

Streaming Three-Dimensional Graphics with Optimized Transmission and Rendering Scalability

A Thesis
Presented to
The Academic Faculty

by

Dihong Tian

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Electrical and Computer Engineering
Georgia Institute of Technology
December 2006

Streaming Three-Dimensional Graphics with Optimized Transmission and Rendering Scalability

Approved by:

Prof. Ghassan AlRegib, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Prof. Yucel Altunbasak
School of Electrical and Computer Engineering
Georgia Institute of Technology

Prof. Russell M. Mersereau
School of Electrical and Computer Engineering
Georgia Institute of Technology

Prof. Anthony J. Yezzi
School of Electrical and Computer Engineering
Georgia Institute of Technology

Prof. Ming Yuan
School of Industrial and Systems Engineering
Georgia Institute of Technology

Date Approved: 10 November 2006

*To my late father, Wujia Tian,
and my mother, Mingxuan Zhang,
who taught me everything that matters in life....*

ACKNOWLEDGEMENTS

At the time of drawing a period for my doctoral endeavor, I would like to express my sincere appreciation and gratitude to my advisor, Prof. Ghassan AlRegib, for his persistent support and invaluable advice during my Ph.D. study. Without his guidance, I would not be able to successfully complete the dissertation. It has been a great pleasure to have him as my advisor. I would also like to express my gratitude to Prof. Yucel Altunbasak, Prof. Russell Mersereau, Prof. Anthony Yezzi, and Prof. Ming Yuan for their input as part of my supervisory committee. Special thanks go to Prof. Yucel Altunbasak for offering me the opportunity to start my Ph.D. at Georgia Tech.

I have many friends and colleagues to thank for their great support. First, I wish to show my deep gratitude to my closest friends Zhifeng Liu, Ning Chen, and Qing Zhao. I truly enjoyed every conversation I had with them about Ph.D. studies, research, future careers, and all other aspects of life. Second, I am grateful to my current and earlier colleagues Junlin Li, Xiaohuan Li, and Dr. Yen-Chi Lee, whom I am pleased to have collaborated with. I am also grateful to many other friends and colleagues: Yuan Li, Chunming Zhao, Qiang Le, Volkan Cevher, Fred Stakem, Nejat Kamaci, Ali C. Begen, Toygar Akgun, Tarik Arici, Mehmet U. Demircin, and Sibel Yaman, to name a few.

My years at the Center for Signal and Image Processing (CSIP) will remain a pleasant memory for its excellent working environments and friendly atmosphere. I want to express my appreciation and respect to our CSIP faculty, who created and are continually improving this unique, nurturing environment at Georgia Tech. I am also indebted to the CSIP staff for their administrative support to my academic pursuit.

Finally, I would like to express the deepest gratitude to my parents and my wife, Shu Li, for their absolute, unqualified love, which kept me going when all else failed.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
LIST OF TABLES	viii
LIST OF FIGURES	ix
SUMMARY	xiii
I MOTIVATION AND OVERVIEW	1
II BACKGROUND	5
2.1 Mesh Compression	6
2.2 Measuring Model Fidelity	9
2.3 Joint Mesh and Texture Optimization	11
2.4 Error-Resilient Streaming	12
III BIT ALLOCATION FOR TEXTURED 3D MODELS	15
3.1 Introduction	15
3.1.1 Progressive Mesh and Texture	16
3.1.2 Fidelity Measure and Bit Allocation	17
3.2 Proposed Bit-Allocation Framework	18
3.2.1 The Fast Quality Measure	21
3.2.2 The Boundary Search Algorithm	26
3.2.3 The Retained Mode	27
3.3 Experimental Results	29
3.3.1 The Equalization Factor	30
3.3.2 The Streamed Mode	31
3.3.3 The Retained Mode	37
3.4 Summary	42
IV HYBRID RETRANSMISSION AND ERROR PROTECTION	44
4.1 Introduction	44
4.2 System Overview	45

4.2.1	The Codec Component	46
4.2.2	The Transmission Component	47
4.2.3	The Transport Layer	48
4.2.4	Simulation Environment	49
4.3	Proposed Transmission Mechanism	51
4.3.1	Distortion-Constrained Transmission	52
4.3.2	Delay-Constrained Transmission	62
4.4	Summary	67
V	MULTI-OBJECT SOURCE AND CHANNEL CODING	69
5.1	Introduction	69
5.2	System Overview	70
5.2.1	Object Weighting	72
5.2.2	LOD Hierarchies	72
5.3	Vector Quantization	74
5.3.1	Weights of Training Vectors	75
5.3.2	Codebook Training Algorithm	75
5.4	Embedded Parity Objects	77
5.4.1	Partially Ordered Packetization	77
5.4.2	Joint Unequal Error Protection	78
5.5	Joint Source and Channel Coding	80
5.5.1	Parity-Object Generation	82
5.5.2	Rate Allocation	83
5.6	Experimental Results	84
5.7	Summary	92
VI	MULTI-STREAMING WITH SCALABLE PARTIAL RELIABILITY 93	
6.1	Introduction	93
6.1.1	The Stream Control Transmission Protocol	94
6.2	Multi-Streaming Framework	95
6.3	Measuring Scene Quality	97
6.4	Transmission Mechanism	102

6.4.1	Rate-Distortion Optimization	103
6.4.2	Fast Heuristics	105
6.5	Performance Evaluation	108
6.5.1	On the Receiving Quality	111
6.5.2	On the Rendering Cost	115
6.5.3	Multi-Streaming versus Single-Streaming	117
6.6	Summary	119
VII	CONCLUSIONS	120
7.1	Summarized Contributions	120
7.1.1	Joint Mesh and Texture Optimization	120
7.1.2	Latency-Minimized Delivery	121
7.1.3	Streaming Content-Rich 3D Scenes	121
7.2	Future Research Directions	123
7.2.1	Joint Application-Transport Scheduling for Virtual Navigation . . .	123
7.2.2	Scalable Coding and Transmission for Animated Geometry	124
APPENDIX A	— SUPPLEMENTARY FOR CHAPTER III	125
APPENDIX B	— SUPPLEMENTARY FOR CHAPTER V	126
APPENDIX C	— SUPPLEMENTARY FOR CHAPTER VI	133
REFERENCES	134
VITA	141

LIST OF TABLES

Table 1	Textured 3D models used in the experiments	30
Table 2	Comparisons of the boundary search and the marginal analysis algorithms under various bit rates	37
Table 3	A summary of the simulation parameters	50
Table 4	Comparative delay results (in seconds) for the selected batches of the VENUS HEAD model	61
Table 5	The test 3D database	85

LIST OF FIGURES

Figure 1	A sample of the triangular mesh and the texture mapping.	5
Figure 2	Illustrations of (a) the half-edge collapse and vertex split operations, the texture deviation metric, and (b) the multi-resolution meshes.	8
Figure 3	Illustrations of the calculation of SSE: (a) An example of 12 camera views surrounding a 3D object, where the viewpoints and view orientations are uniformly distributed; (b) The 24 vertices of the small rhombicuboctahedron are used as the viewpoints in the evaluation of SSE. These pictures are courtesy of Linstrom [63].	11
Figure 4	An illustration of the considered system.	16
Figure 5	The bit structure of an encoded vertex-split operation in the multi-resolution hierarchy. The to-be-split vertex has e incident edges; (x, y, z) and (s, t) denote geometry and texture coordinates, respectively.	17
Figure 6	An illustration of the bit-allocation framework. The streamed mode and the retained mode deal with different data sets: $(M^i, T^j)_{[0,0] \leq (i,j) \leq [n,m]}$ are generated levels-of-details (LODs), while (χ_G, χ_T) denote the enhancement data that transfers a certain LOD to the full resolution.	19
Figure 7	An example where SSE fails to capture the quality difference of simplified textured models. For comparison, quality measures given by the fast quality measure (FQM) (to be detailed in Section 3.2.1) are also presented.	21
Figure 8	With normalized scales, the mean-square texture deviation (MSD) and the mean-square surface distance measured by Metro [27] (MetroMSE) generate close distortion-rate curves.	23
Figure 9	An illustration on estimating the equalization factor λ . (M^k, M^{k-1}) and (T^l, T^{l-1}) are selected <i>turning</i> points on the $(\mathcal{R}_G, \mathcal{Q}_G)$ and $(\mathcal{R}_T, \mathcal{Q}_T)$ curves, respectively, which construct lines that have a 45° -angle with the x -axis. The selected pairs are rendered in the screen space for estimating the equalization factor λ	25
Figure 10	The texture images of the MAP-SPHERE and the MARBLE-BALL models.	30
Figure 11	Rendered results of the MAP-SPHERE ($\lambda = 0.5269$) and the Marble-Ball ($\lambda = 0.9127$) models. (a-d) and (e-h) have the same four pairs of mesh and texture resolutions, but are presented respectively in the order of descending FQM measures.	32
Figure 12	The rate-quality curve and surface of the MAP-SPHERE model. The dashed (red) curve in (a) plots the convex hull of the rate-quality curve.	33
Figure 13	The optimal pair of resolutions found for the MANDRILL model: (a) is the rate-quality curve; (b-e) have decreasing fidelity and are correspondingly marked in (a) under the bit budget 25 KB.	35

Figure 14	The optimal pair of resolutions found for the ZEBRA model: (a) is the rate-quality curve; (b-e) correspond to the marked points along the envelope in (a), with the bit budget varying from 40KB to 8KB.	36
Figure 15	A comparison of the boundary search and the marginal analysis algorithms: (a) the execution paths for the ZEBRA model under a bit budget 15 KB, (b-c) the corresponding subjective results, and (d-e) the subjective results of MANDRILL under a 20 KB rate budget.	38
Figure 16	A comparison of the MxQ-BA algorithm with the two simplest heuristics that transmit all the mesh (texture) data first and then the other.	40
Figure 17	A comparison of the MxQ-BA and λ -BA algorithms with the constant-ratio bit allocation.	41
Figure 18	Sample results of the SALAMANDER model in the retaining process when five data units are decoded.	42
Figure 19	The proposed transmission system consists of three components.	46
Figure 20	The simulation topology in ns-2. There are totally f parallel flows sharing the bottleneck bandwidth. All the background traffic is FTP traffic transmitted over TCP.	50
Figure 21	The distortion-rate performance of four test models.	51
Figure 22	A simple demonstration on the proposed hybrid UEP/SR method, where the packets marked with ‘RS’ are parity-check packets and ‘*’ indicates packet losses during transmission. Batch 2 is retransmitted because it is among the selected χ batches and has not been correctly received.	53
Figure 23	An illustration of the steepest decent algorithm for finding $\mathbf{c}_{\chi opt}$. The double-circled nodes indicate the searching path of the steepest decent algorithm, and the solid one represents the optimal operating point which has the minimum expected transmission delay.	55
Figure 24	The computed minimum expected delay versus the number of parity-check packets that are added. The computation is performed for the HORSE model with a setting of these parameters: packet size $s = 500$ bytes, packet-loss rate $p = 2\%$, and round-trip time $r = 100$ ms.	56
Figure 25	The number of parity-check packets versus the packet-loss rate. The packet size and the round-trip time in the computation are set to be $s = 500$ bytes and $r = 100$ ms, respectively.	57
Figure 26	Average transmission delays for sending various portions of data reliably using REP and 3TP, respectively. The number of parallel flows in the network is $f = 12$, and the packet sizes are $s = 1000$ bytes in (a-b) and $s = 500$ bytes in (c-d).	58
Figure 27	Delay results when the distortion constraint D_{max} is 36 dB. The packet sizes are $s = 500$ bytes in (a-b) and $s = 1000$ bytes in (c-d).	60

Figure 28	Average decoding distortion of received meshes for various delay constraint in a typical network situation: the number of parallel flows is $f = 12$, the packet size is $s = 500$ bytes, and the resulting packet-loss rate is $p = 5.5\%$.	64
Figure 29	Traces of the decoding distortion for each received mesh: (a-b) $\tau_{max} = 2$ sec; (c-d) $\tau_{max} = 7$ sec.	65
Figure 30	Rendering quality near to 40 dB is obtained by REP when $\tau_{max} = 7$ to 10 seconds, with a 10 dB or higher gain compared to the simple heuristic (FA) under the same situation.	66
Figure 31	A diagram of the presented joint source and channel coding system.	71
Figure 32	A diagram of the LOD-hierarchy construction/reconstruction process and the VQ-based multi-resolution coding for one mesh object.	73
Figure 33	The weighted codebook training algorithm.	76
Figure 34	An illustration of the chunk-based transmission and the use of parity objects for joint unequal error protection. Note that the transmission of multiple objects is interleaved to allow equally fast access to the objects.	78
Figure 35	Rate allocation using the parity-object generation algorithm, where $R(\cdot)$ denotes the bit rate of a particular LOD of an object measured in chunks and $\mathbf{I}(\cdot)$ gives the indices of corresponding data chunks in the entire sequence.	84
Figure 36	Distortion-rate performance of different coding schemes in an error-free environment: (a) Distortion of the database; (b) Distortion of selected objects.	88
Figure 37	Subjective comparison of decoded HORSE and VENUS HEAD models under an overall bit rate of 300 KB.	89
Figure 38	Receiving distortion of the test database for the data-chunk distribution under 300 KB: (a) Distortion vs. parity redundancy for $p = 0.05$ and $p = 0.10$; (b) Distortion vs. chunk loss rates with 5% parity redundancy.	90
Figure 39	Receiving distortion of the test database with rate allocation between source and parity objects under a chunk loss rate $p = 0.05$: (a) The overall distortion; (b) Two individual cases under $\mathcal{C} = 300$ KB and $\mathcal{C} = 500$ KB. Objects are indexed in accordance with Table 5 from left to right, top down.	91
Figure 40	The block diagram of the proposed multi-streaming framework. Notations are explained as follows: $\{v, \mathcal{S}\}$ denote the viewpoint and the 3D scene, $\{R, D, \omega\}$ denote the measured rate-distortion performance and the object weights, $\{M_k^0, B_k^i\}$ are the base meshes and the enhancement batches for the objects in the scene, and finally, $\{K_{TX}\}$ denotes the maximum number of transmissions allowed for the data batches. Note that $(K_{TX} - 1)$ is the maximum number of <i>retransmissions</i> upon data losses.	95

Figure 41	A view frustum is defined by six planes, $\pi_{VF,i} : \mathbf{n}_i \cdot x + d_i = 0, i = 0..5$, where \mathbf{n}_i is the normal and d_i is the offset of plane $\pi_{VF,i}$, and x is an arbitrary point on the plane. If x is outside $\pi_{VF,i}$, then $\mathbf{n}_i \cdot x + d_i > 0$ and vice versa.	99
Figure 42	A 2D illustration on defining the grid weight by the distance from the grid to the viewpoint.	101
Figure 43	An illustration on the projection-map based object weighting.	102
Figure 44	A simple example of the steepest decent search for a single object \mathcal{O}_k with $L_k = 4$	106
Figure 45	A sample 3D scene with a demonstrated view space. Objects outside the view space will not be displayed on the user's terminal while other objects may be partially visible due to occlusion.	109
Figure 46	The objects within the view frustum and their estimated weights. These objects are selected to be transmitted in PROpt and the two comparing heuristical methods.	109
Figure 47	Comparisons of the receiving quality with respect to bit-rate constraints on the rendering application.	111
Figure 48	Comparisons of the rendered results between PROpt and the Equal-Weight heuristic with the same bit rate (300 KB). Flat shading is used to enhance the facet effect of the displayed models.	113
Figure 49	Comparisons of the receiving quality with constraints on the transport: (a) Quality versus delay requirements with a fixed packet loss rate $p = 0.05$; (b) Quality versus packet loss rates with a fixed delay requirement $\tau = 20$ sec.	114
Figure 50	A performance evaluation on the processing load at the client: (a) Number of polygons selected for given transport latency; (b) Number of polygons versus the level of quality.	116
Figure 51	Comparisons of receiving quality between multi-streaming and single-streaming under a delay requirement $\tau = 10$ sec: (a) quality versus packet loss rate with a fixed buffer size $\chi = 20$ KB; (b) quality versus size of the receiving buffer with a fixed packet loss rate $p = 0.02$	118
Figure 52	Equal error protection for four data streams using parity objects (streams) and separate FEC, comparatively.	127
Figure 53	Numerical comparisons of decoded error probabilities between in-stream FEC and parity streams in the burst-loss case.	132

SUMMARY

Distributed three-dimensional (3D) graphics applications exhibit both resemblance and uniqueness in comparison with conventional streaming media applications. The resemblance relates to the large data volume and the bandwidth-limited and error-prone transmission channel. The uniqueness is due to the polygon-based representation of 3D geometric meshes and their accompanying attributes such as textures. This specific data format introduces sophisticated rendering computation to display graphics models and therefore places an additional constraint on the streaming application.

The objective of this research is to provide scalable, error-resilient, and time-efficient solutions for high-quality 3D graphics applications in distributed and resource-constrained environments. Resource constraints range from rate-limited and error-prone channels to insufficient data-reception, computing, and display capabilities of client devices. Optimal resource treatment with transmission and rendering scalability is important under such circumstances. The proposed research consists of three milestones. In the first milestone, we develop a joint mesh and texture optimization framework for scalable transmission and rendering of textured 3D models. Then, we address network behaviors and develop a hybrid retransmission and error protection mechanism for the on-demand delivery of 3D models. Next, we advance from individual 3D models to 3D scene databases, which contain numerous objects interacting in one geometric space, and study joint application and transport approaches. By properly addressing the properties of 3D scenes represented in multi-resolution hierarchies, we develop a joint source and channel coding method and a multi-streaming framework for streaming the content-rich 3D scene databases toward optimized transmission and rendering scalability under resource constraints.

CHAPTER I

MOTIVATION AND OVERVIEW

Networked multimedia applications are expanding from streaming video/audio to distributed three-dimensional (3D) graphics, driven by growing demands on various applications such as online entertainment, e-commerce, navigation systems, virtual environments, and medical and scientific visualization. Similar to image and video contents, 3D graphical objects have large volumes of data. Unlike image and video, which are naturally formatted as pixel grids, 3D objects are represented by polygonal geometry parameterized with textures or attributes such as colors and normals. This specific data format requires sophisticated *rendering* operations to be converted into pixel-based presentations. Consequently, distributed 3D graphics require considerable network bandwidth to be transmitted and computing power to be displayed on a remote terminal. Although computing technology in the graphics industry has advanced rapidly in the past years, many platforms today still lack sufficient support for rendering power and/or networking capability.

In the foreseeable future, disparate computing devices and heterogenous networks will remain a constraint for high-quality 3D applications in a distributed environment. When disparate devices communicate through a heterogenous network, different preferences on the complexity of 3D objects may be placed by different clients. In such a situation, sending objects with large data dimensions to a client may not only overload the network link, but also overload the client device when the objects are rendered. On the other hand, graphic objects in general have unequal rendering importance depending on many factors such as view perspectives, object interactions, and application semantics. For a certain device and a given quality requirement, transmitting all mesh geometry with high tessellation and all textures with high resolution to the device may be beyond the client's actual need and therefore become unnecessary. For example, one object may be occluded by another in a 3D scene, or may be outside of the view space and will be culled away in the client's rendering

pipeline. A coarse representation for the object would be sufficient while more bandwidth and computation can be spent on objects for which high levels-of-details (LODs) are desired. To involve platforms with limited or disparate networking and rendering capabilities in interactive graphics applications, transmission and rendering scalability is desired, and optimal treatment of the networking and computation resources becomes important.

The proposed research arises in such a context. It has the objective of providing scalable, error-resilient, and time-efficient solutions for enabling high-quality 3D graphics applications in distributed and resource-constrained environments. Resource constraints range from the insufficient rendering power and data-reception ability of the client device to the limited transmission rate and the error proneness of the transport link, given the timely essence of the application. With regard to the resource constraints and the properties of 3D data, the proposed research addresses the following challenges:

Scalable Coding: The state of the art in scalable coding of 3D graphics resides in the multi-resolution analysis of individual mesh objects [9,28,38,39,44,52,53,60,62,68,79,80,93]. When multiple objects or graphic components, e.g., geometry meshes and texture images, are jointly coded under a limited overall bit rate, their interactions and relative importance need to be taken into account to achieve optimal rate-distortion performance. The proposed research presents solutions for optimal bitstream organization in coding multiple graphic components or objects under constrained bit rates.

Transmission Latency: One major aspect in reducing response time¹ in distributed graphics systems is to minimize the transmission latency [3,4,10,14,22–24,43]. Accounting for the properties of 3D graphic objects and the essences of 3D applications, e.g., the view dependency and the object interaction, the proposed research develops approaches for the latency-minimized delivery of 3D models or apparatuses in bandwidth-limited networking scenarios.

Error Resilience: Network links and communications channels exhibit lossy and noisy features in addition to the limited transmission rate. Despite a few pioneering studies [2,5–7,

¹Response time is defined as the latency between the user input and the response from the system, which can be typically considered as the scene displayed on the user’s terminal.

11, 14, 16, 18, 21, 50, 94, 97–99], an efficient means for providing error resilience to 3D objects has not been thoroughly addressed. The proposed research investigates the outstanding issues and presents solutions toward more efficient and channel-adaptive error resilience for streaming 3D graphics.

Rendering Cost: Platforms such as handheld devices place an extra constraint on the 3D application because of their limited computing power and display ability. Under such constraints, coding and transmission of the 3D database should be designed to minimize the amount of graphic primitives that need to be processed by the client device’s rendering pipeline, with the least loss of application quality. The proposed research also addresses this aspect.

The aforementioned challenges, although presented separately, often require joint considerations because of the concurrent application and transport constraints. For example, sending 3D data over an error-prone bottleneck link will promote the joint consideration of error resilience and latency-minimized delivery in fulfilling a certain level of application quality. Methods proposed in this research will cover both coding and transmission perspectives, with the ultimate goal of providing high-quality visualization for on-demand 3D applications in the resource-constrained environment.

Specifically, this dissertation is organized as follows:

Chapter 2 presents the background of scalable coding and error-resilient streaming of 3D graphics. A brief review of the relevant work in the literature is provided, and the outstanding challenges are stated.

Chapter 3 and Chapter 4 focus on individual 3D models. In particular, Chapter 3 addresses the scalable coding of textured 3D models and tackles the problem of joint mesh and texture optimization. A fast quality measure is proposed to effectively captures the visual fidelity of scalably coded textured 3D models. Based on the proposed quality measure, a bit-allocation framework is developed, which properly selects the resolutions of the mesh and the texture such that the best rendering quality is achieved under limited bit rates. Chapter 4 step further to address network behaviors and study on-demand delivery of multi-resolution 3D models over lossy networks. A latency-minimized transmission mechanism is

presented. Regarding a lossy and rate-constrained environment, the proposed mechanism exploits retransmission and unequal error protection jointly to minimize the transmission latency while guaranteeing a certain level of display quality.

Chapter 5 and Chapter 6 address the properties of 3D scenes, which comprise pluralities of objects in one geometric space with potentially unequal rendering importance. From the source and channel coding perspective, Chapter 5 presents a multi-resolution coding method for 3D scene databases using vector quantization and a forward error protection mechanism, which performs rate allocation between source and parity data and protects multiple graphic objects optimally while preserving their decoding independencies. From a transport perspective, Chapter 6 presents a multi-streaming mechanism to transmit multiple graphic objects in partially ordered transport sequences and with scalable partial reliability, under a rate-distortion optimization framework. The multi-streaming framework takes into account both the independencies and interactions of multiple graphic objects in a 3D scene to maximize the display quality, while minimizing the amount of data that needs to be rendered by the client's rendering engine.

Chapter 7 concludes the dissertation with a summary of contributions and future research directions.

CHAPTER II

BACKGROUND

Before proceeding to the proposed research, we summarize in this chapter the background of scalable coding and error-resilient streaming of 3D graphics. We briefly review the relevant work in the literature and state the outstanding challenges.

Three-dimensional (3D) graphic objects typically consist of polygonal meshes and textures, which apply 1D, 2D, or 3D bitmaps to the mesh surface to assign colors and other attributes such as normals. The parametrization process between the mesh surface and the texture is referred to as texture mapping [56, 59, 64, 74]. Texture maps add realism to 3D models and are most effective when desired surface details are impossible or expensive to achieve by solely using geometry. Figures 1(a–c) provide an example.

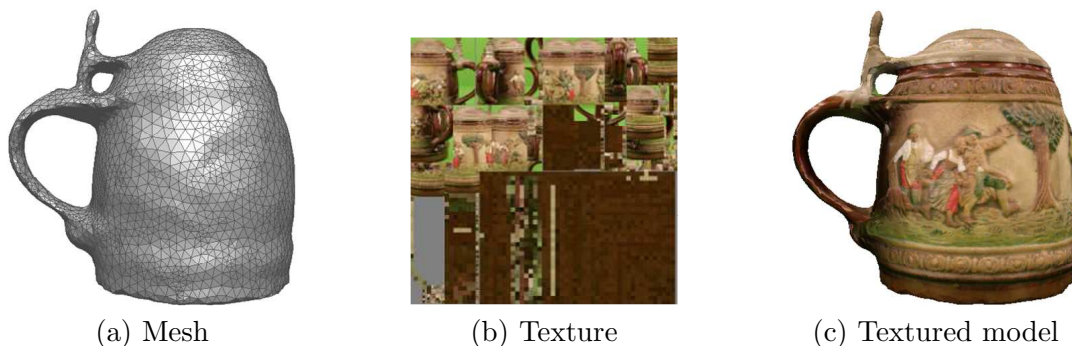


Figure 1: A sample of the triangular mesh and the texture mapping.

As shown in Figure 1(a), polygonal meshes are commonly represented by a collection of points tessellated as triangles in the 3D space. Regarding tessellation (triangulation) there are three different types, referred to as (i) *regular*: every point has degree six, (ii) *irregular*: points have arbitrary degrees, and (iii) *semi-regular*: formulated by regular subdivision starting from a coarse, irregular triangulation [41]. The 3D spatial coordinates of the points are the *geometry* data while the tessellation topology contains the *connectivity* information. In mathematical forms, a triangle mesh is denoted by a pair $(\mathcal{G}, \mathcal{C})$, where $\mathcal{G} = \{\mathbf{p}_i \in \mathbb{R}^3 | 1 \leq$

$i \leq N$ is a set of N points and \mathcal{C} is a complex containing the topological information. \mathcal{C} is a union of three sets defined on the index set of \mathcal{G} : vertices $\mathcal{V} = \{i\}$, edges $\mathcal{E} = \{i, j\}$, and faces $\mathcal{F} = \{i, j, k\}$, such that $\mathcal{C} = \mathcal{V} \cup \mathcal{E} \cup \mathcal{F}$. A vertex-based 2D texture introduces another N -point set \mathcal{T} in \mathbb{R}^2 and defines a one-to-one mapping function $f : \mathcal{U} \mapsto \mathcal{V}$ where \mathcal{U} is the index set of the texture points \mathcal{T} .

In this dissertation, a majority of research efforts are devoted to irregularly sampled meshes as they are the most essential data format in graphics applications. Textures will also be considered, for which we focus on 2D images. Generality will be properly addressed during the presentation of proposed approaches.

2.1 Mesh Compression

The two components of a textured 3D model, i.e., the geometric mesh and the texture image, differ in nature. Compression algorithms applied to these two components are therefore different. Because image compression has been widely understood for its early development and standardization [72, 75, 76, 95], this section summarizes the background of 3D mesh compression, which has a relatively short history.

During the past ten years, mesh compression has undergone a considerable amount of research. Single-resolution compression was first studied and various algorithms were proposed, among which [30, 71, 81, 91] are several selected representatives. Generally, a single-resolution mesh compression algorithm includes traversing through the mesh topology, encoding the connectivity following the traversal order, and compressing the geometry data by vertex quantization, vertex prediction, and entropy coding.

Although single-resolution compression has shown its efficacy in reducing bit rates for representing 3D meshes, it produces single-level bitstreams that are not scalable to network bandwidth or rendering capability, meaning that the represented object will not be displayed on the client’s screen until the entire bitstream is fully downloaded, decoded, and rendered. To provide scalable bitstreams for 3D data, multi-resolution compression techniques were investigated [9, 28, 38, 39, 44, 52, 53, 60, 68, 79, 80, 93]. Using multi-resolution encoders, the server can select the appropriate resolution for a particular client according to the quality

requirement or initially send a coarse representation of the 3D model to the client for quick reconstruction and rendering, and then transmit upon necessity refinement layers, which gradually increase model fidelity toward higher resolutions.

Multi-resolution mesh coding methods can be categorized with the terminology introduced at the beginning of the chapter. Wavelet-based compression algorithms, e.g., [52,53], are not directly applicable to irregular meshes, because they require an irregular mesh to be converted to a semi-regular mesh by re-sampling operations [31, 56] before compression, which permanently alter the connectivity. To preserve the connectivity, multi-resolution compression for irregular meshes is generally performed by simplifying (also referred to as downsampling) the mesh using *half-edge collapse* operations, predicting the coordinates of the collapsed vertices, and coding the prediction residuals along with the cut edges that are required to recover the connectivity [44,60,68,80]. As illustrated by Figure 2(a), a half-edge collapse operation merges one vertex of an edge to the other and alters the neighborhood of the collapsed vertex as a result. Among the irregular mesh compression methods, the progressive forest split (PFS) algorithm [80] has become the core of the 3D mesh coding (3DMC) tool in MPEG-4 [48]. With several improvements, the compressed progressive mesh (CPM) algorithm [68] reported the higher compression ratio compared to its preceding schemes. A compressed mesh stream generated by CPM is composed of a base mesh, M^0 , and L enhancement batches, $\{B^i\}_{i=1,\dots,L}$, each of which encodes a set of *vertex split* operations, which perform the inverse of edge collapses, as shown in Figure 2(a). Sequentially, batch B^i refines mesh M^{i-1} to a higher level-of-detail M^i until the full resolution M^L is reached. Figure 2(b) gives a demonstration of the multi-resolution hierarchy. This hierarchical representation is the basis of our discussion when referring to multi-resolution meshes in the dissertation.

When generating a multi-resolution hierarchy, the edge collapse operations are performed successively in the order of increasing error according to a certain error metric. Such an error metric evaluates the variation of the local curvature caused by the edge collapse. The most generally used local error metric is the *quadric error* metric proposed by Garland et al. [38], which is also deployed by the CPM algorithm [68]. For 3D models with

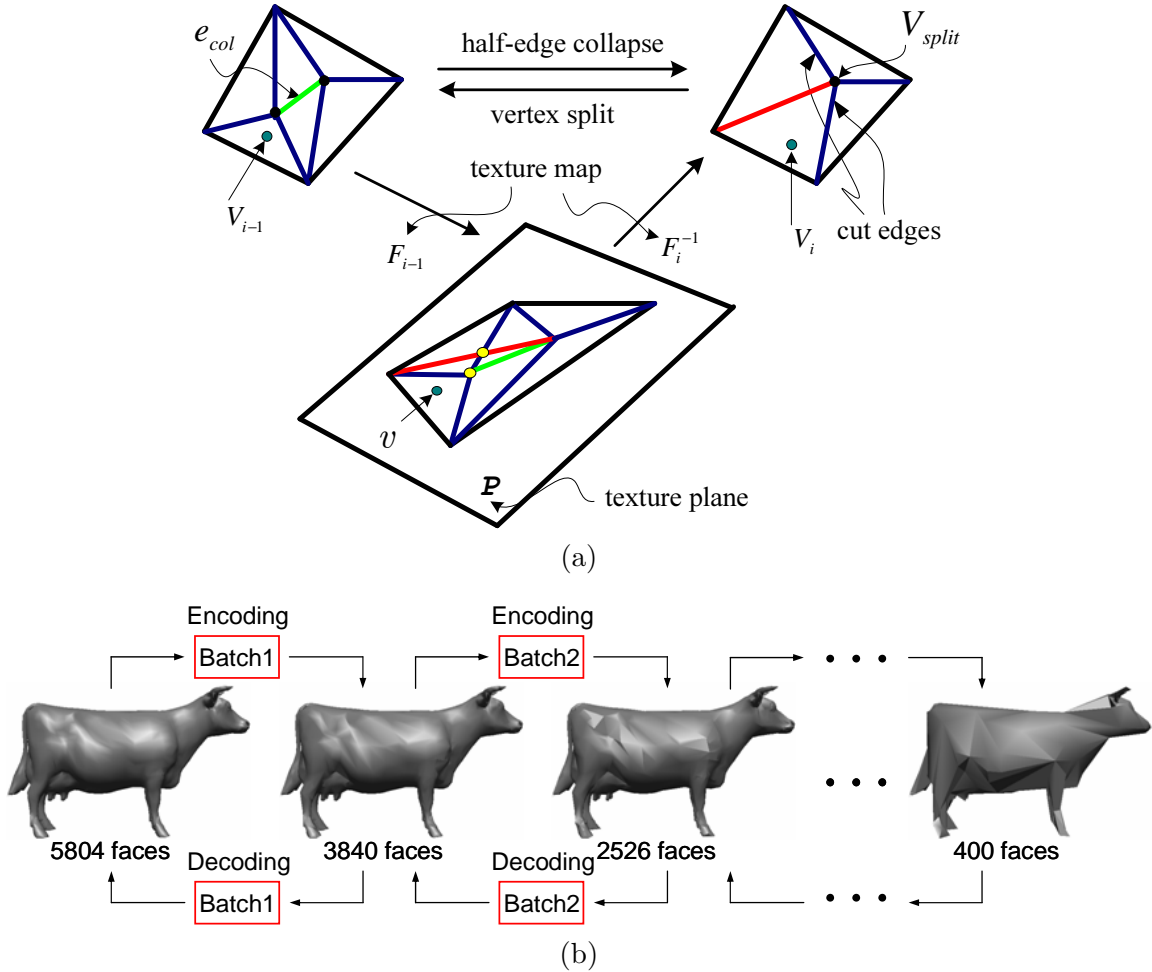


Figure 2: Illustrations of (a) the half-edge collapse and vertex split operations, the texture deviation metric, and (b) the multi-resolution meshes.

appearance attributes such as colors and texture maps, one of the challenges in simplifying the mesh surface is to preserve the appearance attributes. Garland et al. [39] proposed to extend the quadric error metric to handle vertex attributes. An improved quadric error metric was also presented by Hoppe [45]. In [28], Cohen et al. proposed the *texture deviation* as a criterion to measure the surface distortion resulting from simplifying meshes with texture maps. The texture deviation incorporates the texture domain and the geometry through the parametric correspondence. It measures the cost of an edge collapse operation as the maximum distance in the geometric space from points on the simplified mesh to their correspondents on the input surface that have the same parametric locations in the texture domain. As illustrated in Figure 2(a), V_{i-1} and V_i are 3D points on the two meshes before

and after an edge collapse, respectively, both having the same texture coordinates v ; the incremental texture deviation of this edge collapse is then defined as

$$\max_{v \in P} E_{i,i-1}(v) = \max_{v \in P} \| F_i^{-1}(v) - F_{i-1}^{-1}(v) \|, \quad (1)$$

where F_i and F_{i-1} are texture mapping functions for the two meshes.

2.2 Measuring Model Fidelity

The local error metrics such as the quadric error and the texture deviation provide fast approximations on the resulting difference of the mesh surface from an edge collapse operation, according to which all edges are sorted and are collapsed sequentially. To capture the quality difference between a simplified mesh and its full-resolution version more accurately, a statistic distortion metric is needed. Essentially, the statistic distortion metric is expected to reflect the quality degradation of the simplified model in the rendering space. In contrast to the mean squared pixel error that is widely used in 2D imageries, measuring the quality of 3D surfaces is relatively complex. For geometric meshes, a commonly adopted statistic distortion metric is the *root-mean-square* (RMS) surface distance [27]. In particular, the RMS distance for two given surfaces, S and S' , is defined as follows. First, a point-to-surface distance $e(v, S)$ is defined as

$$e(v, S) = \min_{v' \in S} d(v, v'), \quad (2)$$

where $d()$ is the Euclidean distance between two points in \mathbb{E}^3 . Then the RMS distance is given by

$$\mathcal{E}_{rms}(S', S) = \left(\frac{1}{S'} \int_{S'} e^2(v, S) ds \right)^{1/2}. \quad (3)$$

Similarly defined by the point-to-surface distance are the mean and maximum surface distances from S' to S :

$$\mathcal{E}_{mn}(S', S) = \frac{1}{S'} \int_{S'} e(v, S) ds. \quad (4)$$

$$\mathcal{E}_{max}(S', S) = \max_{v \in S'} e(v, S). \quad (5)$$

In practice, these surface distances can be calculated using the Metro tool [27], which performs fast sampling to accelerate the computation.

A study of techniques for measuring and predicting the visual fidelity of 3D meshes was conducted by Watson et al. in [96], where they examined experimental (subjective) techniques as well as several automatic (computational) techniques. Judged by human ratings and naming times¹, both the RMS and mean surface distances were verified successful predictors of visual fidelity for simplified mesh geometry.

Measuring model fidelity becomes more difficult when textures are mapped to 3D meshes, as the geometric inaccuracy of the mesh may either be masked or be highlighted after texture mapping. Conventionally, the screen-space error (SSE) was used in the literature [17, 63, 100] to measure the quality of a textured model by calculating the mean-square pixel error of the image captured from the rendering space. Because a single image cannot capture the entire appearance of a 3D object, Lindstrom et al. [63] proposed to take virtual snapshots of the model from a number of different viewpoints around the object and combine the image differences into a single error measure. In particular, given two sets of images, $\mathcal{Y} = \{Y_k\}$ and $\mathcal{Y}' = \{Y'_k\}$, $k = 1, \dots, K$, with resolution $M \times N$ pixels, which correspond to snapshots taken for two compared models respectively, the mean-square pixel error between these two sets of images is computed by

$$\sigma^2(\mathcal{Y}, \mathcal{Y}') = \frac{1}{KMN} \sum_{k=1}^K \sum_{j=1}^M \sum_{i=1}^N (y_{ijk} - y'_{ijk})^2. \quad (6)$$

The SSE evaluation is usually denoted using the peak signal-to-noise ratio (PSNR):

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{\sigma^2} \right). \quad (7)$$

To ensure an even coverage of image samples, the viewpoints need to be arranged to approximate a sphere of camera positions surrounding the object and be near-equidistant from each other. There are many possible configurations in practice, and one representative is to use the 24 vertices of the small rhombicuboctahedron as the viewpoints, as shown in Figure 3. In this dissertation, unless otherwise noted, the computation of SSE will refer to the definition in (6) and use the small rhombicuboctahedron. A resolution of 512×512 pixels will be used for images captured in the screen space.

¹Human ratings and naming times are two subjective measures widely used in the experimental sciences of visual fidelity.



Figure 3: Illustrations of the calculation of SSE: (a) An example of 12 camera views surrounding a 3D object, where the viewpoints and view orientations are uniformly distributed; (b) The 24 vertices of the small rhombicuboctahedron are used as the viewpoints in the evaluation of SSE. These pictures are courtesy of Linstrom [63].

2.3 Joint Mesh and Texture Optimization

Because of the inter-effect between the mesh and the texture, joint considerations of the two components become necessary when a textured model is coded, transmitted, and/or rendered in a rate-constrained environment [1,17,67,83,100]. Although view-dependent texture coding based on the MPEG-4 visual texture coding (VTC) tool [55] partially alleviates the resource constraint, a scalable bit-allocation framework is desired to explore the effects of both geometry and texture on the resulting model fidelity. To the best of our knowledge, the most closely related effort that addressed this challenge is that by Balmelli [17], where he studied joint mesh and texture compression for terrain models. Both the mesh and texture of the terrain model are simplified to generate various resolutions, and their proper combination is determined under screen-space error evaluation. Even though this is inspiring work, it is limited to the terrains. For a generic 3D object, computing SSE requires rendering the model from various viewpoints and averaging the image errors, which is a computationally demanding task as it requires a number of rendering operations for each pair of mesh and texture resolutions. Beside the complexity, it has been observed that under certain circumstances, the SSE presents a non-convex rate-distortion behavior and fails to reflect the quality difference of textured models properly [82]. The recent research

presented in [100] is a generalized scheme of [17], dealing with generic 3D models. In doing so, a viewing mesh is constructed with its vertices selected as sampling viewpoints for calculation of SSE, which can be considered as a sophisticated extension from using uniform polyhedrons with fixed numbers of vertices such as the small rhombicuboctahedron. Because the framework in [100] sticks to the use of SSE for distortion evaluation, it also encounters the complexity and singularity problems described above.

2.4 *Error-Resilient Streaming*

Compared to joint mesh and texture optimization, a larger research effort was invested in developing time-efficient and error-resilient streaming systems for 3D meshes over rate-constrained and/or lossy networks [2–7, 10, 11, 14, 16, 18, 21–24, 43, 50, 94, 97–99]. According to the adopted error-resilient mechanism, these systems can be categorized into pre-processing, error correction, and transport-layer protocols. In MPEG-4 [48], pre-processing-based error resilience is achieved by data partitioning [16, 97, 98], i.e., partitioning the bitstream into segments and decoding each segment independently. In [43], inter-dependent partitions are included, and the partitions are ordered in a tree structure according to their interdependencies and are transmitted accordingly.

Using forward error correction (FEC) to provide error resiliency to multi-resolution 3D meshes was first investigated by AlRegib *et al.* [2, 5–7, 11], where unequal error protection (UEP) is applied to different LODs of a multi-resolution mesh according to their distortion-rate properties. In [2, 7, 11], bit-allocation algorithms are developed to distribute source and channel coding bits under a total bit budget. Given a multi-resolution hierarchy and a bit budget, the algorithm selects the proper number of LODs to transmit and protects them with the remaining number of bits using UEP such that the expected decoding distortion is minimized. This joint source and channel coding framework was extended in [5, 6] to find an optimal tradeoff between the quantization parameter and the number of LODs when selecting the multi-resolution hierarchy to transmit.

On the transport side, the transmission control protocol (TCP) and the user datagram

protocol (UDP) are generally used for error-sensitive and delay-sensitive streams, respectively. In [3, 4, 10, 22–24], hybrid TCP/UDP transport protocols have been proposed to stream 3D models over lossy networks with reduced transmission latency, where the common idea is to send important data reliably using TCP and the remaining, less important, data using UDP. The system presented in [22–24] is based on the relative importance of different multi-resolution mesh hierarchies. The *3TP* protocol proposed in [3, 4, 10] involves another property that is specific to multi-resolution irregular meshes. In particular, this property indicates that the connectivity information is more error sensitive than the geometry data because the vertex split process is not able to continue without knowing the cut edges (Figure 2[a]). In contrast, it is not the case when any geometry data is missing as geometry contains only prediction errors of vertex positions, which do not prevent the decoding process from proceeding to the next levels, although with additional distortion.

One difficulty of using hybrid TCP and UDP is the synchronization between the two transport associations. In addition, rate control [33, 34, 49] is not considered when using UDP in the hybrid TCP/UDP protocols, which makes the application irresponsive to network congestion and unfair to other streams that share the network link. The lack of a rate control mechanism also results in large variation of the transmission throughput and therefore of the receiving quality over time. All the aforementioned techniques address the efficient transmission of 3D data separately. Yet an appropriate combination of such techniques is desired to achieve better performance. Interaction and tradeoffs among the selected techniques need to be investigated, taking into account the distortion-rate performance of the 3D data and the network characteristics. Finally, only the transmission for individual 3D objects was investigated. Three-dimensional (3D) scenes, which comprise multiple objects interacting in one geometric space, have not been considered.

The outstanding challenges stated above have been addressed in the proposed research, which will be detailed in the next chapters. In particular, the proposed research consists of three milestones. As the first milestone, a joint mesh and texture optimization framework is developed for coding textured 3D models with optimized scalability under limited bit

rates. Then, we address network behaviors and develop a hybrid retransmission and error protection mechanism for delivering 3D models over lossy networks with minimized latency. In the last milestone, we advance from individual 3D models to 3D scene databases and study joint application and transport approaches. By properly addressing the properties of 3D scenes, we develop scalable source and channel coding methods for multiple 3D objects and a multi-streaming framework for streaming 3D scenes in a partially ordered and partially reliable fashion.

CHAPTER III

BIT ALLOCATION FOR TEXTURED 3D MODELS

3.1 Introduction

The first milestone of our proposed research is joint mesh and texture optimization, which relates to the construction of an optimized scalable representation for textured 3D models. Progressive compression methods, as reviewed in the previous chapter, provide scalability for sole geometry or texture. To achieve optimal transmission and rendering scalability, there lacks an effective method that takes into account the inter-effect of the geometry and the texture on the resulting quality of the textured model and generates a multiplexed bitstream. Apparently, sending all bits for the mesh first and then bits for the texture or vice versa would not be efficient, as either a full-resolution mesh with a coarse texture or a full-resolution texture with a coarse mesh will not provide high-resolution visualization for the textured model. In a bit-rate constrained environment, such organization of the bitstream implies large transmission latency before satisfactory visualization is obtained by the client. To achieve time efficiency in the multiplexed bitstream, it is vital for the server to optimally distribute source bits between the mesh and the texture so that progressively decoding received portions of the bitstream maintains visual quality of the textured model at a maximal level.

In this chapter, a joint mesh and texture optimization framework is presented for the rate-constrained coding of textured 3D models [82, 83, 88]. To overcome the deficiency of the earlier methods, we develop a fast quality measure to estimate the quality difference of textured models with simplified meshes and resolution-reduced textures. Based on the proposed quality measure, bit-allocation algorithms are developed to find optimal bit distributions between the mesh and the texture under constrained bit rates.

The system that is under consideration is diagramed in Figure 4. For a textured 3D

model, progressive compression is first performed to generate multi-resolution representations for the mesh and the texture, respectively. Then, with a fidelity measure, the bit-allocation block organizes bits in transmitted data units so that at the receiver, the quality of the displayed model can be maximized by decoding the received bits for the mesh and the texture (separately) and rendering the resulting resolutions.

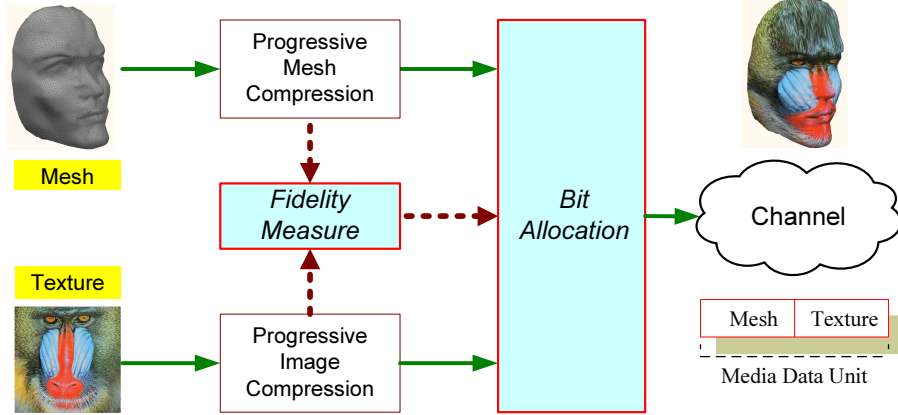


Figure 4: An illustration of the considered system.

3.1.1 Progressive Mesh and Texture

In this work, we are not proposing new compression algorithms but instead we study how to distribute the source bits between the mesh and the mapped texture in order to maximize the quality of the transmitted model. Without loss of generality, we implement the CPM algorithm proposed by Pajarola and Rossignac in [68] for the progressive compression of a mesh surface. Referring to Figure 2(a), instead of using the quadric error, we integrate the texture deviation metric [28] in CPM for the better preservation of appearance. Each vertex on the mesh is treated as a vector $V \in \mathbf{R}^5$. The first three components of V consist of spatial coordinates, and the remaining two components are texture coordinates. We then compress the collapsed vertices using vertex prediction followed by entropy coding of the prediction error. Figure 5 shows the complete binary format of an encoded vertex split operation, where five variable-length-codewords (VLC) are appended after the split status¹,

¹*split status*: a one-bit flag that specifies whether or not a vertex is to be split.

the split direction², and the cut edges³ are respectively encoded.

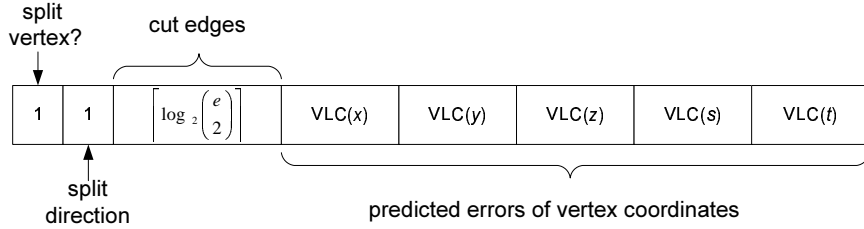


Figure 5: The bit structure of an encoded vertex-split operation in the multi-resolution hierarchy. The to-be-split vertex has e incident edges; (x, y, z) and (s, t) denote geometry and texture coordinates, respectively.

For the texture, we limit our discussion to typical 2D images although the framework can be extended to other textures such as video sequences, and apply the wavelet-based compression algorithm known as SPIHT (Set Partitioning in Hierarchical Trees) [72] to encode the texture into a progressive bitstream. It is worth to point out that these specific mesh and texture codecs are selected only for the ease of discussion. The proposed bit-allocation framework is generally applicable to any other progressive mesh and texture compression methods.

3.1.2 Fidelity Measure and Bit Allocation

Apparently, an essential aspect of the bit-allocation framework is to properly predict the visual fidelity when substituting resolution-reduced textures to simplified mesh surfaces. To quantify the distortion of an approximated model from its full-resolution version, the screen-space error (SSE) is commonly used in the literature [17, 63, 100], which has deficiencies, as reviewed in Section 2.3. In contrast to the traditional screen-space error, we propose a fast quality measure (FQM) to estimate the visual fidelity of multi-resolution textured models. In FQM, the visual fidelity of a simplified textured model is predicted by a weighted combination of errors measured in the geometric and texture domains, respectively, through an *equalization* factor. For a particular model, depending on the features of the geometry and the mapped texture, the equalization factor is estimated as a constant using error

²*split direction*: a one-bit flag that specifies the direction of the vertex split.

³*cut edges*: $\left\lceil \log_2 \binom{e}{2} \right\rceil$ bits that specify the two cut edges among the e incident edges for a vertex.

samples measured in the screen space. FQM works drastically faster than the traditional screen-space error as it greatly reduces the number of rendering operations, while it properly measures quality difference of multi-resolution textured models.

The bit-allocation algorithms are developed based on the proposed quality measure. As the first stage, a pair of mesh and texture resolutions is selected upon an initial bit budget, which provides the best approximated model under an initial bit budget to quickly start display on the user’s screen. Thereafter, enhancement data is optimally organized into (size-limited) data units, which maximally increase visual fidelity of the displayed model at each instant when a data unit is received and decoded. These two stages are referred to as two transmission modes: the *streamed* mode and the *retained* mode. We study both optimal algorithms and fast heuristics that provide optimal or near-optimal solutions with linear computation time. Our empirical analysis shows that not only the bit-allocation algorithm maximizes the receiving quality of the textured 3D model but also it avoids sending additional information to indicate bit boundaries between the mesh and the texture in the multiplexed bitstream, which makes the streaming application more robust to bit errors that may occur randomly during transmission.

The rest of the chapter is organized as follows. Section 3.2 provides details of the proposed framework, the fast quality measure, and the bit-allocation algorithms. In Section 3.3, we empirically investigate the effectiveness and efficiency of the algorithms in different transmission modes. We present both objective and subjective results. Section 3.4 concludes the chapter.

3.2 Proposed Bit-Allocation Framework

Figure 6 presents an illustration of the bit-allocation framework in transmitting a progressively encoded model. At the first stage, namely the streamed mode, the user subscribing to the application desires to quickly display the model in a limited time frame. In a rate-constrained environment, this imposes a bit budget on sending a possibly approximated model with certain resolutions for the mesh and the texture, respectively. With a quality measure \mathcal{Q} , the server searches among all combinations of mesh and texture resolutions

and finds the optimal pair, $(M^k, T^l)_{opt}$, that has the best fidelity while the bit rate satisfies the bit-budget constraint. After sending $(M^k, T^l)_{opt}$ for initial display, enhancement data (χ_G, χ_T) that respectively transfer (M^k, T^l) to their full resolutions (M^n, T^m) are organized into an interleaved bitstream. The bitstream consists of a sequence of data units, each of which conveys a certain number of vertex splits for the mesh and a portion of enhancement bits for the texture.

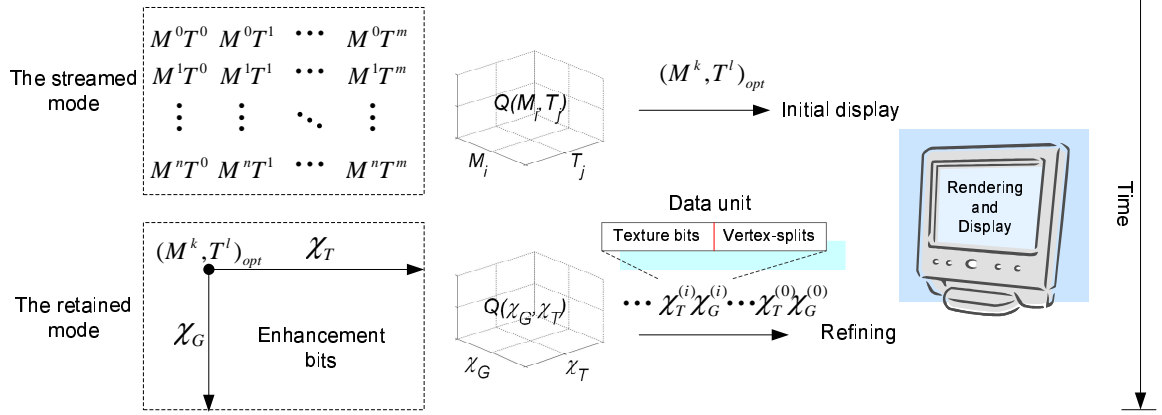


Figure 6: An illustration of the bit-allocation framework. The streamed mode and the retained mode deal with different data sets: $(M^i, T^j)_{[0,0] \leq (i,j) \leq [n,m]}$ are generated levels-of-details (LODs), while (χ_G, χ_T) denote the enhancement data that transfers a certain LOD to the full resolution.

For a further discussion we consider the streamed mode as an example. According to the description above, the rate-distortion framework for this problem can be mathematically stated as: *given a bit budget, \mathcal{C} , and a quality measure, \mathcal{Q} , the best representation of the model obtained in the set of meshes, $\{M^i\}_{i=0\dots n}$, and the set of textures, $\{T^j\}_{j=0\dots m}$, is given by*

$$(M^k, T^l)_{opt} = \arg \max_{(i,j): \mathcal{R}_G(M^i) + \mathcal{R}_T(T^j) \leq \mathcal{C}} \mathcal{Q}(M^i, T^j), \quad (8)$$

where $\mathcal{R}_G(M^i)$ and $\mathcal{R}_T(T^j)$ denote the bit rates of the compressed mesh M^i and texture T^j , respectively, and $\mathcal{Q}(M^i, T^j)$ is the quality when mapping texture T^j onto mesh M^i .

The solution of (8) can be obtained by an exhaustive search over the space of solutions, i.e., comparing all possible pairs of mesh and texture resolutions under the bit budget and finding the one that maximizes the quality measure \mathcal{Q} . It is apparent that this process has computational complexity of $O(n \times m) \cdot O(\mathcal{Q})$, where $O(\mathcal{Q})$ denotes the computation cost

of the quality measure. Due to this factor, using SSE as the quality measure will make the process computationally expensive, for it requires a number of rendering operations for calculating $Q(M^i, T^j)$ and rendering is costly in most of today’s computing systems. For a solution space that has dimensions 40×30 , for example, calculating SSE using the small rhombicuboctahedron needs to capture a total number of $24 \times 40 \times 30 = 28,800$ displayed images. On a computer with a Pentium IV 1.7 GHz CPU, 512 MB RAM, and an NVIDIA Vanta™ graphics card, and using OpenGL as the API, performing this number of display and capture operations consumes approximately 10 hours⁴ for a textured model that has 40,000 triangles and a 512×512 -pixel texture.

More essentially, the quality measure Q is desired to reflect the visual fidelity of the displayed model. For multi-resolution meshes without texture mapping, SSE has been confirmed working successfully in evaluating visual fidelity [96]. However, it has been discovered that when substituting simplified mesh geometry with multi-resolution texture images, SSE may not reflect the perceptual quality properly. Figure 7 shows an example, where we generate a simplified mesh for the MAP-SPHERE model, and then map it with several texture images with different resolutions. The texture resolution is represented by the number of bits per pixel (*bpp*). As can be seen in Figures 7(b-d), the model with a higher resolution texture (higher bit rate) has a lower PSNR value than the model with the same geometry but a lower-resolution texture (lower bit rate), even though the former apparently has better visual fidelity than the latter. This observation shows the existing inaccuracy of using SSE as the fidelity measure for jointly simplified textured models. According to our experiments, such pathologies are generally observed for textured models when mesh geometry is substantially simplified (simplification ratio varies among different models). To guarantee that proper solutions are obtained in (8), a meaningful while more efficient quality measure is required.

⁴A dominant portion of this time measurement is the time consumption of file loading and API-call initialization for every display. If we exclude the file loading and call initialization, rendering operations will take roughly 20–30 minutes.

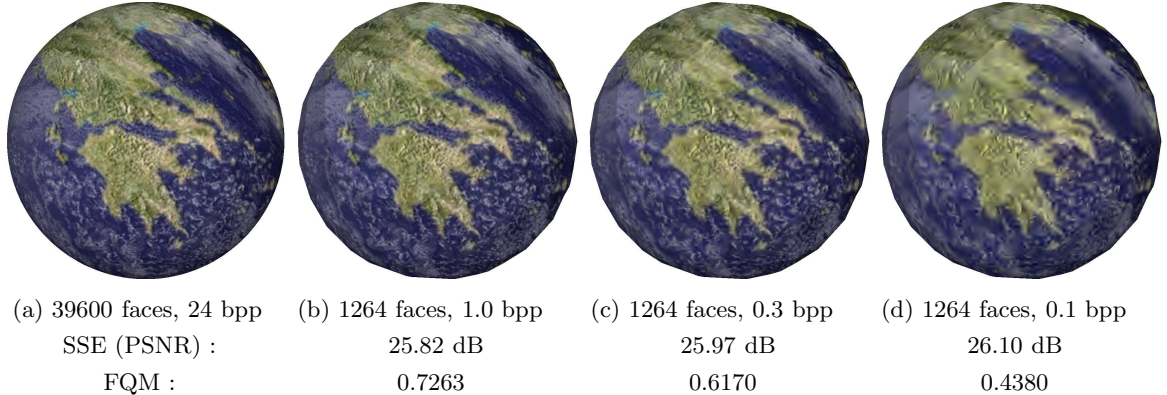


Figure 7: An example where SSE fails to capture the quality difference of simplified textured models. For comparison, quality measures given by the fast quality measure (FQM) (to be detailed in Section 3.2.1) are also presented.

3.2.1 The Fast Quality Measure

To effectively measure the quality of a jointly simplified textured model, we first consider the effects of geometry and texture simplification processes separately. Watson et al. in [96] showed that both the screen-space error and the surface distance successfully predict the perceptual fidelity in the simplified geometry if *no* texture is mapped. On the other hand, the mean-square pixel error is most widely used in imaging to measure the quality of compressed 2D images [32]. Driven by these observations, we propose to combine the mean-square texture deviation computed during the mesh simplification process and the mean-square pixel error incurred by the texture coding, and construct a novel quality measure for textured 3D models. Mathematically, we define the quality (\mathcal{Q}) of a model with texture T^j mapped on mesh M^i as

$$\mathcal{Q}(M^i, T^j) = \lambda \mathcal{Q}_G(M^i) + (1 - \lambda) \mathcal{Q}_T(T^j) \quad (9)$$

for $[0, 0] \leq (i, j) \leq [n, m]$, where $\lambda \in [0, 1]$ is introduced as an *equalization* factor between the mesh and the texture, and will be discussed later in this section. Respectively, \mathcal{Q}_G and \mathcal{Q}_T are computed using errors measured in the geometric and texture domains:

$$\mathcal{Q}_G(M^i) = \text{norm} \left[\log_{10} \left(1 - \frac{\text{MSE}_G(M^i)}{L^2} \right) \right], \quad i \in [0, n], \quad (10)$$

and

$$\mathcal{Q}_{\mathcal{T}}(T^j) = \text{norm} \left[\log_{10} \left(1 - \frac{\text{MSE}_{\mathcal{T}}(T^j)}{255^2} \right) \right], \quad j \in [0, m]. \quad (11)$$

L in (10) is the diagonal of the bounding box of the mesh. Function $\text{norm}[\cdot]$ is defined as a *normalization* operator that normalizes the scales of $\mathcal{Q}_{\mathcal{G}}$ and $\mathcal{Q}_{\mathcal{T}}$ into $[0, 1]$, respectively. In other words, the scales of $\mathcal{Q}_{\mathcal{G}}$ and $\mathcal{Q}_{\mathcal{T}}$ are normalized in their respective coordinate systems so that the full resolutions of mesh and texture have measures $\mathcal{Q}_{\mathcal{G}}(M^n) = \mathcal{Q}_{\mathcal{T}}(T^m) = 1$, and their coarsest versions have quality rated as $\mathcal{Q}_{\mathcal{G}}(M^0) = \mathcal{Q}_{\mathcal{T}}(T^0) = 0$. The scaling effect between two coordinate systems will be accounted for by the equalization factor λ and therefore it will not affect the eventual quality measures.

In the above equations, $\text{MSE}_{\mathcal{G}}$ and $\text{MSE}_{\mathcal{T}}$ are the mean-square errors measured in the geometric and texture domains, respectively⁵. More explicitly, $\text{MSE}_{\mathcal{T}}$ is the mean-square pixel error between the simplified and the full-resolution texture images, calculated using SPIHT [72]. For $\text{MSE}_{\mathcal{G}}$, the Metro tool [27] can be employed to compute the mean-square surface distance between two meshes. For fast computation, however, in our implementation we compute $\text{MSE}_{\mathcal{G}}$ using the mean-square texture deviation (MSD) instead of the Metro error (MetroMSE), as texture deviation has been measured during the progressive compression process and therefore it does not require additional computation. Moreover, it is noticed in our experiments that the mean-square texture deviation provides close scales to the Metro error in a normalized coordinate system, as depicted in Figure 8.

Equations (10) and (11) quantify the quality of the coarser mesh and the lower-resolution texture in their own domains, while the relative effect of the geometry inaccuracy and the texture distortion on visual fidelity is modeled in (9) by the equalization factor λ . For two extreme cases where the model has either sole mesh geometry or texture image, we simply have $\lambda = 1$ and $\lambda = 0$, and (9) returns to be $\mathcal{Q}_{(\lambda=1)} = \mathcal{Q}_{\mathcal{G}}$ and $\mathcal{Q}_{(\lambda=0)} = \mathcal{Q}_{\mathcal{T}}$, correspondingly. These results are meaningful since both $\text{MSE}_{\mathcal{G}}$ and $\text{MSE}_{\mathcal{T}}$ are successful measures of visual fidelity in their respective domains [32, 96].

⁵For differentiation purpose, in the rest of the chapter we specifically use $\text{MSE}_{\mathcal{T}}$ to refer to the mean-square image error measured in the texture domain, while using SSE to denote the mean-square error computed for the displayed models in the rendering space.

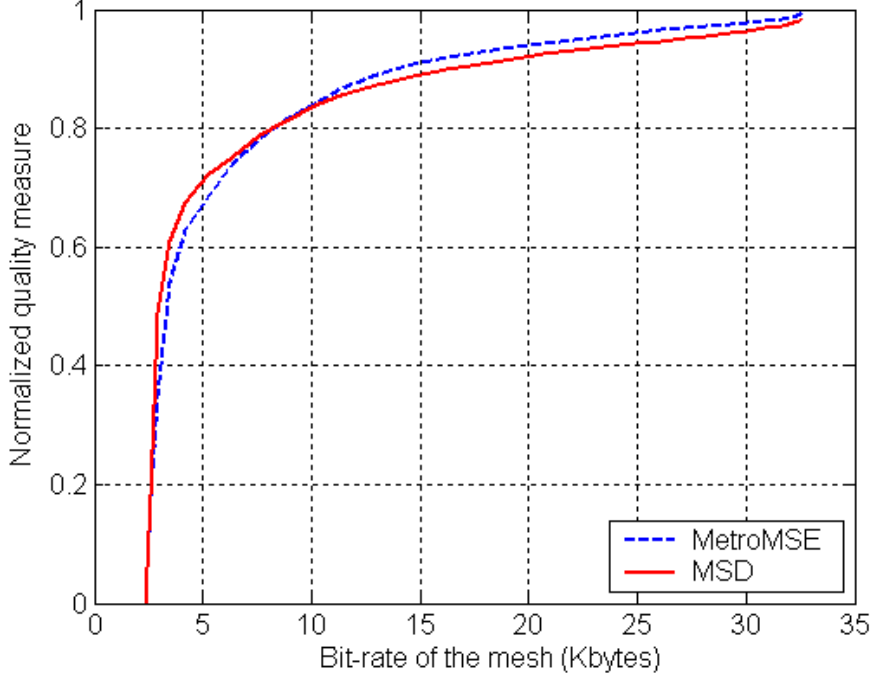


Figure 8: With normalized scales, the mean-square texture deviation (MSD) and the mean-square surface distance measured by Metro [27] (MetroMSE) generate close distortion-rate curves.

In general, it is natural that we expect λ for a particular model to be strongly dependent on the characteristics of that model. Potential factors include the spatial and spectral distribution of the texture image, the fineness scale of the triangular mesh, and the masking effect of substituting the texture for the surface, etc. Unfortunately, finding a computational method for λ that fully describes all the possibilities will not be practical concerning computational efficiency.

To explore a computationally efficient method, we take partial derivatives of (9) to get

$$\lambda = \frac{\partial Q}{\partial Q_G} = \frac{\partial Q}{\partial \mathcal{R}_G} / \frac{\partial Q_G}{\partial \mathcal{R}_G}, \quad (12)$$

and

$$(1 - \lambda) = \frac{\partial Q}{\partial Q_T} = \frac{\partial Q}{\partial \mathcal{R}_T} / \frac{\partial Q_T}{\partial \mathcal{R}_T}, \quad (13)$$

where \mathcal{R}_G and \mathcal{R}_T are the bit rates of the mesh and texture, respectively. Q_G, \mathcal{R}_G and

$\mathcal{Q}_T, \mathcal{R}_T$ are measured quantities in mesh and texture simplification processes. Combining (12) and (13) into a single equation, we obtain

$$\frac{\lambda}{1-\lambda} = \frac{\rho_G}{\rho_T} \Rightarrow \lambda = \frac{\rho_G}{\rho_G + \rho_T}, \quad (14)$$

where

$$\rho_G \triangleq \frac{\partial \mathcal{Q}}{\partial \mathcal{R}_G} / \frac{\partial \mathcal{Q}_G}{\partial \mathcal{R}_G}, \text{ and } \rho_T \triangleq \frac{\partial \mathcal{Q}}{\partial \mathcal{R}_T} / \frac{\partial \mathcal{Q}_T}{\partial \mathcal{R}_T}.$$

Based on (12)-(14), we have developed a simple but effective computational method of estimating the equalization factor. In doing so, we estimate $\partial \mathcal{Q}$ in the rendering space by carefully selecting sample pairs of the mesh and texture resolutions so as to obtain a meaningfully quantified quality difference between them. In particular, the resolutions that we choose to render in the screen space will fall around the *turning* points on the curves of $(\mathcal{R}_G, \mathcal{Q}_G)$ and $(\mathcal{R}_T, \mathcal{Q}_T)$, respectively. Figure 9(a) is a demonstration of this selection conducted in the $(\mathcal{R}_G, \mathcal{Q}_G)$ coordinate system. In doing so, we find an index, k , such that the line constructed by two points, $(\mathcal{R}_G^{(k)}, \mathcal{Q}_G^{(k)})$ and $(\mathcal{R}_G^{(k-1)}, \mathcal{Q}_G^{(k-1)})$, makes a 45° -angle (or the nearest if not exact) with the x -axis. We denote

$$\Delta \mathcal{Q}_G^{(k)} = \mathcal{Q}_G^{(k)} - \mathcal{Q}_G^{(k-1)}, \Delta \mathcal{R}_G^{(k)} = \mathcal{R}_G^{(k)} - \mathcal{R}_G^{(k-1)}.$$

Likewise, we find an index l for the texture such that $(\mathcal{R}_T^{(l)}, \mathcal{Q}_T^{(l)})$ is the *turning* point on the curve of $(\mathcal{R}_T, \mathcal{Q}_T)$ (Figure 9[b]), and denote

$$\Delta \mathcal{Q}_T^{(l)} = \mathcal{Q}_T^{(l)} - \mathcal{Q}_T^{(l-1)}, \Delta \mathcal{R}_T^{(l)} = \mathcal{R}_T^{(l)} - \mathcal{R}_T^{(l-1)}.$$

By choosing the resolutions around the turning points, we expect to attain well bounded numerical values for $\frac{\Delta \mathcal{Q}_G}{\Delta \mathcal{R}_G}$ and $\frac{\Delta \mathcal{Q}_T}{\Delta \mathcal{R}_T}$, the denominators in (12) and (13), hence minimizing potential arithmetic errors in the computation. In addition, according to our observation, rendering thus selected textured models avoids encountering singularities and provides the meaningful measurements of the screen-space error, which lead to a good estimate of $\partial \mathcal{Q}$ in (12) and (13).

Assume that we have selected M^k, M^{k-1} and T^l, T^{l-1} on respective curves. Three pairs of mesh and texture resolutions, (M^k, T^l) , (M^{k-1}, T^l) , and (M^k, T^{l-1}) , as well as the

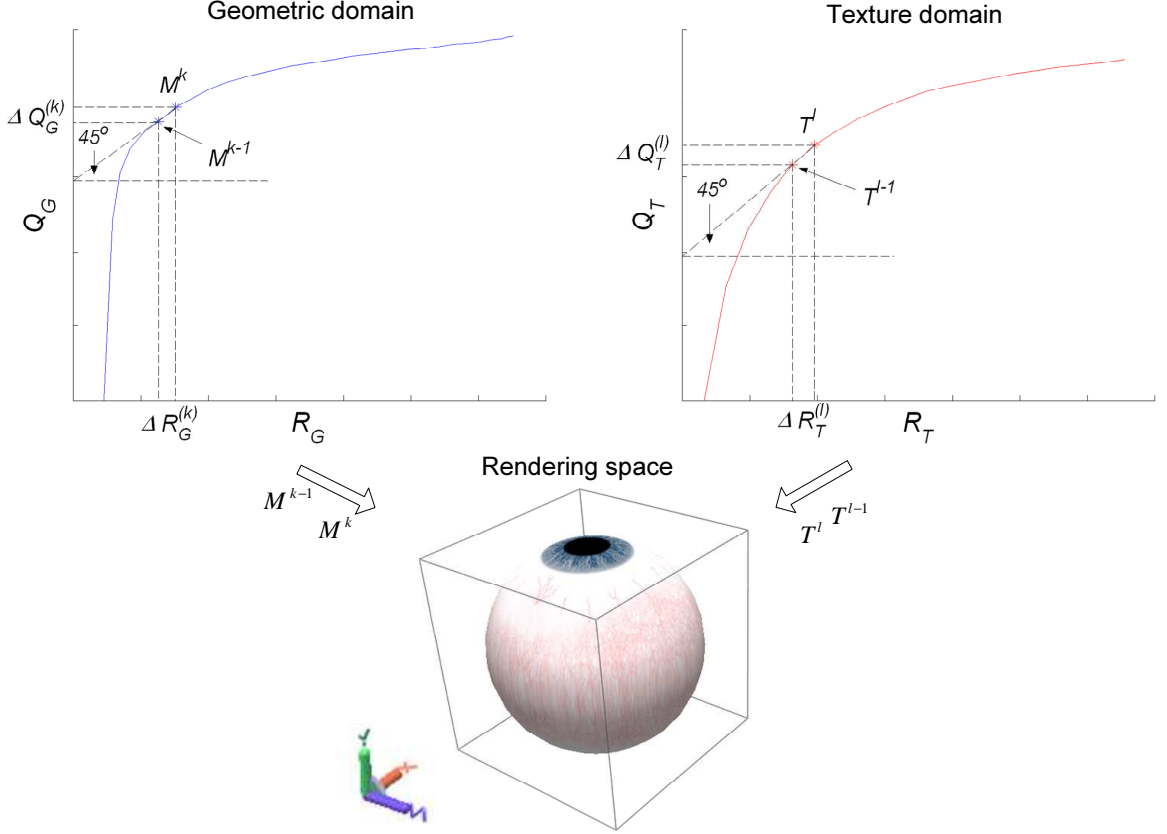


Figure 9: An illustration on estimating the equalization factor λ . (M^k, M^{k-1}) and (T^l, T^{l-1}) are selected *turning points* on the $(\mathcal{R}_G, \mathcal{Q}_G)$ and $(\mathcal{R}_T, \mathcal{Q}_T)$ curves, respectively, which construct lines that have a 45° -angle with the x -axis. The selected pairs are rendered in the screen space for estimating the equalization factor λ .

coarsest representation, (M^0, T^0) , are then rendered in the screen space and their errors are measured respectively. The corresponding numerical results are represented by $\sigma_{k,l}^2$, $\sigma_{k-1,l}^2$, $\sigma_{k,l-1}^2$, and $\sigma_{0,0}^2$. All the quantities are computed in the screen space following Equation (6).

We then define

$$\mathcal{P} = \log_{10}\left(1 - \frac{\sigma^2}{255^2}\right) \quad (15)$$

for the measured mean-square errors and again normalize the results so that $\mathcal{P}_{0,0} = 0$ for the coarsest representation, (M^0, T^0) , and consequently, $\mathcal{P}_{n,m} = 1$ for the full-resolution model, (M^n, T^m) . Denoting the corresponding results for (M^k, T^l) , (M^{k-1}, T^l) , and (M^k, T^{l-1}) by $\mathcal{P}_{k,l}$, $\mathcal{P}_{k-1,l}$, and $\mathcal{P}_{k,l-1}$, respectively, we estimate ∂Q in (12) and (13) by

$$\Delta Q' = \mathcal{P}_{k,l} - \mathcal{P}_{k-1,l}, \text{ and } \Delta Q'' = \mathcal{P}_{k,l} - \mathcal{P}_{k,l-1}. \quad (16)$$

Using (14), the equalization factor, λ , is finally computed by

$$\lambda = \frac{\Delta \mathcal{Q}' / \Delta \mathcal{Q}_{\mathcal{G}}^{(k)}}{\Delta \mathcal{Q}' / \Delta \mathcal{Q}_{\mathcal{G}}^{(k)} + \Delta \mathcal{Q}'' / \Delta \mathcal{Q}_{\mathcal{T}}^{(l)}}. \quad (17)$$

Note that $\Delta \mathcal{R}_{\mathcal{G}}^{(k)}$ and $\Delta \mathcal{R}_{\mathcal{T}}^{(l)}$ have been cancelled in (17) as common factors.

The above process can be repeated by choosing more sample pairs in the screen space. For instance, in our experiments presented in the chapter, we choose to render two more pairs, (M^{k+1}, T^l) and (M^k, T^{l+1}) , and follow the computation described above but using (M^{k+1}, T^l) , (M^k, T^{l+1}) together with (M^k, T^l) and (M^0, T^0) . We then take the average as the final estimate of λ . Combining (9) with (17) gives the complete expression of the predicted visual fidelity for a simplified textured model. Clearly, FQM is computationally efficient, as only a few measurements are required in the screen space. In addition, experimental results presented in Section 3.3 confirm that FQM properly captures the visual fidelity of jointly simplified models, although it is not a rigorously proven metric.

3.2.2 The Boundary Search Algorithm

As discussed before, given the quality measure \mathcal{Q} , the solution to (8) can be obtained by an exhaustive search over the space of solutions, which has computational complexity of $O(n \times m)$. In contrast to SSE, using FQM makes the exhaustive search affordable for many applications as only numerical comparisons are needed after a small (and constant) overhead for estimating the equalization factor.

For applications that deal with large 3D models or require precise bit allocation between the mesh and the texture, the space of solutions may get substantially large. Thus the exhaustive search becomes inefficient and heuristic methods are desirable. One of such heuristics is the *marginal analysis* introduced in [17] using a resource reduction approach. The algorithm starts from the full-resolution mesh and texture and searches in the steepest decent direction while reducing the resolutions of the mesh and the texture. In other words, at each step, either the mesh resolution or the texture resolution is reduced, depending on which action results in a smaller increment of distortion, and the process is repeated until the bit rate satisfies the constraint.

The marginal analysis gains linear computational complexity at the price of reaching possibly suboptimal solutions. Nevertheless, it is noted that the quality function defined in (9) has monotonicity with both mesh and texture resolutions, which ensures that the estimated rate-quality surface has a convex property. With this convex property, instead of performing exhaustive search over the space of solutions, optimal bit-allocation solutions can be found on the boundary of the feasible region, namely,

$$(M^k, T^l)_{opt} = \arg \max_{(i,j):(M^i, T^j) \in \mathcal{B}_C} \mathcal{Q}(M^i, T^j), \quad (18)$$

where \mathcal{B}_C denotes the boundary of the feasible region determined by the bit budget C . The pseudo-code algorithm of procedure *Boundary*(C) is presented in Appendix A.1, where we assume $m < n$ without loss of generality. We learn from Algorithm 1 that finding the boundary of feasible region has computational complexity of $O(n + m)$, and obviously, $|\mathcal{B}_C| = \min(n, m)$. Therefore the overall complexity of (18) remains linear with respect to the decoupled dimensions of the solution space.

3.2.3 The Retained Mode

The previous discussion on the bit-allocation framework has presented a complete solution on finding an optimal pair of mesh and texture resolutions for a given bit budget that provides an approximated model with maximum quality (measured by FQM). Particularly, this solution corresponds to the streamed mode where a quick display on the user’s screen is desired when the user first subscribes to the application. We now extend the framework to the retained mode where having displayed a coarse version of the model, the client accommodates longer download duration and wishes to refine the displayed model for a more accurate interactive study. As illustrated in Figure 6, enhancement data is organized into a sequence of data units with interleaved mesh and texture data. At the client, rendering resolution is improved every time when a new data unit is received and decoded. The objective of this retaining process is to optimally packetize every successive data unit such that greatest increment of visual fidelity is attained at each instant when the displayed model is refined. The size of the data unit is in general constrained by a *maximum data unit* (MDU), which may vary among different application/networking environments but is

considered a constant parameter in a certain environment.

Let $(\chi_{\mathcal{G}}^{(i)}, \chi_{\mathcal{T}}^{(i)})$ ($i \geq 0$) denote the data bits that have been decoded by the client for the mesh and the texture, respectively, after the i -th refinement. Specifically, $(\chi_{\mathcal{G}}^{(0)}, \chi_{\mathcal{T}}^{(0)})$ represents the initial model that has been viewed by the client before the retaining process begins, which may be the lowest-resolution version of the model or a pair of mesh and texture resolutions determined in the streamed mode. Assuming that the size of MDU is \mathcal{K} bits, the bit-allocation solution for the $(i + 1)$ -th refinement level is given by

$$\begin{cases} (\Delta\chi_{\mathcal{G}}, \Delta\chi_{\mathcal{T}})_{opt} = \arg \max_{\Delta\chi_{\mathcal{G}} + \Delta\chi_{\mathcal{T}} \leq \mathcal{K}} \mathcal{Q}(\chi_{\mathcal{G}}^{(i)} + \Delta\chi_{\mathcal{G}}, \chi_{\mathcal{T}}^{(i)} + \Delta\chi_{\mathcal{T}}; \lambda), \\ (\chi_{\mathcal{G}}^{(i+1)}, \chi_{\mathcal{T}}^{(i+1)}) = (\chi_{\mathcal{G}}^{(i)}, \chi_{\mathcal{T}}^{(i)}) + (\Delta\chi_{\mathcal{G}}, \Delta\chi_{\mathcal{T}})_{opt}, \end{cases} \quad (19)$$

where \mathcal{Q} is the FQM function and λ is the equalization factor associated with a particular model. $(\Delta\chi_{\mathcal{G}}, \Delta\chi_{\mathcal{T}})_{opt}$ denote the numbers of bits for the mesh and the texture, respectively, according to which a new data unit will be packetized for the next transmission. Starting from $(\chi_{\mathcal{G}}^{(0)}, \chi_{\mathcal{T}}^{(0)})$, the computation in (19) is repeated until all the enhancement data is transmitted or the retaining process is terminated by the client. The algorithm presented by (19) is referred to as the *maximum quality bit-allocation* (MxQ-BA) algorithm hereafter.

For each decoding opportunity, finding the optimal bit-allocation solution of (19) has a similar formulation as (18) while the solution space is constructed by all possible combinations of a certain number of vertex splits for refining the mesh and a portion of enhancement bits for the texture. Practically, it is neither efficient nor necessary to investigate the problem for individual vertex splits or few texture bits as they perform small and local changes, which normally do not result in perceivable difference. For this sake, in our experimental study presented in Section 3.3, we organize the refinement data of the mesh and the texture into small batches with a size of 100 bytes per batch. Then, at each transmission opportunity, we apply the MxQ-BA algorithm to decide the numbers of batches that should be allocated to the remaining mesh and texture data, respectively, and be packed into the to-be-sent data unit.

The MxQ-BA algorithm provides the best-effort solution for maintaining receiving quality at a maximal level during transmission. Nonetheless, it should be pointed out that in the MxQ-BA algorithm, additional information needs to be sent for each transmitted data

unit to indicate bit boundaries between the mesh and the texture. This header information is critical and has to be correctly received or the entire data unit will not be decodable, which may further affect decoding successive data units because of error propagation. In a typical network environment where bit errors may occur randomly during transmission, error resiliency for the bit-boundary information is important. While error protection bits can be added to provide error resiliency for the data unit, it introduces additional bandwidth requirement as a tradeoff. A detailed study on error-resilient transmission of textured 3D models is beyond the scope of the present work. As an alternative method to achieve better error resiliency, we consider a heuristical bit-allocation algorithm according to the estimated relative importance of the mesh and the texture. More explicitly, we consider performing constant-ratio bit allocation for every data unit using the equalization factor λ , where the numbers of bits for the mesh and the texture are simply given by

$$\begin{cases} (\Delta\chi_{\mathcal{G}}, \Delta\chi_{\mathcal{T}})_{\lambda} = \mathcal{K} \cdot (\lambda, 1 - \lambda), \\ (\chi_{\mathcal{G}}^{(i+1)}, \chi_{\mathcal{T}}^{(i+1)}) = (\chi_{\mathcal{G}}^{(i)}, \chi_{\mathcal{T}}^{(i)}) + (\Delta\chi_{\mathcal{G}}, \Delta\chi_{\mathcal{T}})_{\lambda}, \end{cases} \quad (20)$$

This simple heuristic is referred to as the λ -based *bit-allocation* (λ -BA) algorithm hereafter.

Using the λ -BA algorithm avoids sending bit boundaries in every transmitted data unit, making the retaining process more robust to random bit errors occurred during transmission. In addition, experimental results presented in Section 3.3 show that the λ -BA algorithm performs closely with MxQ-BA, given a good estimate on the equalization factor. Therefore, it provides a good alternate to the optimal algorithm in the situations where error resiliency is of importance.

3.3 *Experimental Results*

In this section we present experimental results for the proposed bit-allocation framework using several test models⁶, as listed in Table 1. In Table 1, T_{full} and T_{base} denote the numbers of polygons (faces) in the full-resolution mesh and in the base mesh, respectively. For all the models, the texture coordinates are quantized using 10 bits. To quantize the geometry coordinates, we use 12 bits for all the models except for MANDRILL, which has

⁶Thanks to Laurent Balmelli and Peter Lindstrom for providing the test models.

the lowest-polygon-count mesh among the test models and its geometry coordinates are quantized with 10 bits. For the texture images, we choose their coding rates in certain ranges such that the decoded images have the meaningful variations of visual fidelity (15–50 dB in general).

Table 1: Textured 3D models used in the experiments

Model	Geometry			Texture			λ
	T_{full}	T_{base}	n	Dimension	coding rate (bpp)	m	
MAP-SPHERE	39,600	784	40	512×512	0.05-1.0	20	0.5269
MARBLE-BALL				512×512			0.9127
MANDRILL	25,644	1176	30	512×512	0.05-1.0	20	0.3977
ZEBRA	58,494	1050	40	1024×1024	0.01-0.4	30	0.6276
SALAMANDER	71,254	1934	40	1024×1024	0.01-0.2	30	0.8990



(a) MAP texture



(b) MARBLE texture

Figure 10: The texture images of the MAP-SPHERE and the MARBLE-BALL models.

3.3.1 The Equalization Factor

In Table 1, the MARBLE-BALL model has the same mesh geometry as the MAP-SPHERE model, which has been shown in Figure 7, while they have different textures as shown in Figure 10. The equalization factors for the two models are found to be $\lambda_{Map} = 0.5269$ and $\lambda_{Marble} = 0.9127$, respectively. Note that λ is close to 0.5 for MAP-SPHERE, whereas λ is close to 1.0 for MARBLE-BALL. These results imply that with the give coding rates, the distortion of the Map texture has significant impact on the resulting visual fidelity of jointly simplified models when it is mapped to the sphere mesh. In contrast, the distortion of the Marble texture has less impact on the model fidelity. Instead, the quality degradation of MARBLE-BALL will be mostly resulted from the mesh error. Subjectively, such difference

can be confirmed by visually comparing the two textures in Figure 10: the MAP texture conveys more appearance detail of the model than the MARBLE image.

Figure 11 presents sampled mesh and texture pairs of the above two models. The FQM measures are calculated using (9) with the corresponding values of λ . Figure 11(a-d) and Figure 11(e-h) have the same four pairs of mesh and texture resolutions, but are ordered differently according to the estimated visual fidelity. From Figure 11(a) to Figure 11(d), visual quality of the models drops as the texture resolution decreases. In contrast, in Figure 11(e-h), difference of the textures is barely noticeable, and visual fidelity degrades as the mesh geometry becomes coarse (under flat shading). These subjective results show that the equalization factor meaningfully reflects the relative effect of the mesh and the texture.

In Figure 12(a), we plot for the MAP-SPHERE model the measured quality of different mesh and texture pairs with respect to their joint bit rate. Each thin curve in the plot represents a constant texture bit rate with gradually increased mesh bit rates. One can observe in Figure 12(a) that for the same mesh resolution, the quality is improved considerably as the texture resolution increases. For instance, the lowest points of the thin curves correspond to mapping the textures with different resolutions to the coarsest mesh geometry. Recall the experiment presented in Figure 7, where SSE fails in reflecting the perceptual fidelity when mapping textures with different resolutions to highly simplified geometry. Marked accordingly in Figure 12(a) are the corresponding quality measures for the models shown in Figure 7(b-d), which provide more meaningful measurements on the perceptual difference. In addition, by eliminating the pathological cases, a convex rate-quality surface is obtained, plotted in Figure 12(b). The X- and Y-axes in Figure 12(b) denote the bit rates of the mesh and the texture, respectively. As discussed in Section 3.2, such a convex property guarantees that optimal bit-allocation solutions can be found in a linear time.

3.3.2 The Streamed Mode

In the rate-quality plot shown in Figure 12(a)), the dashed (red) line denotes the optimal envelope of the convex hull, which gives the upper bound of the model quality that could be achieved under certain bit budgets. Similar rate-quality curves have been generated for the

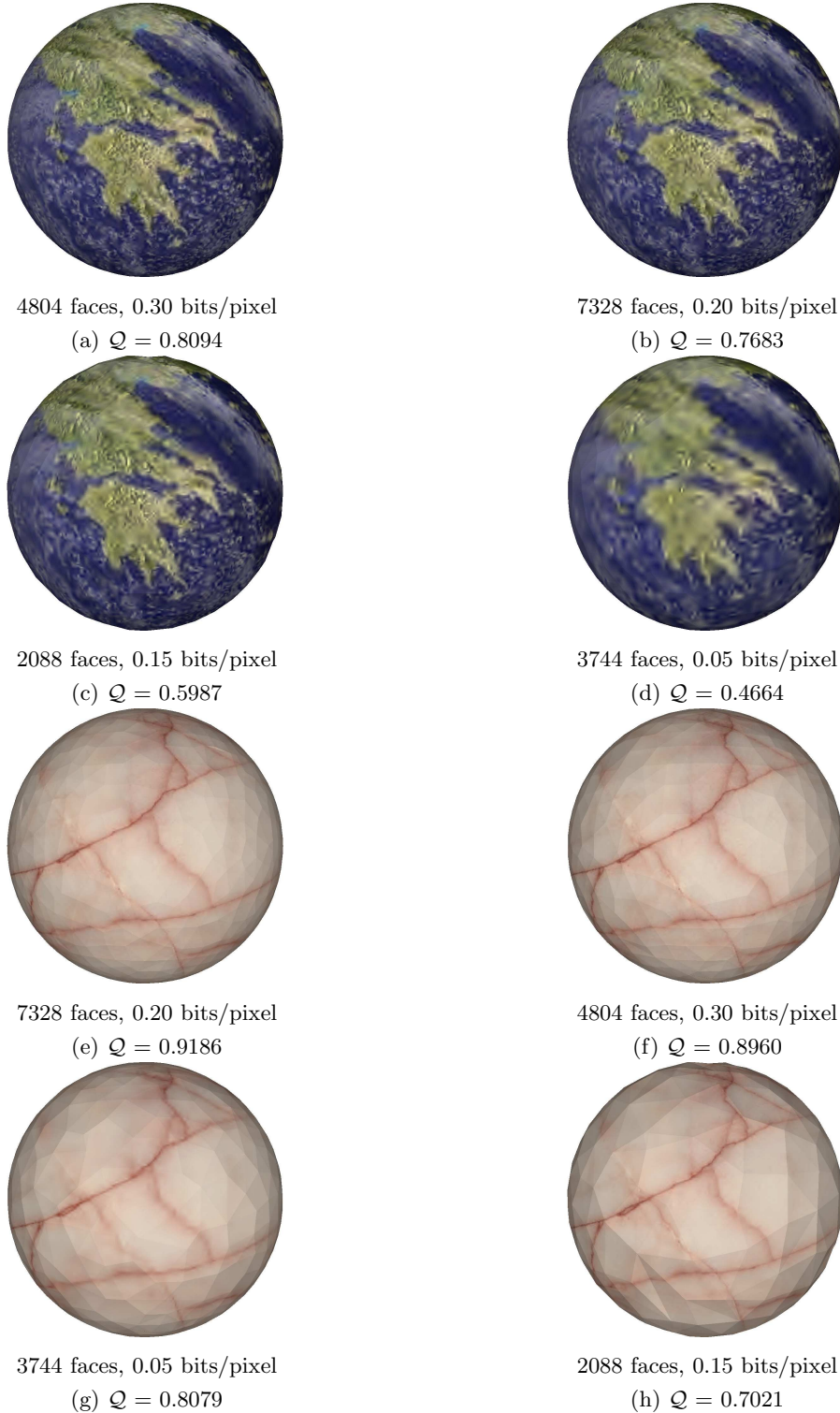


Figure 11: Rendered results of the MAP-SPHERE ($\lambda = 0.5269$) and the Marble-Ball ($\lambda = 0.9127$) models. (a-d) and (e-h) have the same four pairs of mesh and texture resolutions, but are presented respectively in the order of descending FQM measures.

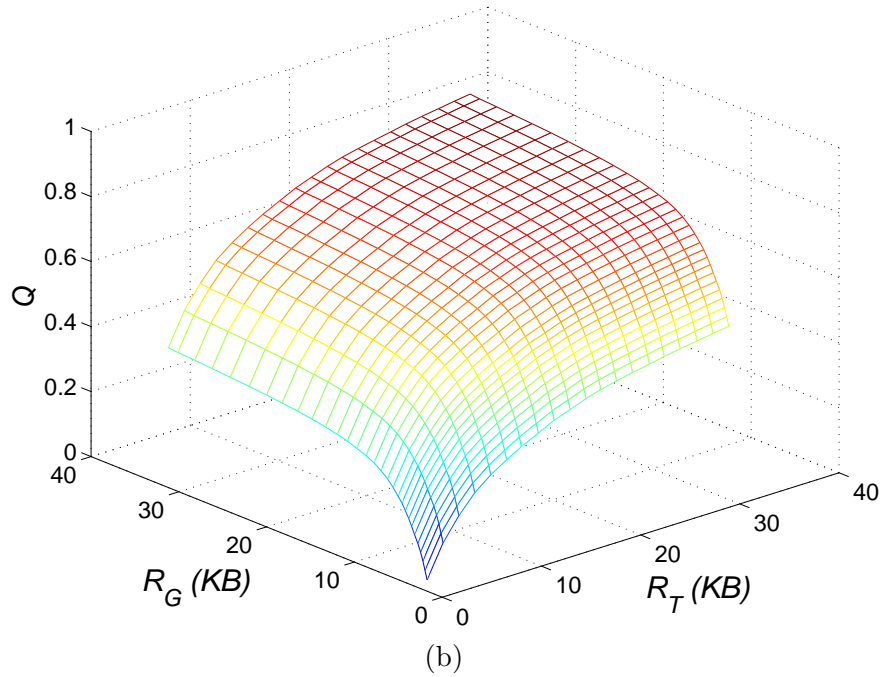
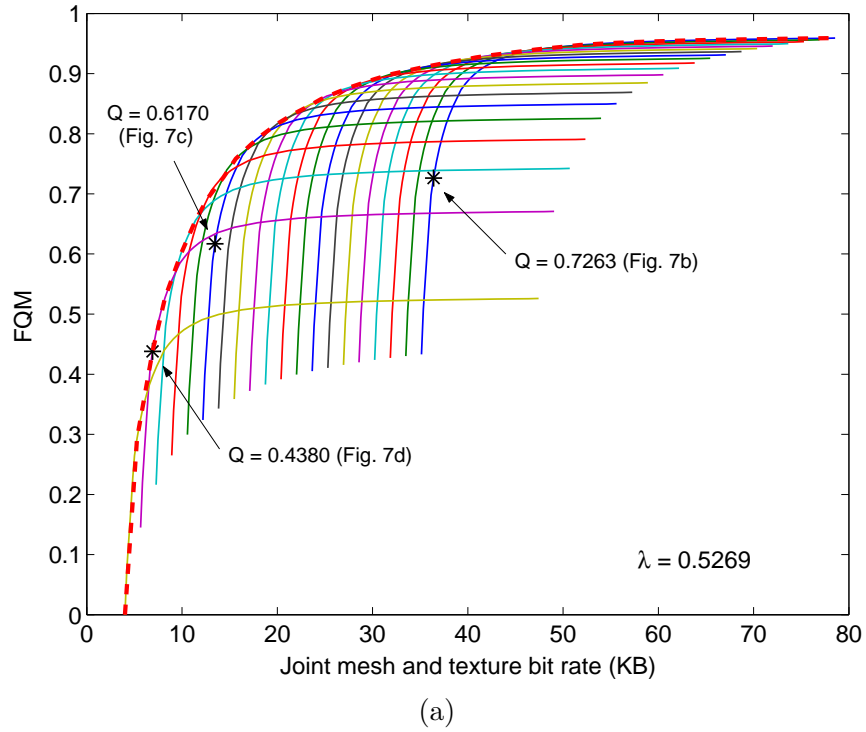


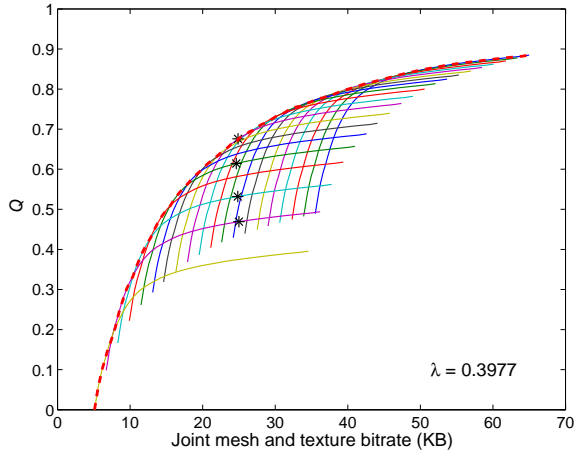
Figure 12: The rate-quality curve and surface of the MAP-SPHERE model. The dashed (red) curve in (a) plots the convex hull of the rate-quality curve.

other models using the estimated equalization factors (as listed in Table 1), respectively. With the rate-quality curve, the optimal pair of mesh and texture resolutions under a given bit budget (the streamed mode) can be found using the boundary search algorithm (18). For example, in Figure 13 we present the rate-quality curve for the MANDRILL model and under a 25 KB bit budget, several sampled results are provided for comparison. The point associated with Figure 13(c) is on the envelope of Figure 13(a), meaning that given a bit budget of 25KB, it provides the best approximation of the MANDRILL model among all the generated resolutions.

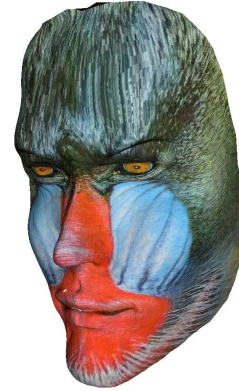
For another demonstration, several bit budgets are imposed on the ZEBRA model ranging from 8 KB to 40 KB, and the optimal results found on the envelope of the rate-quality curve (Figure 14(a)) are captured in Figures 14(b-e). A high-resolution model ($\mathcal{Q} = 0.8355$) is obtained when the bit budget is large (40 KB). The model fidelity decreases as the bit budget drops from 40 KB to 8 KB. Nevertheless, by reducing both mesh and texture resolutions in a *balanced* manner, visual fidelity is maintained at the maximal level under the corresponding bit-rate constraint. Also, the quality measures properly reflect the degradation in visual fidelity.

In Table 2, we present a comparative study on the boundary search algorithm with the marginal analysis using the data sets of the MANDRILL and ZEBRA models. For a sequence of bit budgets, the quality measures found by the boundary search (\mathcal{Q}_B) and the quality measures obtained by the marginal analysis (\mathcal{Q}_M) are presented, appended with the corresponding bit-allocation percentages of the mesh and the texture. Table 2 shows that the boundary search algorithm always finds the maximum quality while the marginal analysis algorithm is suboptimal even though it provides close-to-maximum results. The difference between \mathcal{Q}_M and the optima, \mathcal{Q}_B , is more apparent for lower bit budgets and/or in the MANDRILL case.

Figure 15 visually compares the two algorithms. In Figure 15(a), the (blue) line with plus signs shows the execution path of the marginal analysis for the ZEBRA model under a bit budget of 15KB, while the (black) line with dots indicates the path of the boundary search. The solutions reached by the two algorithms are marked on the contour by circles,



(a) RATE-QUALITY CURVE



25,644 faces, 24 bits/pixel
MANDRILL



5,132 faces, 0.45 bits/pixel
 $(R_G, R_T) = (10.48, 14.40)$ KB
(b) $Q = 0.6758$



8,820 faces, 0.25 bits/pixel
 $(R_G, R_T) = (16.63, 8.00)$ KB
(c) $Q = 0.6142$

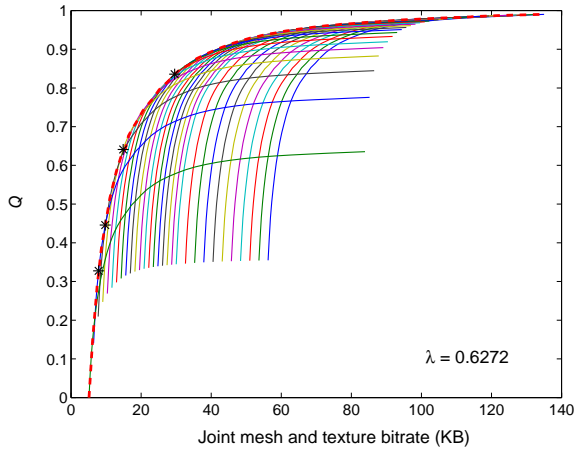


10,932 faces, 0.15 bits/pixel
 $(R_G, R_T) = (20.04, 4.79)$ KB
(d) $Q = 0.5319$



12,168 faces, 0.10 bits/pixel
 $(R_G, R_T) = (21.80, 3.20)$ KB
(e) $Q = 0.4693$

Figure 13: The optimal pair of resolutions found for the MANDRILL model: (a) is the rate-quality curve; (b-e) have decreasing fidelity and are correspondingly marked in (a) under the bit budget 25 KB.



(a) Rate-quality curve



58,494 faces, 24 bits/pixel
ZEBRA



11,806 faces, 0.09 bits/pixel
 $(R_G, R_T) = (20.40, 9.18)$ KB
(b) $Q = 0.8355$



3,370 faces, 0.04 bits/pixel
 $(R_G, R_T) = (9.64, 5.24)$ KB
(c) $Q = 0.6410$



2,330 faces, 0.02 bits/pixel
 $(R_G, R_T) = (7.16, 2.62)$ KB
(d) $Q = 0.4459$



1,586 faces, 0.02 bits/pixel
 $(R_G, R_T) = (5.27, 2.62)$ KB
(e) $Q = 0.3276$

Figure 14: The optimal pair of resolutions found for the ZEBRA model: (a) is the rate-quality curve; (b-e) correspond to the marked points along the envelope in (a), with the bit budget varying from 40KB to 8KB.

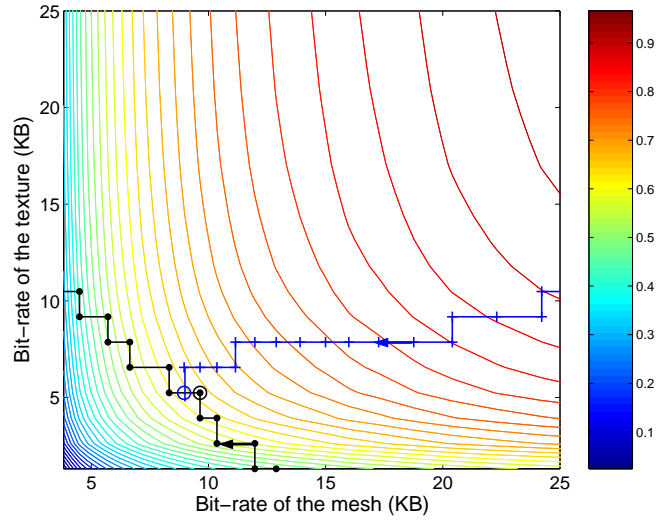
Table 2: Comparisons of the boundary search and the marginal analysis algorithms under various bit rates

MANDRILL		
\mathcal{C} (KB)	\mathcal{Q}_B	\mathcal{Q}_M
8	0.2051 (80%, 20%)	0.0984 (52%, 48%)
10	0.3035 (67%, 33%)	0.2222 (35%, 65%)
12	0.3887 (59%, 41%)	0.2614 (31%, 69%)
14	0.4649 (54%, 46%)	0.3458 (29%, 71%)
16	0.5168 (60%, 40%)	0.3720 (26%, 74%)
20	0.5986 (44%, 56%)	0.5140 (27%, 73%)
24	0.6630 (40%, 60%)	0.6304 (36%, 64%)
28	0.7048 (42%, 58%)	0.6989 (35%, 65%)
32	0.7465 (40%, 60%)	0.7366 (34%, 66%)
36	0.7746 (42%, 58%)	0.7681 (36%, 64%)
ZEBRA		
\mathcal{C} (KB)	\mathcal{Q}_B	\mathcal{Q}_M
8	0.3276 (67%, 33%)	0.2092 (49%, 51%)
10	0.4459 (73%, 27%)	0.3605 (55%, 45%)
12	0.5395 (66%, 34%)	0.5258 (56%, 44%)
14	0.6030 (71%, 29%)	0.5991 (61%, 39%)
16	0.6595 (66%, 34%)	0.6425 (58%, 42%)
20	0.7281 (60%, 40%)	0.7281 (60%, 40%)
28	0.8215 (67%, 33%)	0.8078 (70%, 30%)
32	0.8480 (71%, 29%)	0.8480 (71%, 29%)
36	0.8713 (63%, 37%)	0.8690 (70%, 30%)
40	0.8922 (67%, 33%)	0.8880 (73%, 27%)

and the corresponding quality measures are found in Table 2 to be $\mathcal{Q}_B = \mathcal{Q}_{opt} = 0.6410$ and $\mathcal{Q}_M = 0.6209$, respectively. The captured views are shown in Figure 15(b-c), where we can see that even though the marginal analysis provides a near-optimal solution, difference in visual fidelity is still noticeable from a certain perspective. Likewise, Figure 15(d-e) captures the results of the two algorithms for the MANDRILL model under a bit budget 20 KB. Not only is the difference between \mathcal{Q}_{opt} and \mathcal{Q}_M larger, but also the perceptual difference between the rendered models is more apparent. The numerical results and the subjective observations also confirm the effectiveness of FQM in predicting visual fidelity.

3.3.3 The Retained Mode

We now present experimental results for the retained mode. In our experiments, without loss of generality, we consider that the initial display on the user’s screen is the coarsest



(a)



$$Q_B = Q_{opt} = 0.6410$$

(b)



$$Q_M = 0.6209$$

(c)



$$Q_B = Q_{opt} = 0.5986$$

(d)



$$Q_M = 0.5140$$

(e)

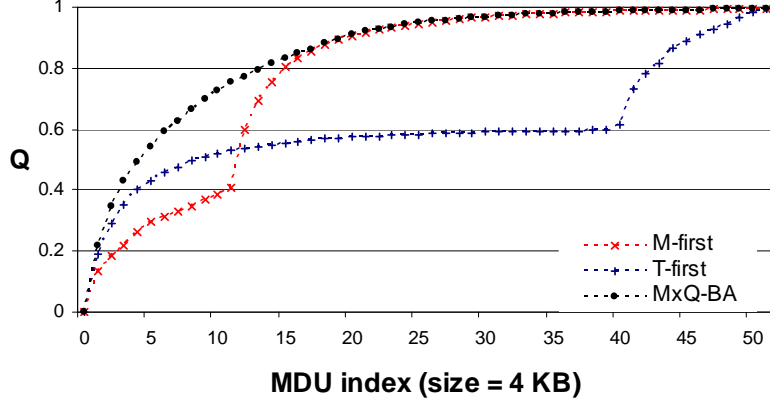
Figure 15: A comparison of the boundary search and the marginal analysis algorithms: (a) the execution paths for the ZEBRA model under a bit budget 15 KB, (b-c) the corresponding subjective results, and (d-e) the subjective results of MANDRILL under a 20 KB rate budget.

version of the progressively compressed model that has the lowest bit rate, and start the computation of the retaining process (19) with $\mathcal{Q}(\chi_G^{(0)}, \chi_T^{(0)}) = 0$. The size of the maximum data unit (MDU) is chosen to be $\mathcal{K} = 2$ KB or 4 KB.

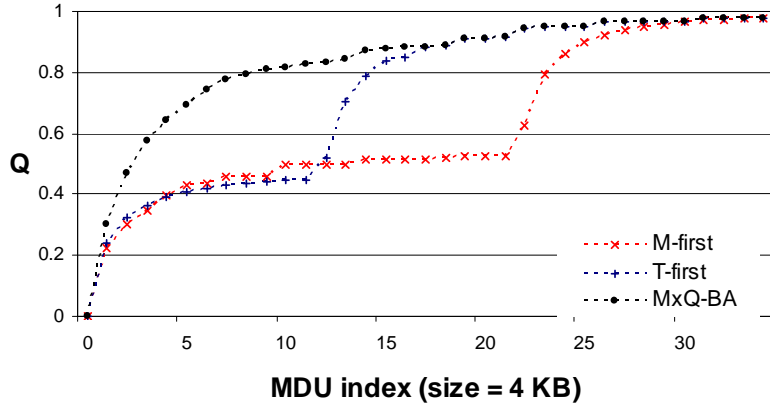
We first compare the MxQ-BA algorithm with two simplest heuristics that transmit first all the mesh (texture) data and then the texture (mesh). These simple heuristics are called *M-first* and *T-first*, respectively. For the MANDRILL and the MAP-SPHERE models, Figures 16(a-b) present the curves of increased quality obtained by gradually decoding refining data units. As expected, the MxQ-BA algorithm significantly outperforms M-first and T-first since the simplest heuristics constantly increase the resolution for the mesh or the texture without refining the other, which in general is very inefficient in increasing the resolution of the textured model. For the MANDRILL model (Figure 16(a)), for example, the quality obtained by the texture-first heuristic after receiving 10 data units is $\mathcal{Q} = 0.3841$, whereas $\mathcal{Q} = 0.7274$ is achieved by MxQ-BA.

As described in Section 3.2.3, MxQ-BA is a best-effort solution which organizes bits in the to-be-sent data unit to maximize the quality predicted by FQM. To avoid sending additional information for bit boundaries in MxQ-BA, the λ -based bit allocation (λ -BA) performs constant-ratio bit allocation using the equalization factor λ as the bit-allocation ratio (λ for the mesh and $(1 - \lambda)$ for the texture), given that λ estimates the relative importance between the mesh and the texture. Next, we investigate the performance of this heuristical method compared with MxQ-BA.

The experimental results for the test models are shown in Figure 17. In Figures 17(a-d), respectively, we plot the rate-quality curves for the λ -BA and the MxQ-BA algorithms as well as two other constant-ratio bit-allocation heuristics which choose bit-allocation percentages different from the equalization factor λ . The bit-allocation percentages for the mesh and the texture are noted in the plots. One can see that both λ -BA and MxQ-BA outperform the comparing constant-ratio bit allocation schemes. The performance upgrade is especially significant when the selected bit-allocation ratio has larger difference compared with the corresponding λ . As the ratio gets closer to λ , the bit-allocation performance



(a) MANDRILL ($\lambda = 0.3977$)



(b) MAP-SPHERE ($\lambda = 0.5269$)

Figure 16: A comparison of the MxQ-BA algorithm with the two simplest heuristics that transmit all the mesh (texture) data first and then the other.

between the constant-ratio scheme and MxQ-BA is less significant. Specifically, for the λ -based constant-ratio bit allocation, i.e., λ -BA, only a slight difference can be observed with MxQ-BA for first transmitted data units, and the quality curves gradually merge together. In Figure 17(d), it is noticed that at some points the λ -BA has even higher quality than MxQ-BA. This is because MxQ-BA always finds the optimal bit allocation that maximizes quality for the next data unit, which is the best-effort solution but possibly lose global optimality after a certain number of iterations. Also in Figures 17(a-d), substantial performance improvement is observed for the ZEBRA and the SALAMANDER models and in the cases where the data unit has the larger size ($\mathcal{K} = 4KB$). This observation is anticipated because the solution space becomes smaller when the size of data units decreases, hence lowering the possible gain that could be achieved by finding the optimal bit-allocation

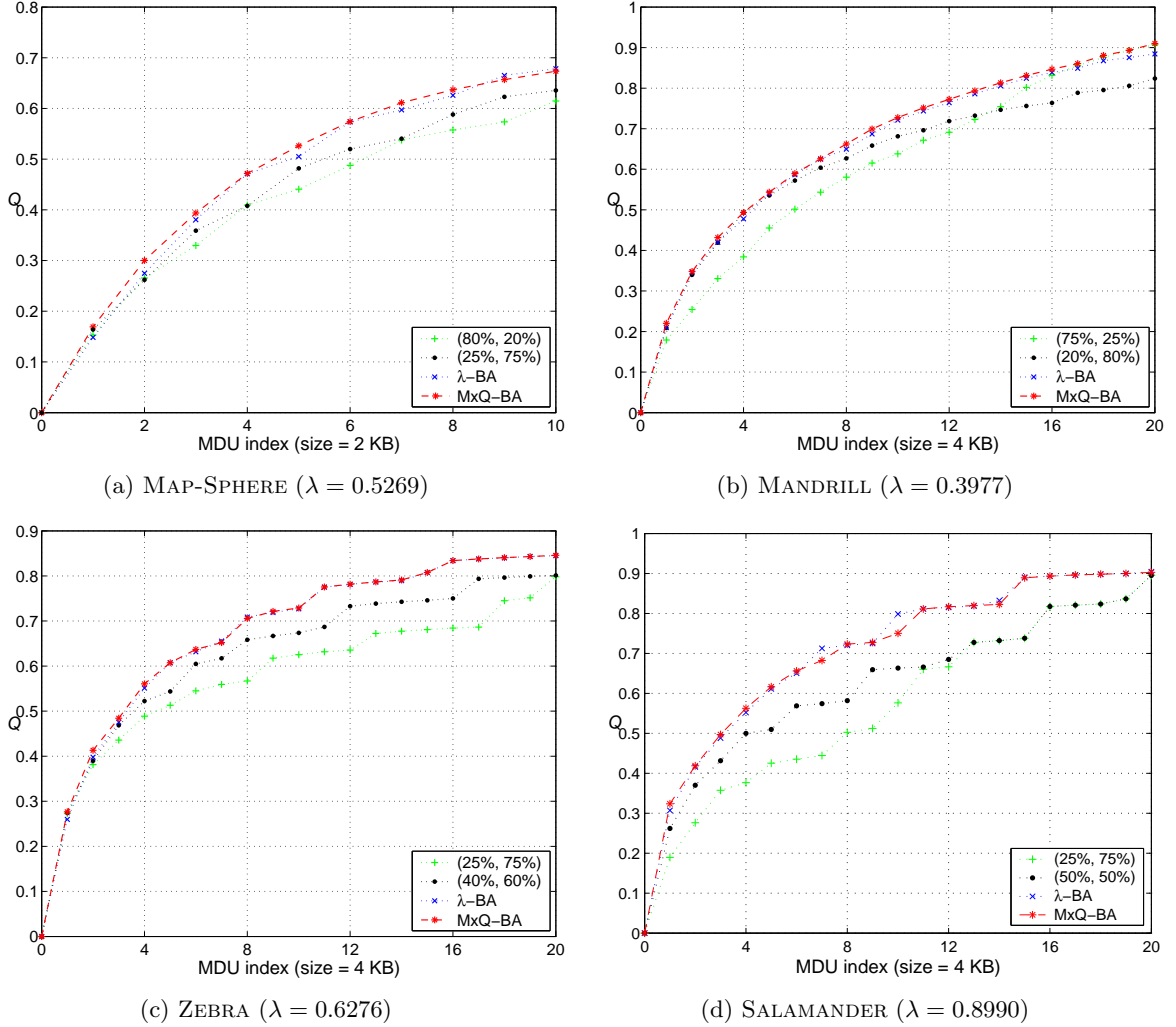


Figure 17: A comparison of the MxQ-BA and λ -BA algorithms with the constant-ratio bit allocation.

solution.

To end our discussion in this section, Figure 18 presents the rendered results of the SALAMANDER model after five data units are decoded, where Figures 18(b-d) correspond to the models obtained by MxQ-BA and the two constant-ratio schemes with bit-allocation percentages (25%, 75%) and (50%, 50%), respectively. Note, for example, how the claws become substantially distorted from Figure 18(b) to Figure 18(d). Such distortion is captured by the quality measure. Compared with the full-resolution model shown in Figure 18(a), the MxQ-BA algorithm provides the approximation that best preserves the details of both the mesh and texture. All the presented results verify the efficacy of the proposed bit-allocation

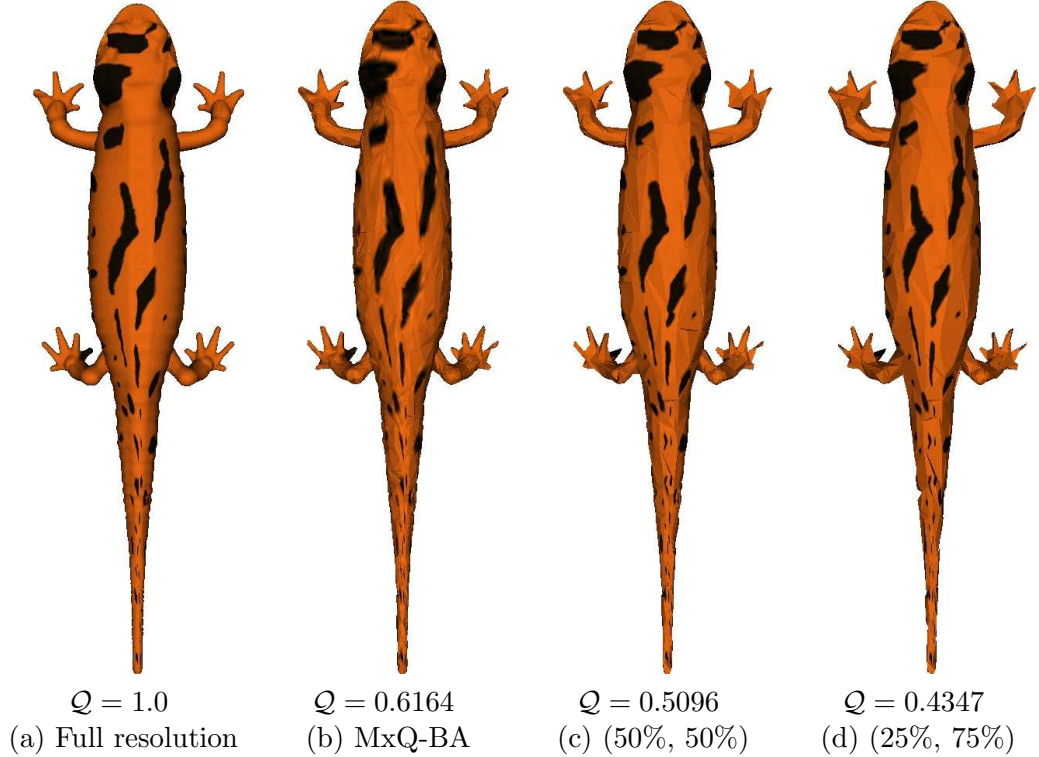


Figure 18: Sample results of the SALAMANDER model in the retaining process when five data units are decoded.

framework.

3.4 Summary

In this chapter, we presented a bit-allocation framework for efficiently streaming textured 3D models, which manages to maximize the quality of the displayed model during progressive transmission. To predict the visual fidelity, a fast quality measure (FQM) is proposed, which combines the distortion of both the mesh geometry and texture image through an equalization factor estimated in the rendering space. FQM provides meaningful prediction on visual fidelity and generates a convex rate-quality surface for the progressively compressed model. Using FQM, bit-allocation algorithms are developed to find the proper distribution of mesh and texture data that maximizes the quality of the transmitted model under a bit-rate constraint. Two transmission stages, namely the streamed mode and the retained mode, are addressed under a rate-distortional framework. The streamed mode corresponds to providing the best initial display on the user’s screen when the progressive

transmission begins, while the retained mode has the objective of maintaining the visual fidelity of the displayed model at the maximal level during transmission. The experimental results justified that by properly estimating the relative importance of the mesh and the texture, the proposed bit-allocation framework quickly presents high-resolution visualization of textured 3D models with limited bit budgets, thus achieving higher efficiency in progressive transmission.

In the retaining process, optimal bit allocation requires sending additional and error-sensitive information to indicate bit boundaries between the mesh and the texture in every data unit. To resolve the problem, we presented a constant-ratio bit-allocation scheme using the equalization factor λ estimated for the given model. This λ -based bit allocation performs closely with the optimal bit-allocation algorithm, making it attractive for real systems where both efficiency and error resiliency are important.

The equalization factor, which reflects the relative importance of the mesh and the texture for a particular model, is estimated through the rendering space. In the presented framework, it is estimated to be view independent by taking multiple virtual snapshots surrounding the object. It is worth to point out, however, that the proposed approach is general enough to allow an easy integration of the bit-allocation framework with view dependent texture coding (e.g., [55]) in MPEG-4. Through the rendering space, different equalization factors can be estimated for different views of the model, each of which will be used in the bit-allocation algorithms following the same process presented in this chapter.

Our discussion in this chapter was focused on the scalable coding of 3D models under rate-constrained environments. Network behaviors such as transmission delay and packet loss, while they play an essential role in streaming graphics applications, have not been considered. In the following chapter, we address network behaviors and develop transmission mechanisms for streaming scalably coded 3D models over lossy and rate-constrained networks.

CHAPTER IV

HYBRID RETRANSMISSION AND ERROR PROTECTION

4.1 Introduction

The objective of scalable streaming graphics applications is to provide real-time display of 3D models on disparate users' terminals with satisfactory quality. To this end, the impact of packet losses and transmission delays on the decoding process need to be explored. Typically, reliable or error-resilient transmission can be achieved by pre-processing techniques such as data partitioning [16, 97, 98], post-processing techniques such as error-concealment, and network-oriented techniques such as forward error correction [2, 7, 11] and retransmission techniques [3, 4, 10, 22–24]. All these techniques address efficient transmission of 3D data separately. Yet an appropriate combination of such techniques is desired to achieve better performance. Interaction and tradeoffs among the selected techniques need to be investigated, taking into account the distortion-rate performance of the 3D data and the network characteristics.

As the second milestone of our proposed research, we develop in this chapter a hybrid mechanism of unequal error protection (UEP) and selective retransmission for streaming 3D models over lossy networks [12, 84]. Hierarchical data batches of the multi-resolution 3D data are protected preferentially according to their distortion-rate performance, network parameters, and channel statistics estimated by the transport layer. To minimize the response time in interaction, the proposed mechanism is designed to have linear computational complexity. In addition, by integrating TCP-friendly congestion control into the system, the proposed mechanism achieves smooth performance over time as well as bandwidth fairness for co-existing applications in the network. For the sake of clarity, we focus on multi-resolution mesh geometry in our presentation. Nevertheless, the presented framework can

be extended to include textured models without difficulty, by integrating our joint mesh and texture optimization presented in the preceding chapter.

The presented work makes four contributions: *(i)* For a given distortion constraint, we derive a simplified expression (with assumptions) of the optimal FEC code that minimizes the expected transmission latency when combined with retransmissions; *(ii)* Observing a semi-infinite space for finding the theoretical optimal solution, we propose an extended steepest decent search algorithm, which quickly reaches the local optima in the solution space; *(iii)* Based on the results for distortion-constrained scenarios, we further develop a transmission mechanism for time-constrained scenarios, which significantly decreases the receiving distortion under a strict delay constraint; *(iv)* By integrating TCP-friendly congestion control into the system, the proposed mechanisms achieves smooth performance over time and bandwidth fairness for parallel applications.

The rest of the chapter is organized as follows. The major aspects of the proposed mesh transmission system are described in Section 4.2. In Section 4.3, we present a detailed study of the hybrid unequal error protection and selective retransmission mechanism. Test results in simulated network environments are also provided in Section 4.3. The chapter is summarized in Section 4.4.

4.2 System Overview

In this section, we provide an overview on the 3D mesh transmission system that is under consideration, and introduce the most important system parameters. As can be seen in Figure 19, the proposed system has three major components: a 3D mesh codec, a hybrid unequal error protection and selective retransmission mechanism at the application level (UEP/SR), and a transport protocol integrated with TCP-friendly rate control (TFRC). Next we briefly describe the system components and the corresponding parameters. Toward the end of the section, we present a simulation environment which will be used to test the performance of the proposed system.

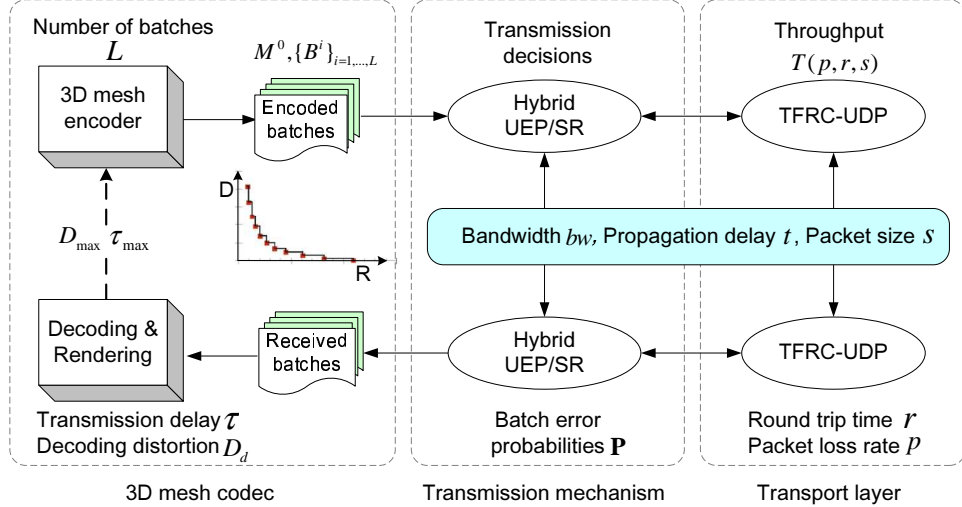


Figure 19: The proposed transmission system consists of three components.

4.2.1 The Codec Component

The 3D mesh codec implemented in the present work closely follows the *compressed progressive mesh* algorithm proposed by Pajarola and Rossignac [68], although the proposed transmission system is general enough to incorporate other multi-resolution compression methods, as will be discussed later in this section. A compressed mesh stream is composed of a base mesh, M^0 , and L enhancement batches, $\{B^i\}_{i=1,\dots,L}$, each of which encodes a set of vertex-split operations that transfer the triangulated mesh surface to a higher resolution¹. Sequentially, batch B^i refines the resolution of mesh M^{i-1} to higher resolution M^i until the full resolution is reached. Each resolution M^i , differs from the full resolution mesh by certain error (distortion) with a certain bit rate. As introduced in Chapter 2, such distortion can be measured by the root-mean-square surface distance between the meshes. To facilitate our discussion, we define a PSNR-like metric to reflect the quality degradation of a multi-resolution mesh. In particular, for a multi-resolution hierarchy $\{M^i\}_{i=0,\dots,L}$ where

¹For those methods that do not require grouping data into batches to achieve high compression ratio such as a wavelet-based scheme [53], we still consider this “batched” organization as a general representation of multi-resolution mesh data for two reasons: (i) isolated refinement operations in the multi-resolution mesh perform small and localized changes which normally do not result in perceivable distortion, and (ii) the refinement data will eventually be packetized when they are transmitted over the network.

M^0 is the base mesh and M^L is the full resolution, we define the quality of mesh M^i as

$$PSNR_m = 20 \log_{10} \frac{\mathcal{E}_{max}(M^0, M^L)}{\mathcal{E}_{rms}(M^i, M^L)} \text{ (dB)}, \quad (21)$$

where $\mathcal{E}_{max}()$ and $\mathcal{E}_{rms}()$ are the measured maximum and root-mean-square surface distances between the corresponding pairs of meshes, respectively. These distances can be calculated using the fast Metro tool [27] in practice.

In all present multi-resolution mesh coding schemes, the base mesh M^0 has both connectivity and geometry information and has to be correctly received or the rendering process will not be able to start (*infinite* distortion), while it in general has a fairly small fraction (1-2% or less) of the entire bitstream. Regarding this, to simplify notation (avoid differentiating the base mesh and the enhancement batches), and for the ease of performance presentation (avoid presenting infinite distortion), in the rest of the chapter we assume that a coarse representation has been received by the client through a reliable channel, and focus our discussion on the transmission of enhancement data.

4.2.2 The Transmission Component

The L enhancement batches, $\{B^i\}_{i=1,\dots,L}$, have different distortion-rate performance and are treated intelligently to achieve the best quality. Herein the best quality is interpreted as either minimized transmission latency τ under a distortion constraint D_{max} or minimized decoding distortion D_d within a limited time frame τ_{max} . Given network statistics reported by the transport layer, the sending application protects the batches with unequal-rate forward error correction (FEC) codes and/or retransmission according to their respective costs, and determines the optimal tradeoff that minimizes the transmission delay τ under the constraint of D_{max} or vice versa. This hybrid unequal error protection and selective retransmission mechanism (UEP/SR) is the core contribution of this work. To illustrate, consider the situation where a distortion constraint D_{max} is imposed. A set of χ batches, $\{B^1, B^2, \dots, B^\chi\}$, are selected under a criterion that their overall bit rate is minimal while their resulting resolution has a distortion level below the constraint, i.e., $D_d \leq D_{max}$. According to the discussion in Section 4.2.1, these selected batches need to be reliably transmitted, or the decoding distortion of the received mesh will not be satisfied (if any batches among the

selected ones are not correctly received, sending higher batches would not help decreasing the decoding distortion). Moreover, instead of solely using feedback-based retransmission to achieve reliable transmission, unequal-rate FEC codes are concurrently used to reduce the potential retransmission cost, and computation is performed to find the optimal FEC codes that minimize the expected transmission delay, $E(\tau)$, with the condition that undecodable batches among the selected ones will be retransmitted. Detailed discussion on the transmission mechanism is presented in Section 4.3.

The Reed-Solomon (RS) code is employed for FEC. We assume an (n, k) RS code with a block size of n packets (i.e., $n \times \text{packet-size}$ symbols) including k ($k < n$) information packets (code rate $\frac{k}{n}$). Considering a lossy channel, an RS code with $(n - k)$ parity-check packets will be able to recover the same number of packet losses. Hence the error probability of a batch using RS code (n, k) is

$$P(n, k) = Pr\{> (n - k) \text{ losses out of } n\}. \quad (22)$$

If the random packet loss is modeled by a Bernoulli process, for example, it is easy to get

$$\begin{aligned} P(n, k) &= 1 - Pr\{\leq (n - k) \text{ losses out of } n\} \\ &= 1 - \sum_{i=0}^{n-k} \binom{n}{i} p^i (1 - p)^{n-i} \end{aligned} \quad (23)$$

where p is the packet-loss rate. Similarly, the calculation of (22) can be performed for more sophisticated channel models, for which a few results have been found in the literature. For a two-state Markov channel model, for example, one may refer to the derivation presented in [46]. Such results can be easily integrated in the computation of transmission decisions, and the simulation results show that even the simple approximation (23) performs fairly well with the proposed algorithms.

4.2.3 The Transport Layer

UDP streams suffer from the lack of congestion control mechanism that prevents them from being reasonably fair² when competing for bandwidth with TCP-based traffic, as

²A flow is “reasonably fair” if its sending rate is generally within a factor of two of the sending rate of a TCP flow under the same condition [42].

TCP throttles its transmission rate against the network congestion. A TCP-friendly system should regulate its data sending rate according to the network condition, typically expressed in terms of the packet size s , the round-trip time r , and the packet-loss rate p . Ideally, the TCP throughput equation is suitable in describing the steady-state sending rate of a TCP-friendly flow [34]:

$$T = \frac{s}{r\sqrt{\frac{2p}{3}} + 4r(3\sqrt{\frac{3p}{8}})p(1 + 32p^2)}, \quad (24)$$

where a recommended choice of the retransmission timeout, $t_{RTO} = 4r$, has already been integrated.

A TCP-friendly rate control protocol (TFRC) based on Equation (24) has been specified in [42]. TFRC is a receiver-based mechanism designed for applications that use a fixed packet size and vary their sending rate in packets per second in response to congestion. The receiver periodically (once per round-trip time) sends feedback reports to the sender, containing the information that allows the sender to adjust its sending rate. Rate control using TFRC provides bandwidth fairness for parallel flows in the network as well as network stability, thus avoiding congestion collapse. In addition, TFRC's rate fluctuation is much lower than TCP, making it more appropriate for streaming applications that desire constant receiving quality. The TFRC implemented in the proposed mesh transmission system is slightly modified so that each data packet is acknowledged by the receiver [69]. The ACKs are used to infer the round-trip time and detect packet losses in order to perform selective retransmission for undecodable batches.

4.2.4 Simulation Environment

In this section, we present a simulation environment on which all our reported results will be based.

4.2.4.1 Network Environment

The simulation of the system is performed using ns-2 [92]. Figure 20 shows the simulated topology. The link between $R1$ and $R2$ is the bottleneck and the bandwidth is shared by f parallel flows. Random Early Detection (RED) [35] gateways are deployed at $R1$ and

$R2$ to improve both fairness and performance of the flows. All the parameters are listed in Table 3.

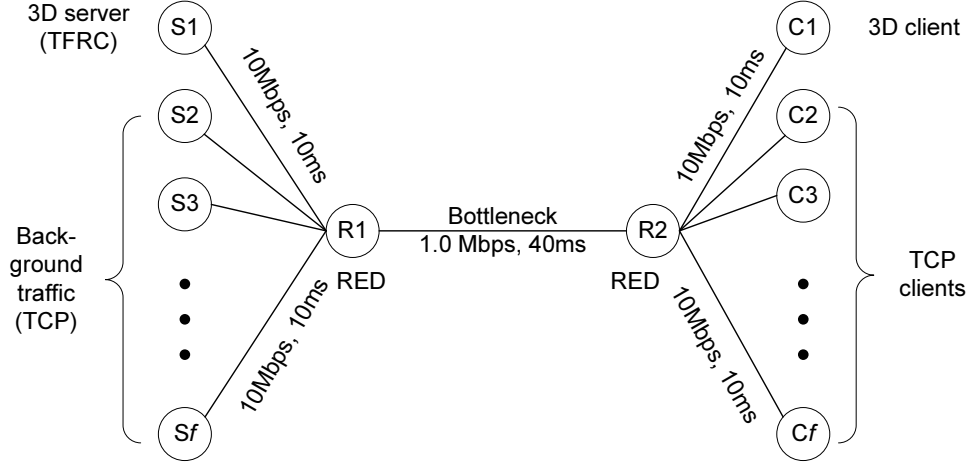


Figure 20: The simulation topology in ns-2. There are totally f parallel flows sharing the bottleneck bandwidth. All the background traffic is FTP traffic transmitted over TCP.

Table 3: A summary of the simulation parameters

Packet size	1000/500 bytes
ACK size	40 bytes
Bottleneck delay	40 ms
Bottleneck bandwidth	1 Mbps
Bandwidth of side links	10 Mbps
Delay of side links	10 ms
Simulation duration	100–200 sec
TCP window	20

RED parameters	
Min threshold	5 packets
Max threshold	$0.5 \times \text{Buffer size}$
Buffer size	1000 packets
q_weight	0.002

4.2.4.2 Test Models

Simulation results, unless otherwise noted, are obtained using the following models (courtesy of Cyberware, Inc): HORSE (39,698 faces), DINOSAUR (28,096 faces), VENUS HEAD (67,170 faces), and BALLJOINT (68,530 faces). All the models are quantized with 12 bits³ and are encoded to generate 10 enhancement batches. Their distortion-rate performance is plotted in Figure 21.

³In general, using 12-14 bits for quantizing 3D models with moderate dimensions introduces invisible distortion [68]. It is worthwhile to be pointed out, however, that the test models and the quantization parameter herein are selected for the illustration purpose. Our proposed transmission mechanism has no dependency on these parameters.

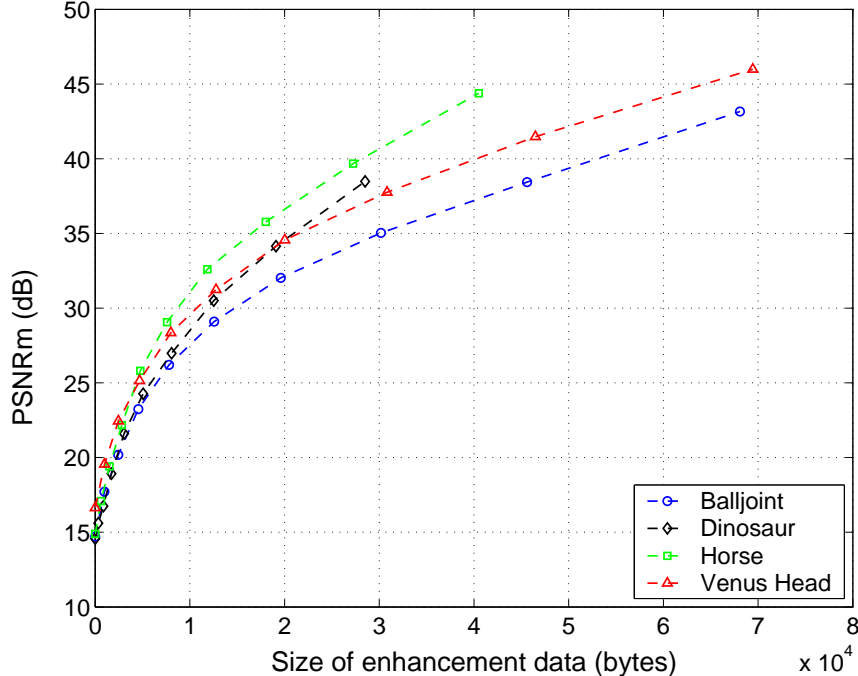


Figure 21: The distortion-rate performance of four test models.

4.3 Proposed Transmission Mechanism

In a hybrid TCP/UDP protocol as reviewed in Section 2.4, transmission latency is reduced by using UDP while distortion is reduced by TCP, i.e., feedback-based retransmission. Error control coding is an alternative to retransmission to improve the channel utilization and provide quality control. Although information theory has shown that sole use of feedback or error control coding can achieve the channel capacity, real systems often have constraints that invalidate ideal assumptions and call for joint considerations on both approaches. Typically, there are many situations where the receiving application has a maximally allowed level on either the transmission latency or the rendering distortion. In such cases, retransmission and error control coding should be treated intelligently so that the lowest distortion level or the minimum transmission latency is obtained while satisfying the corresponding constraint. In this section, we study in detail the hybrid unequal error protection and selective retransmission mechanism for multi-resolution meshes. This proposed mesh transmission mechanism is referred to as REP (retransmission and error protection) hereafter.

4.3.1 Distortion-Constrained Transmission

We first look at the problem of minimizing the transmission latency with a given distortion constraint, D_{max} . For a multi-resolution mesh stream, if a lower batch is not decoded correctly, perceptual quality cannot be improved by decoding higher batches. Henceforth, satisfying D_{max} is equivalent to selecting the least number of batches that need to be transmitted reliably. Denoted by χ , this least number of batches is expressed as

$$\chi = \min\{x | D_d(x) \leq D_{max}\}, \quad (25)$$

where $D_d(x)$ denotes the decoding distortion of the first x batches.

Once the number of selected batches is determined, the challenge is to find the optimal tradeoff between forward error protection and retransmission such that the user-requested distortion level can be satisfied with minimum transmission latency. To do so, an optimal distribution of parity-check packets for the selected batches is computed, taking into account their unequal distortion-rate performance and potential retransmission costs. The selected data is then protected with the corresponding parity-check packets, and is transmitted (with possible retransmissions) until the batches are correctly decoded. A simple illustration of this transmission mechanism is presented in Figure 22, where χ batches are selected to be reliably transmitted with a calculated distribution of the parity-check packets (noted with ‘RS’). Packets marked with ‘*’ indicate the losses occurred during transmission. As depicted in Figure 22, Batch 2 is not decodable due to packet losses and is retransmitted in order to satisfy the distortion constraint. After the χ batches are correctly received, the remaining data is transmitted (with or without error resilience, provided that the requested distortion level has been met).

Using \mathbf{s}_χ and \mathbf{c}_χ to represent the vectors (with length χ) of source and parity-check packets, respectively, for the selected χ batches, the expectation of the transmission latency τ is given as

$$\begin{aligned} E(\tau | \mathbf{s}_\chi, \mathbf{c}_\chi) &= E(\tau_{\mathbf{s}_\chi} + \tau_{\mathbf{c}_\chi} + \tau_R | \mathbf{s}_\chi, \mathbf{c}_\chi) \\ &= E(\tau_{\mathbf{s}_\chi} + \tau_{\mathbf{c}_\chi} | \mathbf{s}_\chi, \mathbf{c}_\chi) + E(\tau_R | \mathbf{s}_\chi, \mathbf{c}_\chi), \end{aligned} \quad (26)$$

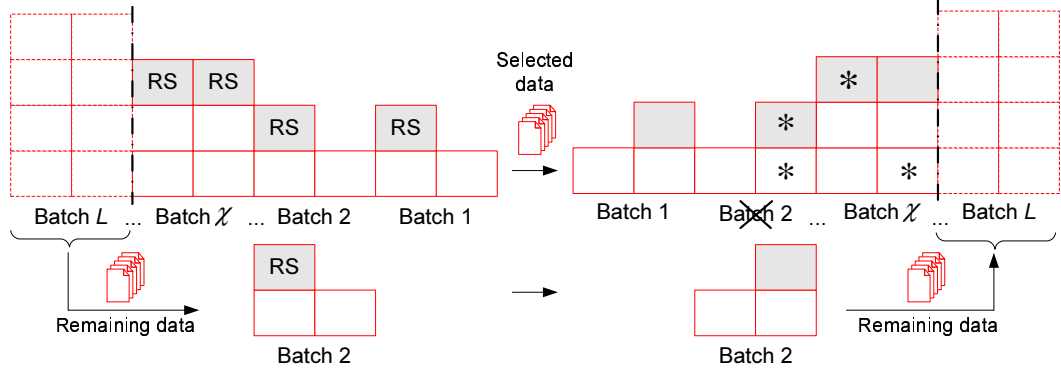


Figure 22: A simple demonstration on the proposed hybrid UEP/SR method, where the packets marked with ‘RS’ are parity-check packets and ‘*’ indicates packet losses during transmission. Batch 2 is retransmitted because it is among the selected χ batches and has not been correctly received.

where $E()$ represents the probabilistic expectation, and τ_R denotes the total latency incurred by retransmission; τ_{s_χ} , τ_{c_χ} denote the transmission costs for all the source and parity-check packets, respectively, in the selected batches, i.e.,

$$\tau_{s_\chi} = |\tau_{s_\chi}| = \sum_{i=1}^{\chi} \tau_{s_\chi}(i), \text{ and } \tau_{c_\chi} = |\tau_{c_\chi}| = \sum_{i=1}^{\chi} \tau_{c_\chi}(i),$$

where $\tau_{s_\chi}(i)$ and $\tau_{c_\chi}(i)$ correspond to the transmission cost for the i -th batch. Given the RS codes $(\mathbf{s}_\chi, \mathbf{c}_\chi)$, the packet-loss rate p and the round-trip time r , the steady-state transmission throughput T is described by Equation (24), and τ_{s_χ} , τ_{c_χ} can be considered as constant-value vectors. With the notations $\boldsymbol{\tau}_0 = (\tau_{s_\chi} + \tau_{c_\chi})$ and $\tau_0 = \tau_{s_\chi} + \tau_{c_\chi} = |\boldsymbol{\tau}_0| = \mathbf{1} \cdot \boldsymbol{\tau}_0$, we have $E(\tau_{s_\chi} + \tau_{c_\chi} | \mathbf{s}_\chi, \mathbf{c}_\chi) = E(\tau_0)$, and Equation (26) is further expressed as

$$\begin{aligned} E(\tau | \mathbf{s}_\chi, \mathbf{c}_\chi) &= E(\tau_0) + E(\tau_R | \mathbf{s}_\chi, \mathbf{c}_\chi) \\ &= |\boldsymbol{\tau}_0| + \mathbf{P}(\mathbf{s}_\chi + \mathbf{c}_\chi, \mathbf{c}_\chi) \cdot [\boldsymbol{\tau}_0 + E(\tau_R | \mathbf{s}_\chi, \mathbf{c}_\chi)] \\ &= \mathbf{1} \cdot \boldsymbol{\tau}_0 + \mathbf{P}(\mathbf{s}_\chi + \mathbf{c}_\chi, \mathbf{c}_\chi) \cdot \boldsymbol{\tau}_0 + \mathbf{P}^2(\mathbf{s}_\chi + \mathbf{c}_\chi, \mathbf{c}_\chi) \cdot \boldsymbol{\tau}_0 \\ &\quad + \cdots + \mathbf{P}^n(\mathbf{s}_\chi + \mathbf{c}_\chi, \mathbf{c}_\chi) \cdot \boldsymbol{\tau}_0 + \cdots \\ &= \frac{1}{1 - \mathbf{P}(\mathbf{s}_\chi + \mathbf{c}_\chi, \mathbf{c}_\chi)} \cdot \boldsymbol{\tau}_0. \end{aligned} \tag{27}$$

Note that in (27), $E(\tau_R | \mathbf{s}_\chi, \mathbf{c}_\chi)$ was expanded as a summation of recursive retransmission costs multiplied by the corresponding probabilities, with an assumption that all processing cost upon data loss is ignorable. $\mathbf{P}(\mathbf{s}_\chi + \mathbf{c}_\chi, \mathbf{c}_\chi)$ is the vector of batch error probabilities

with each element computed according to (22), and $\mathbf{P}^n(\mathbf{s}_\chi + \mathbf{c}_\chi, \mathbf{c}_\chi)$ is defined as

$$\mathbf{P}^n(\mathbf{s}_\chi + \mathbf{c}_\chi, \mathbf{c}_\chi) = \mathbf{P}^{(n-1)}(\mathbf{s}_\chi + \mathbf{c}_\chi, \mathbf{c}_\chi) \cdot \mathbf{P}(\mathbf{s}_\chi + \mathbf{c}_\chi, \mathbf{c}_\chi), \quad n > 1.$$

The optimal distribution of parity-check packets, $\mathbf{c}_{\chi opt}$, is then given by

$$\mathbf{c}_{\chi opt} = \arg \min_{\mathbf{c}_\chi} E(\tau | \mathbf{s}_\chi, \mathbf{c}_\chi). \quad (28)$$

Equations (27-28) with (25) provide the theoretical solution to the problem. Yet an operational solution should also take into account the computation complexity as exhaustive search will not be feasible (the choice of \mathbf{c}_χ in (28) is arbitrary and therefore the solution space is not a close space). To accommodate real-time applications, a generalized *steepest decent* algorithm is used in this work to find the optimal (or a possibly suboptimal) solution. This fast algorithm starts with $\mathbf{c}_\chi = \emptyset$, i.e., no parity-check packets are added at the initial point. At each iteration, the algorithm adds one more parity-check packet to either one of the χ batches, which results in χ possibilities. It then finds amongst the one that decreases the expected delay $E(\tau | \mathbf{s}_\chi, \mathbf{c}_\chi)$ the most, or stops the computation if $E(\tau | \mathbf{s}_\chi, \mathbf{c}_\chi)$ increases for all cases, where $E(\tau | \mathbf{s}_\chi, \mathbf{c}_\chi)$ is calculated using (27). Figure 23 presents a simple illustration on this fast algorithm. The double-circled nodes indicate the searching path of the steepest decent algorithm, which have smaller expected transmission delays than their parents and siblings; the solid one represents the optimal operating point that has the minimum expected transmission delay, meaning that all its descendants (skipped in the plot) have larger expected delays. To simplify notation, the source vector \mathbf{s}_χ is ignored in Figure 23 and only the vector of parity-check packets is indicated. From Figure 23, it is apparent that the steepest decent algorithm has a linear computational complexity $O(K \cdot \chi)$, or more generally, $O(K \cdot L)$, where K is the total number of parity-check packets that are added (the depth of the tree) and L is the number of generated batches (the maximum width of the tree).

Figure 24 presents a further illustration of the algorithm. For the HORSE model (with 10 batches), it shows the computed minimum expected transmission delay with respect to the total number of parity-check packets that are added to the batches. In the computation,

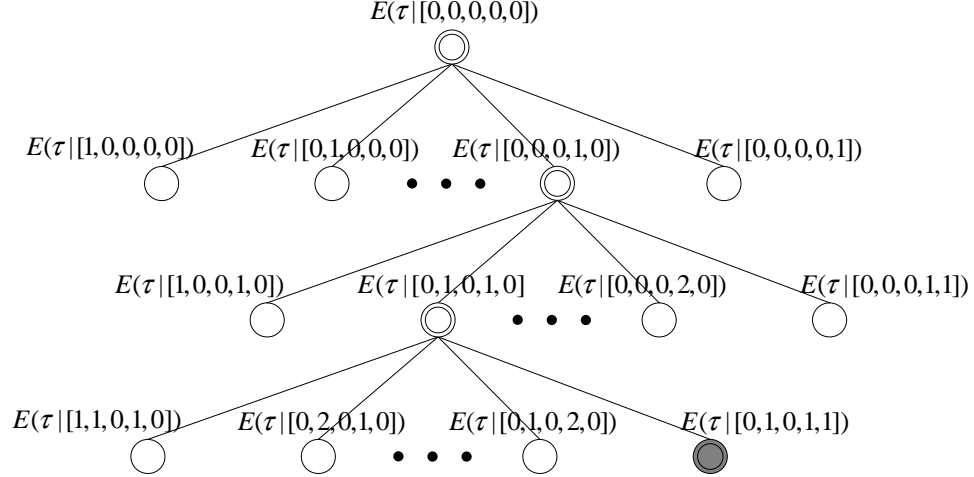


Figure 23: An illustration of the steepest decent algorithm for finding $\mathbf{c}_{\chi opt}$. The double-circled nodes indicate the searching path of the steepest decent algorithm, and the solid one represents the optimal operating point which has the minimum expected transmission delay.

the packet size, the packet-loss rate, and the round-trip time are set to be $s = 500$ bytes, $p = 2\%$, and $r = 100$ ms, respectively. As can be seen in Figure 24, adding parity-check packets at the beginning quickly decreases the expected transmission delay, which reaches a minimal at a certain point ($K = 9$ in this case). Thereafter, adding more parity-check packets gradually increases the expected transmission latency, implying that the additional transmission for sending parity-check packets has exceeded the potential retransmission cost.

To study the performance of the algorithm for different network conditions, we compute the optimal number of parity-check packets with respect to various packet-loss rates⁴ and for different models. The results are shown in Figure 25. The total number of parity-check packets that are needed for the minimum expected transmission delay is plotted with the packet-loss rate varying from 1% to 15%. As one may expect, when the network becomes heavily congested (higher packet-loss rates), the retransmission cost increases, and Figure 25 shows that more parity-check packets are needed to minimize the expected transmission latency. An approximately linear relation is observed between the number of parity-check

⁴We keep the same round-trip time in all the computations based on the following fact: although the packet-loss rate increases quickly as the network becomes more congested, the round-trip time is observed with slight variation in all cases because of the regulation by RED gateways and the rate control mechanism.

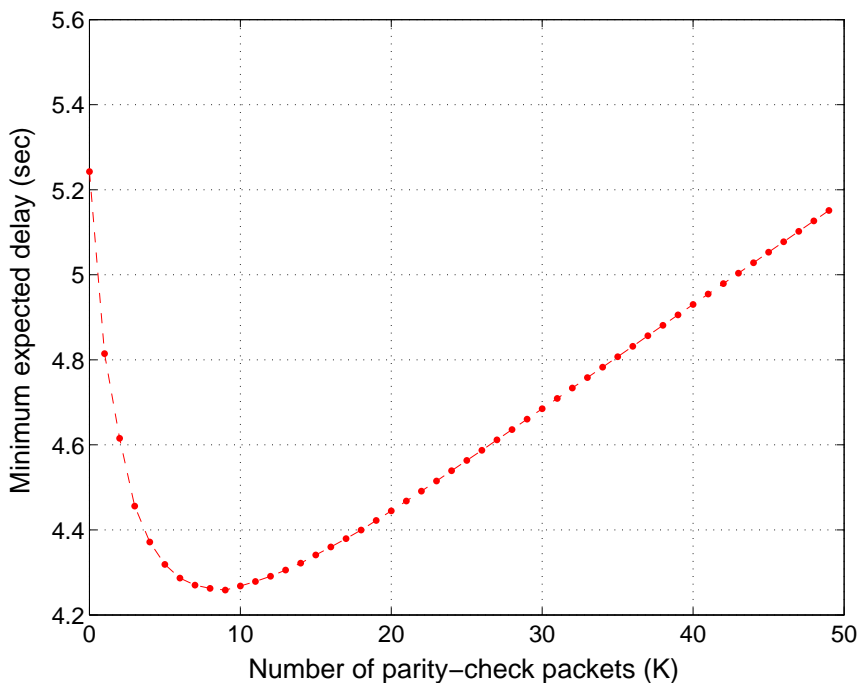


Figure 24: The computed minimum expected delay versus the number of parity-check packets that are added. The computation is performed for the HORSE model with a setting of these parameters: packet size $s = 500$ bytes, packet-loss rate $p = 2\%$, and round-trip time $r = 100$ ms.

packets (K) and the packet-loss rate (p), which implies that the computation complexity is linearly increased as the packet-loss rate gradually increases, according to the expression $O(K \cdot L)$. In Figure 25, it is noted that the VENUS HEAD model requires more parity-check packets (larger K) than the HORSE and DINOSAUR models with the same packet-loss rate, simply because its batches are encoded with larger sizes. In general, batch sizes can be reduced while a higher number of resolutions (larger L) will occur with potential loss/reduction in compression efficiency [79].

We now present simulation results of the proposed hybrid mechanism (REP) for distortion-constrained transmission, and compare it with the 3TP [10]. For fair comparison, in 3TP, we replace UDP with TFRC so the transport layer of 3TP is also TCP-friendly. Thus, in both REP and 3TP, the remaining data other than the selected batches is transmitted under the same mechanism, i.e., sole TFRC-UDP without FEC or retransmission. For this sake, in our presented results, we compare *only* the transmission delays for the batches that

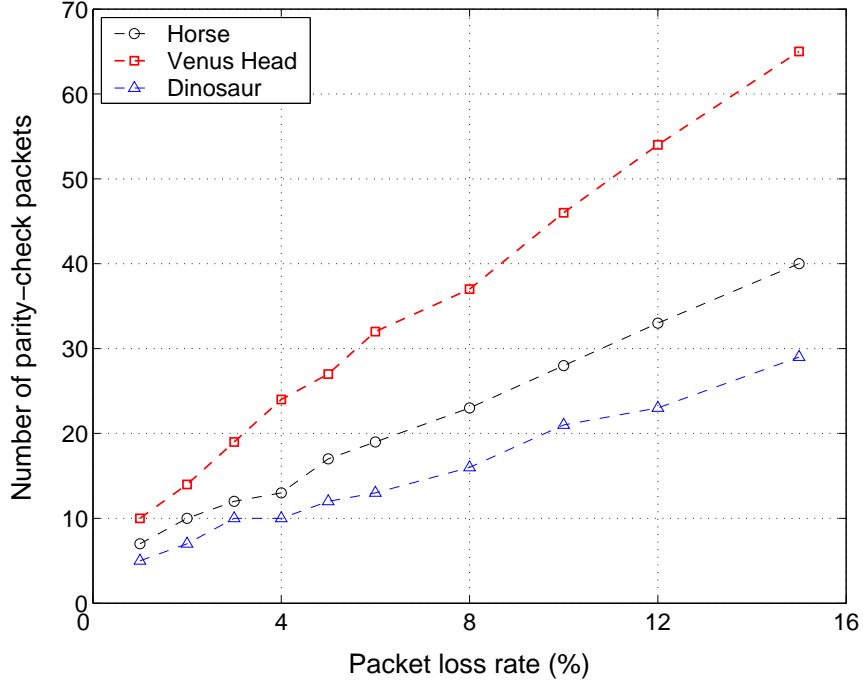


Figure 25: The number of parity-check packets versus the packet-loss rate. The packet size and the round-trip time in the computation are set to be $s = 500$ bytes and $r = 100$ ms, respectively.

are selected to be transmitted reliably under a given distortion constraint, and are treated differently in 3TP and REP (in particular, 3TP uses TCP and REP uses hybrid UEP/SR).

Figures 26(a-d) plot the transmission delay with respect to different numbers of batches that are selected to be reliably transmitted, determined upon the distortion constraint. The delay results have been averaged over all received meshes during the simulation period. The number of parallel flows in the network is set to be $f = 12$, which results in a moderate network congestion situation as reflected by the measured packet-loss rate ($p \approx 6-7\%$). The packet sizes are $s = 1000$ bytes for Figures 26(a-b) and $s = 500$ bytes for Figures 26(c-d), respectively. As can be seen from these plots, REP considerably reduces the transmission latency compared to 3TP. Specifically, when the full resolution model is required by the receiving application, all the enhancement batches need to be reliably transmitted ($\chi = 10$). Simply, 3TP returns to be sole TCP in this case. In contrast, REP still achieves substantial delay reduction by using hybrid UEP/SR. For example, in Figures 26(a-b) where the packet size is $s = 1000$ bytes, 27% reduction in the average delay is observed for the HORSE model

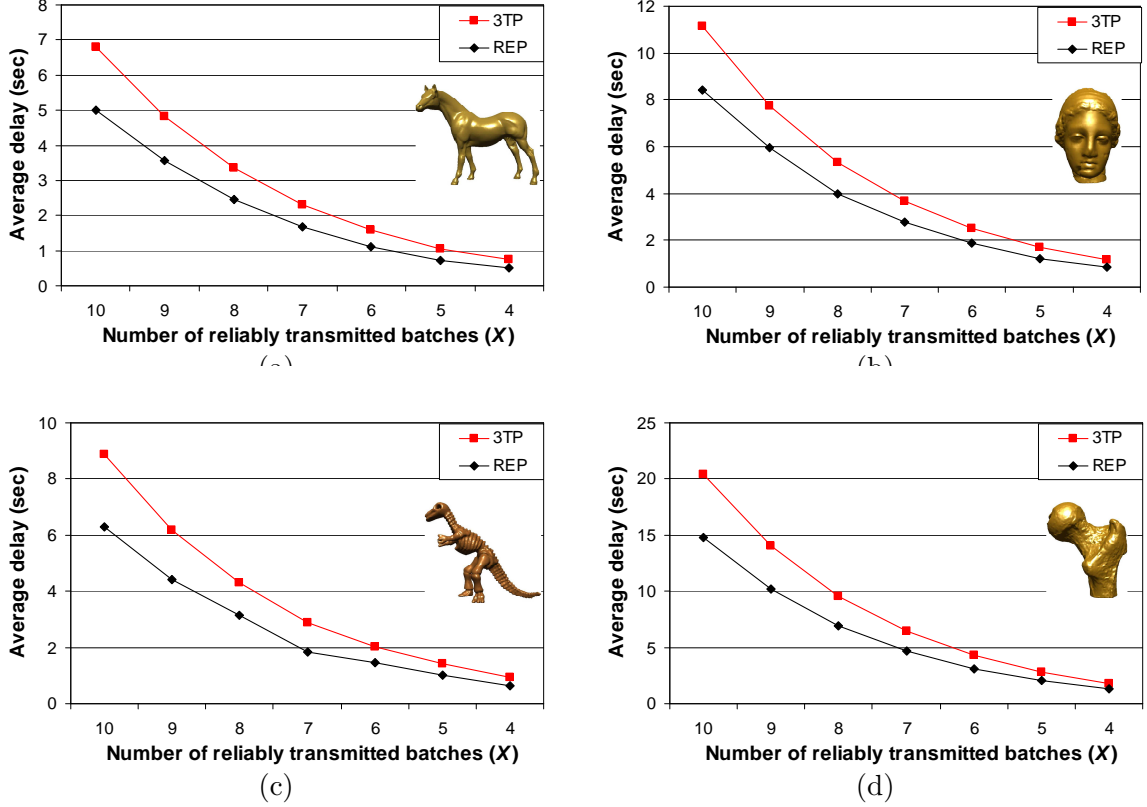


Figure 26: Average transmission delays for sending various portions of data reliably using REP and 3TP, respectively. The number of parallel flows in the network is $f = 12$, and the packet sizes are $s = 1000$ bytes in (a-b) and $s = 500$ bytes in (c-d).

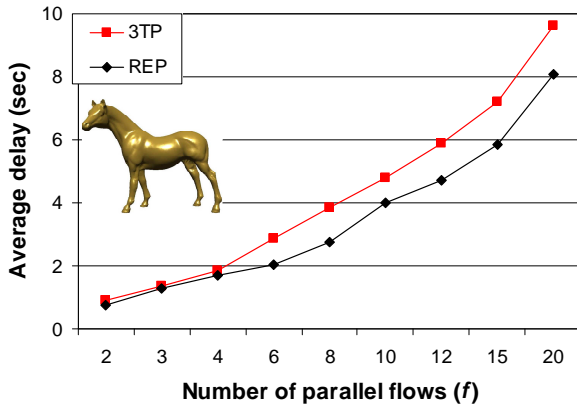
and 25% is observed for the VENUS HEAD model.

In Figure 26(a-d), it is observed that as the portion of reliably transmitted data becomes smaller (i.e., smaller χ), which corresponds to a lower constraint on decoding distortion, the performance of REP gradually merges to 3TP. This behavior is anticipated because as the distortion constraint is lowered, both mechanisms transmit most of the data using TFRC-UDP without FEC or retransmission. Yet, considering a distributed online presentation of 3D scenes, a small reduction in the average delay provided by REP may still result in significant improvement on overall performance when a number of 3D meshes are transmitted on demand.

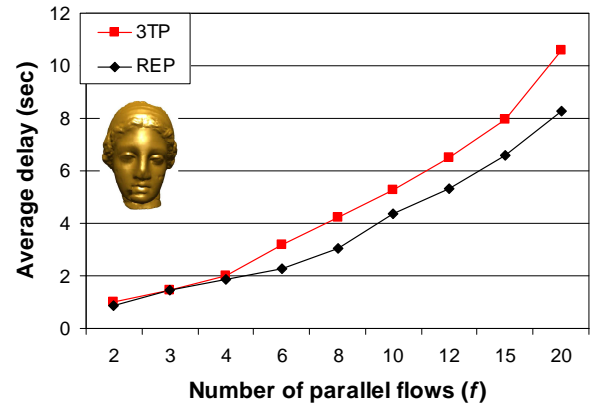
In Figure 27, the variation of the average delay under different network congestion situations is investigated. Suppose an upper bound of the decoding distortion, $D_{max} \geq 36$ dB, is requested for the models to be rendered. To satisfy D_{max} , $\chi = 7$ to 9 enhancement

batches⁵ are selected to be reliably transmitted according to Equation (28) and Figure 21. Figures 27(a-d) present the delay results of REP and 3TP for the selected batches for the test models. It is shown that REP outperforms 3TP under most of the network conditions, and the gain is especially significant when the network encounters moderate or heavy congestion ($f \geq 6$ in the figures). For the network with light traffic load ($f \leq 4$), REP and 3TP perform similarly. When only two flows exist in the network and the packet size is $s = 1000$ bytes (Figure 27(c-d)), one may notice that 3TP provides slightly better results than REP. This is resulted from TCP's advantage of quick adaptation to the changes in available bandwidth compared with TFRC [42]. A more complete numerical comparison of REP with 3TP in various network situations and with different parameters is presented in Table 4, which confirms our conclusions obtained from Figure 26 and Figure 27.

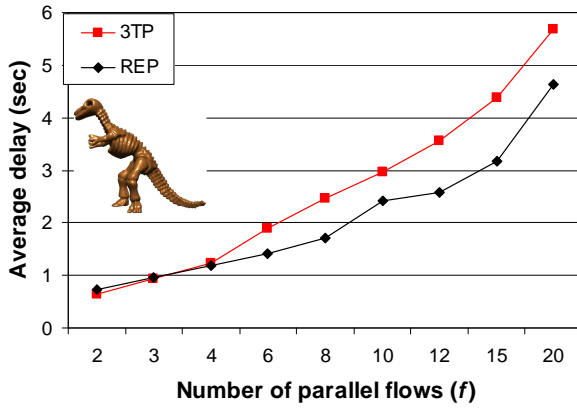
⁵This number depends on the distortion-rate curve of the model.



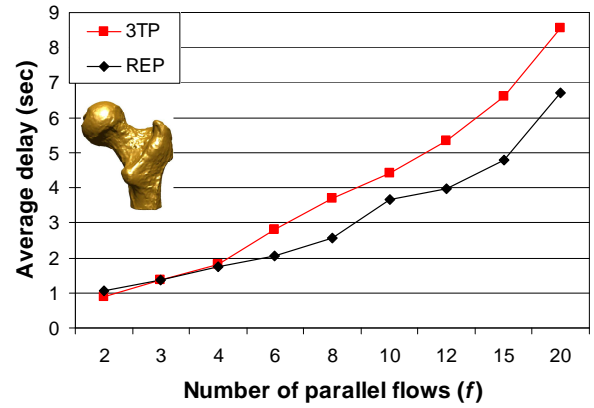
(a) $\chi = 8$



(b) $\chi = 7$



(c) $\chi = 9$



(d) $\chi = 8$

Figure 27: Delay results when the distortion constraint D_{max} is 36 dB. The packet sizes are $s = 500$ bytes in (a-b) and $s = 1000$ bytes in (c-d).

Table 4: Comparative delay results (in seconds) for the selected batches of the VENUS HEAD model

Packet size: $s = 1000$ bytes												
f	$\chi = 10$		$\chi = 9$		$\chi = 8$		$\chi = 7$		$\chi = 6$		$\chi = 5$	
	3TP	REP	3TP	REP	3TP	REP	3TP	REP	3TP	REP	3TP	REP
2	1.78	2.12	1.26	1.48	0.89	1.05	0.64	0.74	0.47	0.53	0.34	0.37
3	2.78	2.84	1.95	1.98	1.37	1.38	0.97	0.98	0.69	0.69	0.49	0.48
4	3.72	3.60	2.59	2.51	1.81	1.75	1.27	1.22	0.89	0.86	0.62	1.59
6	5.81	4.28	4.04	2.96	2.81	2.07	1.95	1.45	1.36	1.02	0.93	0.69
8	7.66	5.36	5.32	3.72	3.68	2.56	2.55	1.78	1.76	1.24	1.19	0.85
10	9.22	7.43	6.39	5.23	4.42	3.65	3.04	2.56	2.10	1.72	1.41	1.18
12	11.2	8.44	7.73	5.95	5.34	3.98	3.67	2.79	2.52	1.88	1.69	1.23
15	13.8	9.99	9.57	6.54	6.59	4.78	4.52	3.40	3.09	2.28	2.06	1.51
20	18.0	13.6	12.5	9.81	8.56	6.72	5.86	4.64	4.00	3.18	2.65	2.20
Packet size: $s = 500$ bytes												
f	$\chi = 10$		$\chi = 9$		$\chi = 8$		$\chi = 7$		$\chi = 6$		$\chi = 5$	
	3TP	REP	3TP	REP	3TP	REP	3TP	REP	3TP	REP	3TP	REP
2	3.04	2.57	2.12	1.78	1.46	1.24	1.01	0.85	0.70	0.58	0.48	0.40
3	4.53	4.38	3.14	3.09	2.16	2.13	1.47	1.45	1.00	0.98	0.68	0.66
4	6.29	5.67	4.35	3.99	2.97	2.73	2.02	1.86	1.36	1.26	0.92	0.84
6	9.99	6.98	6.89	4.85	4.70	3.39	3.17	2.28	2.12	1.54	1.40	1.02
8	13.5	9.35	9.27	6.65	6.30	4.51	4.24	3.04	2.83	2.03	1.86	1.34
10	16.7	13.4	11.5	9.21	7.83	6.40	5.26	4.37	3.50	2.96	2.29	1.94
12	20.7	16.4	14.2	11.4	9.65	7.99	6.48	5.32	4.29	3.53	2.80	2.35
15	25.4	20.1	17.5	14.0	11.9	9.96	7.94	6.58	5.25	4.62	3.42	2.99
20	34.1	26.6	23.4	18.6	15.9	12.3	10.6	8.28	7.00	5.73	4.54	3.77

4.3.2 Delay-Constrained Transmission

So far, we detailed the proposed mechanism for distortion-constraint applications. In this part, we consider a time-critical situation where a delay constraint τ_{max} is imposed by the application and a minimum decoding distortion is desired. A hybrid TCP/UDP protocol is not suitable for this problem because of the automatic retransmission behavior of TCP. Although having the application conservatively send a small portion of data via TCP and the remainder via UDP may possibly satisfy the delay constraint, it is difficult to perform the *conservative* selection in TCP because it halves the transmission rate in response to a single packet drop, which results in frequent abrupt changes in throughput. In contrast, TFRC has lower variation of throughput over time [42], making it favorable in the delay-constraint application that is under consideration.

Utilizing the smooth sending rate variations in TFRC, we have developed in REP an operational algorithm with low computation complexity for delay-constrained transmission. In particular, for a given time frame, the algorithm selects the maximum number of batches to transmit provided that their minimum expected transmission latency does not exceed the allowable time frame. The minimum expected transmission latency is achieved by finding the optimal tradeoff between UEP and retransmission, and is computed using the proposed steepest decent algorithm as shown in the previous subsection. We consider the selected data (with the corresponding RS codes) as a *greedy conservative* solution at each transmission opportunity. After sending the selected source data and the corresponding parity-check packets, the server updates its sending buffer according to the feedback. The same procedure is repeated before all the batches are correctly received or the deadline has been reached.

We denote the source bit rate of the L batches by vector \mathbf{s} , and use $\mathbf{s}^{(i)}$, $\tau^{(i)}$ to represent the source vector and the time instant for i -th transmission opportunity, respectively. Major steps of the algorithm are then presented as follows.

Note that in step (iii), $\chi \in [1, L]$ is the maximum number of batches with corresponding RS codes $(\mathbf{s}_\chi^{(i)}, \mathbf{c}_{\chi opt}^{(i)})$ whose expected transmission latency falls within the remaining time frame, $(\tau_{max} - \tau^{(i-1)})$, multiplied by a fraction factor α . Ideally, we have $\alpha = 1.0$ given that

-
- (i) $i = 0$, $\mathbf{s}^{(0)} = \mathbf{s}$, $\tau^{(0)} = 0$;
 - (ii) $i = i + 1$;
 - (iii) update $\mathbf{s}^{(i-1)}$ to $\mathbf{s}^{(i)}$ by removing those batches that have been acknowledged;
 - (iv) if $\tau^{(i-1)} < \tau_{max}$, find $(\chi, \mathbf{c}_{\chi opt}^{(i)})$ such that the following conditions are satisfied:

$$\left\{ \begin{array}{l} 1 \leq \chi \leq L, \\ E(\tau^{(i)} | \mathbf{s}_{\chi}^{(i)}, \mathbf{c}_{\chi opt}^{(i)}) < \alpha \cdot (\tau_{max} - \tau^{(i-1)}), \quad 0 < \alpha \leq 1, \\ \text{if } \chi < L, \text{ then } E(\tau^{(i)} | \mathbf{s}_{\chi+1}^{(i)}, \mathbf{c}_{\chi+1, opt}^{(i)}) > \alpha \cdot (\tau_{max} - \tau^{(i-1)}), \end{array} \right. \quad (29)$$

where $\mathbf{c}_{\chi opt}^{(i)} = \arg \min E(\tau | \mathbf{s}_{\chi}^{(i)}, \mathbf{c}_{\chi}^{(i)})$ is given by Equations (27-28);

- (v) send $(\mathbf{s}_{\chi}^{(i)} + \mathbf{c}_{\chi opt}^{(i)})$ using TFRC;
 - (vi) if τ_{max} has not been reached, set $\tau^{(i)}$ with the current time; REDO (ii).
-

smooth transmission throughput is provided by TFRC, in which case satisfaction of (29) can be considered as the greediest conservative selection of data that is to be sent. In practice, α may be chosen to be close (but not equal) to 1.0 for first few transmissions in order for the application to be more robust to abrupt drops of data sending rate occurred during transmission, thus avoiding large variation of the decoding distortion among received 3D models. The particular choice of the factor α is a design issue. In our simulation, we have found that a simple choice with $\alpha = 0.8$ for the initial transmission ($i = 1$) and $\alpha = 1.0$ for the rest transmissions ($i > 1$) works fairly well.

Performance of the above algorithm has been investigated compared with a heuristic mechanism that is also based on TFRC. It utilizes the time frame maximally by performing a simple scheme: the maximum number of batches that satisfy the timing condition are first selected based on the transmission throughput; the remaining time slot (if any) is then filled with the first few batches that have the least bit rate but are generally most important. Pseudo-code of this simple Filling Algorithm (FA) is shown in the following, where T is the transmission throughput and $R(\chi)$ denotes the total bit rate of χ batches, $\{B^1, B^2, \dots, B^{\chi}\}$.

```

while  $\tau < \tau_{max}$  and  $\mathbf{s}_{\chi} \neq \emptyset$  do
  find the maximum  $\chi \in [1, L]$  s.t.  $\frac{R(\chi)}{T} \leq \tau_{max} - \tau$ ;
  send  $\mathbf{s}_{\chi} = \{B^1, B^2, \dots, B^{\chi}\}$  using TFRC;
   $\tau = \tau + \frac{R(\chi)}{T}$ ;
end while

```

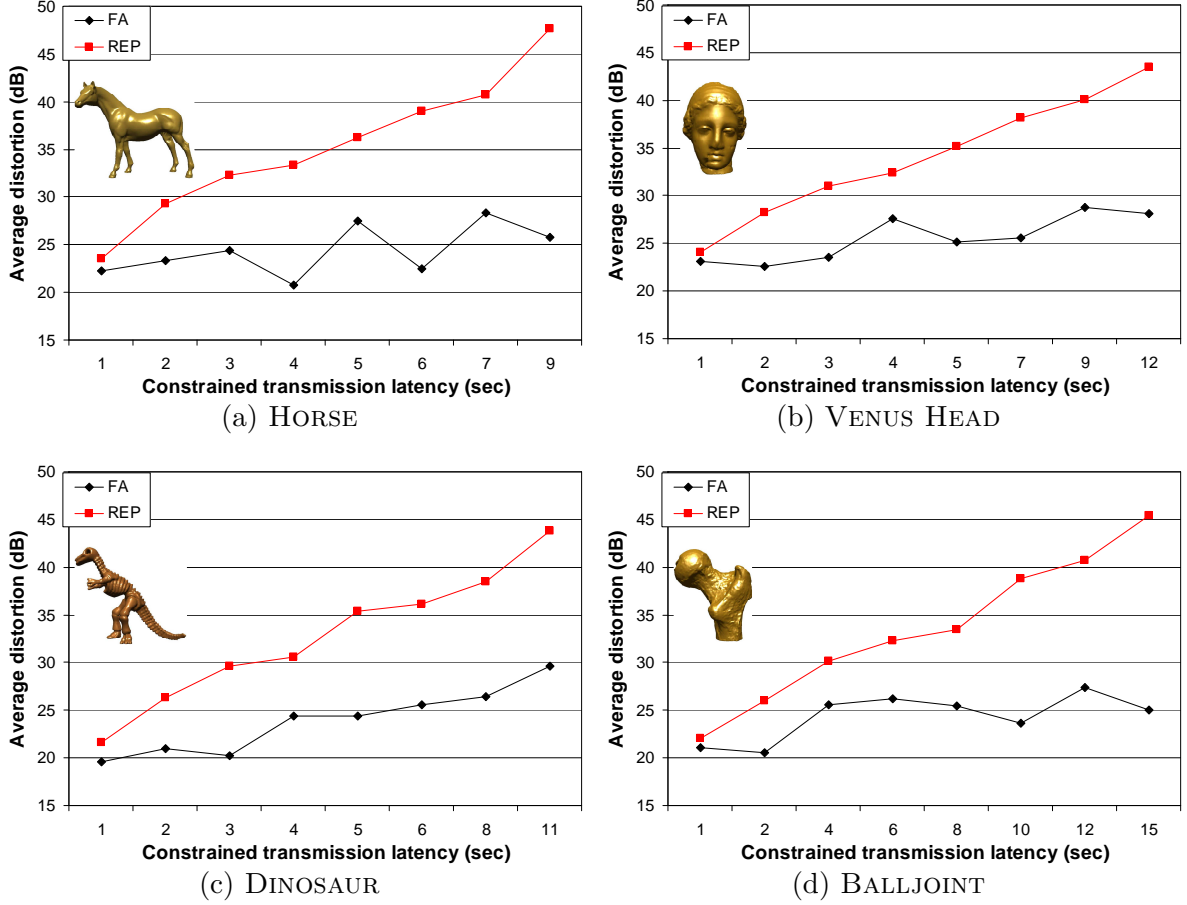
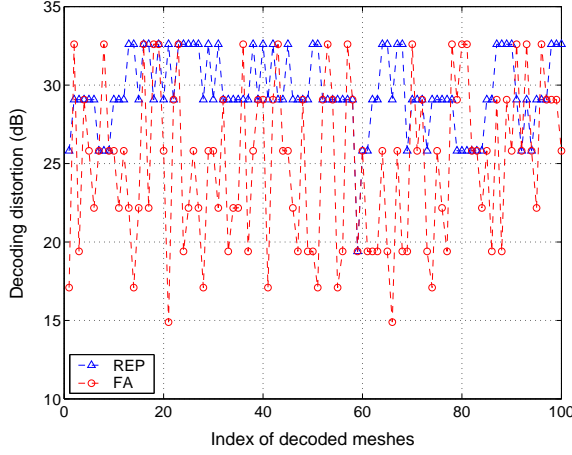


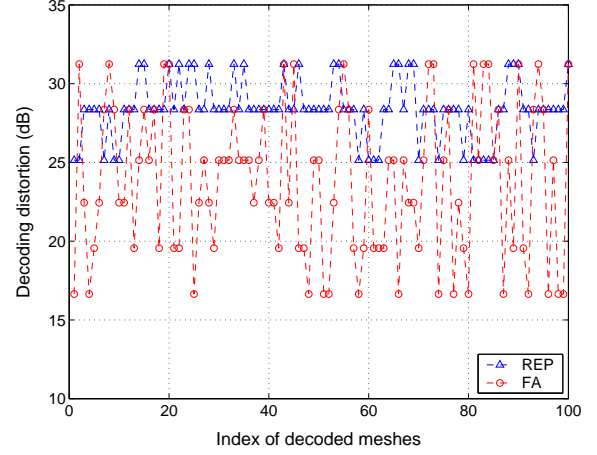
Figure 28: Average decoding distortion of received meshes for various delay constraint in a typical network situation: the number of parallel flows is $f = 12$, the packet size is $s = 500$ bytes, and the resulting packet-loss rate is $p = 5.5\%$.

Simulation results are presented in Figure 28 and Figure 29, where the number of parallel flows in the simulated network is set to be $f = 12$ (moderate traffic load) and the packet size is $s = 500$ bytes. Figures 28(a-d) plot the decoding distortion averaged over all received meshes with respect to various delay constraints, i.e., different τ_{max} . One can see that REP greatly improves the decoding quality within the same time frame compared to the FA algorithm. In addition, as τ_{max} increases, the decoding quality provided by REP quickly arises, whereas for FA, the resulting effect is observed to be unpredictable. It is a natural consequence of random losses as in the heuristic scheme, all the enhancement batches, except those few ones that are used to fill the the remaining time slot, are treated equally without error resilience.

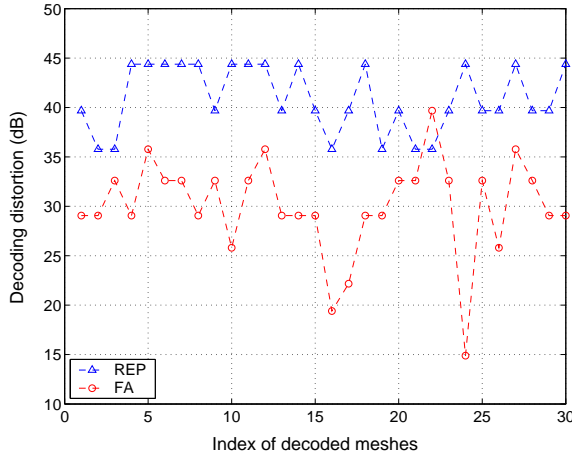
In Figure 29, we trace the decoding distortion of every received HORSE or VENUS HEAD



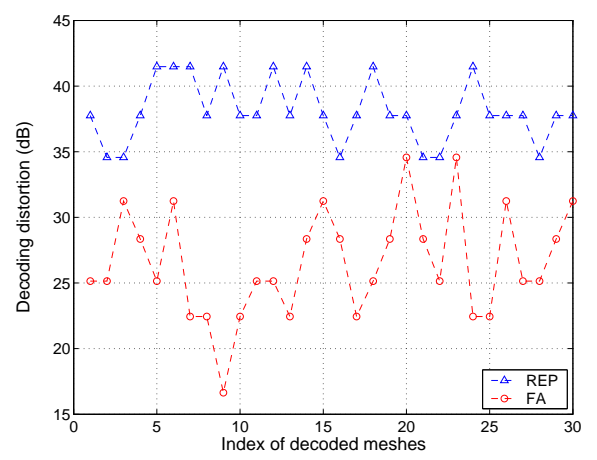
(a) HORSE, $\tau_{max} = 2$ sec



(b) VENUS HEAD, $\tau_{max} = 2$ sec



(c) HORSE, $\tau_{max} = 7$ sec



(d) VENUS HEAD, $\tau_{max} = 7$ sec

Figure 29: Traces of the decoding distortion for each received mesh: (a-b) $\tau_{max} = 2$ sec; (c-d) $\tau_{max} = 7$ sec.

mesh for two individual cases: (a-b) $\tau_{max} = 2$ seconds, and (c-d) $\tau_{max} = 7$ seconds. The (blue) triangles-marked curve denotes the $PSNR_m$ values of decoded meshes in REP, while the (red) circles-marked corresponds to the FA mechanism. As expected, REP has much lower variation of the decoding quality in addition to a lower average distortion level (higher $PSNR_m$) than FA. Difference in the average quality is especially significant when larger transmission latency is allowed. For example, in Figures 29(c-d), REP provides a median-level quality of 39.68 dB (for HORSE) or 37.76 dB (for VENUS HEAD), whereas FA has only 29.07 dB or 28.35 dB, correspondingly. Such difference in the average quality has actually been shown in Figure 28. For the test models, average quality near to 40 dB is



(a) REP: 39.68 dB



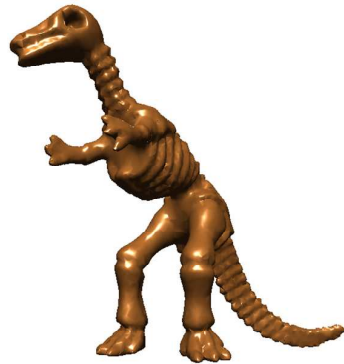
(b) FA: 29.07 dB



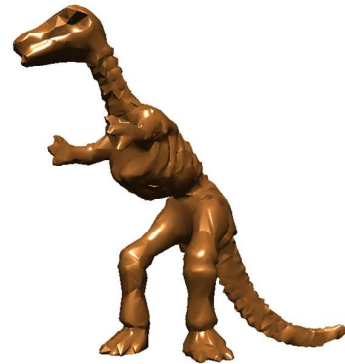
(c) REP: 37.76 dB



(d) FA: 28.35 dB



(e) REP: 38.49 dB



(f) FA: 26.98 dB



(g) REP: 38.43 dB



(h) FA: 26.20 dB

Figure 30: Rendering quality near to 40 dB is obtained by REP when $\tau_{max} = 7$ to 10 seconds, with a 10 dB or higher gain compared to the simple heuristic (FA) under the same situation.

typically observed for REP when $\tau_{max} = 7$ to 10 seconds, with a 10 dB or higher gain over the comparing heuristic. The perceptual quality difference of the rendered models is as significant as reflected by the quantities, captured by Figures 30(a-d). All the above results, along with those presented in Section 4.3.1, have verified REP a successful transmission mechanism for streaming 3D meshes over a lossy network.

4.4 *Summary*

In this chapter, we developed a hybrid unequal error protection and selective retransmission mechanism (REP) for streaming scalably coded 3D models over lossy networks. We considered both distortion- and time-constrained scenarios. To minimize the transmission latency under a distortion constraint, a portion of multi-resolution mesh data is selected to be reliably transmitted. Then, the best tradeoff between forward packet-loss resilience and retransmission is calculated in linear time, using a steepest decent algorithm. For time-constrained applications, we developed an operational algorithm based on the proposed steepest decent algorithm to maximize the model quality. The proposed algorithms have low computational complexity, making them attractive for distributed graphics applications.

TCP-friendly rate control (TFRC) is integrated in REP, which regulates the data sending rate in response to network congestion so that REP fairly competes for bandwidth with co-existing applications in the network. In addition, because TFRC has much lower variations of throughput over time than TCP, it is a favorable mechanism for REP to achieve smooth performance in interactive and time-critical situations.

During the development of REP, packet losses are assumed to be primarily caused by buffer overflow (congestion). For wireless links, where packets can be corrupted by channel errors at the physical layer, packet losses caused by bit errors should be considered. Not only rate control needs to be elaborated accordingly, but also an accurate model of the packet-loss process becomes important. Furthermore, the feedback channel is also error-prone because of the fading effect, which complicates the process of finding optimal transmission decisions. All these features need to be addressed to improve the performance of streaming 3D data over wireless channels.

Our research efforts so far have been focused on the scalable coding and transmission of individual 3D models. Three-dimensional graphical scenes, which contain numerous objects interacting in one geometric space, present several unique properties. Taking into account those properties can improve the transmission and rendering scalability for the streaming application. In the next chapters, we advance from individual 3D models to 3D scene databases and explore coding and transmission mechanisms that address the properties of 3D scenes.

CHAPTER V

MULTI-OBJECT SOURCE AND CHANNEL CODING

5.1 Introduction

We begin our exploration with scalable source and channel coding. A 3D scene database, as aforementioned, synthesizes various objects in one geometric space, which have potentially unequal contributions to the display quality depending on such factors as the objects' geometric complexities, interactions, and the application's semantics. When coding a scene database under limited bit rates, objects' unequal display importance should be considered to present the best scene quality. To do so, one may apply the conventional algorithms to code all objects separately into multiple resolutions and then select proper LODs for different objects such that the rate constraint is satisfied. This straightforward solution, however, assumes a fixed quantization precision for all the objects and for the entire LOD hierarchy of each object. Intuitively, with the same overall bit rate, higher scene quality may be achieved by coding more important objects or more important LODs of an object with higher quantization precisions and less important portions with lower quantization precisions. In general, an adaptive coding scheme should intelligently determine the quantization precisions for different objects and different LODs of each object regarding their unequal decoding importance.

Adaptive quantization aims to properly distribute source bits among various LOD hierarchies to obtain improved distortion-rate performance. When channel coding is involved in a lossy environment, two aspects regarding transmission efficiency and error resilience need to be addressed concurrently. In one aspect, multiple objects are coded independently and therefore are desired to be delivered in respective packet sequences. Thus, at the receiving end, losing one packet will only corrupt or delay the decoding of a particular object, while other objects can still be decoded and manipulated. In the other aspect, unequal error resilience is desired for multi-resolution objects to provide preferential error protection for

more important objects as well as for more important layers of each object. Conventionally, unequal error protection (UEP) schemes were devised for single mesh or image object, e.g., [5, 11, 46, 54, 66], or multiple image objects in a separate fashion [37, 65]. Joint while preferential error protection for various multi-resolution graphic objects has not been well addressed in the literature.

The joint source and channel coding system presented in this chapter properly accounts for the above aspects [61, 85, 90]. The presented work makes three contributions. First, we propose an adaptive vector-quantization (VQ) scheme for coding a 3D scene database into multi-resolution hierarchies. Second, we propose an object-oriented streaming mechanism, which treats graphic objects jointly in error protection while preserving their independencies in transport. In doing so, we novelly generate parity data as additional objects embedded in the stream of graphic objects. Finally, we develop a rate-distortion framework that allocates bit rates between source and parity objects and generates parity data optimally. Our experimental results demonstrate that the proposed system improves the quality of decoded 3D databases significantly compared to conventional methods.

The rest of the chapter is organized as follows. The next section provides an overview of the proposed coding system. Section 5.3 describes the adaptive vector quantization for 3D scene databases. In Section 5.4, we address joint unequal error protection for multiple scalably coded objects with independent transport. In Section 5.5, we present the joint source and channel coding scheme under a rate-distortion optimization framework. We provide experimental results in Section 5.6 and conclude the chapter in Section 5.7.

5.2 System Overview

A diagram of the proposed joint source and channel coding system is depicted in Figure 31. We exploit vector quantization (VQ) to perform adaptive quantization in multi-resolution scene compression. In doing so, we generate the VQ codebook using all mesh objects contained in the input scene database. Each training vector, which is a geometry prediction residual produced by progressive mesh simplification [68], is assigned a weight factor to

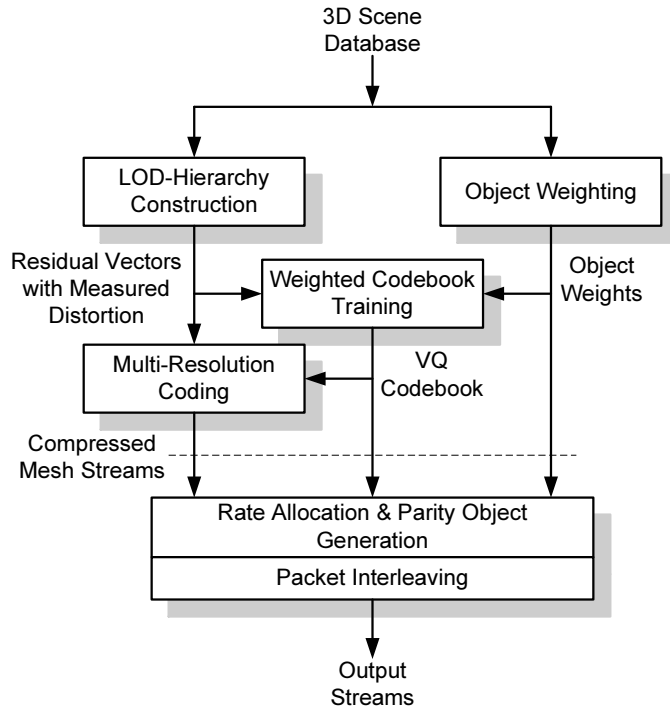


Figure 31: A diagram of the presented joint source and channel coding system.

reflect its relative importance in decoding the scene. This weight factor is determined by incorporating application-dependent object importance with measured distortion in the mesh simplification process. A weighted codebook training algorithm is then executed toward higher quantization precisions for more important vectors. Using thus produced codebook, multiple objects are coded into LOD hierarchies separately and are organized into respective packet sequences to preserve their manipulation independency during packet delivery. In contrast to the independent transport, for error resilience, the packets of graphic objects are protected concurrently while also preferentially by a plurality of FEC codes. The parity data of each FEC code is treated as a separate object parallel to graphic objects. Based on weighted distortion-rate properties, an optimization framework performs rate allocation between graphic objects and parity objects and generates parity data correspondingly. Finally, all the objects are transmitted in an interleaved manner to allow equally fast access to each object at the receiving end.

5.2.1 Object Weighting

As the first component of the presented coding system, object weighting assigns a weight factor to each object that is expected to reflect the relative importance of the object in the displayed 3D scene. To this end, a straightforward heuristic is to assign object weights according to the relative volumes of the objects. More sophisticatedly, weights assigned to objects need to reflect objects' geometric complexities and/or their inherent importance determined upon the semantics of the application. For example, merchandizes should be more important than shelves in a virtual shop, and paints would be more important than walls in a virtual gallery. In general, modeling the relative importance of the objects is application specific. For this reason, we do not intend to specify an object weighting process in this chapter. Rather, we assume that each object κ in a given scene database, \mathcal{S} , has been assigned a weight factor, $\omega_\kappa \in [0, 1]$, to reflect its application-specific importance. All the weight factors are normalized so that their summation is equal to 1, i.e.,

$$\sum_{\kappa \in \mathcal{S}} \omega_\kappa = 1. \quad (30)$$

In the rest of the chapter, we discuss the other components of the proposed coding system based on given object weights.

5.2.2 LOD Hierarchies

In parallel to the object weighting, a multi-resolution hierarchy is constructed for each mesh object in the input scene database. To be explicit, Figure 32 shows a diagram of the VQ-based multi-resolution coding procedure for a single object, which can be summarized in three steps: (i) The full-resolution mesh M^L is downsampled to generate a sequence of LODs, M^{L-1}, \dots, M^0 , by performing successive *half-edge* collapse operations [44] at each level. Every half-edge collapse operation merges one vertex of an edge to the other, which alters the neighborhood of the collapsed vertex. The corresponding connectivity information of the collapsed vertices, $\mathcal{C}^L, \mathcal{C}^{L-1}, \dots, \mathcal{C}^1$, are encoded [68] following a traverse order, while the geometry data, $\mathcal{G}^L, \mathcal{G}^{L-1}, \dots, \mathcal{G}^1$, are buffered; (ii) The base mesh is encoded using single-resolution mesh compression algorithms, e.g., [91]; (iii) Starting from the decoded

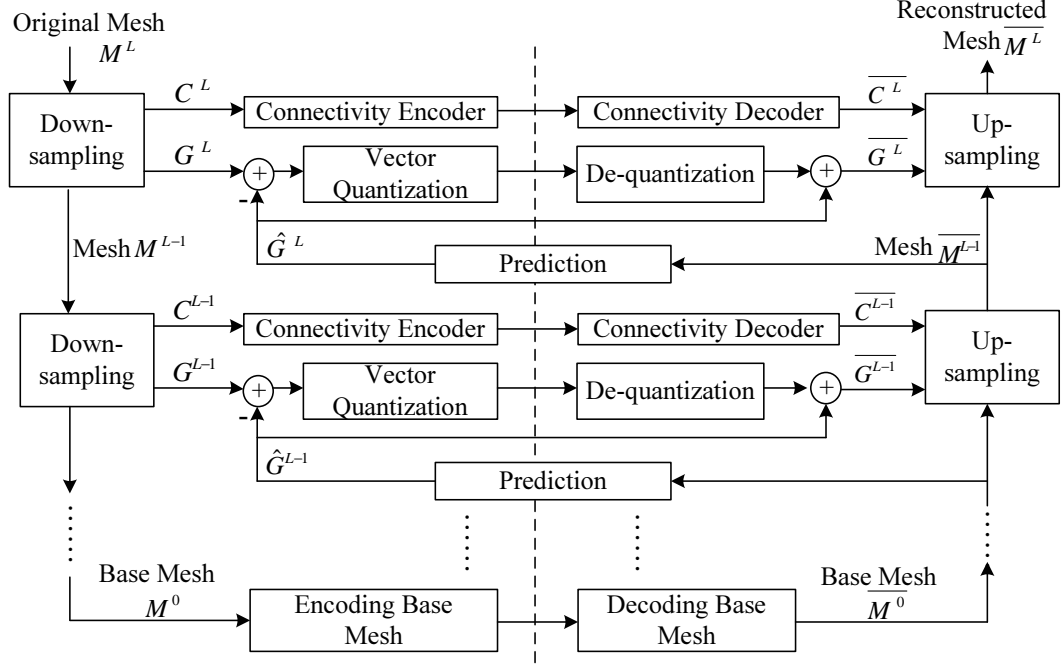


Figure 32: A diagram of the LOD-hierarchy construction/reconstruction process and the VQ-based multi-resolution coding for one mesh object.

base mesh, the enhancement geometry data, $\mathcal{G}^1, \mathcal{G}^2, \dots, \mathcal{G}^L$, are sequentially predicted based on previous reconstructed LODs, and the prediction residuals are compressed by vector quantization. Different from the existing algorithms, in this chapter we exploit a scene-adaptive vector quantization scheme to compress the geometry residuals, which is detailed in Section 5.3.

For a single object, the *normalized distortion* of an LOD M^i in the constructed LOD hierarchy is measured following the definition in Chapter 4. In particular, we define the normalized distortion of an LOD M^i as

$$\mathcal{D}(M^i) = \frac{\mathcal{E}_{rms}(M^i, M^L)}{\mathcal{E}_{max}(M^0, M^L)}. \quad (31)$$

In (31), $\mathcal{E}_{rms}(M^i, M^L)$ and $\mathcal{E}_{max}(M^0, M^L)$ are the root-mean-square and maximum distances, respectively, as specified by (3) and (5) in Chapter 2. Given a scene database with N multi-resolution objects, $\{M_\kappa^i\}$, $\kappa = 1 \dots N, i = 0 \dots L_\kappa$, and the object weights, $\{\omega_\kappa\}$, we define the distortion of a decoded scene as:

$$\mathcal{D}_s(\{M_\kappa^i\}) = \sum_{\kappa=1}^N \omega_\kappa \cdot \mathcal{D}(M_\kappa^i), \quad 0 \leq \omega_\kappa \leq 1, \quad \sum_{\kappa=1}^N \omega_\kappa = 1, \quad (32)$$

where $\mathcal{D}(M_\kappa^{i_\kappa})$ denotes the measured distortion for mesh M_κ decoded at a resolution i_κ .

5.3 Vector Quantization

Entropy coding was commonly used in coding coordinate residuals in separate spatial dimensions. Vector quantization (VQ) was first introduced in single-resolution mesh compression [26,57] to code the vertex geometry jointly. Recently, Li et al. studied the incorporation of vector quantization with a multi-resolution hierarchy and proposed a VQM algorithm [62] for multi-resolution mesh compression, which showed improved compression efficiency compared to its preceding algorithms that use scalar quantization and entropy coding.

In [62], vector quantization was adopted for the multi-resolution compression of single mesh objects, where the VQ codebook is generated by using a separate set of training models and the codebook training algorithm is independent from the coding process. Such a codebook independency is necessary for coding a single mesh object as otherwise the codebook must be transmitted with the object, which can incur a considerable overhead to the bitstream. When coding a scene database comprising a number of mesh objects, however, the overhead of the codebook is small compared to the entire bitstream. Meanwhile, using prediction residual vectors generated from all mesh objects in the scene database as the training set of the codebook is expected to yield improved compression efficiency. Furthermore, by partitioning the training vectors into various sets in accordance with their decoding importance, adaptive quantization precisions can be allocated to different objects as well as different LODs of each object.

Based on the discussion above, we develop a weighted training algorithm to produce the VQ codebook for compressing the geometry data of a scene database. The algorithm is adaptive in three senses: (i) it is adaptive to the contents of the input 3D scene database, as the training set is taken from all objects in database, (ii) it is adaptive to the application-specified object importance, and (iii) it is adaptive to the decoding importance of the different LODs of each object.

5.3.1 Weights of Training Vectors

The proposed codebook training process takes prediction residuals from all mesh objects in the input scene database as training vectors. We denote the entire set of training vectors by a union of subsets, $\bigcup_{\kappa, i_\kappa} \{\mathbf{x}_{\kappa, j}^{i_\kappa} | j = 1, 2, \dots, n_{\kappa}^{i_\kappa}\}$, where each subset $\{\mathbf{x}_{\kappa, j}^{i_\kappa}\}$ is a batch of j residual vectors for decoding mesh $M_{\kappa}^{i_\kappa}$ toward its higher LOD $M_{\kappa}^{i_\kappa+1}$. To account for the unequal importance of different objects and different LODs of each object in vector quantization, we assign each training vector $\mathbf{x}_{\kappa, j}^{i_\kappa}$ a weight factor, $\gamma(\mathbf{x}_{\kappa, j}^{i_\kappa})$, which incorporates both the object weight and the measured distortion of the corresponding LOD. In particular, we have

$$\gamma(\mathbf{x}_{\kappa, j}^{i_\kappa}) = \omega_{\kappa} \cdot \frac{\Delta D_{\kappa}^{i_\kappa}}{n_{\kappa}^{i_\kappa}}, \quad (33)$$

where ω_{κ} is the weight of the object and

$$\Delta D_{\kappa}^{i_\kappa} = D(M_{\kappa}^{i_\kappa}) - D(M_{\kappa}^{i_\kappa+1}) \quad (34)$$

denotes the *reduction* of normalized distortion when the higher LOD $M_{\kappa}^{i_\kappa+1}$ is successfully decoded.

Two heuristics are implied by the weight allocation given in (33). First, vectors of lower LODs are allocated larger weights than those of higher LODs for the same object. This is because decoding lower LODs in general results in more significant distortion reduction (larger $\Delta D_{\kappa}^{i_\kappa}$) than decoding higher LODs. Second, within each batch, all training vectors have the same weight. In other words, they will be treated equally importantly in minimizing quantization errors in the codebook training process.

5.3.2 Codebook Training Algorithm

By introducing vector weights we aim to quantize various batches of geometry data with different precisions to attain higher distortion-rate performance. To this end, we develop a weighted codebook training algorithm based on the stochastic relaxation (SR) algorithm proposed in [102]. In the following, we present a complete description of the weighted codebook training algorithm. In our presentation, we consider a q -bit vector quantizer. Therefore the codebook contains 2^q code vectors. We use $\{\mathbf{x}_i, \gamma_i\}$, $i = 1, 2, \dots, K$, to denote the entire set of training vectors and their corresponding weights allocated by (33).

(i) Code vector initialization:

$$y_1^{(1)}, \dots, y_{2^q}^{(1)}, m = 1.$$

(ii) Nearest neighbor repartition ($1 \leq i \leq K$):

$$j = \arg \min_{1 \leq l \leq 2^q} \|\mathbf{x}_i - \mathbf{y}_l^{(m)}\|, \quad (35)$$

Put $\mathbf{x}_i \rightarrow \mathcal{R}_j^{(m)}$,

$$\mathcal{D}_m = \mathcal{D}_m + \gamma_i \cdot \|\mathbf{x}_i - \mathbf{y}_j^{(m)}\|^2. \quad (36)$$

(iii) Centroid computation ($1 \leq l \leq 2^q$):

$$\mathbf{y}_l^{(m)} = \frac{\sum_{i: \mathbf{x}_i \in \mathcal{R}_l^{(m)}} \gamma_i \cdot \mathbf{x}_i}{\sum_{i: \mathbf{x}_i \in \mathcal{R}_l^{(m)}} \gamma_i}. \quad (37)$$

(iv) Code vector jiggling ($1 \leq l \leq 2^q$):

$$\begin{aligned} T_m &= \sigma_x^2 \left(1 - \frac{m}{I}\right)^3, \\ \mathbf{y}_l^{(m)} &= \mathbf{y}_l^{(m)} + \xi_l(T_m). \end{aligned} \quad (38)$$

(v) Stopping condition:

If $m \geq I$, Stop,
Otherwise, $m = m + 1$, Goto (ii).

Figure 33: The weighted codebook training algorithm.

The major steps of the codebook training algorithm are shown in Figure 33, which are performed in an iterative fashion. In the algorithm, $\mathcal{R}_j^{(m)}$, $\mathbf{y}_j^{(m)}$, and \mathcal{D}_m denote the j -th partition region, the j -th code vector, and the calculated distortion, respectively, at the m -th iteration. The procedure of code vector jiggling, as given in (38), is used to prevent the algorithm from staying in poor local optimal results [102]. In particular, $\xi_l(T_m)$ denotes a perturbation noise added to the code vector, and the noise variance is dictated by the so-called temperature T_m , which gradually decreases as m increases. σ_x^2 indicates the variance of the code vector components, and finally, I defines the number of iterations to be run. More detailed discussions on these parameters can be found in [102].

The presented codebook training algorithm accounts for vector weights in discrepancy evaluation (36) and centroid computation (37). As a result, the algorithm tends to enclose more important vectors in smaller partitions to reduce their quantization errors. Although

this weighted training process is not theoretically justified to yield optimal results, its effectiveness is confirmed by empirical success, as will be shown in Section 5.6.

5.4 *Embedded Parity Objects*

One essence of the adaptive vector quantization is the preferential treatment on coding the LOD hierarchies of various mesh objects with respect to their unequal importance. When we consider error-resilient delivery of the scalably coded scene database under a lossy and rate-constrained environment, it is natural to embrace the same spirit and devise unequal error protection (UEP) schemes. Unlike conventional UEP schemes designed for single mesh or image objects [5, 11, 46, 54, 66], the change from a single-object context to a multi-object one invokes two fundamental aspects that need to be properly addressed together with UEP. In one aspect, because multiple mesh objects are coded independently, they are desired to be delivered in respective transport sequences to preserve the independence at the receiving application. In the other, with the same amount of parity redundancy, the highest error-protection efficiency is expected to be accomplished by joint considerations on multiple objects. We address these issues with UEP in this section.

Because the VQ codebook and the compressed base meshes in general correspond to a small fraction of the entire bitstream, the network and the receiver are mainly loaded by the transmission and processing of the enhancement layers of the scene database. For this sake, throughout our discussion we assume that the VQ codebook and the compressed base meshes are delivered via a reliable channel. We focus our discussion on the error-prone transmission of the enhancement layers of the scene database.

5.4.1 Partially Ordered Packetization

Earlier transmission schemes proposed for multi-object media applications organize the bitstream of multiple objects into a strictly ordered packet sequence, where each packet contains various numbers of bytes for the objects [37, 65]. In those cases, packetization is essentially byte-oriented, and one lost packet may corrupt or delay (if retransmission is allowed) the decoding of multiple objects. To address the coding independency properly, a partially ordered packetization scheme is employed. As shown in Figure 34, instead

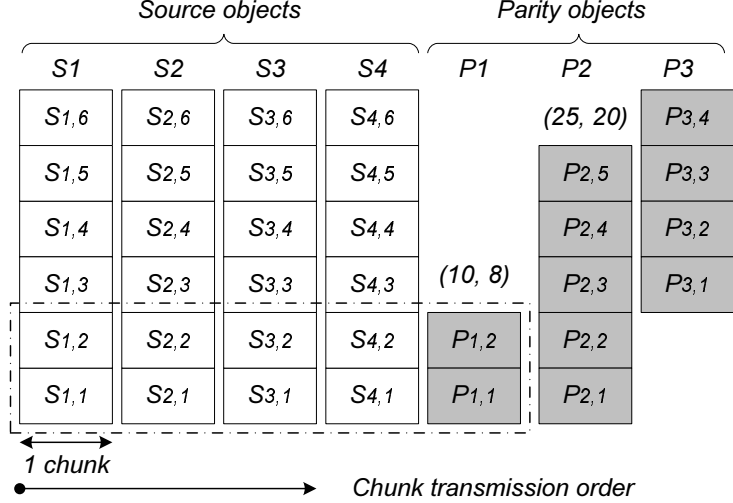


Figure 34: An illustration of the chunk-based transmission and the use of parity objects for joint unequal error protection. Note that the transmission of multiple objects is interleaved to allow equally fast access to the objects.

of packetizing the entire bitstream into a strict sequence, the bitstream of each object is respectively organized into a sequence of *chunks*. At the sending side, chunks of multiple objects are transmitted in an interleaved or round-robin fashion. At the receiving end, multiple chunks from different objects may be received in arbitrary orders, as far as the respective sequence of each object is maintained.

The respective order of each object ensures that the manipulation independency of multiple objects is well preserved in packet delivery under random network behaviors, as in this case, one lost or delayed packet will only affect the sequenced data reception and decoding of a particular object, leaving no impact on the reception or decoding of other objects.

5.4.2 Joint Unequal Error Protection

We now study chunk-level forward error correction (FEC) for the error resilience of multiple objects. To this end, we consider common FEC codes such as Reed-Solomon codes, where for every block of k data chunks $h = (n - k)$ parity chunks are generated to form an n -chunk FEC block. Such an (n, k) systematic code can recover up to h chunk losses (erasures), as positions of the lost chunks in the chunk sequence are known. Principles and implementation issues of such block erasure codes can be found in many literatures, e.g., reference [70].

One natural implementation of the chunk-level FEC is to append parity chunks to data chunks for different objects (streams)¹, respectively, similar to the schemes in [37, 65]. Although such *in-stream* FEC can easily distribute unequal parity redundancies to multiple objects as well as multiple layers of each object, it is not efficient as one parity chunk only provides error resilience to a certain object while it introduces redundancy to all the objects. In this work, we propose to use “parity objects” to overcome the deficiency. Each *parity object* with a size of h chunks refers to an (n, k) systematic code where k data chunks from one or many objects are protected by $h = n - k$ parity chunks. Upon transmission, this parity object is assigned a separate object identification number (OID), as shown in Figure 34. Since both data and parity chunks are uniquely indexed, the receiving end can recognize the locations of lost chunks and recover the code block under the code’s correcting capability. Meanwhile, partially ordered transport are still preserved for all the streams. In addition, with parity objects, unequal error protection can be implemented for arbitrary sets of data chunks by generating multiple parity objects. In Figure 34, for example, chunks $(S_{i,j}, P_{1,j})$, $i, j = 1, 2$, form a $(10, 8)$ FEC code, and $(S_{i,j}, P_{2,j})$, $i, j = 1, 2, \dots, 5$ form a $(25, 20)$ code. Same as source objects, parity objects are respectively ordered, and are embedded into the chunk stream through interleaving.

It is evident that, with the same amount of redundance, using parity objects can present concurrent and hence improved error resilience to multiple source objects compared to in-stream FEC. We use a scenario of equal error protection to clarify this statement, where for a number of s source objects, in-stream FEC appends h parity chunks for every block of k data chunks in each object, respectively, while a parity object generates hs parity chunks for the total number of ks data chunks concurrently. In both cases, the entire FEC block under consideration has a total number of ns chunks. Let $\gamma_I(s)$, $\gamma_P(s)$ denote the probabilities that the entire block is correctly decoded in the separate-FEC and the parity-object cases, respectively. We have $\gamma_I(s) = \Pr\{\leq h \text{ losses out of } n \text{ chunks in each of } s \text{ objects}\}$, and $\gamma_P(s) = \Pr\{\leq hs \text{ losses out of } ns \text{ chunks}\}$.

¹Regarding each object as a chunk stream from the transport perspective, we refer to objects and streams interchangeably in our presentation.

Proposition 1. *Independent from the random loss process in the transmission of the ns -chunk block, $\gamma_I(s) \leq \gamma_P(s)$ for $s \geq 1$.*

Proof. Divide the ns -chunk block into s groups according to the transmission sequence, where chunks with index $i \pmod n$ belong to the same group. Suppose that there are $\nu_1, \nu_2, \dots, \nu_s$ ($0 \leq \nu_i \leq n, i = 1, 2, \dots, s$) losses out of n chunks in each group. In the separate FEC case, the event that the ns -chunk block is recoverable is equivalent to $\nu_i \leq h$ for $i = 1, 2, \dots, s$. Hence $\sum_{i=1}^s \nu_i \leq hs$, which means that the entire block is also recoverable with an hs -chunk parity object regardless of the loss pattern in each group. The inverse of the above reasoning is not true. Therefore $\gamma_I(s) \leq \gamma_P(s)$. \square

Proposition 1 provides a loose though general comparison between the loss-recovery probabilities of the two FEC schemes. As a supplementary result, more analytical comparisons are given in Appendix B.1, where explicit loss-recovery probabilities are derived for particular channel models.

It should be mentioned that using a larger FEC block will require increased coding complexity, which may be upper bounded by resource limitations in a real implementation. Beside that, in both FEC schemes, certain side information needs to be transmitted for the receiver to decode the FEC codes correctly. Efficient coding of such side information is not studied in this chapter. Instead, we assume that overhead for sending the side information is negligible compared to the entire bitstream. In the rest of the chapter, we use triple (n, k, \mathbf{I}) to represent an (n, k) -code parity object, with \mathbf{I} denoting the indices of the k data chunks. Apparently, $k = |\mathbf{I}|$ is the size of set \mathbf{I} .

5.5 Joint Source and Channel Coding

While the larger FEC block in general provides the higher error-correcting ability, data chunks of graphic objects have unequal decoding importance and hence desire preferential error resilience to achieve optimized rate-distortion performance. Next, we study in detail how parity objects should be generated for a set of scalably coded graphic objects and how a given rate budget should be distributed amongst all the source and parity objects, with the

goal of providing maximized expected receiving quality. More explicitly, we aim to tackle the following problem:

Given a set of N scalably coded mesh objects with pre-assigned object weights $\{\omega_\kappa\}_{\kappa=1}^N$, an index set \mathbf{I}_N for all data chunks, a chunk-erasure channel, and a rate constraint \mathcal{C} measured in chunks, find a rate allocation $\Psi = \{k_t, h_t; \mathbf{I}_t\}$, $\mathbf{I}_t \subseteq \mathbf{I}_N$, and a set of χ FEC codes,

$$F = \{(n_1, k_1, \mathbf{I}_1), (n_2, k_2, \mathbf{I}_2), \dots, (n_\chi, k_\chi, \mathbf{I}_\chi)\}, \quad (39)$$

such that the following conditions are satisfied:

- (a) $k_t + h_t \leq \mathcal{C}$, where $k_t = |\mathbf{I}_t|$,
- (b) $\mathbf{I}_i \subseteq \mathbf{I}_t$, for $i = 1, \dots, \chi$,
- (c) $\sum_{i=1}^{\chi} (n_i - k_i) \leq h_t$, and
- (d) the expected receiving distortion $E[D|F]$ is minimized.

In (39), one should note that in principle $\mathbf{I}_i \cap \mathbf{I}_j \neq \emptyset$, $i \neq j$, as one data chunk may be protected by more than one FEC code. The index set \mathbf{I}_t indicates the distribution of all k_t data chunks among the N multi-resolution objects.

Suppose that there are L_κ chunks for each object κ . The expected receiving distortion $E[D|F]$ is given by

$$E[D|F] = \sum_{\kappa=1}^N \omega_\kappa \left(\sum_{l=0}^{L_\kappa} r(\kappa, l|F) D_\kappa(l) \right), \quad (40)$$

where $D_\kappa(l)$ is the resulting distortion from decoding the first l chunks of object κ , and $r(\kappa, l|F)$ is the probability that up to l chunks are recovered given the FEC codes F . Considering that a chunk is decodable if and only if all its preceding chunks in the same object are successfully decoded, we have

$$r(\kappa, l|F) = \begin{cases} \varepsilon(\kappa, l+1|F) \prod_{i=1}^l [1 - \varepsilon(\kappa, i|F)], & l < L_\kappa, \\ \prod_{i=1}^{L_\kappa} [1 - \varepsilon(\kappa, i|F)], & l = L_\kappa, \end{cases} \quad (41)$$

where $\varepsilon(\kappa, i|F)$ is the error probability of chunk i in object κ , with given F and the chunk-erasure channel. For simplicity, we assume that the channel has an independent chunk loss

process with a loss probability p . Then $\varepsilon(\kappa, i|F)$ can be easily calculated for given FEC codes [70].

To avoid overwhelming computation in finding a globally optimal solution, we solve the described problem by an iterative search consisting in two major steps. As one step, we conduct a *parity-object generation* algorithm that finds, for a certain rate allocation $\Psi = \{k_t, h_t; \mathbf{I}_t\}$, a set of FEC codes that satisfies conditions (b–d). As another step, a steepest decent search algorithm is performed, which finds the proper rate distribution between source and parity objects under the rate constraint. The detailed solution is presented below.

5.5.1 Parity-Object Generation

To find the parity-object solution for given $\{k_t, h_t; \mathbf{I}_t\}$ that satisfies conditions (b–d), we develop a heuristic algorithm based on a fact revealed by Proposition 1. Namely, a joint FEC code is generally more efficient than separate FEC codes with the same proportion of redundancy. This fact inspires us to perform a search starting from a single parity object and gradually “split” the parity objects toward decreasing expected receiving distortion until an optimal point is reached.

Several notations are used in our presentation of the algorithm: (i) $\pi_{\kappa,l}$: *importance* of chunk l in object κ , initially assigned to be

$$\pi_{\kappa,l} = \omega_{\kappa} D_{\kappa}(l) \cdot p, \quad (42)$$

(ii) Π : the sequence of k_t data chunks in a *decreasing* order of $\pi_{\kappa,l}$, and (iii) $f(\chi, h, \Pi)$: the optimal FEC solution for a fixed number of χ parity objects, a number of h parity chunks, and a data chunk sequence Π . For $\chi = 1$, we have $f(1, 0, \Pi) = \emptyset$ and

$$f(1, j, \Pi) = \arg \min_{k=1,2,\dots,k_t} E[D|(k+j, k, \mathbf{I}_k)], j = 1, 2, \dots, h_t, \quad (43)$$

where \mathbf{I}_k denotes data chunks $0, 1, \dots, k-1$ in sequence Π . For $\chi > 1$, we recursively compute $f(\chi, h, \Pi)$, $\chi \leq h \leq h_t$, by

$$\begin{cases} f(\chi, h, \Pi) = \arg \min_{F_j: j=1,2,\dots,h-\chi+1} E[D|F_j], \\ F_j = f(\chi-1, h-j, \Pi) \cup f(1, j, \Pi'), \end{cases} \quad (44)$$

where Π' denotes the reordered chunk sequence with updated chunk importance after applying FEC codes $f(\chi - 1, h - j, \Pi)$.

The algorithm then finds the optimal FEC solution by performing the following iteration,

$$F_{opt}^{(1)} = f(1, h_t, \Pi), \quad (45)$$

$$F_{opt}^{(\chi+1)} = \arg \min_{F \in \{F_{opt}^{(\chi)}, f(\chi+1, h_t, \Pi)\}} E[D|F], \quad (46)$$

$$\pi_{\kappa, l} = \omega_{\kappa} D_{\kappa}(l) \cdot \varepsilon(\kappa, l | F_{opt}^{(\chi+1)}), \quad (47)$$

until for a certain χ there exists

$$F_{opt}^{(\chi+1)} = F_{opt}^{(\chi)}.$$

Without much difficulty, the foregoing algorithm can be implemented using dynamic programming. It has a worst-case computation complexity of $O(h_t k_t) \cdot O(k_t \log k_t)$, accounting for the reordering operations of the chunk sequence needed in (44). In practice, the algorithm reaches optima in a faster computation time, as we expected from Proposition 1.

5.5.2 Rate Allocation

We use *ParityObject*(\cdot) to denote the parity-object generation algorithm described above, which returns the optimal FEC solution F_{Ψ} and the corresponding expected distortion $E[D|F_{\Psi}]$ for a given rate distribution $\Psi = \{k_t, h_t; \mathbf{I}_t\}$. The optimal rate-allocation solution, Ψ_{opt} , under the rate constraint \mathcal{C} is then given by

$$\Psi_{opt} = \arg \min_{\Psi: k_t + h_t \leq \mathcal{C}} E[D|F_{\Psi}] \quad (48)$$

The solution to (48) is found by a steepest decent search algorithm starting from the lowest resolution for each object. The steepest decent search is performed in an incremental manner. At each step, the resolution of either one of the N objects is increased; the remaining bit rate is allocated to parity redundancy, and the corresponding optimal FEC solution is computed using the parity-object generation algorithm. Among N possibilities, the one that results in the maximum ratio of expected distortion reduction over rate increment is selected. The resolution of the corresponding object is then increased. This

(i) *Initialization*: $m = 1, i_\kappa = 0, 1 \leq \kappa \leq N$,

$$\begin{aligned} k_t^{(1)} &= \sum_{\kappa=1}^N R(M_\kappa^0), h_t^{(1)} = \mathcal{C} - k_t^{(1)}; \mathbf{I}_t^{(1)} = \bigcup_{\kappa=1}^N \mathbf{I}(M_\kappa^{(1)}), \\ \{F^{(1)}, E[D|F^{(1)}]\} &\leftarrow \text{ParityObject}(k_t^{(1)}, h_t^{(1)}; \mathbf{I}_t^{(1)}). \end{aligned}$$

(ii) *Incremental allocation* ($1 \leq \kappa \leq N$):

$$\begin{aligned} \delta_\kappa &= R(M_\kappa^{i_\kappa+1}), \\ k_{t,\kappa} &= k_t^{(m)} + \delta_\kappa, h_{t,\kappa} = h_t^{(m)} - \delta_\kappa; \\ \mathbf{I}_{t,\kappa} &= \mathbf{I}_t^{(m)} \cup \mathbf{I}(M_\kappa^{i_\kappa+1}), \\ \{F_\kappa, E[D|F_\kappa]\} &\leftarrow \text{ParityObject}(k_{t,\kappa}, h_{t,\kappa}; \mathbf{I}_{t,\kappa}). \end{aligned}$$

(iii) *Steepest decent search*:

$$\begin{aligned} \kappa_{opt} &= \arg \max_{\kappa=1,2,\dots,N} \left(E[D|F^{(m)}] - E[D|F_\kappa] \right) / \delta_\kappa, \\ i_{\kappa_{opt}} &= i_{\kappa_{opt}} + 1, \\ \{E[D|F^{(m+1)}], F^{(m+1)}\} &= \{E[D|F_{\kappa_{opt}}], F_{\kappa_{opt}}\}, \\ \{k_t^{(m+1)}, h_t^{(m+1)}; \mathbf{I}_t^{(m+1)}\} &= \{k_{t,\kappa_{opt}}, h_{t,\kappa_{opt}}; \mathbf{I}_{t,\kappa_{opt}}\}. \end{aligned}$$

(iv) *Stopping condition*:

$$\begin{aligned} &\text{If } E[D|F^{(m+1)}] > E[D|F^{(m)}], \text{ Stop,} \\ &\text{Otherwise, } m = m + 1, \text{ Goto (ii).} \end{aligned}$$

Figure 35: Rate allocation using the parity-object generation algorithm, where $R(\cdot)$ denotes the bit rate of a particular LOD of an object measured in chunks and $\mathbf{I}(\cdot)$ gives the indices of corresponding data chunks in the entire sequence.

procedure is repeated until no distortion reduction is attained by any of the N possibilities.

We summarize the major steps of the rate-allocation algorithm in Figure 35.

One special case of the rate-allocation in (48) is $h_t \equiv 0$, which corresponds to scenarios where there is no data loss or the rate constraint is solely imposed on the source coding. In such cases, parity object is not generated, and the rate-allocation algorithm returns to be distributing bits among all source objects toward the best distortion-rate performance.

5.6 Experimental Results

In this section, we present empirical results for evaluating the performance of the proposed joint coding system. A 3D database containing 10 mesh models (courtesy of Cyberware,

Inc.) is used in our tests, which have pre-assigned object weights ranging from 0.02 to 0.25 as listed in Table 5. Each model is progressively simplified to generate various LODs, and the compressed LOD hierarchy is packetized with a chunk size of 500 bytes. We use 13 bits for vector quantization.

Table 5: The test 3D database

Model	# Faces	ω_κ	Model	# Faces	ω_κ
EYEBALL	39,600	0.05	HORSE	39,698	0.25
SCREWDRIVER	54,300	0.05	DINOSAUR	56,192	0.02
TEETH	58,300	0.06	VENUS HEAD	67,170	0.15
BALL JOINT	68,530	0.10	SANTA	75,778	0.20
ISIS	93,820	0.08	SHELL	97,928	0.04

As the first study, we consider an error-free environment and investigate the distortion-rate performance of the proposed coding system compared with a conventional method that generates the VQ codebook using a training model set independent from the test database. This independent training set has a comparable size with the test database and is generated using the same parameters in the mesh simplification process.

Figure 36(a) presents the obtained distortion-rate performance for the two comparing methods (the solid lines), where the x -axis is the overall bit-rate constraint² for all the objects and the y -axis gives the PSNR value. Herein we calculate

$$PSNR = -20 \log_{10} \mathcal{D} \text{ (dB)}$$

with \mathcal{D} given by (31) and (32). It is shown that the distortion-rate performance of the scene database is significantly improved by using adaptive vector quantization. At an overall rate of 300 KB, for example, adaptive VQ increases the quality of the database by 3 dB compared to independent VQ. Respectively, Figure 36(b) plots the distortion of several objects in the database for an individual comparison. As can be seen in the plots, although both schemes effectively distribute bit rates among the objects according to their weights, adaptive VQ greatly outperforms its independent counterpart. For the objects with higher weights, e.g.,

²The overall rate budget here does not include the bit rate of base meshes, which are compressed by the same algorithm in all coding schemes. The overall bit rate of adaptive VQ includes the bit rate of the VQ codebook, as it is required to be transmitted with the compressed bitstream.

HORSE and VENUS HEAD, adaptive VQ quickly reaches a close-to-full resolution as the overall bit rate increases. Subjectively, this significant quality difference is confirmed by visually comparing the two pairs of models shown in Figures 37(a-d), where the HORSE and VENUS HEAD models decoded under the rate of 300 KB are captured.

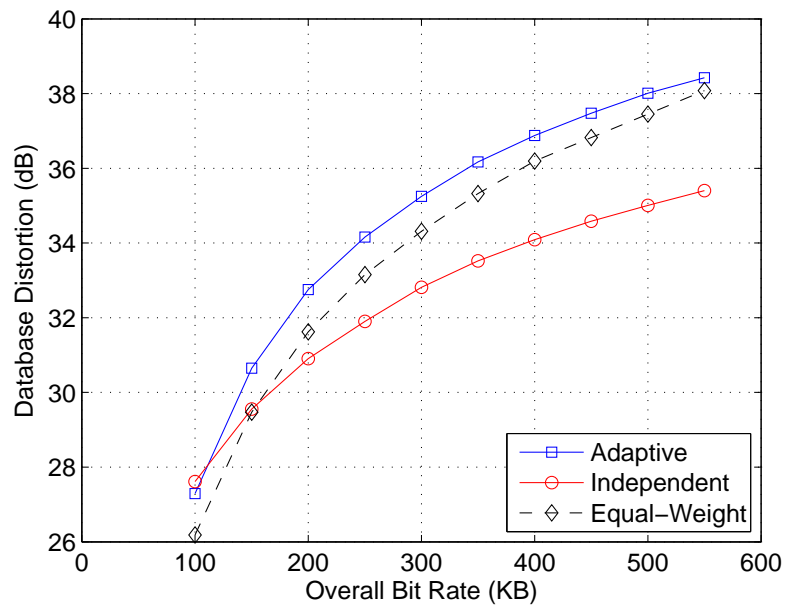
In Figure 36(a), we further compare the proposed coding scheme with a method that treats multiple objects and their various LODs equally importantly (the dashed line). Same as the proposed scheme, the equal-weight method uses prediction residuals from all mesh objects in the input database as training vectors. Nevertheless, the VQ codebook is trained by the original SR algorithm [102] instead of the weighted training, and equal weights are deployed in the rate allocation for all the objects under given rate constraints. From the plots in Figure 36(a), one can observe that the proposed scheme also outperforms the equal-weight method considerably, especially at low bit rates. A typical gain of 0.5–1.5 dB is achieved for the investigated bit-rate range. This improvement comes from two parts: (i) the weighted codebook training which allocates higher quantization precisions to higher-weight objects and lower LODs of each object, and (ii) preferential rate allocation among the weighted objects.

To study the performance of the proposed coding system under a lossy environment, we compare our parity-object based transmission mechanism with the conventional object-oriented UEP scheme, which implements UEP for multiple objects respectively and appends parity data to the stream of each object. For a given parity budget, the algorithm incrementally distributes parity chunks among source objects until the maximum redundancy is reached. At each time, the assignment of one additional parity chunk to either of the multiple objects is determined by minimizing the expected receiving distortion. We refer to this comparing mechanism as *separate UEP* in our presentation.

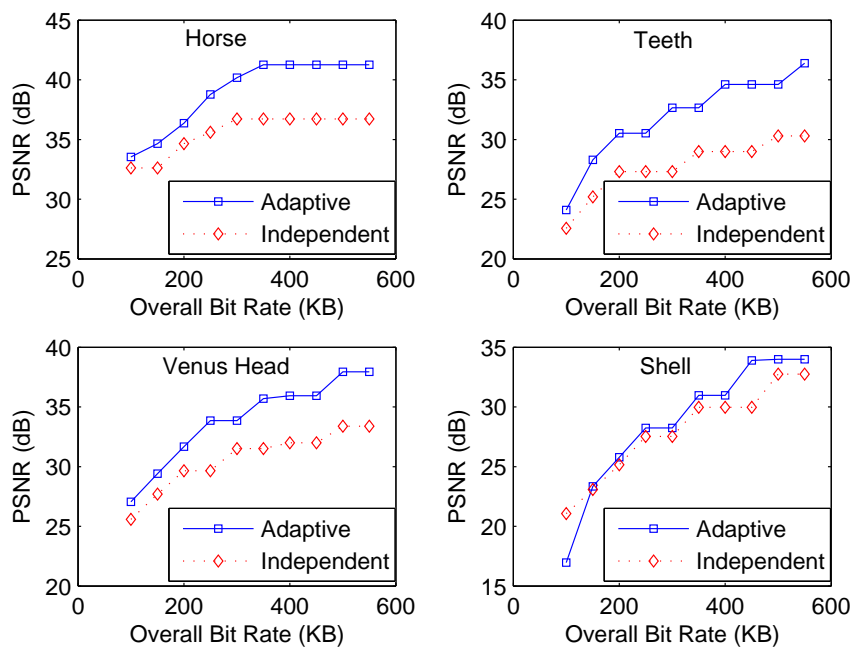
We conduct the performance evaluation following the two stages presented in Section 5.5. In the first stage, we apply the parity-generation algorithm for a fixed source-rate allocation obtained under 300 KB but with different budgets on parity redundancies or various chunk loss rates. The results of averaged receiving distortion are presented in Figures 38(a-b). In particular, Figure 38(a) plots the averaged receiving distortion of the two UEP schemes

with gradually increased parity redundancy budgets under chunk loss rates $p = 0.05$ and $p = 0.10$, while Figure 38(b) shows the receiving distortion for various loss rates under a fixed parity redundancy ratio of 5%. As we anticipate, the parity-object method stably outperforms separate UEP under all conditions. An approximately 2-dB improvement is quickly reached as the chunk loss rate and/or the parity redundancy ratio increase.

We then place overall bit-rate constraints on source and channel coding and perform joint rate allocation following the algorithm described in Figure 35. Figure 39(a) presents the receiving distortion of the scene database under different rate constraints. Similar to what we have perceived in the preceding results, using parity objects improves the receiving distortion by roughly 2 dB compared to separate UEP. The performance difference between the two comparing schemes reaches a stable level with an overall rate $\mathcal{C} \geq 300$ KB. In Figure 39(a), the proportion of bit rate allocated to parity under the corresponding rate constraint is also marked. It is shown that, for a certain chunk loss rate, the parity-allocation ratio gradually decreases and converges to a constant as the overall bit rate increases. Figure 39(b) provides, for two individual cases: $\mathcal{C} = 300$ KB and $\mathcal{C} = 500$ KB, the distortion of each mesh object. The results confirm the effectiveness of both schemes in protecting objects with higher weights, whereas the proposed joint coding method provides more efficient error recovery for the transmitted data chunks, hence resulting in higher receiving quality for every object in the database.



(a)



(b)

Figure 36: Distortion-rate performance of different coding schemes in an error-free environment: (a) Distortion of the database; (b) Distortion of selected objects.



PSNR = 40.18 dB
Adaptive
(a)



PSNR = 36.73 dB
Independent
(b)

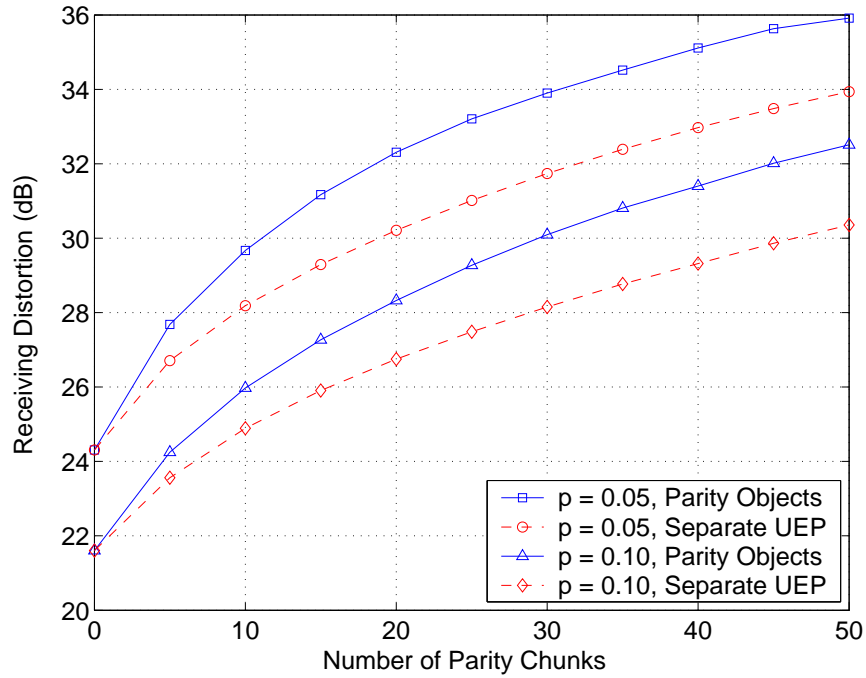


PSNR = 33.85 dB
Adaptive
(c)

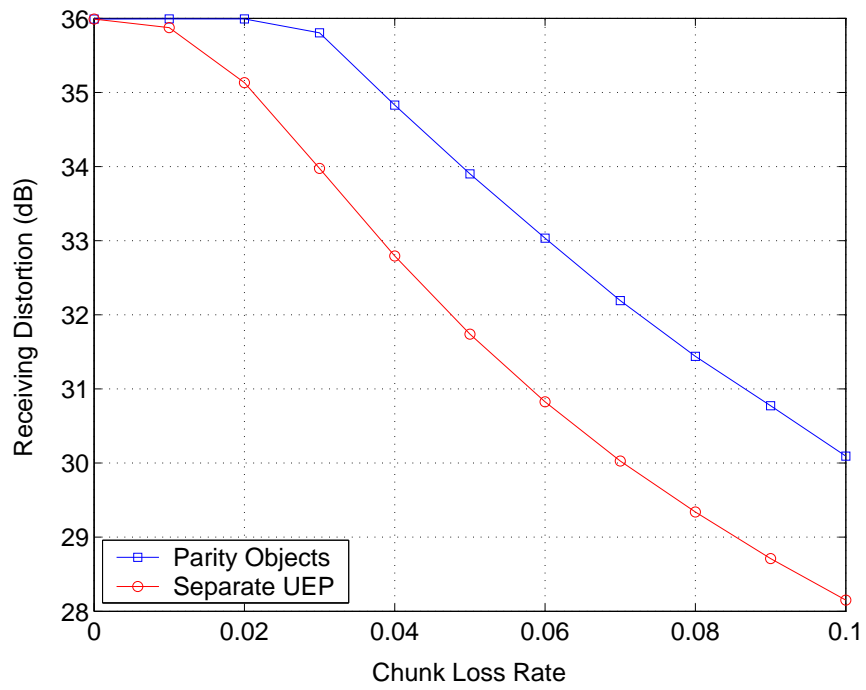


PSNR = 31.51 dB
Independent
(d)

Figure 37: Subjective comparison of decoded HORSE and VENUS HEAD models under an overall bit rate of 300 KB.

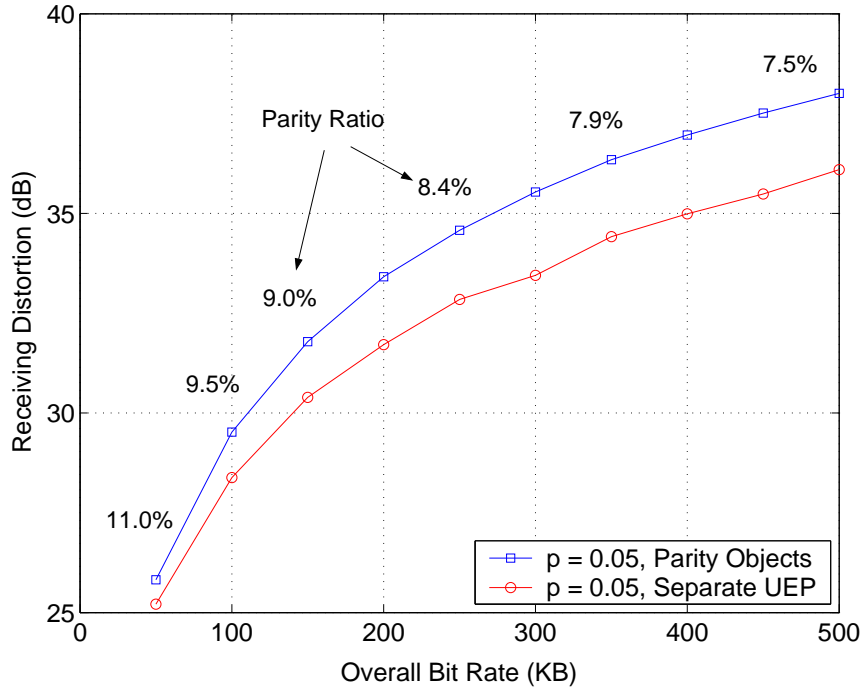


(a)

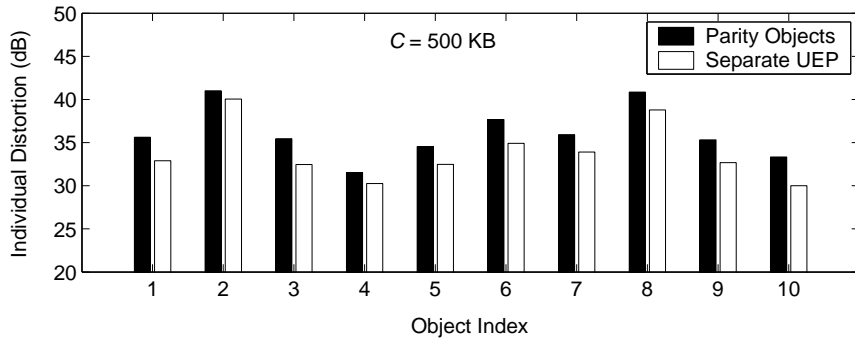
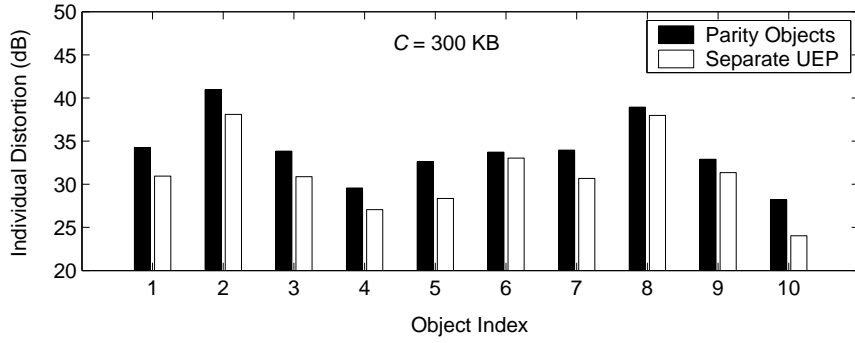


(b)

Figure 38: Receiving distortion of the test database for the data-chunk distribution under 300 KB: (a) Distortion vs. parity redundancy for $p = 0.05$ and $p = 0.10$; (b) Distortion vs. chunk loss rates with 5% parity redundancy.



(a)



(b)

Figure 39: Receiving distortion of the test database with rate allocation between source and parity objects under a chunk loss rate $p = 0.05$: (a) The overall distortion; (b) Two individual cases under $C = 300$ KB and $C = 500$ KB. Objects are indexed in accordance with Table 5 from left to right, top down.

5.7 *Summary*

The coding system presented in this chapter consist of three major components: adaptive vector quantization, embedded parity objects, and joint source and channel coding. Adaptive vector quantization constructs a multi-resolution hierarchy for each object in the 3D scene database. For the best distortion-rate performance, variable quantization precisions are allocated to different objects and different layers of each object based on a weighted distortion model. For packet loss resilience, a plurality of FEC codes are generated as parity objects parallel to graphic objects, which protect the graphic objects concurrently and also preferentially in regard to their unequal decoding importance. A rate-distortion optimization framework is then developed, which performs rate allocation between graphic objects and parity objects and generates the parity data accordingly.

The presented system addressed the properties of 3D scenes from a point of view of source and channel coding. It is natural to also consider these properties from a transport perspective. In addition, in this work we assumed pre-determined object weights for a given 3D scene database. It is desired to have a computational model that estimates the relative importance of the objects in rendering the scene. To do so, interactions of the objects with and within the view space need to be taken into account. Depending on factors such as the object coordinates and the view space, the objects may require different LODs to be displayed with desired quality, or may not need to be rendered at all if, for example, the object falls outside of the view space. We study these issues in the next chapter.

CHAPTER VI

MULTI-STREAMING WITH SCALABLE PARTIAL RELIABILITY

6.1 Introduction

We have observed that scalably coded 3D scenes have three essential properties: (i) objects in a scene have potentially unequal importance regarding display; (ii) LODs of an individual object have decoding dependencies, while (iii) LODs for different objects can be decoded independently. From the transport perspective, these properties suggest that a strictly sequenced delivery for the multiple objects is not necessary. Instead, transmission efficiency may be improved by relaxing the packet-reception order on the transport level. Furthermore, reliable transfer is necessary for the lower LODs of more important objects, by decoding which a significant increment on the scene quality will be obtained. On the other end, unreliable transfer for the higher LODs of less important objects is preferred for improved time efficiency. Between the two ends, a continuous spectrum of partial reliability levels are desired, which refer to selective and differentiated retransmission schemes for various portions of data upon their importance.

In this chapter, we develop an application and transport cross-layer mechanism for streaming 3D scenes in a partially ordered and partially reliable fashion [13, 86, 87, 89]. Incorporating a relatively new IP transport protocol named the stream control transmission protocol (SCTP) [77, 78], we present a multi-streaming framework for scalably encoded 3D scenes with rate-distortion optimized transmission strategies. In doing so, we first present a weighted distortion metric to measure the quality of a scene rendered with multi-resolution objects, modeling objects' unequal importance regarding display. To preserve the manipulation independency of multiple objects in data delivery while provide preferential treatment for different objects as well as different layers of each object, transmission of the

objects is performed over respectively sequenced streams. A rate-distortion optimization framework is then developed, which determines an optimal level of reliability for every chunk of data in each stream, taking into account the rendering importance of the object, the distortion-rate performance of the data chunks, and the statistics of the network link. Compared with heuristical methods, simulation results show that the proposed framework maximizes the display quality of the scene while minimizing the amount of data that needs to be processed by the client's rendering engine.

6.1.1 The Stream Control Transmission Protocol

The stream control transmission protocol (SCTP) [78] is a state-of-the-art IP transport protocol, which embeds ordered and unordered delivery in one connection (called *association* in SCTP) and always retains TCP-friendly congestion control and congestion avoidance¹. SCTP allows separated data streams from the upper-layer application to be multiplexed into one association, and maintains respectively the order of data units on different streams. Thus, if one data unit belonging to a certain stream is lost, succeeding data units from that stream will be stored in the receiver's stream buffer until the lost data is retransmitted from the source. In the meantime, data from other streams can still be passed to upper-layer applications. This partially ordered multi-streaming mechanism avoids the head of line (HOL) blocking in TCP due to the strictly sequenced delivery. Because SCTP is a relatively new protocol, research results on multi-streaming are preliminary [15]. Particularly, joint considerations on the transport mechanism and the graphical media application, although being desirable, have not received research attention.

Partial reliability has been introduced as an extension of SCTP (PR-SCTP) [77], which allows the sender to specify the retransmission behavior for different streams, in a continuous spectrum from TCP-like reliability with multiple retransmissions to UDP-like unreliability with no retransmission at all. The selection of the reliability level, however, has been considered as an application behavior and not been addressed in the previous efforts. In addition, in the traditional SCTP implementation, the reliability level is fixed within one

¹For a good overview on SCTP the readers are referred to [19,36].

stream, which is not optimal for scalable in-order transmission. In contrast to the traditional SCTP, a transport layer with scalable in-stream partial reliability is included in the proposed transmission mechanism, which allows different chunks of data in a stream to have different reliability levels while they are delivered in sequence.

The rest of the chapter is organized as follows. Section 6.2 provides an overview on the proposed multi-streaming framework. In Section 6.3, a distortion measure for 3D scenes is presented, followed by a fast algorithm for estimating the relative weights of the objects. Streaming mechanisms are studied in detail in Section 6.4, where the partially reliable transfer is addressed by a rate-distortion optimization framework. Test results in a simulated network environment are given in Section 6.5. Finally, Section 6.6 concludes the chapter.

6.2 Multi-Streaming Framework

The proposed streaming method for 3D scenes is a joint application and transport mechanism and is composed of the following major components: multi-resolution object coding, object weighting, rate-distortion optimized reliability allocation, and multi-streaming. A block diagram of the framework is presented in Figure 40. Below we briefly describe the major components, leaving algorithm details to the next sections.

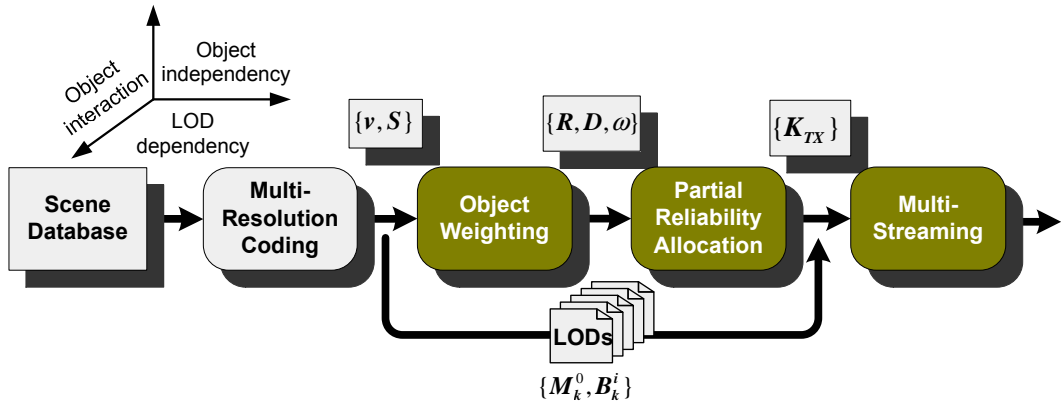


Figure 40: The block diagram of the proposed multi-streaming framework. Notations are explained as follows: $\{v, S\}$ denote the viewpoint and the 3D scene, $\{R, D, \omega\}$ denote the measured rate-distortion performance and the object weights, $\{M_k^0, B_k^i\}$ are the base meshes and the enhancement batches for the objects in the scene, and finally, $\{K_{TX}\}$ denotes the maximum number of transmissions allowed for the data batches. Note that $(K_{TX} - 1)$ is the maximum number of *retransmissions* upon data losses.

Objects in the 3D scene are first encoded as multiple resolutions to provide transmission and rendering scalability. Although 3D objects in general include both mesh surfaces (polygons) and attributes such as textures, throughout the discussion we focus on geometric data for the simplicity of presentation. Nonetheless, the proposed multi-streaming framework and the algorithms are general enough to integrate the attribute data. Same as the previous chapters, the 3D mesh codec employed in the presented work closely follows the CPM algorithm [68].

As introduced at the beginning of this chapter, objects in a 3D scene interact with the view space and with each other as well. In addition, different coordinates of the objects result in potentially unequal importance in visualization. To account for these factors in the rate-distortional streaming framework, an object weighting algorithm is performed by the server before transmission. Note that, the object weighting algorithm does not target at a complete model of the perceptual quality, which is a subjective issue and therefore is out of the scope of this work. Instead, it is designed to provide an estimate on the relative importance of the objects regarding display. A light weight algorithm is desired as complex computation may not be affordable by the server for a real-time application. The algorithm consists in two passes. It first performs a fast test to detect the objects that are not in the current view space. Then, for the objects inside or intersected with the view space, the algorithm creates a *bounding volume projection map* (BVPM) for the objects and estimates the weight factors based on that. Finally, the distortion for a certain resolution of the scene is evaluated as a weighted sum of the distortion for the individual objects that construct the scene. Detailed description on the algorithm is presented in Section 6.3.

In addition to the interactions between objects, LODs of each object have strong dependencies. A lower LOD of an object is more important than a higher LOD because decoding the latter requires successful decoding of the former. This property is noticed as the core of the prior art on scalable transmission of 3D models, e.g, the hybrid TCP/UDP protocol developed in [24], where reliable transport (by TCP) is applied to more important LODs while unreliable transport (by UDP) is used for the rest. The present work steps further in accounting for the dependencies between LODs. A rate-distortion optimization algorithm is

performed to assign for each LOD of every object a level of reliability (LOR), which ranges in a continuous spectrum from full reliability to unreliability.

In contrast to the interactions and dependencies, respective objects in the scene have decoding independencies. In other words, the LODs of one object can be decoded independently regardless of the status of other objects. This property implies that more efficient transport could be achieved by using unordered delivery for separate objects while preserving ordered delivery for each object. To do so, the multi-streaming feature from SCTP is incorporated. Objects are organized into multiple streams according to their weights and based on the network conditions. We discuss on the detailed strategy in Section 6.4.

When performing multi-streaming, every batch of data from the upper-layer application is fragmented into multiple transport data units, which are referred to as *chunks* according to the taxonomy of SCTP [78]. A round-robin scheduling scheme is used to deal with the transmission of multiple streams. In the original design of SCTP, depending on the size of the data chunk and the path MTU (maximum transmission unit) on the network, chunks from multiple streams may be multiplexed into one IP packet for transmission. Although the multiplexing mechanism provides certain flexibility to the transport, it somewhat cancels the impact of having multiple streams as on packet loss in this case is equivalent to simultaneous losses in the corresponding streams. As a result, partially ordered delivery is invalidated. For this reason, chunk bundling is not considered in this work and we persist on one chunk per packet in the analysis and the performance evaluation as well.

Throughout the rest of our discussion on the streaming mechanisms, we refer to a simplified network model consisting in two SCTP endpoints. An error-prone channel with an average packet error rate p is assumed, and we consider an independent error process in our analysis for simplicity. Extension of the results to correlated link-error models such as Markov models will be briefly discussed at the end of the chapter.

6.3 Measuring Scene Quality

In Chapter 5, we have introduced the normalized distortion for a mesh with multiple resolutions $\{M^i\}_{i=0,\dots,L}$ where M^0 is the base mesh and M^L is the full resolution. We rewrite

the definition here for the presentation convenience:

$$\mathcal{D}(M^i) = \frac{\mathcal{E}_{rms}(M^i, M^L)}{\mathcal{E}_{max}(M^0, M^L)}, \quad (49)$$

where $\mathcal{E}_{rms}()$ and $\mathcal{E}_{max}()$ are the measured maximum and root-mean-square surface distances between the corresponding pairs of meshes.

The metric in (49) is further extended to measure the distortion of a 3D scene that has multiple meshes with potential difference in rendering.

Definition 1. *Given a 3D scene \mathcal{S} with N multi-resolution objects, $\{M_k^i\}$, $k = 1 \dots N, i = 1 \dots L_k$, we define the distortion of the scene \mathcal{S} as*

$$\mathcal{D}_s = \sum_{M_k^{i_k} \in \mathcal{S}} \omega_k \cdot \mathcal{D}(M_k^{i_k}), \quad 0 \leq \omega_k \leq 1, \quad (50)$$

where $\mathcal{D}(M_k^{i_k})$ denotes the measured distortion for mesh M_k with resolution i_k ; $\{\omega_k\}_{k=1 \dots N}$ are defined as normalized weight factors to reflect the relative importance of the objects, $\sum_{k=1}^N \omega_k = 1$.

Along with the distortion measure, we introduced a metric with a similar formulation to the peak signal-to-noise ratio (PSNR) that is commonly used in imaging:

Definition 2. *For the given scene \mathcal{S} we have*

$$PSNR = -20 \log_{10} \mathcal{D}_s (dB). \quad (51)$$

The weight factor of an object is expected to reflect the relative importance of the object in rendering the scene. In general, the rendering importance may depend on many factors such as the object coordinates, the view space, the user’s operating focus, and/or the inherent importance of the model according to the semantics of the application². A thorough discussion on these issues is beyond the scope of this dissertation. For the completeness of this work, we employ here a simple view-dependent scheme to estimate the relative importance of objects in a 3D scene. It should be pointed out, however, that the object

²For example, merchandizes would be more important than shelves in an online virtual shop; paints and antiques would be more important than furniture in a virtual museum.

weighting method adopted in the chapter serves as a sample procedure. The weighted distortion metric presented above and the streaming mechanism proposed in the next section are general enough to incorporate other weighting schemes without difficulty.

As shown in Figure 41, given a view perspective, a *view frustum* (VF) is defined by six planes where two are parallel to each other. Intuitively, determined upon the coordinates, objects falling outside of the view frustum will not be visible and hence have no contribution to the display quality. Also, some objects may be partially or completely occluded by other objects that are “in front of” them. In general, an object closer to the viewpoint and with a smaller view angle is subject to a higher weight of importance than an object located farther or with a larger angle to the viewpoint.

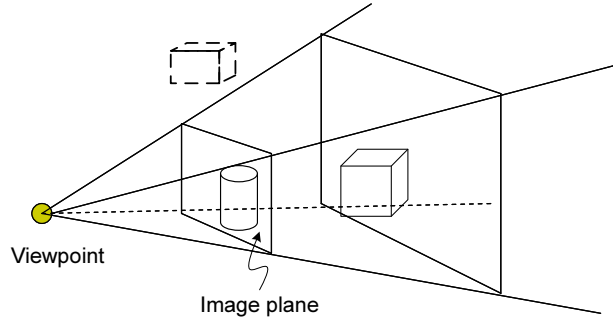


Figure 41: A view frustum is defined by six planes, $\pi_{VF,i} : \mathbf{n}_i \cdot x + d_i = 0, i = 0 \dots 5$, where \mathbf{n}_i is the normal and d_i is the offset of plane $\pi_{VF,i}$, and x is an arbitrary point on the plane. If x is outside $\pi_{VF,i}$, then $\mathbf{n}_i \cdot x + d_i > 0$ and vice versa.

Determination on the object weights, therefore, shares closely the heuristics with the fundamental visibility culling problem [29] in computer graphics, which aims at quickly rejecting invisible geometry before more expensive rendering operations are performed. Nonetheless, there are still essential issues that differ the object weighting (OW) in our streaming context from the visibility culling (VC), as detailed in the following.

The first essential difference between OW and VC is the application target. In particular, the object weighting is performed on the server side to provide an estimate on the scene quality and for efficiently streaming the 3D database. The object weights may be updated dynamically during streaming using feedback from the client. Expensive computations toward accurate decisions are therefore not the key but prompt decisions and operating

flexibility are. For instance, considering the increasing size of database in 3D scenes while many objects may be with small dimensions, the server may organize closely located objects into *object sets* and consider each object set as a “hyper-object” for the weighting and the streaming processes. The visibility culling, on the other hand, is incurred by the client when the 3D scene is actually being rendered to a frame. Visibility culling needs to be accurate in the sense that visible portions should never be discarded from rendering. It may tolerate relatively sophisticated computation as far as the rendering process can be accelerated. For this sake, visibility culling operates not only on objects but mainly on geometry primitives, and is executed at multiple stages in the rendering pipeline.

Beside the preference on light-weight computation to accuracy, the object weighting needs to account for some perspectives that are not in the interests of visibility culling. Explicitly, it needs to *quantify* the relative importance of the objects instead of solely selecting them into a visible set. For example, objects (or object sets) with different dimensions or coordinates should be differentiated from each other, although they may be all visible to the viewpoint. From a mathematical point of view, the object weighting function should therefore be defined as a map from the set of objects, \mathcal{S} , to a continuous set of output $[0, 1]$, i.e., $OW : \mathcal{S} \mapsto [0, 1]$. In contrast, the visibility culling is a binary function defined as $VC : \mathcal{G} \mapsto \{0, 1\}$ where \mathcal{G} denotes the set of geometry primitives.

Driven by the above discussion, our object weighting algorithm is composed of two passes. The first step of the algorithm is to perform a fast view-frustum testing and its output are two sets of objects: $\mathcal{S}_1 \equiv \{\text{Objects outside of VF}\}$, and $\mathcal{S}_2 \equiv \{\text{Objects within or intersected with VF}\}$. Apparently, we have $\omega(\mathcal{O}_j) = 0$ for $\mathcal{O}_j \in \mathcal{S}_1$. For the objects in \mathcal{S}_2 , three actions are taken prior to the estimate of the weight factors: First, the bounding volume (BV) of each object is constructed and the center of the BV is calculated. Then, the distance between the viewpoint and the center of the BV is calculated and used as an estimate of the distance from the viewpoint to the object. Finally, these distances are sorted with the nearest objects first and the centering objects first, if any objects have equal distances to the viewpoint.

The remaining challenge is to evaluate the sorted objects with a meaningful estimate of

the weights. To do so, we uniformly divide the image plane into $h \times w$ grids. The weight of a grid is defined by the reciprocal of its distance to the viewpoint, i.e.,

$$\omega_{\mathbf{v}} \propto \frac{1}{\|\mathbf{v} - \mathbf{v}_0\|} \quad (52)$$

where \mathbf{v} and \mathbf{v}_0 denote the center coordinates of the grid and the coordinates of the viewpoint, respectively (Figure 42). Initially, all the grids are marked as “unused for weighting”. Then, for each object in the sorted list, a *bounding volume projection map* (BVPM) is generated which, as the name implies, is the projection of the object’s bounding volume on the image plane. The overall weight of the grids covered by the BVPM is used for evaluation of the object weight. Once a grid is counted, it is marked as “used for weighting” and becomes no longer usable for the succeeding objects. At the end of the process, normalization is performed so the summation of the object weights is equal to 1.

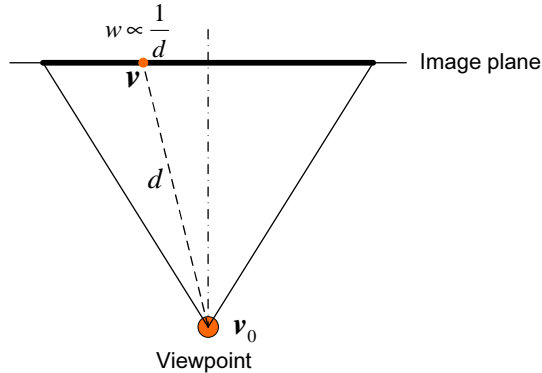


Figure 42: A 2D illustration on defining the grid weight by the distance from the grid to the viewpoint.

Figure 43 gives a demonstration on the projection maps. Note that in Figure 43, each TEAPOT object has several partitions and the bounding box for each partition is created, which formulates an overall bounding volume for the entire object. When computing the object weights, the bounding boxes for the partitions can be processed independently, and their summarized weight will be used as the weight for the original object. Dividing objects into partitions and calculating tight-fitting bounding volumes in general provide more accurate estimates on the object weights³. In addition, computation complexity is controllable

³The incorporation of data partitioning with the streaming framework can also provide higher error

as object partitioning can be performed flexibly following a hierarchical structure [40]. In an extreme case, an occlusion map [103] will be generated for each object, which yields the most accurate coverage of visible portions while requires processing every geometry primitive.

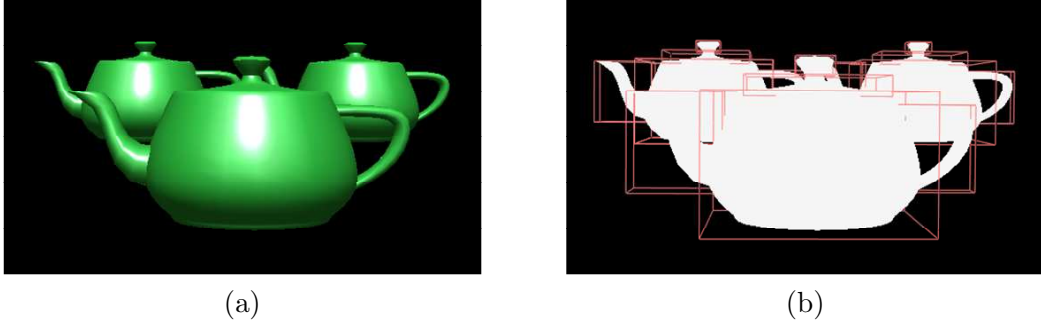


Figure 43: An illustration on the projection-map based object weighting.

6.4 *Transmission Mechanism*

The object weighting process differentiates the objects into several categories: (i) objects rejected by the view-frustum test (with $\omega = 0$, obviously), (ii) objects in the view frustum but assigned $\omega = 0$, and (iii) objects with $0 < \omega < 1$. Transmission for objects in the first category will be skipped and the base representations will be sent for objects in the second category. For the objects in the third category, enhancement batches will be properly chosen to be sent in addition to the base layers.

The transmission procedure therefore consists in two phases. During the first phase base layers of the selected objects are organized into multiple streams and are transmitted with full reliability. Because the base layers in general have a fairly small fraction (1-2% or less) of the entire bitstream, the network is mainly loaded by the transmission of the remaining data, which is the focus of our discussion in this section. In particular, we study a rate-distortion optimization algorithm which performs selective transmission and partial-reliability (PR)

resilience and improved transmission performance when each partition is coded independently [98], in which case each partition is indeed treated as a separate object. Hereinafter we refer to an “object” as the smallest unit regarding multi-resolution coding and transmission.

allocation toward maximal rendering quality under rate constraints. This optimal partial-reliability allocation algorithm is referred to as **PROpt** hereafter. Based on the distortion-rate performance of the objects, their weight factors, and the link error statistics, PROpt determines which enhancement batches of the objects are to be transmitted, and assigns for each selected batch a proper level of reliability (i.e., a maximum number of retransmissions upon data loss). After the scalable PR-allocation, multi-streaming is performed for the selected batches of data with the corresponding reliability levels.

Throughout the rest of our discussion, we refer to a simplified network model consisting in two endpoints. An error-prone link with an average packet loss rate p is assumed, and we consider an independent error process for simplicity. Extension of the results to correlated link-error models such as Markov models will be investigated in our future work.

6.4.1 Rate-Distortion Optimization

All the objects with $\omega > 0$ are considered in the second phase of transmission. Without loss of generality, we assume a total set of N objects $\{\mathcal{O}_k\}_{k=1\dots N}$ with corresponding non-zero weights $\{\omega_k\}_{k=1\dots N}$. Each object \mathcal{O}_k has a number of L_k batches and their distortion-rate performance are denoted by $\{\Delta\mathcal{D}_{kj}, \Delta\mathcal{R}_{kj}\}_{k=1\dots N, j=1\dots L_k}$, where $\Delta\mathcal{R}_{kj}$ is the number of data chunks (packets) for sending batch B_{kj} and $\Delta\mathcal{D}_{kj}$ denotes the distortion reduction by successfully decoding batch B_{kj} from the previous LOD, i.e.,

$$\Delta\mathcal{R}_{kj} = \mathcal{R}_{kj} - \mathcal{R}_{k,j-1}, \text{ and } \Delta\mathcal{D}_{kj} = \mathcal{D}_{kj} - \mathcal{D}_{k,j-1}. \quad (53)$$

In (53), \mathcal{R}_{kj} is the overall rate in chunks and \mathcal{D}_{kj} is the distortion measured as described in Section 6.3. Also note that we have $\Delta\mathcal{D}_{kj} < 0$. For each batch, we compute the residual error rate ϵ_{kj} as a function of $\Delta\mathcal{R}_{kj}$ and the packet loss rate p on the network. In particular, for an independent error process as considered in the chapter, it is obvious that we have

$$\epsilon_{kj} = 1 - (1 - p)^{\Delta\mathcal{R}_{kj}}. \quad (54)$$

Using ρ_{kj} to denote the maximum number of transmissions that is allocated to j -th batch of object \mathcal{O}_k , we define $\boldsymbol{\pi}_k \equiv \{\rho_{kj}\}_{j=1\dots L_k}$ as the transmission policy for object \mathcal{O}_k , and define $\boldsymbol{\Pi} \equiv \{\boldsymbol{\pi}_k\}_{k=1\dots N}$ as the overall transmission policy for all the objects. For a

given transmission policy Π , two bit-rate results are observed from the application and the transport points of view, respectively. The bit rate seen by the application is an *effective* rate that represents the amount of data needed to be processed by the decoding and rendering pipeline, and consequently, the level of quality. We denote the rate by $\mathcal{R}_e(\Pi)$. On the transport layer, the expected transmission rate, $E(\mathcal{R}|\Pi)$, is concerned as it accounts for potential retransmission cost in addition to the effective rate. The expected transmission rate reflects the latency of successful delivery of the data under given network conditions, as the network conditions essentially transfer to a steady-state transport throughput, denoted by G .

Apparently, the above two rates can be expressed, respectively, as the summation of the corresponding rates for the selected batches of data. Namely,

$$\mathcal{R}_e(\Pi) = \sum_{k=1}^N \mathcal{R}_e(\boldsymbol{\pi}_k) = \sum_{k=1}^N \sum_{j=1}^{L_k} \Delta \mathcal{R}_{kj}, \quad (55)$$

and

$$E(\mathcal{R}|\Pi) = \sum_{k=1}^N E(\mathcal{R}|\boldsymbol{\pi}_k) = \sum_{k=1}^N \sum_{j=1}^{L_k} E(\mathcal{R}|\rho_{kj}). \quad (56)$$

Without much difficulty, we can derive the expected transmission rate, $E(\mathcal{R}|\rho_{kj})$, as follows.

$$\begin{aligned} E(\mathcal{R}|\rho_{kj}) &= \Delta \mathcal{R}_{kj} [(1 - \epsilon_{kj}) + 2(1 - \epsilon_{kj})\epsilon_{kj} + \cdots \\ &\quad + (\rho_{kj} - 1)(1 - \epsilon_{kj})\epsilon_{kj}^{(\rho_{kj}-2)} \\ &\quad + \rho_{kj}(1 - \epsilon_{kj})\epsilon_{kj}^{(\rho_{kj}-1)} + \rho_{kj}\epsilon_{kj}^{\rho_{kj}}] \\ &= \Delta \mathcal{R}_{kj} \left[(1 - \epsilon_{kj}) \sum_{n=1}^{\rho_{kj}-1} n \epsilon_{kj}^{(n-1)} + \rho_{kj} \epsilon_{kj}^{(\rho_{kj}-1)} \right]. \end{aligned} \quad (57)$$

A given transmission policy Π also results in an expected decoding distortion, $E(\mathcal{D}|\Pi)$, derivation on which is slightly different from the transmission rate. As described in Section 6.3, we model the overall distortion of the scene as a weighted summation of the expected distortion of all the objects. With the weight factors, $\{\omega_k\}$, for the objects, the overall expected distortion is given by

$$E(\mathcal{D}|\Pi) = \sum_{k=1}^N \omega_k E(\mathcal{D}|\boldsymbol{\pi}_k). \quad (58)$$

To compute $E(\mathcal{D}|\boldsymbol{\pi}_k)$ for each policy $\boldsymbol{\pi}_k$, we define an event Γ_{kj} as

$$\Gamma_{kj} \triangleq \{ \text{delivery of } j\text{-th batch for object } \mathcal{O}_k \text{ fails while all} \\ \text{preceding batches have been successfully received} \}.$$

Because the random error process on the link has been assumed to be an independent process, events $\{\Gamma_{kj}\}_{j=1\dots L_k}$ are independent from each other. Therefore, the expected distortion for policy $\boldsymbol{\pi}_k$ is the summation of the expected distortion for each event Γ_{kj} . Explicitly, we can write

$$\begin{aligned} E(\mathcal{D}|\boldsymbol{\pi}_k) &= \sum_{j=1}^{L_k} E(\mathcal{D}|\Gamma_{kj}) \\ &= \sum_{j=1}^{L_k} \mathcal{D}_{kj} \epsilon_{kj}^{\rho_{kj}} \prod_{l=0}^{j-1} (1 - \epsilon_{kl}^{\rho_{kl}}). \end{aligned} \quad (59)$$

Generally, we model a client with limited networking and computing resource by a bit-rate constraint, \mathcal{R}_c , on the application, and a delay requirement, τ , for data transport, which transfers to a transport rate constraint $\mathcal{R}_T = \tau \cdot G$ on a steady-state network. By summarizing (53)-(59) we can write the rate-distortion formulation and the solution in the following:

$$\Pi_{opt} = \arg \min_{\mathcal{R}_e(\Pi) \leq \mathcal{R}_c \& E(\mathcal{R}|\Pi) \leq \mathcal{R}_T} E(\mathcal{D}|\Pi), \quad (60)$$

where $\mathcal{R}_e(\Pi)$, $E(\mathcal{R}|\Pi)$, and $E(\mathcal{D}|\Pi)$ are given by the equations (55), (56-57), and (58-59), respectively.

To find the solution for (60), the most straightforward method is a brute force algorithm that searches over the entire solution space. Unfortunately, brute force minimization requires exponential computation time, and is computationally infeasible for solution spaces of even modest size.

6.4.2 Fast Heuristics

Heuristics to reduce the computation complexity of (60) are therefore required. To do so, we notice that the distortion-rate performance of the multi-resolution objects have a common characteristic: the data batch for a lower LOD is more important than that for a higher

LOD while the former has also a lower bit-rate. In other words, decoding the batch for a lower LOD results in more significant reduction in distortion with less increment on the bit-rate. Mathematically, for object \mathcal{O}_k with distortion-rate performance $(\Delta\mathcal{D}_{kj}, \Delta\mathcal{R}_{kj})$, the above heuristic stands for the following relation:

$$\Delta\mathcal{R}_{kj} \geq \Delta\mathcal{R}_{k,j-1} \text{ and } |\Delta\mathcal{D}_{kj}| \leq |\Delta\mathcal{D}_{k,j-1}|, 1 \leq j \leq L_k. \quad (61)$$

Using (61) greatly reduces the complexity of finding the optimal PR-allocation for individual objects as it suggests that a gradient-based *steepest decent* algorithm can be employed. Note that a steepest decent algorithm requires only linear computation time. Figure 44 demonstrates the procedure of steepest decent search for a single object \mathcal{O}_k with four batches of data, i.e., $L_k = 4$. The solid nodes denote the PR-allocation with the minimum expected distortion under the rate constraint, while the dashed branches indicate nodes that are cut off from the brute force optimization.

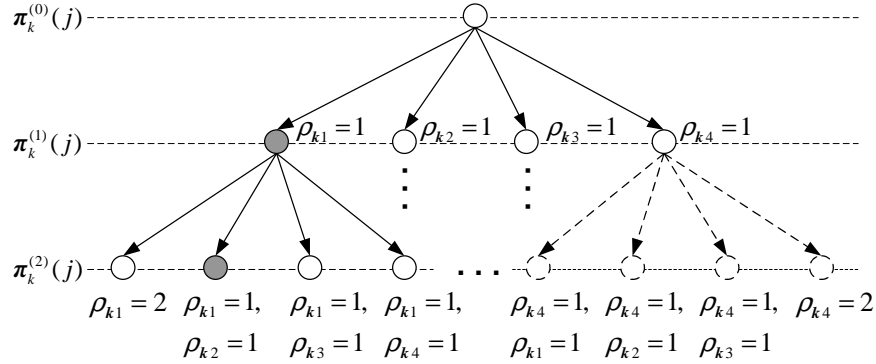


Figure 44: A simple example of the steepest decent search for a single object \mathcal{O}_k with $L_k = 4$.

We use $\tilde{\pi}_k^{(i)} = \{\rho_{k1}, \rho_{k2}, \dots, \rho_{kL_k}\}$ to denote the determined PR-allocation for object \mathcal{O}_k at stage i , and define

$$\begin{aligned} \pi_k^{(i)}(j) &= \tilde{\pi}_k^{(i-1)} \cup \{\rho_{kj} = 1\} \\ &= \{\rho_{k1}, \rho_{k2}, \dots, \rho_{kj} + 1, \dots, \rho_{kL_k}\}, j = 1, \dots, L_k, \end{aligned} \quad (62)$$

which denote the L_k nodes (possibilities) for the next stage. At each stage, the steepest algorithm calculates for each node the gradient of the expected distortion,

$$\delta_k^{(i)}(j) \equiv \left| \frac{\Delta E(\mathcal{D}|\pi_k^{(i)}(j))}{\Delta\mathcal{R}_{kj}} \right| = \left| \frac{E(\mathcal{D}|\pi_k^{(i)}(j)) - E(\mathcal{D}|\tilde{\pi}_k^{(i-1)})}{\Delta\mathcal{R}_{kj}} \right|, \quad (63)$$

and then finds among the L_k nodes the one that maximizes $\delta_k^{(i)}(j)$:

$$\tilde{\pi}_k^{(i)} = \arg \max_{1 \leq j \leq L_k} \delta_k^{(i)}(j). \quad (64)$$

Applying the relation defined in (62), we can substitute the quantities in (63) with the results that we have obtained in (57) and (59), and reach the following result through simple derivations:

$$\delta_k^{(i)}(j) = \left| \frac{\alpha \mathcal{D}_{kj} \prod_{l=0}^{j-1} \gamma_{kl} + \beta \sum_{t=j+1}^{L_k} \mathcal{D}_{kt} \epsilon_{kt}^{\rho_{kt}} \prod_{l=0}^{t-1} \gamma_{kl}}{\Delta \mathcal{R}_{kj}} \right|, \quad (65)$$

where

$$\alpha = \epsilon_{kj}^{\rho_{kj}} (\epsilon_{kj} - 1), \text{ and } \beta = \epsilon_{kj}^{\rho_{kj}} \frac{1 - \epsilon_{kj}}{1 - \epsilon_{kj}^{\rho_{kj}}}.$$

The derivation to the above results is simple and has been skipped in (63). Interested readers are referred to Appendix C.1 for further details.

The computation of (63) and (64) is performed iteratively and follows a cumulative fashion. In other words, if the rate constraint on sending the object is increased, the algorithm does not need to restart from the initial point where no transmission has been assigned to the batches of the object, and calculate again the optimal PR-allocation for the object. Instead, it can start with the prior optimal point that was computed with the lower rate constraint, and continue the steepest decent search toward a new optimum.

With the fast PR-allocation for individual objects, the optimal PR-allocation denoted by (60) can be decoupled. Namely, a gradient search is performed in the LOD plane for each object while a brute force optimization is conducted among the objects. This decomposition reduces the complexity of the problem and makes the computation feasible for 3D scenes with a small number of objects. For relatively complex scenes or servers with limited computing power, however, light-weight computation is still desired.

Under such situations, we extend the steepest decent search in the LOD plane of each object to the object space. In an iterative fashion, the algorithm performs steepest decent search *vertically* in each LOD plane while *horizontally* among the objects. As one step for a particular object, PR-allocation for the object proceeds one stage further to find the new “optimal” allocation for that object. The resulting reduction in expected distortion for each object is properly weighted, and the object with the maximum reduction of weighted

distortion is selected and its PR-allocation is updated. Note that, except for the first step, at each time the steepest decent search proceeds only for the selected object, leaving the remaining objects unchanged (until one of them is selected in next steps). This procedure is repeated until the rate constraint is reached. The algorithm attains computing complexity linear to the space dimension at the cost of reaching a possibly local-optimal solution. Nonetheless, according to our empirical study, such a suboptimal solution is indeed sufficient in fulfilling our main goal of developing an intelligent transmission mechanism with differential reliability for 3D objects, while it favors light-weight computation.

6.5 Performance Evaluation

We present a performance evaluation in this section on the multi-streaming framework described in the previous sections. The *ns-2* network simulator [92] is employed for simulation. We consider a simple topology with two SCTP endpoints connected by a bottleneck link with transmission rate $r = 384$ kbps, maximum segment size $s = 1000$ bytes, and propagation delay $d = 120$ msec. Unless otherwise noted, the size of the receiving buffer is chosen to be $\chi = 20$ KB, and the packet error rate is $p = 0.05$. For transport, the sending buffer is assumed to be sufficiently large while the receiver buffer has a limited size of χ bytes. We perform the simulation over a sufficiently long period and present the averaged statistics.

The sample scene shown in Figure 45, which contains 10 objects overall, is used as a benchmark for our performance evaluation. Each object in the scene is encoded to generate 10 enhancement batches using multi-resolution coding [68]. The full-resolution objects have 39,698 polygons for the HORSE model and 56,192 polygons for the DINOSAUR while their lowest-resolution counterparts have 710 and 1022 polygons, respectively. With the benchmark, following the object weighting process described in Section 6.3 produces a result where 3 objects are rejected by the view-frustum test. The remaining objects and their estimated weights are shown in Figure 46, where for simplicity object partitioning is not performed and a single bounding box is used for each object.

We compare PROpt with two heuristical methods that also perform multi-streaming after discarding objects by a view-frustum test. In particular, the first heuristic transmits

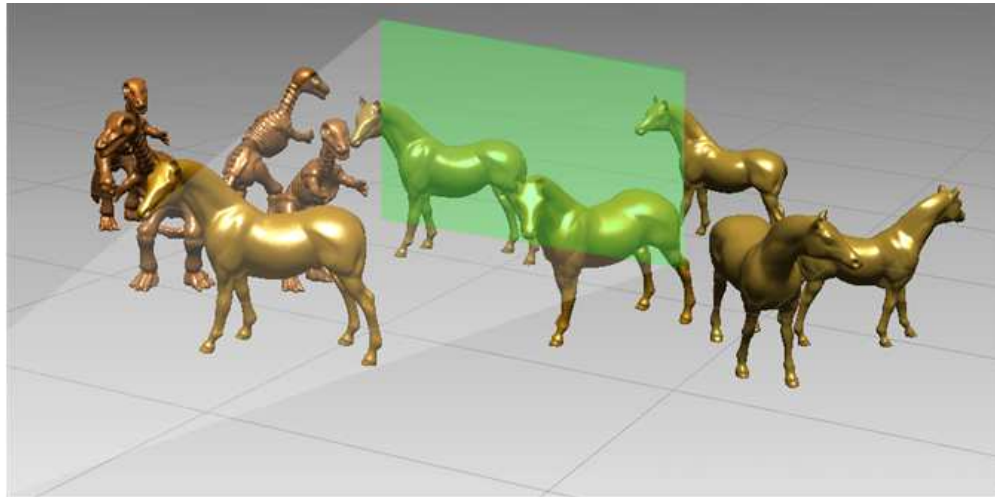


Figure 45: A sample 3D scene with a demonstrated view space. Objects outside the view space will not be displayed on the user’s terminal while other objects may be partially visible due to occlusion.

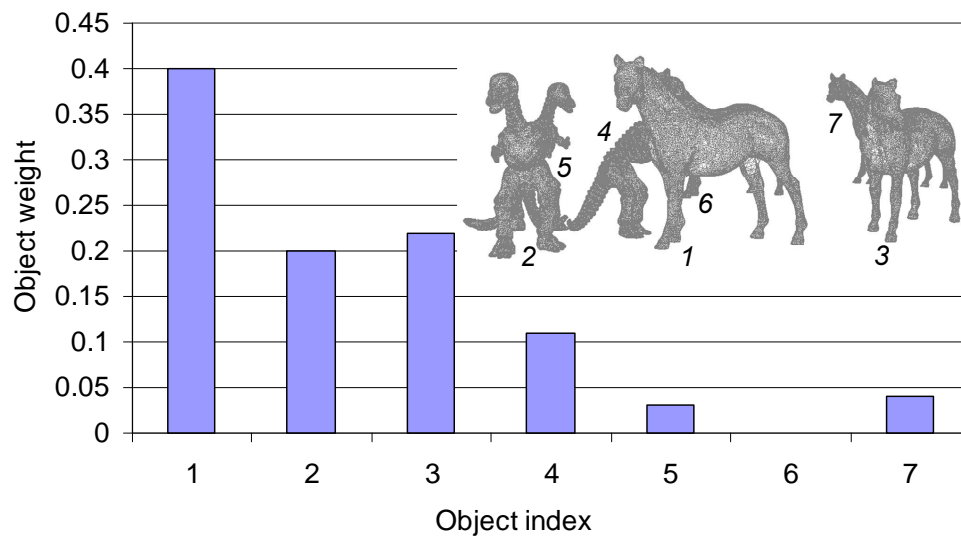


Figure 46: The objects within the view frustum and their estimated weights. These objects are selected to be transmitted in PROpt and the two comparing heuristical methods.

data batches of the selected objects in a *round-robin* fashion. A fixed one-time retransmission opportunity is allocated to each data chunk to provide partial reliability. The second heuristic performs scalable PR-allocation independently for the selected objects. In other words, it assigns equal weights to every object in the view space. These two heuristical methods are referred to as Round-Robin and Equal-Weight, respectively.

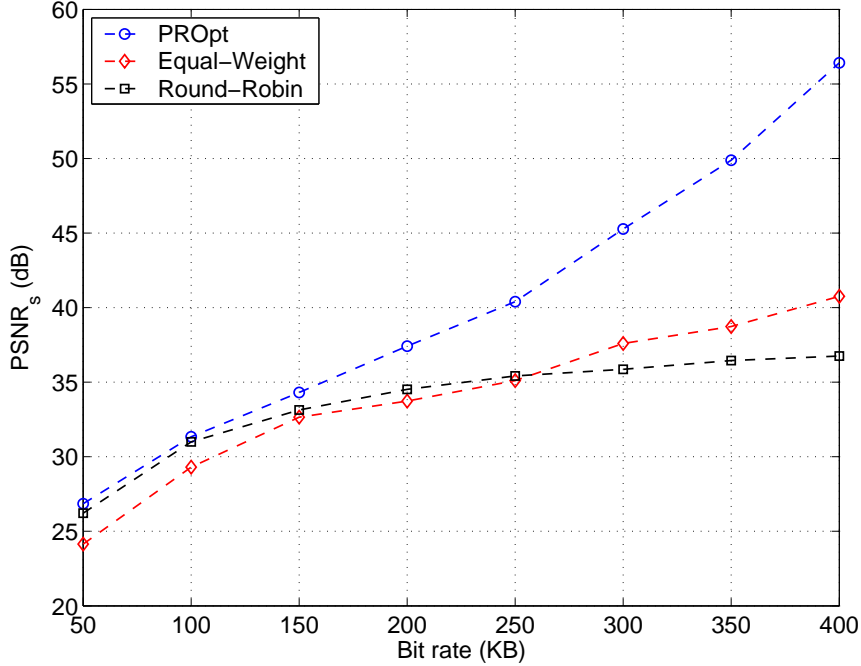


Figure 47: Comparisons of the receiving quality with respect to bit-rate constraints on the rendering application.

6.5.1 On the Receiving Quality

In the first study, we use a relaxed delay requirement on the transport and focus on the receiving quality for different bit-rate constraints on the client application. The results are plotted in Figure 47. From the plots, it can be seen that PROpt outperforms the comparing heuristical methods over the entire bit-rate range. The quality improvement is especially significant at moderate and relatively large bit rates. At very low bit rates ($\mathcal{R}_c \leq 100$ KB), Round-Robin performs closely with PROpt while Equal-Weight presents the lowest quality. Both heuristics and PROpt increase the receiving quality stably as the bit rate increases. Comparatively, Equal-Weight improves the quality more quickly than Round-Robin, resulting from its rate-distortion preferential treatment for LODs of each object. By performing partial-reliability allocation jointly among objects based upon their proper weights, PROpt further increases the receiving quality drastically and quickly approaches a level close to the full resolution ($PSNR_s > 45$ dB). At the rate of 300 KB, for example, PROpt outperforms Equal-Weight by approximately 7.7 dB in the measured quality. Figures 48 presents two rendered results that confirm such a significant quality

difference in a subjective comparison. In Figures 48, please note how PROpt preserves full or close-to-full resolutions for the objects that are likely receiving the most visual attention while presents the lowest resolution for the object that is hardly visible, in accordance with the provided object weights (Figure 46).

In Figure 49(a), we relax the bit-rate constraint on the application and study the receiving quality with different delay requirements on transport. Similar results to Figure 47 are observed in Figure 49(a), which is anticipated as the delay requirement essentially transfers to a bit-rate constraint for overall transmission. Different from Figure 47, where PROpt outperforms the heuristical methods slightly at very low bit rates whereas significantly as the rate increases, a considerable and relatively constant quality improvement by PROpt is observed in Figure 49(a) under various delay constraints. This is because in the delay-constrained situation, different transport decisions made by the comparing mechanisms result in substantial differences on data reception at the upper-layer application.

Both Figure 47 and Figure 49(a) correspond to a packet loss rate $p = 0.05$. In Figure 49(b), we conduct a study on the receiving quality under different packet loss rates with a fixed delay requirement $\tau = 20$ sec. For all the three methods, increasing packet loss rates degrades the receiving quality, which is straightforward since a higher loss rate results in a lower throughput, and consequentially, places a stricter bit-rate constraint on the transport. Again, the results in Figure 49(b) confirm our prior observations. Namely, the Equal-Weight and Round-Robin heuristics perform similarly in most situations, while PROpt greatly outperforms both heuristics over entire investigated ranges.

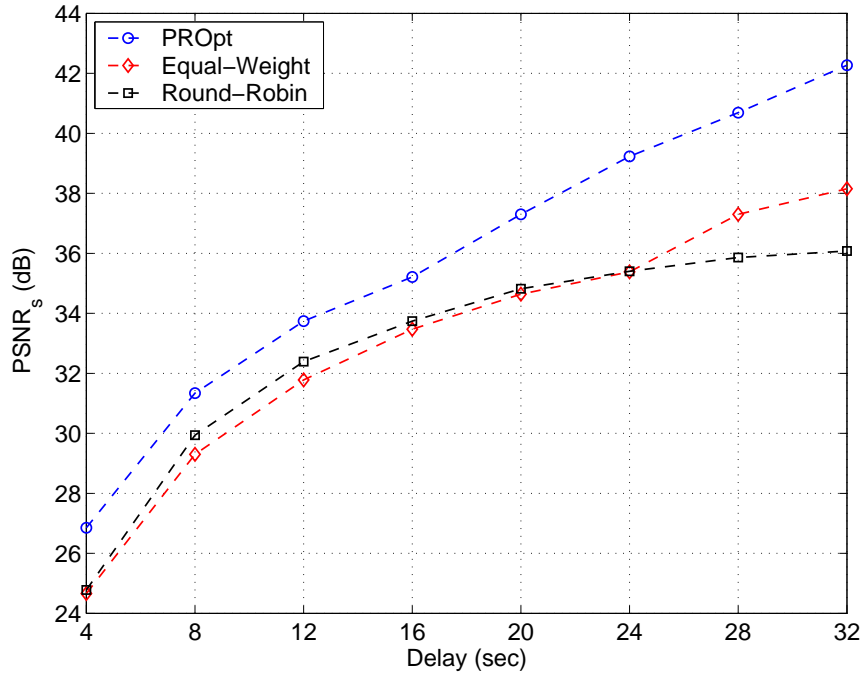


(a) PROpt

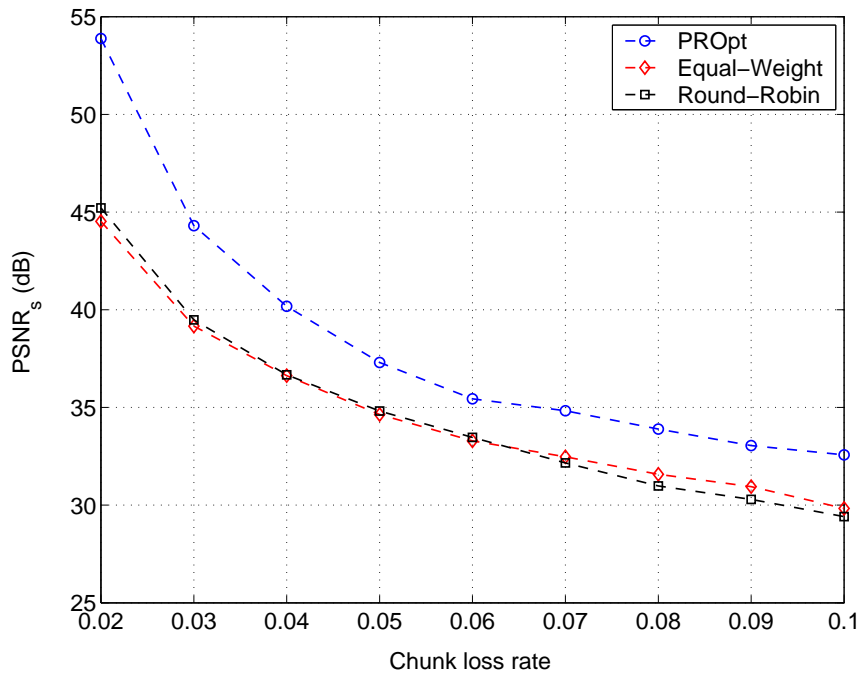


(b) Equal-Weight

Figure 48: Comparisons of the rendered results between PROpt and the Equal-Weight heuristic with the same bit rate (300 KB). Flat shading is used to enhance the facet effect of the displayed models.



(a)



(b)

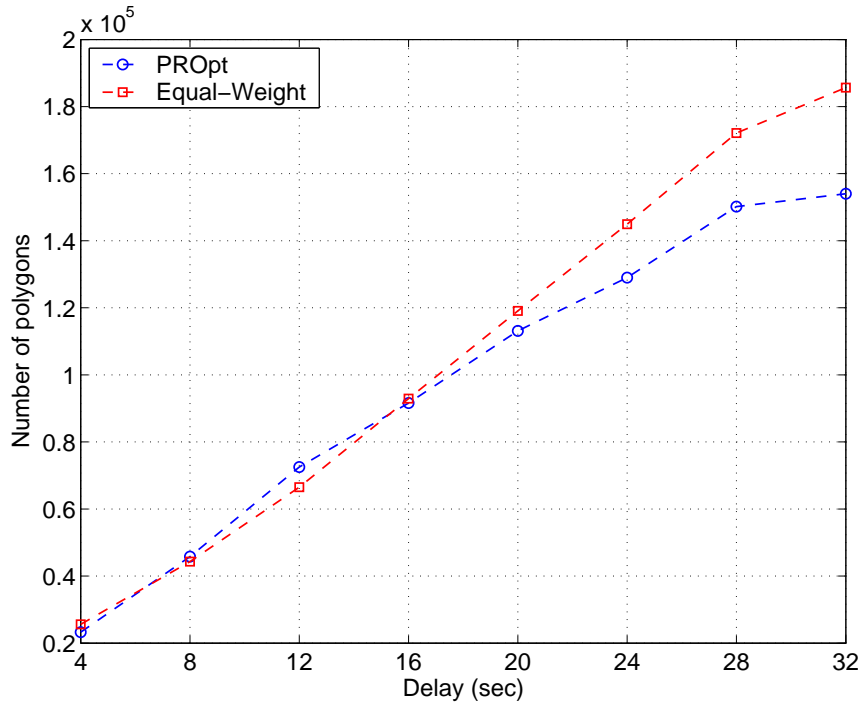
Figure 49: Comparisons of the receiving quality with constraints on the transport: (a) Quality versus delay requirements with a fixed packet loss rate $p = 0.05$; (b) Quality versus packet loss rates with a fixed delay requirement $\tau = 20$ sec.

6.5.2 On the Rendering Cost

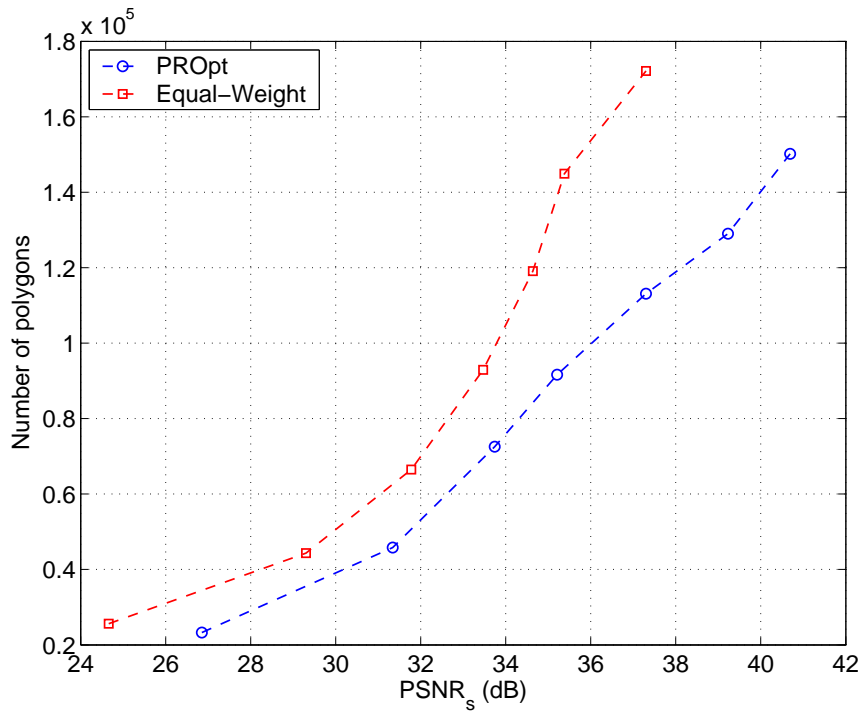
We have compared the performance of the mechanisms from a quality point of view. From another perspective, in the following study we evaluate the processing load placed on the client’s application by the different mechanisms. The processing load is evaluated by the selected number of polygons sent to the client, which will be decoded and rendered by the client’s application. We perform the comparison between PROpt and Equal-Weight to illustrate the efficacy of PROpt with proper object weighting on saving the amount of 3D data that needs to be processed by the application. Such savings are meaningful, as processing a larger number of polygons implies a larger consumption on the computation power of the client, and hence an increased time response for the overall process involving data delivery and rendering.

Figure 50(a) shows the numbers of polygons selected by the two mechanisms for given delay requirements. It is observed that both methods have approximately linear slopes on increasing the number of polygons transmitted when a larger transport latency is allowed. Although, the Equal-Weight heuristic performs closely with PROpt (or even slightly better) when the transmission latency is relatively small, it grows more quickly once the delay requirement is relaxed. This is because as the transport rate increases, the Equal-Weight heuristic tends to select higher LODs for all the objects in the view space approximately equally. In contrast, PROpt spares the increased bit rate to provide further protection on the lower LODs of the more important objects, which reduces the expected receiving distortion while not increasing the maximum amount of 3D data that needs to be rendered.

Figure 50(b) shows the performance difference between the two methods more clearly, where we plot the number of polygons with respect to the level of quality that can be achieved by rendering all the selected polygons. One can see that with the same level of quality, PROpt significantly reduces the number of rendered polygons compared to Equal-Weight. For example, at a quality level of 35 dB, the heuristic transmits 144,932 polygons to the client while PROpt requires solely 91,612 polygons, which implies a saving of approximately 37% on the client’s process.



(a)



(b)

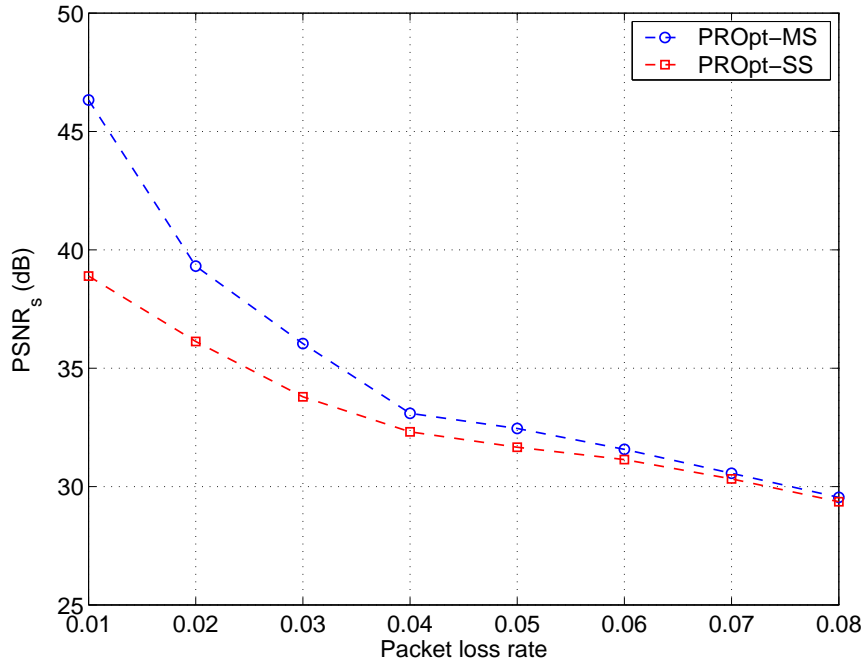
Figure 50: A performance evaluation on the processing load at the client: (a) Number of polygons selected for given transport latency; (b) Number of polygons versus the level of quality.

6.5.3 Multi-Streaming versus Single-Streaming

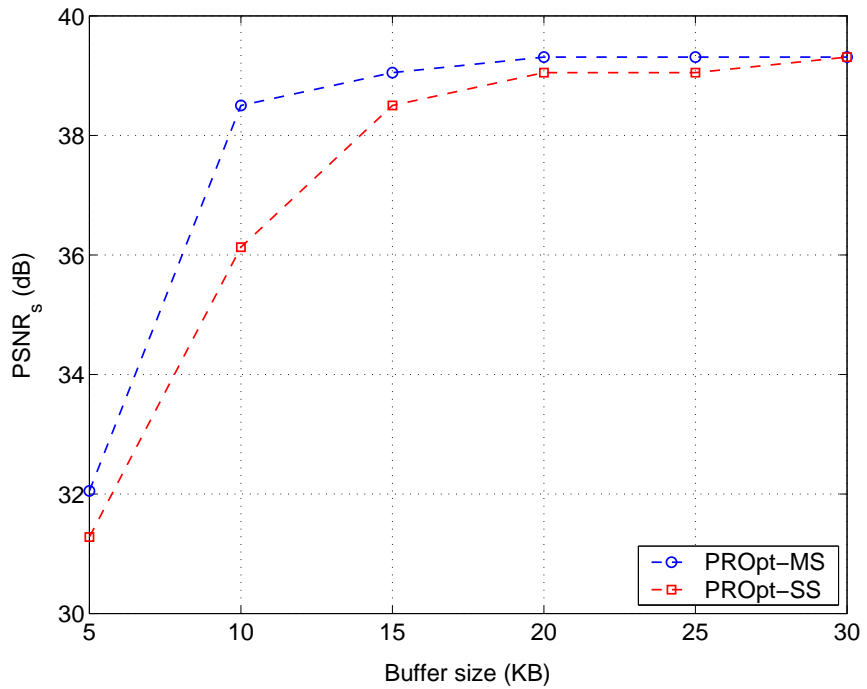
Multi-streaming has always been enabled in the previous studies. Toward the end of our performance evaluation, we present a comparison between PROpt using multi-streaming, as the default mechanism, and a variation that sends all objects through a single stream. For differentiation purpose we refer to them as PROpt-MS and PROpt-SS, respectively.

Figure 51(a) shows the receiving quality of the two schemes under different packet loss rates, where the receiving buffer has a fixed size of 20 KB. As one can see, multi-streaming always outperforms single-streaming and the gain is fairly significant when the packet loss rate is relatively low. This phenomenon is because the partially order delivery in multi-streaming provides a more efficient use on the receiving buffer, which results in higher throughput than single-streaming. When the packet loss rate increases, more streams are likely to be affected by packet losses and the throughput decreases. Therefore, the performance of two schemes gradually merges.

The above explanation also applies to the results presented in Figure 51(b), where the receiving quality is plotted for different sizes of the receiving buffer with a fixed packet loss rate ($p = 0.02$). Again, due to the more efficient use of the receiving buffer, PROpt with multi-streaming attains a relaxed bit-rate constraint on the transport with the same delay requirement, hence improving the streaming quality. All the presented results have verified the optimal PR-allocation (with multi-streaming) a preferable mechanism under the environment with low bit rate and limited computation resource.



(a)



(b)

Figure 51: Comparisons of receiving quality between multi-streaming and single-streaming under a delay requirement $\tau = 10$ sec: (a) quality versus packet loss rate with a fixed buffer size $\chi = 20$ KB; (b) quality versus size of the receiving buffer with a fixed packet loss rate $p = 0.02$.

6.6 *Summary*

The multi-streaming framework proposed in this chapter is an application and transport cross-layer design, with a joint consideration on multi-resolution coding, scene-quality modeling, and partially sequenced and partially reliable transport. The presented work makes two main contributions. First, we proposed a computational and meaningful distortion metric, which evaluates the quality degradation of a multi-resolution 3D scene by a weighted summation of the distortion of individual objects that construct the scene. Second, more importantly, we proposed a multi-streaming mechanism with scalable partial-reliability allocation (PROpt) for transmitting 3D scene databases. This rate-distortion optimized framework greatly improves both streaming efficiency and application quality while providing transmission and rendering scalability, making it preferable in resource-constrained environments.

Although a view-dependent algorithm was employed to capture the relative weights of different objects in our presentation, the proposed framework is general and can incorporate different object weighting schemes depending upon particular applications. Our evaluation so far focused on 3D scenes with certain view perspectives and fixed object weights. Further investigations involving real-time updates of object weights will be conducted to accommodate adaptive 3D applications.

In PROpt, data chunks from the upper-layer application are treated preferentially by assigning unequal numbers of transmission opportunities, and retransmissions are performed at the transport level through the automatic repeat request (ARQ) mechanism [77]. During our presentation, we assumed an independent error process on the end-to-end network link. As another extension of the framework, correlated link-error models, e.g., Markov models, will be considered, which is important for evaluating the framework on wireless links. When a correlated link is considered, an application-level mechanism that schedules the transmission of multiple streams at each transmission opportunity may become desirable. Incorporating such application-level scheduling with partially reliable transport is another research topic to be investigated in our future work.

CHAPTER VII

CONCLUSIONS

In this chapter, we conclude the thesis with a summary of contributions and future research directions. In particular, in Section 7.1, we summarize how the proposed approaches fit within a general framework of providing scalable, error-resilient, and time-efficient solutions for streaming graphics applications. In Section 7.2, we outline two future research topics that expand the scopes of our proposed technologies.

7.1 Summarized Contributions

The proposed research followed three milestones. First, we tackled the problem of joint mesh and texture optimization. Then, we provided a solution for latency-minimized delivery of multi-resolution 3D models by developing a joint retransmission and unequal error protection system. Finally, we advanced from individual 3D models to 3D scene databases and proposed two object-based application and transport approaches. A detailed summary of the proposed approaches is given below.

7.1.1 Joint Mesh and Texture Optimization

In Chapter 3, we developed a bit-allocation framework for efficiently streaming textured 3D models in rate-constrained environment with maximal display quality. A novel and fast quality measure (FQM) was proposed to predict the visual fidelity of textured 3D models, which combines the distortion of both mesh geometry and texture image through an equalization factor estimated in the rendering space. FQM provides meaningful prediction on visual fidelity and generates a convex rate-quality surface for the progressively compressed model. Using FQM, bit-allocation algorithms were developed to find the proper distribution of mesh and texture data that maximizes the quality of the transmitted model under a bit-rate constraint. Two transmission stages, namely the streamed mode and the retained mode, are addressed under a rate-distortional framework. The streamed mode corresponds

to providing the best initial display on the user’s screen when the progressive transmission begins, while the retained mode has the objective of maintaining the visual fidelity of the displayed model at the maximal level during transmission. The experimental results justified that by properly estimating the relative importance of the mesh and the texture, the proposed bit-allocation framework quickly presents high-resolution visualization of textured 3D models with limited bit budgets, thus achieving higher efficiency in progressive transmission.

7.1.2 Latency-Minimized Delivery

In Chapter 4, we addressed the latency-minimized delivery of scalable 3D data by developing a hybrid retransmission and unequal error protection mechanism (named REP). The presented work made several contributions: *(i)* We analyzed intensively the property of multi-resolution 3D meshes and presented a TCP-friendly transmission system for multi-resolution meshes including a novel and meaningful quality measure; *(ii)* Given a distortion constraint, we derived a simplified expression (with assumptions) of the optimal FEC code that minimizes the expected transmission latency when combined with retransmissions; *(iii)* Observing a semi-infinite space for finding the theoretical optimal solution, we proposed an extended steepest decent search algorithm which quickly reaches the local optima in the solution space; *(iv)* Based on the above results for quality-critical scenarios, we further developed a time-critical streaming algorithm which significantly decreases the receiving distortion upon a strict delay constraint.

7.1.3 Streaming Content-Rich 3D Scenes

When moving from individual 3D models to content-rich 3D scene databases, we addressed three important properties in regard to the rendering and transmission scalability: *(i)* multiple objects in a 3D scene have unequal display importance; *(ii)* LODs of each individual object have decoding dependencies, and *(iii)* LODs for different objects have manipulation independencies.

Multi-Object Coding: The joint coding system presented in Chapter 5 addresses the properties of 3D scene databases by three components: adaptive vector quantization, embedded parity objects, and joint source and channel coding. Modeling the unequal decoding importance of mesh objects in a scene database, the proposed coding system provides preferential treatments among multiple objects and among various LODs of each object while preserving their decoding independencies in packet delivery. We showed that, by treating graphic objects jointly and preferentially in source and channel coding while preserving their decoding independencies, the proposed system reduces the receiving distortion of the 3D database significantly compared to conventional methods.

Multi-Streaming: The multi-streaming framework developed in Chapter 6 is an application and transport cross-layer design, with a joint consideration on multi-resolution coding, scene-quality modeling, and partially sequenced and partially reliable transport. The presented work made two main contributions. First, we proposed a computational and meaningful distortion metric, which evaluates the quality degradation of a multi-resolution 3D scene by a weighted summation of the distortion of individual objects that construct the scene. Second, more importantly, we proposed a multi-streaming mechanism with scalable partial-reliability allocation for transmitting 3D scene databases. This rate-distortion optimized framework greatly improves both streaming efficiency and application quality while providing transmission and rendering scalability, making it preferable in resource-constrained environments.

It should be noted that, by incorporating the fast quality measure (FQM) and the equalization factor presented in Chapter 3, the weighted distortion metric used for multi-resolution 3D scene databases and the multi-streaming framework can be extended to involve textured 3D models. In addition, following a similar philosophy as in Chapter 4, the multi-streaming framework can be further integrated with the joint coding system presented in Chapter 5 to obtain a hybrid retransmission and forward error correction system.

7.2 *Future Research Directions*

For all the proposed techniques, we have summarized possible improvements or extensions at the end of the corresponding chapters, which are not replicated here. At the end of this thesis, we present two future research topics that are relatively independent and desire advanced techniques to be built atop our proposed technologies.

7.2.1 **Joint Application-Transport Scheduling for Virtual Navigation**

View-dependent 3D streaming allows the navigation of large virtual environments without replicating the entire database at the client, which in general has limited computing power and storage. In the proposed research, we introduced multi-streaming to account for the decoding independencies of different objects, and used scalable partial reliability and joint source-channel coding to reflect the unequal rendering importance of the objects and their multi-resolution hierarchies. Implicitly, the streaming process was assumed to follow a round-robin fashion in the presented systems. In other words, transmission slots are allocated equally among multiple streams. Nevertheless, because of the continuous change of the viewpoint during navigation, objects in the virtual scenes have different decoding and rendering time lines, depending on their relative locations to the viewpoint and the user's interaction. As such, at a certain time instance, some objects may require more prompt delivery in order to be rendered on time, for which a larger number of transmission slots need to be allocated such that those objects can be displayed with a satisfactory quality.

Optimal scheduling of transmission is therefore desired for high-quality virtual navigation. With known or predicted motion of the viewpoint, the scheduling mechanism will infer the display time lines of the objects, and allocate the transmission slots accordingly under a certain optimization criterion. Apparently, implementing the scheduling mechanism requires joint actions of both application and transport layers, as a proprietary treatment should also be followed between multiple retransmission requests and between retransmission and newly issued transmission. This joint application-transport scheduling mechanism may further coordinate with the recently developed prefetching and caching techniques [20, 25] to provide the optimal use of the client device's computing resource.

7.2.2 Scalable Coding and Transmission for Animated Geometry

Animations exist in many 3D applications such as computer games and films, where relative vertex positions change over time while connectivity remains constant. Simple animations, e.g. rigid-body motion, can be described by equations that model the trajectories of the vertices. To provide realism, life-like soft-body animations are often needed, which involve human animators and physical simulation to build highly complex motions. Although it is possible to use all static compression and transmission schemes also on a frame-by-frame basis for such animations, temporal and spatial coherence should be explored to achieve higher compression efficiency as well as transmission and rendering scalability.

With the constant connectivity, the animated geometry can be described by a matrix, G_T , which stores the vertex positions of each frame in its columns [73]. The compression of the animated geometry can be achieved by several ways, with a common goal of reducing the dimension of the animation matrix. In the spatial domain, the geometry may be segmented into partitions and the animation of each partition can be approximated by rigid-body motion [58]. In the temporal domain, linear predictors or linear projections with principal component analysis (PCA) were used to exploit the inter-frame coherence [8, 47, 51]. Naturally, a synthesis of the two spirits provides higher compression efficiency [73].

The principal component analysis resembles vector quantization in the sense of regarding the animation matrix as a vector space and representing it with a small number of basis functions (or codewords in the vector quantization case). From the streaming perspective, it can be foreseen that the basis functions must be transmitted reliably or artifacts will be incurred in the decoded animation sequence. To provide transmission and rendering scalability, a progressive coding scheme is desirable to represent the basis functions in coarse-to-fine scales. Furthermore, in both spatial and temporal domains, a preferential treatment may be applied by taking into account the possibly unequal importance of different partitions and different frames, respectively, based upon the application properties. Our proposed approaches in this dissertation will serve as initial technologies toward the development of advanced techniques for streaming animation geometry with optimized transmission and rendering scalability.

APPENDIX A

SUPPLEMENTARY FOR CHAPTER III

A.1 Pseudo-Code of Finding the Boundary of a Feasible Region

Algorithm 1 $\mathcal{B}_{\mathcal{C}} = \text{Boundary}(\mathcal{C})$

```
 $\mathcal{B}_{\mathcal{C}} \leftarrow \Phi, i \leftarrow 0, j \leftarrow m$   
while  $i \leq n$  and  $\mathcal{R}_{\mathcal{G}}^{(i)} + \mathcal{R}_{\mathcal{T}}^{(j)} > \mathcal{C}$  do  
   $i \leftarrow i + 1$   
end while  
if  $i \leq n$  then  
  while  $j > 0$  and  $\mathcal{R}_{\mathcal{G}}^{(i)} + \mathcal{R}_{\mathcal{T}}^{(j-1)} \leq \mathcal{C}$  do  
     $j \leftarrow j - 1$   
  end while  
   $\mathcal{B}_{\mathcal{C}} \leftarrow \mathcal{B}_{\mathcal{C}} \cup (M^i, T^j)$   
  while  $i \leq n$  and  $j > 0$  do  
     $j \leftarrow j - 1$   
    while  $i \leq n$  and  $\mathcal{R}_{\mathcal{G}}^{(i)} + \mathcal{R}_{\mathcal{T}}^{(j)} > \mathcal{C}$  do  
       $i \leftarrow i + 1$   
    end while  
    if  $i \leq n$  then  
      while  $j > 0$  and  $\mathcal{R}_{\mathcal{G}}^{(i)} + \mathcal{R}_{\mathcal{T}}^{(j-1)} \leq \mathcal{C}$  do  
         $j \leftarrow j - 1$   
      end while  
       $\mathcal{B}_{\mathcal{C}} \leftarrow \mathcal{B}_{\mathcal{C}} \cup (M^i, T^j)$   
    end if  
  end while  
end if
```

APPENDIX B

SUPPLEMENTARY FOR CHAPTER V

B.1 Loss Recovery Probabilities

Proposition 1 in Chapter 5 presented a loose comparison of recovery probabilities between the parity-object-based joint FEC and the conventional FEC scheme that generates parity data for each source object separately. More analytically, in this appendix we derive recovery probabilities of the two FEC schemes for particular channel models. For simplicity, we refer to an equal error protection case. Nonetheless, the derivation can be easily applied to unequal error protection with modifications. We consider common FEC codes such as Reed-Solomon codes, where for every block of k data chunks $h = (n - k)$ parity chunks are generated to form an n -chunk FEC block. Such an (n, k) systematic code can recover up to h chunk losses (erasures) as positions of the lost chunks in the chunk sequence are known.

Figure 52 recaps the considered scenario where equal error protection is applied to s data streams. The conventional FEC appends h parity chunks for every block of k data chunks in each stream, respectively, which is referred to as in-stream FEC hereafter. Comparatively, a parity object generates hs parity chunks for the total number of ks data chunks concurrently. The solid and dashed arrows indicate chunk transmission orders in the two cases, respectively, following a round-robin scheme. In both cases, the entire FEC block under consideration has a total number of ns chunks. Let $\gamma_I(s)$, $\gamma_P(s)$ denote the probabilities that the entire block is correctly decoded in the separate-FEC and the parity-stream cases, respectively. Then $\gamma_I(s) = \Pr\{\leq h \text{ losses out of } n \text{ chunks in each of } s \text{ streams}\}$, and $\gamma_P(s) = \Pr\{\leq hs \text{ losses out of } ns \text{ chunks}\}$.

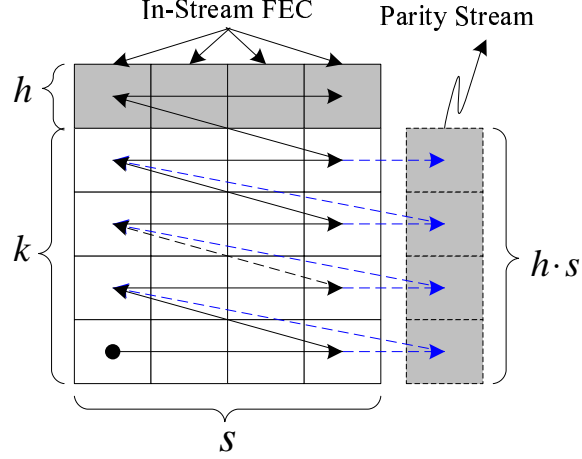


Figure 52: Equal error protection for four data streams using parity objects (streams) and separate FEC, comparatively.

B.1.1 Bernoulli Channel

If chunk losses in the ns -chunk block are modeled by a Bernoulli process with a mean chunk loss rate p , we have

$$\begin{aligned} \gamma_I(s) &= \left[\sum_{j=0}^h \binom{n}{j} p^j (1-p)^{n-j} \right]^s, \\ \gamma_P(s) &= \sum_{j=0}^{hs} \binom{ns}{j} p^j (1-p)^{ns-j}. \end{aligned} \quad (66)$$

It is easy to show that $\gamma_P(s)$ is monotonically increasing with the variable s , using the well known equality:

$$\binom{m}{\nu} = \binom{m-1}{\nu} + \binom{m-1}{\nu-1}, \quad 2 \leq \nu \leq m.$$

Therefore the following inequality holds:

$$\gamma_I(s) < \gamma_I(1) = \gamma_P(1) < \gamma_P(s), \quad s > 1, \quad (67)$$

which means that a higher recovering probability is attained with the parity object compared to separate FEC.

B.1.2 Gilbert Channel

For burst chunk losses, reaching a similar conclusion as (67) is not straightforward because of the interleaved transmission of each stream. Consider a simplified Gilbert model, which

is a two-state Markov model with transition probability matrix $\mathbf{P} = \begin{pmatrix} \beta & 1 - \beta \\ 1 - \alpha & \alpha \end{pmatrix}$ where α and β are the probabilities that chunk reception remains in failure and success states, respectively. The mean chunk loss rate p , the average burst-loss length r , and the correlation of two consecutive losses ρ are

$$p = \frac{1 - \beta}{1 - \alpha + 1 - \beta}, r = \frac{1}{1 - \alpha}, \text{ and } \rho = \alpha + \beta - 1. \quad (68)$$

Let $P(\nu, m)$ be the probability of ν losses in a sequence of m chunks, $P_G(\nu, m)$ be the probability of ν losses in m chunks ending with a successfully received chunk, and $P_B(\nu, m)$ be the probability of ν losses in m chunks ending with a lost chunk. Following the literature [101], we have

$$P(\nu, m) = P_G(\nu, m) + P_B(\nu, m). \quad (69)$$

For $m = 1, 2, 3, \dots$ and $\nu = 0, 1, 2, \dots, m$,

$$P_G(\nu, m) = \beta P_G(\nu, m - 1) + (1 - \alpha) P_B(\nu, m - 1), \quad (70)$$

$$P_B(\nu, m) = \alpha P_B(\nu - 1, m - 1) + (1 - \beta) P_G(\nu - 1, m - 1). \quad (71)$$

The initial conditions for the recursion are

$$P_G(0, 0) = \frac{1 - \alpha}{1 - \alpha + 1 - \beta}, P_B(0, 0) = \frac{1 - \beta}{1 - \alpha + 1 - \beta}, \quad (72)$$

and $P_G(\nu, 0) = P_B(\nu, 0) = P_B(0, m) = 0$ for $\nu \neq 0, m \neq 0$.

In the parity-stream case, since all the ns chunks form one FEC block, the recovering probability can be calculated by

$$\gamma_P(s) = \sum_{\nu=0}^{hs} P(\nu, ns). \quad (73)$$

The calculation of $\gamma_I(s)$, i.e., the recovering probability of the ns -chunk block in the case of in-stream FEC, is a bit complex as multiple streams are transmitted interleavingly (Figure 52) and chunk losses among the interleaved streams are inter-dependent. To account for the interleaving degree, it is necessary to know transition probabilities for two chunks

that are spaced apart by an arbitrary d -chunk time in transmission, which are shown in [101] to be

$$\alpha_d = p + \rho^d(1 - p), \text{ and } \beta_d = (1 - p) + \rho^d p. \quad (74)$$

Now, let $\tilde{Q}_I(s; \kappa_1, \kappa_2, \dots, \kappa_l)$ be the probability that streams $\kappa_1, \kappa_2, \dots, \kappa_l$ are correctly decoded among all s streams, and $\tilde{Q}_I(s; \kappa_i | \kappa_1, \kappa_2, \dots, \kappa_l)$ be the conditional probability that stream κ_i is recoverable given streams $\kappa_1, \kappa_2, \dots, \kappa_l$ are all recoverable. Herein we assume $\kappa_1 < \kappa_2 < \dots < \kappa_l$ without loss of generality. With the renewal property of a Markov process, it is easy to show that the following equality holds:

$$\tilde{Q}_I(s; \kappa_i | \kappa_1, \kappa_2, \dots, \kappa_l) = \tilde{Q}_I(s; \kappa_i | \kappa_1, \kappa_l), \quad \kappa_l < \kappa_i \leq s. \quad (75)$$

Applying (75), the probability $\gamma_I(s)$ can be expressed as

$$\begin{aligned} \gamma_I(s) &= \tilde{Q}_I(s; 1) \prod_{\kappa=2}^s \tilde{Q}_I(s; \kappa | 1, \kappa - 1) \\ &= \tilde{Q}_I(s; 1) \prod_{\kappa=2}^s \tilde{Q}_I(s; 1, \kappa - 1) \tilde{Q}_I(s; 1, \kappa - 1, \kappa). \end{aligned} \quad (76)$$

To evaluate (76) we need to calculate three types of probabilities: $\tilde{Q}_I(s; 1)$, $\tilde{Q}_I(s; 1, \kappa - 1)$ and $\tilde{Q}_I(s; 1, \kappa - 1, \kappa)$ for $3 \leq \kappa \leq s$. $\tilde{Q}_I(s; 1)$ can be calculated using (69)–(73) by replacing the transition probabilities with α_s and β_s , as given in (74), and setting $s = 1$ in (73). Similarly, the latter two quantities can be evaluated if *joint-loss* probabilities of multiple streams are known. Generally denoting such probabilities by $\tilde{P}_s(\nu_{\kappa_1}, \nu_{\kappa_2}, \dots, \nu_{\kappa_l}, n)$ where ν_{κ_j} , $j = 1, \dots, l$, is the number of losses in stream κ_j and n is the total number of chunks in each stream, we can rewrite $\gamma_I(s)$ as

$$\begin{aligned} \gamma_I(s) &= \sum_{\nu_1=0}^h \tilde{P}_s(\nu_1, n) \prod_{\kappa=2}^s \left[\sum_{\nu_1=0}^h \sum_{\nu_{\kappa-1}=0}^h \tilde{P}_s(\nu_1, \nu_{\kappa-1}, n) \right. \\ &\quad \left. \times \sum_{\nu_1=0}^h \sum_{\nu_{\kappa-1}=0}^h \sum_{\nu_{\kappa}=0}^h \tilde{P}_s(\nu_1, \nu_{\kappa-1}, \nu_{\kappa}, n) \right]. \end{aligned} \quad (77)$$

$\tilde{P}_s(\nu_1, n)$ in (77) is given by (69)–(72) with properly replaced transition probabilities. Following a similar logic as (69)–(72), and using the results in (74), one can also derive recursive expressions for $\tilde{P}_s(\nu_1, \nu_{\kappa-1}, n)$ and $\tilde{P}_s(\nu_1, \nu_{\kappa-1}, \nu_{\kappa}, n)$ by listing all possible ending states of the last chunks in the considered streams.

The evaluation of (77) needs to calculate probabilities $\tilde{P}(\nu_1, \nu_\kappa, m)$ and $\tilde{P}(\nu_1, \nu_\kappa, \nu_{\kappa+1}, m)$ for $2 \leq \kappa \leq s-1$. As an example, we show how a recursive expression is acquired for $\tilde{P}(\nu_1, \nu_\kappa, m)$. Using $\tilde{P}_{GG}(\nu_1, \nu_\kappa, m)$, $\tilde{P}_{GB}(\nu_1, \nu_\kappa, m)$, $\tilde{P}_{BG}(\nu_1, \nu_\kappa, m)$, and $\tilde{P}_{BB}(\nu_1, \nu_\kappa, m)$ to denote the probabilities of ν_1, ν_κ out of m losses in the two streams, respectively, with different ending states of the last two chunks, we have

$$\tilde{P}(\nu_1, \nu_\kappa, m) = \tilde{P}_{GG}(\nu_1, \nu_\kappa, m) + \tilde{P}_{GB}(\nu_1, \nu_\kappa, m) + \tilde{P}_{BG}(\nu_1, \nu_\kappa, m) + \tilde{P}_{BB}(\nu_1, \nu_\kappa, m). \quad (78)$$

For $m = 1, 2, 3, \dots$ and $\nu_1, \nu_\kappa = 0, 1, 2, \dots, m$,

$$\begin{aligned} \tilde{P}_{GG}(\nu_1, \nu_\kappa, m) &= \tilde{P}_{GG}(\nu_1, \nu_\kappa, m-1)\beta_{s+1-\kappa}\beta_{\kappa-1} \\ &+ \tilde{P}_{BG}(\nu_1, \nu_\kappa, m-1)\beta_{s+1-\kappa}\beta_{\kappa-1} \\ &+ \tilde{P}_{GB}(\nu_1, \nu_\kappa, m-1)(1-\alpha_{s+1-\kappa})\beta_{\kappa-1} \\ &+ \tilde{P}_{BB}(\nu_1, \nu_\kappa, m-1)(1-\alpha_{s+1-\kappa})\beta_{\kappa-1}, \end{aligned} \quad (79)$$

$$\begin{aligned} \tilde{P}_{GB}(\nu_1, \nu_\kappa, m) &= \tilde{P}_{GG}(\nu_1, \nu_\kappa-1, m-1)\beta_{s+1-\kappa}(1-\beta_{\kappa-1}) \\ &+ \tilde{P}_{BG}(\nu_1, \nu_\kappa-1, m-1)\beta_{s+1-\kappa}(1-\beta_{\kappa-1}) \\ &+ \tilde{P}_{GB}(\nu_1, \nu_\kappa-1, m-1)(1-\alpha_{s+1-\kappa})(1-\beta_{\kappa-1}) \\ &+ \tilde{P}_{BB}(\nu_1, \nu_\kappa-1, m-1)(1-\alpha_{s+1-\kappa})(1-\beta_{\kappa-1}), \end{aligned} \quad (80)$$

$$\begin{aligned} \tilde{P}_{BG}(\nu_1, \nu_\kappa, m) &= \tilde{P}_{GG}(\nu_1-1, \nu_\kappa, m-1)(1-\beta_{s+1-\kappa})(1-\alpha_{\kappa-1}) \\ &+ \tilde{P}_{BG}(\nu_1-1, \nu_\kappa, m-1)(1-\beta_{s+1-\kappa})(1-\alpha_{\kappa-1}) \\ &+ \tilde{P}_{GB}(\nu_1-1, \nu_\kappa, m-1)\alpha_{s+1-\kappa}(1-\alpha_{\kappa-1}) \\ &+ \tilde{P}_{BB}(\nu_1-1, \nu_\kappa, m-1)\alpha_{s+1-\kappa}(1-\alpha_{\kappa-1}), \end{aligned} \quad (81)$$

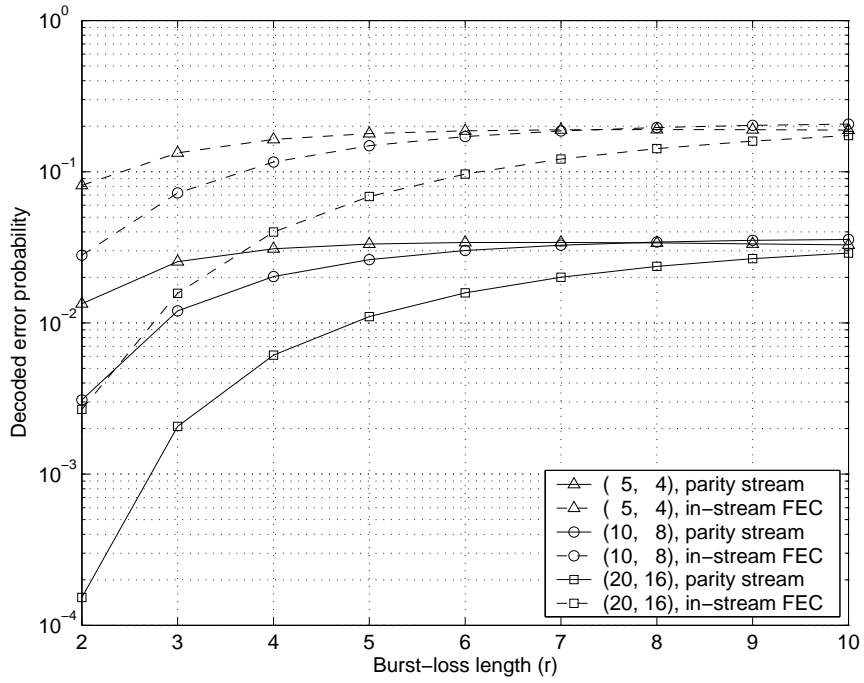
$$\begin{aligned} \tilde{P}_{BB}(\nu_1, \nu_\kappa, m) &= \tilde{P}_{GG}(\nu_1-1, \nu_\kappa-1, m-1)(1-\beta_{s+1-\kappa})\alpha_{\kappa-1} \\ &+ \tilde{P}_{BG}(\nu_1-1, \nu_\kappa-1, m-1)(1-\beta_{s+1-\kappa})\alpha_{\kappa-1} \\ &+ \tilde{P}_{GB}(\nu_1-1, \nu_\kappa-1, m-1)\alpha_{s+1-\kappa}\alpha_{\kappa-1} \\ &+ \tilde{P}_{BB}(\nu_1-1, \nu_\kappa-1, m-1)\alpha_{s+1-\kappa}\alpha_{\kappa-1}. \end{aligned} \quad (82)$$

The initial conditions for the recursion are

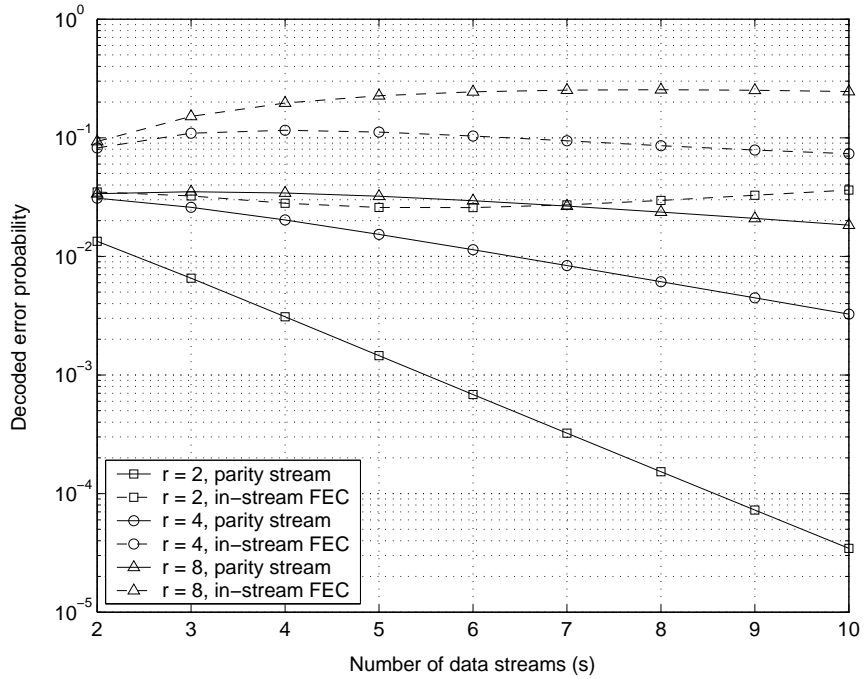
$$\begin{aligned}
\tilde{P}_{GG}(0,0,0) &= \frac{(1-\alpha)\beta_{\kappa-1}}{1-\alpha+1-\beta}, \\
\tilde{P}_{GB}(0,0,0) &= \frac{(1-\alpha)(1-\beta_{\kappa-1})}{1-\alpha+1-\beta}, \\
\tilde{P}_{BG}(0,0,0) &= \frac{(1-\beta)(1-\alpha_{\kappa-1})}{1-\alpha+1-\beta}, \\
\tilde{P}_{BB}(0,0,0) &= \frac{(1-\beta)\alpha_{\kappa-1}}{1-\alpha+1-\beta},
\end{aligned} \tag{83}$$

and $\tilde{P}_{NN}(\nu_1, \nu_\kappa, m) = 0$ for all other selections of (ν_1, ν_κ, m) that are not included in (79)–(83), with N being replaced by G or B accordingly.

Using (73) and (77), we perform numerical comparisons between $\gamma_I(s)$ and $\gamma_P(s)$ and present the results in Figure 53, where decoded error probabilities, $1 - Q_z(s)$ with $z = P$ or I , are plotted for different parameters. In particular, Figure 53(a) shows the variation of decoded error probability with respect to the average burst-loss length, while Figure 53(b) shows the variation of decoded error probability with respect to the number of data streams. Interestingly, it is observed that $\gamma_P(s)$ and $\gamma_I(s)$ may exhibit non-monotonicity upon different selection of parameters, particularly when the average burst-loss length increases or the FEC code has lower error-correcting ability. Nevertheless, under same conditions, using parity streams always achieves much lower error probabilities than in-stream FEC, confirming the efficacy of parity streams in providing error resilience for multiple data streams.



(a) $s = 4, p = 0.02$



(b) $(n, k) = (10, 8), p = 0.02$

Figure 53: Numerical comparisons of decoded error probabilities between in-stream FEC and parity streams in the burst-loss case.

APPENDIX C

SUPPLEMENTARY FOR CHAPTER VI

C.1 Proof of Equality (63)

$\tilde{\pi}_k^{(i-1)} = \{\rho_{k1}, \rho_{k2}, \dots, \rho_{kj}, \dots, \rho_{kL_k}\}$, and $\pi_k^{(i)}(j) = \{\rho'_{k1}, \rho'_{k2}, \dots, \rho'_{kj}, \dots, \rho'_{kL_k}\}$, where $\rho'_{kl} = \rho_{kl}, l \neq j$, and $\rho'_{kj} = \rho_{kj} + 1$.

$$\begin{aligned}
 \Delta E(\mathcal{R}|\pi_k^{(i)}(j)) &= E(\mathcal{R}|\pi_k^{(i)}(j)) - E(\mathcal{R}|\tilde{\pi}_k^{(i-1)}) \\
 &= \left[E(\mathcal{R}|\rho_{kj} + 1) + \sum_{l=1, l \neq j}^{L_k} E(\mathcal{R}|\rho_{kl}) \right] - \left[E(\mathcal{R}|\rho_{kj}) + \sum_{l=1, l \neq j}^{L_k} E(\mathcal{R}|\rho_{kl}) \right] \\
 &= E(\mathcal{R}|\rho_{kj} + 1) - E(\mathcal{R}|\rho_{kj}) \\
 &= \Delta \mathcal{R}_{kj} \left[(\rho_{kj} + 1)(1 - \epsilon_{kj})\epsilon^{\rho_{kj}} + (\rho_{kj} + 1)\epsilon^{(\rho_{kj}+1)} - \rho_{kj}\epsilon^{\rho_{kj}} \right] \quad (\text{Eqn. [57]}) \\
 &= \epsilon_{kj}^{\rho_{kj}} \Delta \mathcal{R}_{kj},
 \end{aligned}$$

$$\begin{aligned}
 \Delta E(\mathcal{D}|\pi_k^{(i)}(j)) &= E(\mathcal{D}|\pi_k^{(i)}(j)) - E(\mathcal{D}|\tilde{\pi}_k^{(i-1)}) \\
 &= \sum_{t=1}^{L_k} \mathcal{D}_{kt} \epsilon_{kt}^{\rho'_{kt}} \prod_{l=0}^{t-1} (1 - \epsilon_{kl}^{\rho'_{kl}}) - \sum_{t=1}^{L_k} \mathcal{D}_{kt} \epsilon_{kt}^{\rho_{kt}} \prod_{l=0}^{t-1} (1 - \epsilon_{kl}^{\rho_{kl}}) \quad (\text{Eqn. [59]}) \\
 &= \sum_{t=0}^{j-1} \left[\mathcal{D}_{kt} \epsilon_{kt}^{\rho'_{kt}} \prod_{l=0}^{t-1} (1 - \epsilon_{kl}^{\rho'_{kl}}) - \mathcal{D}_{kt} \epsilon_{kt}^{\rho_{kt}} \prod_{l=0}^{t-1} (1 - \epsilon_{kl}^{\rho_{kl}}) \right] \\
 &\quad + \left[\mathcal{D}_{kj} \epsilon_{kj}^{\rho'_{kj}} \prod_{l=0}^{j-1} (1 - \epsilon_{kl}^{\rho'_{kl}}) - \mathcal{D}_{kj} \epsilon_{kj}^{\rho_{kj}} \prod_{l=0}^{j-1} (1 - \epsilon_{kl}^{\rho_{kl}}) \right] \\
 &\quad + \sum_{t=j+1}^{L_k} \left[\mathcal{D}_{kt} \epsilon_{kt}^{\rho'_{kt}} \prod_{l=0}^{t-1} (1 - \epsilon_{kl}^{\rho'_{kl}}) - \mathcal{D}_{kt} \epsilon_{kt}^{\rho_{kt}} \prod_{l=0}^{t-1} (1 - \epsilon_{kl}^{\rho_{kl}}) \right] \\
 &= 0 + \left[\mathcal{D}_{kj} \epsilon_{kj}^{(\rho_{kj}+1)} \prod_{l=0}^{j-1} (1 - \epsilon_{kl}^{\rho_{kl}}) - \mathcal{D}_{kj} \epsilon_{kj}^{\rho_{kj}} \prod_{l=0}^{j-1} (1 - \epsilon_{kl}^{\rho_{kl}}) \right] \\
 &\quad + \sum_{t=j+1}^{L_k} \left[\frac{1 - \epsilon^{(\rho_{kj}+1)}}{1 - \epsilon^{\rho_{kj}}} \mathcal{D}_{kt} \epsilon_{kt}^{\rho_{kt}} \prod_{l=0}^{t-1} (1 - \epsilon_{kl}^{\rho_{kl}}) - \mathcal{D}_{kt} \epsilon_{kt}^{\rho_{kt}} \prod_{l=0}^{t-1} (1 - \epsilon_{kl}^{\rho_{kl}}) \right] \\
 &= (\epsilon_{kj} - 1) \cdot \mathcal{D}_{kj} \epsilon_{kj}^{\rho_{kj}} \prod_{l=0}^{j-1} (1 - \epsilon_{kl}^{\rho_{kl}}) + \epsilon_{kj}^{\rho_{kj}} \frac{1 - \epsilon_{kj}}{1 - \epsilon_{kj}^{\rho_{kj}}} \cdot \sum_{t=j+1}^{L_k} \mathcal{D}_{kt} \epsilon_{kt}^{\rho_{kt}} \prod_{l=0}^{t-1} (1 - \epsilon_{kl}^{\rho_{kl}}).
 \end{aligned}$$

REFERENCES

- [1] ABASOLO, M. J. and PERALES, F., “Geometric-textured bitree: Transmission of a multiresolution terrain across the internet,” *Journal of Computer Science and Technology*, vol. 2, no. 7, pp. 34–41, 2002.
- [2] AL-REGIB, G. and ALTUNBASAK, Y., “An unequal error protection method for packet loss resilient 3-D mesh transmission,” in *Proc. IEEE INFOCOM*, vol. 2, pp. 743–752, 2002.
- [3] AL-REGIB, G. and ALTUNBASAK, Y., “3TP: An application-layer protocol for streaming 3-D graphics,” in *Proc. IEEE Int. Conf. on Multimedia and Expo*, vol. 1, pp. 421–424, 2003.
- [4] AL-REGIB, G. and ALTUNBASAK, Y., “3TP: 3-D models transport protocol,” in *Proc. Web3D*, pp. 155–162, 2004.
- [5] AL-REGIB, G., ALTUNBASAK, Y., and MERSEREAU, R. M., “Bit allocation for joint source and channel coding of progressively compressed 3-D models,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 15, no. 2, pp. 256–268, 2005.
- [6] AL-REGIB, G., ALTUNBASAK, Y., and ROSSIGNAC, J., “A joint source and channel coding approach for progressively compressed 3d mesh transmission,” in *Proc. IEEE International Conference on Image Processing*, vol. 2, pp. 161–164, 2002.
- [7] AL-REGIB, G., ALTUNBASAK, Y., ROSSIGNAC, J., and MERSEREAU, R. M., “Encoding of 3D animations for efficient delivery,” in *Proc. IEEE Int. Conf. on Image Processing*, vol. 1, pp. 353–356, 2002.
- [8] ALEXA, M. and MULLER, W., “Representing animations by principal components,” *Computer Graphics Forum*, vol. 19, no. 3, pp. 411–418, 2000.
- [9] ALLIEZ, P. and DESBRUN, M., “Progressive compression for lossless transmission of triangle meshes,” in *Proc. Computer Graphics and Interactive Techniques*, pp. 195–202, 2001.
- [10] ALREGIB, G. and ALTUNBASAK, Y., “3TP: An application-layer protocol for streaming 3-D graphics,” *IEEE Trans. Multimedia*, vol. 7, no. 6, pp. 1149–1156, 2005.
- [11] ALREGIB, G., ALTUNBASAK, Y., and ROSSIGNAC, J., “An unequal error protection method for progressively compressed 3D models,” *IEEE Trans. Multimedia*, vol. 7, no. 4, pp. 766–776, 2005.
- [12] ALREGIB, G. and TIAN, D., “Latency-minimized delivery of multi-resolution mesh geometry,” in *Proc. IEEE Int. Conf. on Acoustic, Speech, and Signal Processing*, vol. 5, (Toulouse, France), pp. 369–372, May 2006.

- [13] ALREGIB, G. and TIAN, D., “Multi-streaming of visual scenes with scalable partial reliability,” in *Proc. IEEE Int. Conf. on Acoustic, Speech, and Signal Processing*, (Atlanta, GA), October 2006.
- [14] ALREGIB, G., “Delay-constrained 3-D graphics streaming over lossy networks.” Ph.D. dissertation, Georgia Institute of Technology, 2002.
- [15] ATIQUZZAMAN, M. and IVANCIC, W., “Evaluation of sctp multistreaming over wireless/satellite links,” in *Proc. Computer Communications and Networks*, pp. 591–594, 2003.
- [16] BAJAJ, C. L., CUTCHIN, S., PASCUCCI, V., and ZHUANG, G., “Error resilient transmission of compressed VRML,” *Technical report, TICAM, The University of Texas at Austin*, 1998.
- [17] BALMELLI, L., “Rate-distortion optimal mesh simplification for communications.” Ph.D. dissertation No 2260, Ecole Polytechnique Federale de Lausanne, Switzerland, 2001.
- [18] BISCHOFF, S. and KOBBELT, L., “Streaming 3D geometry data over lossy communication channels,” in *Proc. IEEE Int. Conf. on Multimedia and Expo*, vol. 1, pp. 361–364, 2002.
- [19] CARO, A. L., IYENGAR, J. R., AMER, P. D., LADHA, S., HEINZ, G. J., and SHAH, K. C., “SCTP: a proposed standard for robust internet data transport,” *IEEE Computer*, vol. 36, no. 11, pp. 56–63, 2003.
- [20] CHAN, A., LAU, R. W. H., and NG, B., “A hybrid motion prediction method for caching and prefetching in distributed virtual environments,” in *Proc. ACM Symposium on Virtual Reality Software and Technology*, pp. 135–142, 2001.
- [21] CHEN, B.-Y. and NISHITA, “Multiresolution streaming mesh with shape preserving and QoS-like controlling,” in *Proc. Web3D*, pp. 35–42, 2002.
- [22] CHEN, Z., BARNES, F. J., and BODENHEIMER, B., “Hybrid and forward error correction transmission techniques for unreliable transport of 3D geometry,” *Multimedia Systems*, vol. 10, no. 2, pp. 230–244, 2005.
- [23] CHEN, Z., BODENHEIMER, B., and BARNES, F. J., “Extending progressive meshes for use over unreliable networks,” in *Proc. IEEE Int. Conf. on Multimedia and Expo*, vol. 3, pp. 253–256, 2003.
- [24] CHEN, Z., BODENHEIMER, B., and BARNES, F. J., “Robust transmission of 3D geometry over lossy networks,” in *Proc. Web3D*, pp. 161–172, 2003.
- [25] CHIM, J., LAU, R. W. H., LEONG, H. V., and Si, A., “CyberWalk: a web-based distributed virtual walkthrough environment,” *IEEE Trans. Multimedia*, vol. 5, no. 4, pp. 503–515, 2003.
- [26] CHOU, P. H. and MENG, T. H., “Vertex data compression through vector quantization,” *IEEE Trans. Visualization and Computer Graphics*, vol. 8, no. 4, pp. 373–382, 2002.

- [27] CIGNONI, P., ROCCHINI, C., and SCOPIGNO, R., “Metro: measuring error on simplified surfaces,” in *Proc. Eurographics*, vol. 17(2), pp. 167–174, 1998.
- [28] COHEN, J., OLANO, M., and MANOCHA, D., “Appearance-preserving simplification,” in *Proc. ACM SIGGRAPH*, pp. 115–122, 1998.
- [29] COHEN-OR, D., CHRYSANTHOU, Y., and SILVA, C., “A survey of visibility for walk-through applications.” *Proc. EUROGRAPHICS’00*, course notes, 2000.
- [30] DEERING, M., “Geometry compression,” in *Proc. ACM SIGGRAPH*, pp. 13–20, August 1995.
- [31] ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., and STUETZLE, W., “Multiresolution analysis of arbitrary meshes,” in *Proc. ACM SIGGRAPH*, pp. 173–182, 1995.
- [32] ESKICIOGLU, A. M. and FISHER, P. S., “Image quality measures and their performance,” *IEEE Trans. Communications*, vol. 43, no. 12, pp. 2959–2965, 1995.
- [33] FLOYD, S. and FALL, K., “Promoting the use of end-to-end congestion control in the internet,” *IEEE/ACM Trans. Networking*, vol. 7, no. 4, pp. 458–472, 1999.
- [34] FLOYD, S., HANDLEY, M., PADHYE, J., and WIDMER, J., “Equation-based congestion control for unicast applications,” in *Proc. ACM SIGCOMM*, pp. 43–56, 2000.
- [35] FLOYD, S. and JACOBSON, V., “Random early detection gateways for congestion avoidance,” *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 397–413, 1993.
- [36] FU, S.-J. and ATIQUZZAMAN, M., “SCTP: state of the art in research, products, and technical challenges,” *IEEE Communications Magazine*, vol. 42, no. 4, pp. 64–76, 2004.
- [37] GAN, T. and MA, K.-K., “Weighted unequal error protection for transmitting scalable object-oriented images over packet-erasure networks,” *IEEE Trans. Image Processing*, vol. 14, no. 2, pp. 189–199, 2005.
- [38] GARLAND, M. and HECKBERT, P., “Surface simplification using quadric error metrics,” in *Proc. ACM SIGGRAPH*, pp. 209–216, 1997.
- [39] GARLAND, M. and HECKBERT, P., “Simplifying surfaces with color and texture using quadratic error metric,” in *Proc. IEEE Visualization*, pp. 287–295, October 19–20 1998.
- [40] GOTTSCHALK, S., LIN, M. C., and MANOCHA, D., “OBBTree: A hierarchical structure for rapid interference detection,” in *Proc. ACM SIGGRAPH*, pp. 171–180, 1996.
- [41] GUSKOV, I., SWELDENS, W., and SCHRÖDER, P., “Multiresolution signal processing for meshes,” in *Proc. ACM SIGGRAPH*, pp. 325–334, 1999.
- [42] HANDLEY, M., FLOYD, S., PADHYE, J., and WIDMER, J., *TCP friendly rate control (TFRC): protocol specification*. Request for Comments (RFC) 3448, The Internet Society, January 2003.

- [43] HARRIS, A. F. and KRAVETS, R., “The design of a transport protocol for on-demand graphical rendering,” in *Proc. NOSSDAV*, pp. 43–49, 2002.
- [44] HOPPE, H., “Progressive mesh,” in *Proc. ACM SIGGRAPH*, pp. 99–108, 1996.
- [45] HOPPE, H., “New quadric metric for simplifying meshes with appearance attributes,” in *Proc. IEEE Visualization*, pp. 59–66, 1999.
- [46] HORN, U., STUHLMULLER, K., LINK, M., and GIROD, B., “Robust internet video transmission based on scalable coding and unequal error protection,” *Signal Processing: Image Communications*, vol. 15, pp. 77–94, 1999.
- [47] IBARRIA, L. and ROSSIGNAC, J., “Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity,” in *Proc. ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 126–135, 2003.
- [48] ISO/IEC-JTC1/SC29/WG11, *Generic coding of audio-visual objects – Part 2: Visual*. ISO/IEC 14496-2, 1999.
- [49] JACOBSON, V., “Congestion avoidance and control,” in *Proc. ACM SIGCOMM*, pp. 314–329, 1988.
- [50] JAROMERSKY, P., WU, X., CHIANG, Y.-J., and MEMON, N., “Multiple-description geometry compression for networked interactive 3D graphics,” in *Proc. IEEE Int. Conf. on Image and Graphics*, pp. 468–471, 2004.
- [51] KARNI, Z. and GOTSMAN, C., “Compression of soft-body animation sequences,” *Computers and Graphics*, vol. 28, pp. 25–34, 2004.
- [52] KHODAKOVSKY, A. and GUSKOV, I., “Normal mesh compression,” in *Geometric Modeling for Scientific Visualization*, Springer-Verlag, 2002.
- [53] KHODAKOVSKY, A., SCHRODER, P., and SWELDENS, W., “Progressive geometry compression,” in *Proc. ACM SIGGRAPH*, pp. 271–278, 2000.
- [54] KIM, J., MERSEREAU, R. M., and ALTUNBASAK, Y., “Error-resilient image and video transmission over the internet using unequal error protection,” *IEEE Trans. Image Processing*, vol. 12, no. 2, pp. 121–131, 2003.
- [55] LAFRUIT, G., DELFOSSE, E., OSORIO, R., RAEMDONCK, W. V., FERENTINOS, V., and BORMANS, J., “View-dependent, scalable texture streaming in 3-D QoS with MPEG-4 visual texture coding,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 14, no. 7, pp. 1021–1031, 2004.
- [56] LEE, A. W. F., SWELDENS, W., SCHRODER, P., COWSAR, L., and DOBKIN, D., “Maps: Multiresolution adaptive parameterization of surfaces,” in *Proc. ACM SIGGRAPH*, pp. 95–104, 1998.
- [57] LEE, E.-S. and KO, H.-S., “Vertex data compression for triangular meshes,” in *Pacific Graphics*, pp. 225–234, 2000.
- [58] LENGYEL, J. E., “Compression of time-dependent geometry,” in *Proc. ACM Symposium on Interactive 3D graphics*, pp. 89–95, 1999.

- [59] LEVY, B., “Constrained texture mapping for polygonal meshes,” in *Proc. ACM SIGGRAPH*, pp. 417–424, 2001.
- [60] LI, J. and KUO, C.-C., “Progressive coding of 3-D graphic models,” *Proceedings of the IEEE*, vol. 86, no. 6, pp. 1052–1063, 1998.
- [61] LI, J., TIAN, D., and ALREGIB, G., “Adaptive multi-resolution coding for 3D scenes using vector quantization,” in *Proc. IEEE Int. Conf. on Acoustic, Speech, and Signal Processing*, (Atlanta, GA), October 2006.
- [62] LI, J., TIAN, D., and ALREGIB, G., “Vector quantization in multi-resolution mesh compression,” *IEEE Signal Processing Letters*, vol. 13, October 2006.
- [63] LINDSTROM, P., “Model simplification using image and geometry-based metrics.” Ph.D. Dissertation, Georgia Institute of Technology, 2000.
- [64] MAILLOT, J., YAHIA, H., and VERROUST, A., “Interactive texture mapping,” in *Proc. ACM SIGGRAPH*, vol. 27, 1993.
- [65] MARKA, M. and FOWLER, J. E., “Unequal error protection of embedded multimedia objects for packet-erasure channels,” in *Proc. Int. Workshop on Multimedia Signal Processing*, pp. 61–64, 2002.
- [66] MOHR, A. E., RISKIN, E. A., and LADNER, R. E., “Unequal loss protection: Graceful degradation of image quality over packet erasure channels through forward error correction,” *IEEE J. Selected Areas in Communications*, vol. 18, no. 6, pp. 819–828, 2000.
- [67] OKUDA, M. and CHEN, T., “Joint geometry/texture progressive coding of 3d models,” in *Proc. IEEE Int. Conf. on Image Processing*, pp. 632–635, 2000.
- [68] PAJAROLA, R. and ROSSIGNAC, J., “Compressed progressive meshes,” *IEEE Trans. Visualization and Computer Graphics*, vol. 6, no. 1, pp. 79–93, 2000.
- [69] REJAIE, R., HANDLEY, M., and ESTRIN, D., “RAP: an end-to-end rate-based congestion control mechanism for realtime streams in the internet,” in *Proc. IEEE INFOCOM*, vol. 3, pp. 1337–1345, 1999.
- [70] RIZZO, L., “Effective erasure codes for reliable computer communication protocols,” *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 2, pp. 24–36, 1997.
- [71] ROSSIGNAC, J., “Edgebreaker: connectivity compression for triangle meshes,” *IEEE trans. visualization and computer graphics*, vol. 5, pp. 47–61, January 1999.
- [72] SAID, A. and PEARLMAN, W. A., “A new, fast, and efficient image codec based on set partitioning in hierarchical trees,” *IEEE Trans. Circuits and Systems for Video Tech.*, vol. 6, no. 3, pp. 243–250, 1996.
- [73] SATTLE, M., SARLETTE, R., and KLEIN, R., “Simple and efficient compression of animation sequences,” in *Proc. ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 209–217, 2005.
- [74] SEGAL, M. and AKELEY, K., *The OpenGL Graphics System: A Specification (Version 1.2.1)*. Silicon Graphics, Inc., April 1999.

- [75] SHAPIRO, J. M., “Embedded image coding using zerotrees of wavelets coefficients,” *IEEE Trans. Signal Processing*, vol. 41, pp. 3445–3462, 1993.
- [76] SKODRAS, A., CHRISTOPOULOS, C., and EBRAHIMI, T., “The JPEG 2000 still image compression standard,” *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, 2001.
- [77] STEWART, R., RAMALHO, M., XIE, Q., TUEXEN, M., and CONRAD, P., *Stream control transmission protocol (SCTP) partial reliability extension*. Request for Comments (RFC) 3758, The Internet Society, May 2004.
- [78] STEWART, R., XIE, Q., MORNEAULT, K., SHARP, C., and ET AL., H. S., *Stream control transmission protocol*. Request for Comments (RFC) 2960, The Internet Society, October 2000.
- [79] TAUBIN, G., “3D geometry compression and progressive transmission,” in *Proc. Eurographics*, 1999.
- [80] TAUBIN, G., GUEZIEC, A., HORN, W., and LAZARUS, F., “Progressive forest split compression,” in *Proc. ACM SIGGRAPH*, pp. 123–132, 1998.
- [81] TAUBIN, G. and ROSSIGNAC, J., “Geometry compression through topological surgery,” *ACM Trans. Graphics*, vol. 17, pp. 84–115, April 1998.
- [82] TIAN, D. and ALREGIB, G., “FQM: a fast quality measure for efficient transmission of textured 3D models,” in *Proc. ACM Multimedia*, (New York, NY), pp. 684–691, October 2004.
- [83] TIAN, D. and ALREGIB, G., “Progressive streaming of textured 3D models over bandwidth-limited channels,” in *Proc. IEEE Int. Conf. on Acoustic, Speech, and Signal Processing*, vol. 2, (Philadelphia, PA), pp. 1289–1291, May 2005.
- [84] TIAN, D. and ALREGIB, G., “On-demand transmission of 3D models over lossy networks,” *Signal Processing: Image Communication*, vol. 21, pp. 396–415, June 2006.
- [85] TIAN, D. and ALREGIB, G., “Parity-object embedded streaming for synthetic graphics,” in *Proc. IEEE Int. Conf. on Acoustic, Speech, and Signal Processing*, (Atlanta, GA), October 2006.
- [86] TIAN, D. and ALREGIB, G., “Parity streams - a novel FEC scheme with the stream control transmission protocol,” *IEEE Communications Letters*, vol. 10, pp. 498–500, June 2006.
- [87] TIAN, D. and ALREGIB, G., “PODS: partially ordered delivery for 3D scenes in resource-constrained environments,” in *Proc. IEEE Int. Conf. on Acoustic, Speech, and Signal Processing*, vol. 5, (Toulouse, France), pp. 413–416, May 2006.
- [88] TIAN, D. and ALREGIB, G., “BATX3: bit-allocation for textured 3D models,” *IEEE Trans. Circuits and Systems for Video Technology*, submitted.
- [89] TIAN, D. and ALREGIB, G., “Multi-streaming of 3D scenes with optimized transmission and rendering scalability,” *IEEE Trans. Multimedia*, to appear.

- [90] TIAN, D., LI, J., and ALREGIB, G., “Joint source and channel coding for 3D scene databases using vector quantization and embedded parity objects,” *IEEE Trans. Image Processing*, submitted.
- [91] TOUMA, C. and GOTSMAN, C., “Triangle mesh compression,” in *Proc. Graphics Interface*, (Vancouver, Canada), June 1998.
- [92] UCB/LBNL/VINT, *network simulator ns (version 2)*. Available at URL <http://www.isi.edu/nsnam/ns/>.
- [93] VALETTE, S. and PROST, R., “Wavelet-based progressive compression scheme for triangle meshes: wavemesh,” *IEEE Trans. Visualization and Computer Graphics*, vol. 10, no. 2, pp. 123–129, 2004.
- [94] VARAKLIOTIS, S., HAILES, S., and OSTERMANN, J., “Optimally smooth error resilient streaming of 3-D wireframe animation,” in *Proc. SPIE Visual Communications and Image Processing*, vol. 5150, pp. 1009–1022, 2003.
- [95] WALLACE, G. K., “The JPEG still picture compression standard,” *Communications of the ACM*, vol. 34, no. 4, pp. 30–34, 1991.
- [96] WATSON, B., FRIEDMAN, A., and MCGAFFEY, A., “Measuring and predicting visual fidelity,” in *Proc. ACM SIGGRAPH*, pp. 213–220, 2001.
- [97] YAN, Z., KUMAR, S., and KUO, C.-C., “Error-resilient coding of 3-D graphic models via adaptive mesh segmentation,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 7, pp. 860–873, 2001.
- [98] YAN, Z., KUMAR, S., and KUO, C.-C., “Mesh segmentation schemes for error resilient coding of 3-D graphic models,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 15, no. 1, pp. 138–144, 2005.
- [99] YANG, S. and KUO, C.-C., “Robust graphics streaming in walkthrough virtual environments via wireless channels,” in *Proc. IEEE Global Telecommunications Conference*, vol. 1, pp. 3191–3195, 2003.
- [100] YANG, S., LEE, C.-H., and KUO, C.-C., “Optimized mesh and texture multiplexing for progressive textured model transmission,” in *Proc. ACM Multimedia*, pp. 676–683, 2004.
- [101] YEE, J. R. and WELDON, E. J., “Evaluation of the performance of error-correcting codes on a Gilbert channel,” *IEEE Trans. Communications*, vol. 43, no. 8, pp. 2316–2323, 1995.
- [102] ZEGER, K., VAISEY, J., and GERSHO, A., “Globally optimal vector quantizer design by stochastic relaxation,” *IEEE Trans. Signal Processing*, vol. 40, no. 2, pp. 310–322, 1992.
- [103] ZHANG, H., “Effective occlusion culling for the interactive display of arbitrary models.” Ph.D. dissertation, Department of Computer Science, UNC-Chapel Hill, 1998.

VITA

Dihong Tian was born and grew up in Chishui, Guizhou, a small city in the southwest of China. With his prize in the National Chemistry Olympiad for high school students, he entered the University of Science and Technology of China (USTC) in 1995 as a waiver of the national college entrance exam. At USTC, he studied in the Special Class for the Gifted Young, known as SCGY, for 4 years and received a B.S.E. degree in computer science with the highest honor. From 1999 to 2002, he was studying toward an M.S.E. degree at USTC, with a major in wireless communications. He received his Master's degree in 2002.

Since August 2002, he has been with the Center for Signal and Image Processing (CSIP) at Georgia Institute of Technology, pursuing a Ph.D. degree in electrical and computer engineering. His Ph.D. research is devoted to multimedia communications and networking, particularly on streaming video and three-dimensional (3D) graphics. He received an M.S. degree in electrical and computer engineering from Georgia Tech in December 2003. He is the recipient of CSIP Outstanding Research Award in Fall 2005.