. 8

(i) DOB Slope, Fr. 4 (j) DOB Slope, Fr. 6 (k) DOB Slope, Fr. 7 (l) DOB Slope, Fr. 8

(m) EOB Slope, Fr. 4 (n) EOB Slope, Fr. 6 (o) EOB Slope, Fr. 7 (p) EOB Slope, Fr. 8

(q) DOB Rate Int=0.025, Fr. 4 (r) DOB Rate Int=0.025, Fr. 6 (s) DOB Rate Int=0.025, Fr. 7 (t) DOB Rate Int=0.025, Fr. 8

(u) EOB Rate Int=0.025, Fr. 4 (v) EOB Rate Int=0.025, Fr. 6 (w) EOB Rate Int=0.025, Fr. 7 (x) EOB Rate Int=0.025, Fr. 8

Figure 7.12: Comparison of visual reconstructions of frames 4,6,7,8 for several methods decoded at 12 bpvf.

7.5 Conclusion

In this work, we have developed quality scalability support for predictive layered animated mesh coding structure, which is also a part of the MPEG-4 FAMC standard. We have introduced two methods to achieve quality scalability by ordering bitplanes, namely DOB and EOB. We have proposed the usage of a trellis structure for optimal bitplane ordering and proposed several methods to decrease computational complexity of optimization process.

Experimental results show that compared to quality scalable achieved by fixed layer/bitplane ordering of reference SPC coder, the proposed DOB and EOB methods achieve rate distortion curves significantly closer to the desired non-scalable RD curve. For the trellis structure in the optimization, trade-off between complexity and RD performance can be achieved by the rate interval based approximation. Comparing DOB and EOB, EOB usually performs better at lower bitrates due to the fact that EOB avoids mismatch whereas mismatch present at DOB affects the performance more at lower bitrates. The performance is similar at higher bitrates. We note that the performance difference is also content dependent. In order to provide basic comparison of DOB and EOB methods, we measured the optimization time in the encoder and observed that DOB takes around 30-50% less time since there exists only one encoding in the DOB optimization. Finally, subjective analysis by visual reconstructions confirmed the objective results and also indicated that optimized quality scalable coding also provides better balanced error distribution among frames.

In this work, the area under the RD curve was chosen as the path metric during the optimization. However, the optimization framework can be modified to support different purposes. For example, the path metric can be modified such that in addition to the RD performance, minimizing the spatial and/or temporal variance are also taken into account. Other visual quality metrics can also be used in path metrics.

CHAPTER 8

CONCLUSION

In this thesis, we studied several aspects of robust transmission of both static and animated 3D meshes. First, we presented our contributions for MDC and error resilient coding of static 3D meshes followed by the contributons for the MDC of animated meshes in the next chapter. Then we introduced improvements for predictive animated mesh coding to improve compression performance. Finally, we presented two methods for optimal quality scalable coding for animated meshes. In summary, the following conclusions are drawn:

In the first work, for the MDC of static meshes, the MDSQ based method is an early work whereas better results are obtained by the other methods we studuied, *TM-MDC* and *FEC*. Both of the methods employ optimization with respect to varying bandwidth and loss rate of the channel, which is not available in neither the only previous work in the literature [76] nor the MDSQ based method. Moreover, number of descriptions and description sizes can be adjusted more flexibly in *TM-MDC* and *FEC* based methods. Comparing *TM-MDC* and *FEC* based methods, the methods show similar performance in terms of expected distortion. However, *FEC* based method has several advantages. While the *FEC* based method generates one compressed bitstream during optimization for any number of descriptions, *TM-MDC* needs to generate different compressed bitstream to optimize for different total number of descriptions. Another advantage of *FEC* based method is that *TM-MDC* needs to include whole coarsest level geometry in each description, which may cause high redundancy for higher number of descriptions whereas *FEC* based method spreads the bitplanes of coarsest level geometry according to their importances in compressed bitstream.

For the usage of the proposed *FEC* based method in packet loss resilient streaming of static meshes, we have presented an extensive analysis of loss resilient coding methods which are

based on optimally protecting compressed bitstreams with FEC codes with respect to given channel bandwidth and packet loss rate constraints. For the CPM based methods, we introduced a general problem definiton with solution and the $kStep$ parameter to iterate protection rates with different steps. The experiments indicates that considerable decrease in optimization time can be achieved by increasing the $kStep$ at the expense of very small PSNR degradation. For the PGC based methods, we proposed RD curve modeling which performed very close to using original RD curve while providing significant decrease in optimization time. Comparing the CPM and PGC based methods, experimental results show that PGC methods achieve approximately 10db better PSNR for all loss rates. This shows that 10dB compression performance difference between the PGC and CPM is preserved in packet loss resilient transmission. Apart from the PSNR performance, PGC based methods have an advantage of flexible packetization due to the embedded structure of the bitstream which needs to be generated only once for the PGC method. Simulation in scenarios where the optimization and channel loss rates mismatch shows that when the model encounters a channel with a loss rate higher than the optimized rate, the performance degradation can be severe. On the other hand, when the encountered channel loss rate is lower than the optimization rate, the loss in the performance is not significant. Therefore when the channel conditions are uncertain or time varying, it is more robust to optimize loss protection with respect to a higher loss rate.

In the second work of MDC of animated meshes work, we analyzed performance of the three proposed methods with redundancy-rate-distortion curves and visual reconstructions. The experimental results show that *Vertex partitioning* performs better at low redundancies for especially spatially dense models. *Temporal subsampling* performs better at moderate redundancies (corresponding to including at least the spatial base layer in both descriptions) as well as low redundancies for spatially coarse models. Layer duplication based MDC can achieve the lowest redundancies with flexible redundancy allocation capability and can be designed to achieve the smallest variance of reconstruction quality between consecutive frames.

In the third work, we developed prediction improvements for scalable predictive animated mesh coding. The improvements are based on weighted spatial prediction, weighted temporal prediction and exploitation of angular relations of triangles between the frames. The methods we introduced depend on the spatial-temporal layer decomposition structure presented in the SPC, which is also a part of the MPEG-4 FAMC standard. The experimental results show that up to 30% bitrate reductions compared to SPC can be achieved with the combination of pro-

posed prediction schemes depending on the content and quantization level. The combination of proposed algorithms also achieve very competitive performance compared to non-scalable MPEG-4 FAMC coder.

In our final work, we introduced quality scalability support for predictive layered animated mesh coding structure, which is also a part of the MPEG-4 FAMC standard. We presented two methods to achieve quality scalability by ordering bitplanes, namely DOB and EOB. We proposed the usage of a trellis structure for optimal bitplane ordering and proposed several methods to decrease computational complexity of optimization process. Experimental results show that compared to quality scalable achieved by fixed layer/bitplane ordering of reference SPC coder, the DOB and EOB methods achieve rate distortion curves significantly closer to the desired non-scalable RD curve. For the trellis structure in the optimization, trade-off between complexity and RD performance can be achieved by the rate interval based approximation. Comparing DOB and EOB, EOB usually performs better at lower bitrates due to the fact that EOB avoids mismatch whereas mismatch present at DOB affects the performance more at lower bitrates. The performance is similar at higher bitrates. We note that the performance difference is also content dependent. In order to provide basic comparison of DOB and EOB methods, we measured the optimization time in the encoder and observed that DOB takes around 30-50% less time since there exists only one encoding in the DOB optimization. Finally, subjective analysis by visual reconstructions confirmed the objective results and also indicated that optimized quality scalable coding also provides better balanced error distribution among frames.

REFERENCES

[1] A. Norkin, M. O. Bici, G. B. Akar, A. Gotchev, and J. Astola, "Wavelet-based multiple description coding of 3-D geometry," in *VCIP'07, Proc. SPIE*, vol. 6508, San-Jose, US, Jan. 2007, pp. 65 082I–1 –65 082I–10.

[2] M. O. Bici and G. Bozdagi Akar, "Multiple description scalar quantization based 3D mesh coding," in *Proc. IEEE Int. Conf. Image Processing*, Atlanta, US, Oct. 2006, pp. 553–556.

[3] A. Norkin, M. O. Bici, A. Aksay, C. Bilen, A. Gotchev, G. Akar, K. Egiazarian, and J. Astola, "Multiple Description Coding and its Relevance to 3DTV," *Three-Dimensional Television*, pp. 371–426, 2008.

[4] M. O. Bici, A. Norkin, G. Bozdagi Akar, A. Gotchev, and J. Astola, "Multiple description coding of 3D geometry with forward error correction codes," in *3DTV-Conference The True Vision - Capture, Transmission and Display of 3D Video*, Kos, Greece, May 2007.

[5] M. O. Bici, A. Norkin, and G. Bozdagi Akar, "Packet loss resilient transmission of 3D models," in *Proc. IEEE Int. Conf. Image Processing (ICIP 2007)*, San Antonio, US, 2007.

[6] M. O. Bici and G. B. Akar, "Joint Source-Channel Coding for Error Resilient Transmission of Static 3D Models," *Accepted for publication in Turkish Journal of Electrical Engineering and Computer Sciences*.

[7] M. O. Bici, N. Stefanoski, and G. B. Akar, "Vertex partitioning based Multiple Description Coding of 3D dynamic meshes," in *3DTV-Conference The True Vision - Capture, Transmission and Display of 3D Video*. IEEE, 2009, pp. 1–4.

[8] M. O. Bici and G. B. Akar, "Multiple description coding of 3D dynamic meshes based on temporal subsampling," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 7543, 2010, p. 10.

[9] M. O. Bici and G. B. Akar, "Multiple description coding of animated meshes," *Signal Processing: Image Communication*, vol. In Press, Corrected Proof, pp. –, 2010.

[10] ——, "Improved prediction for layered predictive animated mesh compression," in *Proc. IEEE Int. Conf. Image Processing (ICIP 2010)*, Hong Kong, 2010.

[11] ——, "Improved Prediction Methods For Scalable Predictive Animated Mesh Compression," *Submitted to Journal of Visual Communication and Image Representation*.

[12] ——, "Optimal Quality Scalability for Predictively Coded Animated Meshes," *Submitted to IEEE Transactions on Multimedia*.

[13] V. Goyal, “Multiple description coding: compression meets the network,” *IEEE Signal Processing Mag.*, vol. 18, pp. 74–93, September 2001.

[14] Y. Wang, A. Reibman, and S. Lin, “Multiple Description Coding for Video Delivery,” *Proceedings of the IEEE*, vol. 93, no. 1, pp. 57–70, 2005.

[15] Y. Wang, M. Orchard, V. Vaishampayan, and A. Reibman, “Multiple description coding using pairwise correlating transforms,” *Image Processing, IEEE Transactions on*, vol. 10, no. 3, pp. 351–366, 2001.

[16] R. R. Blahut, *Theory and Practice of Error Control Codes*. Reading: Addison-Wesley, 1983.

[17] P. Alliez and C. Gotsman, “Recent advances in compression of 3D meshes,” in *Proc. Symposium on Multiresolution in Geometric Modeling*, 2003.

[18] J. Peng, C.-S. Kim, and C.-C. J. Kuo, “Technologies for 3D mesh compression: A survey,” *Journal of Visual Communication and Image Representation*, vol. 16, pp. 688–733, Dec. 2005.

[19] A. Khodakovsky, P. Schröder, and W. Sweldens, “Progressive geometry compression,” in *Siggraph 2000, Computer Graphics Proceedings*, 2000, pp. 271–278.

[20] R. Pajarola and J. Rossignac, “Compressed progressive meshes,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 79–93, Jan. 2000.

[21] A. Khodakovsky and I. Guskov, “Compression of normal meshes,” in *Geometric Modeling for Scientific Visualization. Springer-Verlag*, 2003.

[22] F. Morán and N. García, “Comparison of wavelet-based three-dimensional model coding techniques,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 7, pp. 937–949, July 2004.

[23] S. Lavu, H. Choi, and R. Baraniuk, “Estimation-quantization geometry coding using normal meshes,” in *DCC ’03: Proceedings of the Conference on Data Compression*. Washington, DC, USA: IEEE Computer Society, 2003, p. 362.

[24] F. Payan and M. Antonini, “3D mesh wavelet coding using efficient model-based bit allocation,” *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, pp. 391 – 394, 2002.

[25] K. Köse, A. E. Çetin, U. Güdükbay, and L. Onural, “3D model compression using connectivity-guided adaptive wavelet transform built into 2D SPIHT,” *J. Vis. Comun. Image Represent.*, vol. 21, pp. 17–28, January 2010. [Online]. Available: http://dx.doi.org/10.1016/j.jvcir.2009.09.007

[26] F. Morán, P. Gioia, M. Steliaros, M. Bourges-Sévenier, and N. García, “Subdivision surfaces in MPEG-4,” in *Proc. IEEE ICIP*, vol. III, Sept 2002, pp. 5–8.

[27] M. Bourges-Sévenier and E. S. Jang, “An introduction to the MPEG-4 animation framework eXtension,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 7, pp. 928–936, July 2004.

[28] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, "MAPS: Multiresolution adaptive parameterization of surfaces," in *Proceedings of SIGGRAPH 98*, July 1998, pp. 95–104.

[29] C. Loop, "Smooth subdivision surfaces based on triangles," *Master's Thesis, Univ. of Utah, Dept. of Mathematics*, 1987.

[30] N. Dyn, D. Levin, and J. A. Gregory, "A butterfly subdivision scheme for surface interpolation with tension control," *ACM Trans. Graph.*, vol. 9, no. 2, pp. 160–169, Apr. 1990.

[31] C. Touma and C. Gotsman, "Triangle mesh compression," in *Proc. Graphics Interface*, Vancouver, BC, Canada, Jun. 1998.

[32] A. Said and W. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, June 1996.

[33] J. E. Lengyel, "Compression of time-dependent geometry," in *Proceedings of the 1999 symposium on Interactive 3D graphics.*, 1999, pp. 89–95.

[34] J. Ahn, C. Kim, C. Kuo, and Y. Ho, "Motion-compensated compression of 3D animation models," *Electronics Letters*, vol. 37, no. 24, pp. 1445–1446, 2001.

[35] J. Zhang and C. Owen, "Octree-based animated geometry compression," in *Data Compression Conference, 2004. Proceedings. DCC 2004*, 2004, pp. 508–517.

[36] ——, "Octree-based animated geometry compression," *Computers & Graphics*, vol. 31, no. 3, pp. 463–479, 2007.

[37] K. Muller, A. Smolic, M. Kautzner, P. Eisert, and T. Wiegand, "Predictive Compression of Dynamic 3D Meshes," *Image Processing, 2005 IEEE International Conference on*, vol. 1, 2005.

[38] K. Müller, A. Smolic, M. Kautzner, P. Eisert, and T. Wiegand, "Rate-distortion-optimized predictive compression of dynamic 3D mesh sequences," *Signal Processing: Image Communication*, vol. 21, no. 9, pp. 812–828, 2006.

[39] K. Mamou, T. Zaharia, and F. Prêteux, "A skinning approach for dynamic 3d mesh compression," *Computer Animation and Virtual Worlds*, vol. 17, no. 3-4, pp. 337–346, 2006.

[40] I. Guskov and A. Khodakovsky, "Wavelet compression of parametrically coherent mesh sequences," in *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2004, pp. 183–192.

[41] F. Payan and M. Antonini, "Wavelet-based compression of 3D mesh sequences," *Proceedings of IEEE ACIDCA-ICMI 2005*, 2005.

[42] ——, "Temporal wavelet-based compression for 3D animated models," *Computers & Graphics*, vol. 31, no. 1, pp. 77–88, 2007.

[43] J. Cho, M. Kim, S. Valette, H. Jung, and R. Prost, "3-d dynamic mesh compression using wavelet-based multiresolution analysis," in *2006 IEEE International Conference on Image Processing*, 2006, pp. 529–532.

[44] Y. Boulfani-Cuisinaud and M. Antonini, "Motion-based geometry compensation for dwt compression of 3D mesh sequence," in *IEEE International Conference in Image Processing (CD-ROM)*, 2007.

[45] H. Briceno, P. Sander, L. McMillan, S. Gortler, and H. Hoppe, "Geometry videos: A new representation for 3D animations," in *Proceedings of ACM Symposium on Computer Animation 2003.*, 2003, pp. 136–146.

[46] X. Gu, S. J. Gortler, and H. Hoppe, "Geometry images," in *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 2002, pp. 355–361.

[47] K. Mamou, T. Zaharia, and F. Prêteux, "Multi-chart geometry video: A compact representation for 3D animations," in *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, 2006, pp. 711–718.

[48] M. Alexa and W. Muller, "Representing Animations by Principal Components," *Computer Graphics Forum*, vol. 19, no. 3, pp. 411–418, 2000.

[49] Z. Karni and C. Gotsman, "Compression of soft-body animation sequences," *Computers & Graphics*, vol. 28, no. 1, pp. 25–34, 2004.

[50] M. Sattler, R. Sarlette, and R. Klein, "Simple and efficient compression of animation sequences," *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 209–217, 2005.

[51] R. Amjoun, R. Sondershaus, and W. Straßer, "Compression of complex animated meshes," *Advances in Computer Graphics*, pp. 606–613, 2006.

[52] R. Amjoun and W. Straßer, "Efficient compression of 3-D dynamic mesh sequences," *J. WSCG*, 2007.

[53] J. Heu, C. Kim, and S. Lee, "SNR and temporal scalable coding of 3-D mesh sequences using singular value decomposition," *Journal of Visual Communication and Image Representation*, 2009.

[54] J. Rossignac, "Edgebreaker: Connectivity compression for triangle meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 1, pp. 47–61, 1999.

[55] L. Váša and V. Skala, "Coddyac: Connectivity driven dynamic mesh compression," in *3DTV-Conference The True Vision - Capture, Transmission and Display of 3D Video*, 2007, pp. 1–4.

[56] ——, "Combined compression and simplification of dynamic 3D meshes," *Computer Animation and Virtual Worlds*, vol. 20, no. 4, pp. 447–456, 2009.

[57] ——, "COBRA: Compression of the Basis for PCA Represented Animations," in *Computer Graphics Forum*, vol. 28, no. 6. John Wiley & Sons, 2009, pp. 1529–1540.

[58] ——, "Geometry-Driven Local Neighbourhood Based Predictors for Dynamic Mesh Compression," in *Computer Graphics Forum*. John Wiley & Sons.

[59] J. Yang, C. Kim, and S. Lee, “Compression of 3-D triangle mesh sequences based on vertex-wise motion vector prediction,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 12, no. 12, pp. 1178–1184, 2002.

[60] L. Ibarria and J. Rossignac, “Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity,” *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 126–135, 2003.

[61] R. Amjoun and W. Straßer, “Predictive-DCT Coding for 3D Mesh Sequences Compression,” *Journal of Virtual Reality and Broadcasting*, vol. 5, no. 6, 2008.

[62] ——, “Single-rate near lossless compression of animated geometry,” *Computer-Aided Design*, vol. 41, no. 10, pp. 711–718, 2009.

[63] N. Stefanoski and J. Ostermann, “Connectivity-Guided Predictive Compression of Dynamic 3D Meshes,” *Image Processing, 2006 IEEE International Conference on*, pp. 2973–2976, 2006.

[64] N. Stefanoski, X. Liu, P. Klie, and J. Ostermann, “Scalable linear predictive coding of time-consistent 3D mesh sequences,” *3DTV-Conference The True Vision - Capture, Transmission and Display of 3D Video*, pp. 1–4, 2007.

[65] N. Stefanoski and J. Ostermann, “Spatially and temporally scalable compression of animated 3D meshes with MPEG-4/FAMC,” *Proceedings of the IEEE International Conference on Image Processing*, pp. 2696–2699, 2008.

[66] ——, “SPC: Fast and efficient scalable predictive coding of animated meshes,” *Computer Graphics Forum*, vol. 29, pp. 101–116(16), March 2010.

[67] D. Marpe, H. Schwarz, and T. Wiegand, “Context-based adaptive binary arithmetic coding in the H. 264/AVC video compression standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, 2003.

[68] K. Mamou, T. Zaharia, and F. Preteux, “Famc: The mpeg-4 standard for animated mesh compression,” *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pp. 2676–2679, Oct. 2008.

[69] H. Kirchhoffer, D. Marpe, K. Muller, and T. Wiegand, “Context-adaptive binary arithmetic coding for frame-based animated mesh compression,” in *2008 IEEE International Conference on Multimedia and Expo*, 2008, pp. 341–344.

[70] P. Cignoni, C. Rocchini, and R. Scopigno, “Metro: Measuring error on simplified surfaces,” *Computer Graphics Forum*, vol. 17, pp. 167–174, 1998.

[71] C. L. Bajaj, S. Cutchin, V. Pascucci, and G. Zhuang, “Error resilient transmission of compressed vrml,” *TICAM, The Univ. Texas at Austin, Austin, TX, Tech. Rep., 1998*.

[72] Z. Yan, S. Kumar, and C.-C. J. Kuo, “Error resilient coding of 3-D graphic models via adaptive mesh segmentation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 860–873, Jul. 2001.

[73] N.-H. Lin, T.-H. Huang, and B.-Y. Chen, “3D model streaming based on jpeg 2000,” *IEEE Transactions on Consumer Electronics*, vol. 53, no. 1, 2007.

[74] "JPEG 2000 image coding system – Part 3: Motion JPEG 2000, ISO/IEC 15444-3:2002," 2004.

[75] H. Li, M. Li, and B. Prabhakaran, "Middleware for streaming 3D progressive meshes over lossy networks," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 2, no. 4, pp. 282–317, 2006.

[76] P. Jaromersky, X. Wu, Y. Chiang, and N. Memon, "Multiple-description geometry compression for networked interactive 3D graphics," in *Proc. ICIG'2004*, Dec. 2004, pp. 468–471.

[77] G. AlRegib, Y. Altunbasak, and J. Rossignac, "An unequal error protection method for progressively transmitted 3-D models," *IEEE Trans. Multimedia*, vol. 7, pp. 766–776, Aug. 2005.

[78] G. AlRegib, Y. Altunbasak, and R. M. Mersereau, "Bit allocation for joint source and channel coding of progressively compressed 3-D models," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 2, pp. 256–268, Feb. 2005.

[79] G. AlRegib, Y. Altunbasak, and J. Rossignac, "Error-resilient transmission of 3-D models," *ACM Trans. on Graphics*, vol. 24(2), pp. 182–208, Apr. 2005.

[80] S. Ahmad and R. Hamzaoui, "Optimal error protection of progressively compressed 3D meshes," in *Proceedings IEEE International Conference on Multimedia and Expo (ICME'06)*, Jul 2006, pp. 201–204.

[81] V. Vaishampayan, "Design of multiple description scalar quantizers," *IEEE Trans. Inform. Theory*, vol. 39, no. 3, pp. 821–834, May 1993.

[82] C. Tian and S. Hemami, "Universal multiple description scalar quantization: analysis and design," *Information Theory, IEEE Transactions on*, vol. 50, no. 9, pp. 2089–2102, 2004.

[83] A. Bogomjakov and C. Gotsman, "Universal rendering sequences for transparent vertex caching of progressive meshes," *Computer Graphics Forum*, vol. 21, pp. 137–148, 2002.

[84] Z. Karni, A. Bogomjakov, and C. Gotsman, "Efficient compression and rendering of multi-resolution meshes," in *Proc. IEEE Visualization*, Boston, US, Oct. 2002.

[85] E. Riskin, "Optimum bit allocation via generalized BFOS algorithm," *IEEE Trans. Inform. Theory*, vol. 37, pp. 400–4002, March 1991.

[86] Y. Charfi, R. Hamzaoui, and D. Saupe, "Model-based real-time progressive transmission of images over noisy channel," in *Proc. WCNC'03*, New Orleans, LA, Mar. 2003, pp. 347–354.

[87] A. E. Mohr, E. A. Riskin, and R. E. Ladner, "Graceful degradation over packet erasure channels through forward error correction," *Proceedings of the 1999 Data Compression Conference (DCC)*, 1999.

[88] A. Mohr, E. Riskin, and R. Ladner, "Approximately optimal assignment for unequal loss protection," in *Proc. ICIP'00*, 2000, pp. 367–370.

[89] R. Puri and K. Ramchandran, "Multiple description source coding using forward error correctioncodes," in *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers*, vol. 1, Oct 1999.

[90] S. Dumitrescu, X. Wu, and Z. Wang, "Globally optimal uneven error-protected packetization of scalable code streams," in *DCC '02: Proceedings of the Conference on Data Compression*, Apr. 2002.

[91] V. Stankovic, R. Hamzaoui, and Z. Xiong, "Packet loss protection of embedded data with fast local search," *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 2, 2002.

[92] S. Dumitrescu and X. Wu, "Globally optimal uneven erasure-protected multi-group packetization of scalable codes," in *Proceedings IEEE International Conference on Multimedia and Expo (ICME'05)*, Jul 2005, pp. 900–903.

[93] B. Girod, K. Stuhlmüller, M. Link, and U. Horn, "Packet loss resilient internet video streaming," *Proceedings of SPIE Visual Communications and Image Processing '99*, pp. 833–844, 1999.

[94] U. Horn, K. Stuhlmüller, M. Link, and B. Girod, "Robust internet video transmission based on scalable coding and unequal error protection," *Image Com.*, vol. 15, no. 1–29, pp. 77–94, Sep 1999.

[95] S. Varakliotis, S. Hailes, and J. Ostermann, "Optimally smooth error resilient streaming of 3D wireframe animations," *Proceedings of SPIE*, vol. 5150, p. 1009, 2004.

[96] A. Reibman, H. Jafarkhani, Y. Wang, M. Orchard, and R. Puri, "Multiple-description video coding using motion-compensated temporal prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 3, pp. 193–204, 2002.

[97] T. Tillo, M. Grangetto, G. Olmo, *et al.*, "Redundant slice optimal allocation for H. 264 multiple description coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 1, pp. 59–70, 2008.

[98] C. Zhu and M. Liu, "Multiple description video coding based on hierarchical B pictures," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 4, pp. 511–521, 2009.

[99] B. Demir, S. Erturk, and O. Urhan, "Improved quality multiple description 3D mesh coding with optimal filtering," in *2009 16th IEEE International Conference on Image Processing (ICIP)*, 2009, pp. 3541–3544.

[100] O. Sorkine and D. Cohen-Or, "Least-squares meshes," in *Proceedings of Shape Modeling International*. IEEE Computer Society Press, 2004, pp. 191–199.

[101] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[102] G. Forney Jr, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

CURRICULUM VITAE

Mehmet Oğuz Bici was born in Ankara, Turkey, in 1983. He received the B.S. degree in Electrical and Electronics Engineering, from Middle East Technical University, Ankara, Turkey in 2005. Between 2005 and 2010, he worked as a researcher in Multimedia Research Group of Electrical and Electronics Engineering Department of Middle East Technical University within 3DTV project, Network of Excellence funded by the European Commission 6th Framework Programme and MOBILE3DTV project, funded by the European Commission 7th Framework Programme. He is currently working as a researcher in Tampere University of Technology, Department of Signal Processing. His research interests include coding and transmission of image, video and 3D meshes. His publications are as follows:

Book Chapters

1. A. Norkin, M. O. Bici, A. Aksay, C. Bilen, A. Gotchev, G. Bozdagi Akar, K. Egiazarian, and J.Astola, ”Multiple description coding and its relevance to 3DTV” in Haldun M. Ozaktas and Levent Onural (Eds.), ”Three-Dimensional Television: Capture, Transmission, and Display.”, Springer, Heidelberg, 2007. In press

Journals

1. M. O. Bici, G. B. Akar, ”Multiple description coding of animated meshes,” Signal Processing: Image Communication, In Press, Corrected Proof, Available online 26 October 2010, ISSN 0923-5965, DOI: 10.1016/j.image.2010.10.004.
2. M. O. Bici, A. Norkin, G. B. Akar, ”Joint Source-Channel Coding for Error Resilient Transmission of Static 3D Models,” Accepted for publication in Turkish Journal of Electrical Engineering and Computer Sciences
3. M. O. Bici and G. B. Akar, “Improved Prediction Methods For Scalable Predictive Animated Mesh Compression,” *Submitted to Journal of Visual Communication and Image Representation*

4. M. O. Bici and G. B. Akar, “Optimal Quality Scalability for Predictively Coded Animated Meshes,” *Submitted to IEEE Transactions on Multimedia*.

International Conference Papers

1. M. O. Bici and G. B. Akar, ”Improved Prediction For Layered Predictive Animated Mesh Compression,” International Conference on Image Processing, ICIP 2010, Hong Kong, Sept 2010.

2. M. O. Bici, D. Bugdayci, G. B. Akar, A. Gotchev, ”Mobile 3D Video Broadcast,” International Conference on Image Processing, ICIP 2010, Hong Kong, Sept 2010.

3. M. O. Bici and G. B. Akar, ”Multiple description coding of 3D dynamic meshes based on temporal subsampling,” in Visual Information Processing and Communication, Proc. SPIE 7543, 75430C (2010), DOI:10.1117/12.839850, San-Jose, US, Jan. 2010

4. A. Aksay, M. O. Bici, D. Bugdayci, A. Tikanmaki, A. Gotchev, G. B. Akar, ”A Study on the Effect of MPE-FEC for 3D Video Broadcasting over DVB-H,” Mobimedia 2009, London, UK, Sept’09.

5. M. O. Bici, N. Stefanoski, G. Bozdagi Akar, ”Vertex Partitioning Based Multiple Description Coding of 3D Dynamic Meshes,” in 3DTV Conference 2009, Potsdam, Germany, 2009

6. M. O. Bici, A. Aksay, A. Tikanmaki, A. Gotchev, G. B. Akar, ”Stereo Video Broadcasting Simulation for DVB-H,” NEM-Summit’08, St Malo, France, Oct. 2008.

7. M. O. Bici, G. Bozdagi Akar, ”Distributed 3D Dynamic Mesh Coding,” in Proc. IEEE Int. Conf. Image Processing, San Diego, US, Oct. 2008

8. M. O. Bici, A. Norkin, G. Bozdagi Akar, and A. Gotchev, ”Optimized multiple description scalar quantization based 3D mesh coding,” in Workshop on Information Theoretic Methods in Science and Engineering (WITMSE) 2008, Tampere, Finland, 2008

9. M. O. Bici, A. Norkin, G. Bozdagi Akar, ”Packet Loss Resilient Transmission of 3D Models,” in Proc. IEEE Int. Conf. Image Processing, San Antonio, US, Sep. 2007

10. M. O. Bici, A. Norkin, G. Bozdagi Akar, A. Gotchev, and J. Astola, ”Multiple Description Coding of 3D Geometry With Forward Error Correction Codes,” in 3DTV Conference 2007, Kos Island, Greece, 2007

11. A. Norkin, M. O. Bici, G. Bozdagi Akar, A. Gotchev, and J. Astola, ”Wavelet-based multiple description coding of 3-D geometry,” in Visual Communications and Image Processing 2007. Proceedings of the SPIE, Volume 6508, pp. 65082I, San-Jose, US, Jan. 2007.

12. M. O. Bici and G. Bozdagi Akar, ”Multiple description scalar quantization based 3D mesh coding,” in Proc. IEEE Int. Conf. Image Processing, Atlanta, US, Oct. 2006.

13. A. Aksay, M. O. Bici, G. Bozdagi Akar, ”Evaluation of Disparity Map Characteristics For Stereo Image Coding,” IEEE International Conference on Image Processing, ICIP-2005, Italy, Sept. 2005.

National Conference Papers

1. D. Bugdayci, M. P. Bici, A. Aksay, M. Demirtas, G. B. Akar, ”Video + Depth based 3D Video Broadcast over DVB-H,” IEEE SIU 2010 (18th National Signal Processing and Applications Conference), Diyarbakir, Turkey, Apr’10. (”DVB-H Üzerinden Video+Derinlik Tabanli 3B Video Yayini”)

2. M. O. Bici, A. Aksay, A. Tikanmaki, A. Gotchev, G. B. Akar, ”Stereo Video Broadcast Over DVB-H,” IEEE SIU 2009 (17th National Signal Processing and Applications Conference), Antalya, Turkey. (”DVB-H Üzerinden Stereo Video Yayini”)

3. C. V. Isik, M. O. Bici, A. Aksay, G. Bozdagi Akar, ”Importance of Side Information Generation for Distributed Video Coding”, IEEE SIU 2008 (16th National Signal Processing and Applications Conference), Didim, Turkey. (”Dagitik Video Kodlama için Yan Bilginin Önemi”)

4. M. Oguz Bici, Andrey Norkin, G. Bozdagi Akar, ”Multiple Description Coding of 3D Meshes Based on Forward Error Correction,” IEEE SIU 2007 (15th National Signal Processing and Applications Conference), Eskisehir, Turkey. (”3B Aglarin Gönderme Yönünde Hata Düzeltimine Dayali Çoklu Anlatim Kodlanmasi”)

5. M. Oguz Bici, G. Bozdagi Akar, ”3D Mesh Coding Using Multiple Description Scalar Quantizers,” IEEE SIU 2006 (14th National Signal Processing and Applications Conference), Antalya, Turkey. (”Çoklu Anlatim Skaler Nicemleyiciler ile 3B Ag Kodlama”)

6. A. Aksay, M. Oguz Bici, G. Bozdagi Akar, ”Using Disparity Map Compensation For Stereo Image Coding,” IEEE SIU 2005 (13th National Signal Processing and Applications Conference), Kayseri, Turkey. (”Stereo Imge Sikistirma Için Aykirilik Haritasi Dengelemesi Kullanimi”)