QoS-ENABLED STREAMING OF ANIMATED 3-D WIREFRAME MODELS

Socrates Varakliotis

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

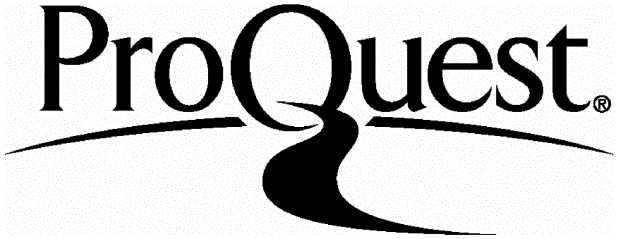
of the

University of London.

Department of Computer Science

University College London

July 7, 2004



To Ioanna and Giorgos.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Stephen Hailes, who provided diligent guidance, support, and feedback on the ideas developed in this work. His insightful comments, suggestions and constructive criticism have helped to finely shape this dissertation. I also thank my past supervisor Vicky Hardman for helping lay the foundations of this work.

My gratitude is expressed to Jörn Ostermann for donating his expertise during endless, and often late night, transatlantic phone discussions. His patience and academic support were inspirational and are reflected in this thesis.

Many thanks are also due to Andrea Basso and other colleagues in AT&T Labs-Research for lively and intriguing discussions. They all contributed to a very pleasant and stimulating research environment during my internship in their Lab.

This thesis is also influenced by comments from many researchers and reviewers around the world. Among them prof. Thomas Sikora and Jörg Widmer made fruitful comments and shared long discussions. Jerome Lengyel and Jeong-Hwan Ahn kindly made their animated 3-D models available to this research. Michael Garland shared his QSlim decimation software. Several colleagues participating in MPEG-4 also provided assistance, in particular Alexandre Cotarmanac'h, Jean-Claude Dufourd and Cyril Concolato.

I am especially indebted to Orion Hodson who provided invaluable mentoring and fresh technical discussions during my early stages in the department.

Fellow researchers and staff of the Networked Multimedia group at UCL Computer Science made numerous comments, suggestions and contributions. Dimitrios Miras, Colin Perkins, and Jon Crowcroft commented on multiple issues in the past years. Adam Greenhalgh was a good desk neighbour and provided considerable technical support. Marcelo Pias also answered many Linux-related questions.

This work was partially supported by the "Alexander Onassis Public Benefit Foundation", AT&T Labs-Research, British Telecom JISC projects, and the EPSRC.

Last, but not least, I owe much of this thesis to my parents, to whom I dedicate it, for their endless patience.

Publications

1. S. Varakliotis, S. Hailes, J. Ostermann.
   "Progressive coding for QoS-enabled streaming of dynamic 3-D meshes."
   In Proc. of *IEEE International Conference on Communications* (ICC).
   June 2004, Paris.

2. S. Varakliotis, S. Hailes, J. Ostermann.
   "Optimally Smooth Error Resilient Streaming of 3-D Wireframe Animations."
   In Proc. of *SPIE Visual Communications & Image Processing* (VCIP).
   July 2003, Lugano.

3. S. Varakliotis, J. Ostermann, S. Hailes.
   "Repair Options for 3-D Wireframe Model Animation Sequences."
   In Proc. of 3^{rd} *IEEE International Conference in Multimedia and Expo* (ICME).
   August 2002, Lausanne.

4. S. Varakliotis, J. Ostermann, V. Hardman.
   "Coding of Animated 3-D Wireframe Models for Internet Streaming Applications."
   In Proc. of 2^{nd} *IEEE International Conference in Multimedia and Expo* (ICME).
   August 2001, Tokyo.

5. A. Basso, S. Varakliotis, R. Castagno, F. Lohan.
   "Transport of MPEG-4 over IP/RTP."
   In *European Transactions on Telecommunications*: special issue on Packet Video 2000.
   April 2001.

6. A. Basso, S. Varakliotis.
   "Transport of MPEG-4 over IP/RTP."
   In Proc. of 1^{st} *IEEE International Conference in Multimedia and Expo* (ICME).
   July 2000, New York.

7. S. Varakliotis, A. Basso, V. Hardman.
   "Delivering MPEG-4 content over IP: An architecture and issues towards standardization."
   In Proc. of *IEEE International Conference on Telecommunications* (ICT).
   May 2000, Acapulco.

8. A. Basso, S. Varakliotis, R. Castagno.
   "Transport of MPEG-4 over IP/RTP."
   In Proc. of *Packet Video Workshop*.
   May 2000, Cagliari.

"...

The most compelling shapes are those near to our hearts: peoples faces, a gracefully moving body, a natural scene with rustling leaves and flowing water. Evolution has tuned us to these sights. By combining vision and graphics, capturing and creating images of these scenes may soon be within reach.

And once we have these powerful tools for creation and manipulation in hand, perhaps we will be one step closer to the best possible tool for the imagination."

JED LENGYEL

The Convergence of Graphics and Vision

IEEE Computer, p.53, July 1998.

Abstract

The rapid growth of end systems processing power and the development of state-of-the-art animation modelling packages gave birth to the field of animated 3-D graphics. However, the considerable amount of data involved in 3-D animations, combined with lack of inherent QoS support in the best-effort Internet, has meant that 3-D animation is, at best, still an emerging network stream type. Indeed, it is currently going through the early stages of the same development process undergone by what are now regarded as 'traditional' multimedia stream types in the 1990s. Naivety in the transmission of 3-D streams means that the network is exposed to a risk of considerably increased congestion, along with serious concerns about possible effects on existing well-behaved bulk data, or legacy audiovisual streaming media traffic, for which a largely acceptable degree of QoS has been achieved over the last decade.

In order for networked animation to become a reality, it is necessary (i) to encode 3-D animation streams in a way that is compact, (ii) to ensure that congestive loss suffered on the network has as little perceptual impact as possible and, (iii) to ensure that the overall perceptual quality of the animation remains acceptable in the presence of fluctuating bandwidth. We present solutions to each part.

In order to ground the individual pieces of practical work that we have undertaken within an appropriate context, we propose and evaluate an end-to-end architecture suitable for streaming the novel data type of 3-D wireframe animations over the Internet. We then explain how we instantiate this architecture in order to (i) allow the representation of time evolving 3-D geometry data at the source in a single resolution stream, (ii) achieve high animation quality by maintaining optimal error control over the model's surface smoothness by adding appropriate redundancy and, (iii) demonstrate Internet-friendly behaviour by finely adjusting the encoded animation rate at the source, thus gracefully adapting to a 3-D model quality corresponding to the available channel bandwidth. The rate controller operates in tandem with a source coding mechanism based on 3-D mesh decimation principles, which trades quality by reducing the number of geometric constructs encoded in the bitstream. The decimated geometric constructs are further predicted by their neighbours in an appropriately designed receiver, which is also capable of recovering from limited frame losses.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over recent years, the increasing power of end systems has driven the growth in the use of multimedia over the Internet. Early text-based communication over low-bandwidth links, has been constantly evolving towards a richer, interactive, real-time, audio-visual collaboration through a virtual world. We experience this transformation of the Internet daily by generating and consuming a range of digital media from a spectrum of applications, at work, in education, and for entertainment.

Visual information exchange in this networked world goes beyond static 2-D images or natural video. Now 3-D graphics add realism and allow interactivity, especially in software applications for distributed industrial design, medical imaging, and even game playing. In parallel, specialised hardware has been developed, and devices such as laser range scanners can facilitate the data acquisition process of intricate physical objects and produce consistent polyhedral 3-D models. As a consequence of the way that such graphics applications and data acquisition devices have been constructed, much effort has been expended in ensuring that large static 3-D models can be stored and transmitted efficiently.

Furthermore, modelling packages can process static 3-D graphics models, or sequences of these, and generate complex synthetic animations. Graphics animations require considerable amounts of storage and bandwidth for transmission, even when they only approximate shapes with limited accuracy. However, the Internet's available resources are not unlimited. Today's streaming applications, which transport mainly audio and natural video, consume a considerable proportion of Internet's bandwidth. Many of these applications to date have been appropriately engineered to achieve some quality of service that the Internet, as a best effort network, did not originally provide. There are a number of open questions that must be addressed before networked 3-D animation can become a practical reality. For example, it is unclear whether the internal mechanisms of existing streaming applications are adequate to transport the emerging streams of animated 3-D graphics. It is likely that we may need to appropriately

process 3-D animation media, and enhance streaming applications with knowledge from the novel 3-D signals we intend to transmit, in order to efficiently deploy them over the Internet, while maintaining the network's stability.

In the remainder of this chapter, the historical context of this work will be briefly outlined, followed by motivating factors and the contributions to knowledge this thesis makes. Since this work draws on research from a number of fields of endeavour, several strands need to be explored.

1.1 Background and Motivation

Systems based on 3-D animation enable intuitive and realistic interaction with virtual models and allow the production of effects that cannot be achieved with conventional 2-D animation realised with natural video processing. Emerging applications for graphics streaming are moving beyond rigid models with animated transformations; today, animation modelling programs are capable of creating extremely complex and expensive to compute time-dependent geometry sequences. Novel animated 3-D graphics stream types are already being used to complement traditional audiovisual material. In particular, streams representing models for 3-D design art, e-commerce virtual salesmen, medical image sequences, recreational cartoons, and educational 3-D content for natural sciences, to name but a few, show the clearest commercial benefits.

Despite superficial similarities of animated 3-D graphics to natural video, both being visual media, the two are significantly different. Thus, although in image and natural video coding, data come as a 2-D regular array where luminance and chrominance signals need to be compressed, 3-D graphic data are not defined on a regular grid. Both the geometric structure (described by connectivity) and vertex data values (expressed as geometry positional matrices) must be coded, as well as the attributes (colours, normals and textures). Additionally, in the case of animated 3-D models, geometry motion needs to be structured and compactly represented.

1.1.1 Representation of 3-D data

1.1.1.1 Static mesh compression

The research field that deals with enabling technology for compact representation and efficient storage of 3-D models is 3-D mesh compression. In its context, algorithms have been developed that code geometry, connectivity, and attributes of a still 3-D model. To be able to recover a model that is as close to the original as possible, connectivity data need to be losslessly

coded, whereas position and attribute[1] information can be largely removed. The aim is to maximally reduce the number of bits needed to represent these models by preserving some minimum level of visual quality, at which any coding artifacts remain imperceptible to the human eye. With state-of-the-art static mesh compression algorithms, the storage requirements of typical complex models made up of a few tens of thousands of vertices can be reduced by two orders of magnitude, whilst maintaining much of their shape and attributes.

1.1.1.2 Progressive coding

Compression of static 3-D models has been complemented by progressive representations that trade compactness for flexibility in manipulating the compressed bitstream. For example, the need for transmission of a static model over a network, rendering it while its parts are still in transit, has resulted in hierarchical (or scalable) representations. These representations can be very compact and each small data fragment (or packet) contributes to the reduction of their coding distortion. The downside of such approaches is that some extra bits need to be allocated to afford this commodity; however, in modern hierarchical encoding this overhead can be negligible.

In a closely related field, multi-resolution, or incremental, representations have been developed in order to manage the level of detail of a model adaptively at run time, or just to reduce its complexity and storage requirements. This need arises at the low-end consumer electronic devices, where computer games or virtual environments must often operate on systems with highly constrained processing resources. But it arises at the high-end as well, where realistic scientific visualisation systems typically construct and use model repositories that far exceed the capacity of powerful workstations, let alone the bandwidth bottleneck in case of their network transmission.

1.1.1.3 Coding of dynamic 3-D media

Compression and transmission of static 3-D models is just one approach of interest in the field of 3-D graphics that drives the evolution of networked graphics media. Whether compression is performed in a single-resolution or a scalable manner, the area has been intensely researched in the past decade. However, very little research has been conducted in compressing dynamic 3-D geometry, a logical extension of static 3-D meshes to the temporal domain. Dynamic geometry coding, also known as time-dependent geometry, deals with the efficient representation of sequences of a 3-D model, which usually maintain the same number of geometric constructs (vertices, faces, and edges) and constant topology (connecting relations between vertices), but

[1] 'Attributes' refer to data representing colour, texture or normals in this thesis.

they have their geometry (vertex positions) displaced between two consecutive instances, or frames, thus generating the effect of 3-D model animation.

1.1.2 The Internet

1.1.2.1 Best effort networks

In packet networks, when a source starts transmitting data, there is usually no a priori knowledge of the available bandwidth along the path to the destination. High-rate sources can potentially bring a network to congestion state, which manifests itself mainly with packet loss. On the other hand, overly conservative sources may leave the network under-utilised. When applications need to stream time-continuous data, as for audio and video, packet networks can only "do their best" and offer no guarantees with respect to the delivered quality of service. Since a large core of the network infrastructure only supports First-in-First-Out (FIFO) queue routers, end-systems are expected to be cooperative and to react to network feedback properly and promptly by adjusting their transmission rates to match the bandwidth that can be made available from the network.

1.1.2.2 Streaming multimedia over the Internet

Lack of QoS support in the Internet, however, did not prevent streaming applications from growing. Early forms of continuous media, those of audio, image, and mainly video, constitute a significant portion of the Internet traffic. These continuous media are transmitted based almost exclusively on the User Datagram Protocol (UDP) at the transport layer, while the Real-time Transport Protocol (RTP) [SCFJ03] is becoming the common standard application level streaming protocol. UDP is the protocol of choice mainly because it has no hard timing constraints, as opposed to Transmission Control Protocol's (TCP) timers and acknowledgement based transport. The downside of using UDP, though, is the lack of reliability, and consequently its vulnerability to packet losses.

On the other hand, the dominant part of the traffic remains TCP-based and is short-lived (e.g., Web, ftp, e-mail, also known as 'mice'), which is well-behaved towards the network; internal mechanisms in the various TCP flavours are designed to ensure network stability. Lack of such mechanisms in UDP has one key consequence: long-lived audio-visual streams (also known as 'elephants'), and 3-D graphics in our case, if not controlled at the end-systems, could reach and sustain aggressive transmission rates that could lead the network to congestion, or even worse, to congestive collapse. Therefore, not only they could bring about packet losses, but they pose threats to well-behaved TCP traffic, which, if congestion persists, may end up being shut out of the network.

1.1.2.3 End-to-end congestion control

The demand for cooperation of the end-systems is the baseline definition of end-to-end congestion control mechanisms in the Internet. Such control mechanisms should not only scale well with the number of sources, without creating substantial overhead to the network, but should also limit the bandwidth usage to a fair share among them. Furthermore, they should ensure stability of the active sources, by allowing them to converge to an equilibrium, should there be no further fluctuations to available bandwidth.

Consequently, protocols for congestion control have arisen that try to mimic TCP's behaviour as a moving target (due to TCP's fluctuations), while exerting social behaviour (being *TCP-friendly*) towards other well-behaved traffic. Almost all such protocols are tailored to the streaming needs of legacy multimedia systems, namely streaming audio and video, and can be divided into two main categories: TCP-modelling and Additive Increase - Multiplicative Decrease (AIMD). In the first category, an equation estimates TCP's transmission rate as a function of packet loss events and round-trip delays (TCP-Friendly Rate Control Protocol, TFRC, is such an example). In the second category, conservative linear increases of the output rate take place at the source when more bandwidth becomes available, or sharp cumulative rate drops are employed otherwise (i.e., Rate Adaptation Protocol or RAP).

1.1.3 Networked animation

1.1.3.1 Addressing the problems of networking 3-D animation

Despite the previously described developments individually in the areas of 3-D mesh coding, congestion control, and error control, animated 3-D graphics is not yet a mainstream networked medium. This is a consequence of the lack of scalable solutions that would allow 3-D animations to survive the best-effort environment afforded by the Internet. In order to produce such a solution, the fundamental problems that arise within the Internet, such as limited availability of bandwidth, unpredictable available rate fluctuations, and packet loss due to congestion, need to be explicitly addressed *ab initio*.

The challenge with modern graphics streaming applications is to determine whether the existing, seemingly generic, congestion control protocols can be safely deployed in 3-D animation transmission systems as is, or whether we can benefit the design of these applications by applying signal processing methods at the source that dynamically fit the output rate to the estimated available bandwidth.

1.1.3.2 Standardisation activities

In the effort to build support for 3-D media over packet networks, standards-based source encoding and communication systems must be actively pursued as solution environments. They should be considered, where possible, as they wrap up state-of-the-art technologies in complete multimedia frameworks.

For example, the ISO/IEC MPEG-4 standard caters for source coding of face and body animation data in the specification of its visual technology, as well as for encoding of whole multimedia presentation scenes in its binary scene format (BIFS), whilst allowing dynamic manipulation of the scene's objects. However, despite being a potentially promising technology, MPEG-4 has failed to achieve widespread use. Unlike developments in its natural video counterpart, animation in MPEG-4 has not been designed with joint source-network considerations in mind. Even after the most recent Animation Effects (AFX) extensions, the standard does not propose any form of rate or error control on dynamic graphics media. These are some of the reasons why to date the degree to which MPEG-4 has been deployed as an industry standard for quality streamed animations is rather limited.

As another example, the Web3D/X3D framework has been designed around an inherently reliable protocol (HTTP/TCP) which, however, for reasons related to their hard real-time constraints and reliable delivery as mentioned earlier, is unsuitable for real-time streaming.

Furthermore, despite efforts in standardisation bodies to provide universal 3-D graphics storage formats, the transport format for this instance of multimedia is simply nonexistent. RTP-based payload formats emerge in abundance through standardisation procedures in the IETF for streaming audio, video, image, or text communications, but specifications for animated graphics transport payloads have never been proposed.

1.1.3.3 Visual perception factors

The impact of using unreliable transport protocols for streaming simplifies, yet weakens, the packet transport process as it renders it susceptible to signal quality deterioration due to packet loss. The application level transport protocol RTP has reached standard status in the IETF, aiming to strengthening the robustness level of audio-visual streams against unreliable transport. However, the transport mechanisms alone, even when they are supplemented with appropriate payload formatting, are inadequate for resilient streaming. The development of end-to-end error control mechanisms and media concealment at the receivers is a complementary necessity. Compared to conventional multimedia data sources, 3-D graphics streaming applications generate different data to pass across the network, loss of which affects signal reconstruction differently to, for example, video.

The perceptual effects of such loss have been poorly addressed in the context of animation to date and much of the work that there has been in this field has relied on objective measures, such as PSNR, in lieu of those that take subjective effects into account. Thus one significant current challenge in this field is to integrate animated 3-D geometry as a first-class citizen in the existing evolving infrastructure of the Internet, in a way that allows for both graceful adaptation to time varying conditions, and optimisation of subjective visual perception factors. By providing solutions towards this way we can realise the full potential of this medium and make animated 3-D geometry as ubiquitous as audio, image, and natural video streaming.

1.1.4 The current state of the art

So far, the predominant forms of quality of service for graphics have been: (i) network adaptivity for progressively coded 3-D models, (ii) error-resilient transmission of static 3-D models, (iii) level-of-detail control. The aim was to enable asynchronous rendering of static 3-D objects whilst their model is being streamed. These three forms of graphics data QoS have been achieved mainly through suitable signal (model) encoding [ISO01b, Hop96, PH97, TGHL98, PR00, LLK97, COLR99, AD01a], multi-resolution signal processing [SZL92, RB93, CVM+96, RR96, Hop96, GH97, CCMS97], forward error correction [RA02, RA01, RARM02] and error resilient model transmission [BCPZ98, BK02a, BK02b, LKSS00], transcoding and packet re-transmissions [Mar02].

State-of-the-art research on animated 3-D models, has addressed compression of dynamic 3-D meshes [Len99, GSK02, AKKH02, YKL02, SPB00]. Transmission of animated 3-D geometry has only been engineered with VRML-like position and orientation node interpolations that allow creation of predefined (stored) animations of face and body figures [GTHL99, ISO01b]. Recent extensions through the AFX framework [ISO02] in the MPEG-4 standard leverage the state of the art in 3-D model animation beyond face and body animations. To accommodate 3-D animations over the best-effort Internet, methods from progressive or level-of-detail based coding need to be enhanced with dynamic 3-D geometry coding, aided, in a controlled fashion, by simplification and refinement techniques. Hardly has any past research considered quality of service control *while* transmitting *animated* 3-D models over a best-effort network such as the Internet.

Also, several congestion control protocols have been proposed to date for use in the Internet, in particular with natural audio and video streaming in mind. AIMD based protocols, like RAP [RHE99], address quality of service for visual media streaming, by managing the number of layers the visual media is split into to handle the effects of long-term bandwidth

variation, or by absorbing short-term rate fluctuations with receiver buffer policing. TFRC [FHPW00], operates a TCP modelling approach at the source, primarily caring for network stability and maintaining stream fairness by mimicking TCP's congestion behaviour. However, none of these protocols blends congestion control with the intricacies of the source signal they are transporting. There is a challenge in building rate control methods for animated 3-D models, using knowledge of the source signal and its encoding.

As a consequence of the level of variability and unpredictability of available resources along network paths of any length, it is desirable to be able finely to *adapt* the transmitted mesh quality to the prevailing network conditions on a short timescale, by appropriately reducing the output bitrate in poor conditions, or increasing the bitrate should more bandwidth become available. In this dissertation, the dimension of quality of service (QoS) for streaming dynamic 3-D meshes that we consider is one based on the general concept of fidelity: We are concerned with the *"end-to-end graceful degradation or enhancement of their geometric detail"*. Such degradation can occur as a result of network congestion or packet loss to which a controlled 3-D model encoder must react, adjusting its coding by limited step simplifications or step refinements that reflect the new network conditions.

1.2 The Research Problem and its Scope

The precise statement of the thesis research problem is the following:

Streaming of animated 3-D wireframe models can be supported in large scale over the Internet such that:

- *The animated model data are compactly represented while in transport from the source to the destination,*
- *The animation streams are resilient to the channel's loss conditions,*
- *The sender and receiver cooperate to deliver high and stable quality animated models that adapt to the network's variable available rate.*

In the diagram of figure 1.1 the work described in this dissertation is depicted as a synthesis of its component parts laid out in the previous section. Milestone technologies of the relevant research scenery are placed on three axes. The projection of key research fields influencing the work presented herein defines the context and limits the scope of the dissertation.

In particular, the dissertation develops and builds knowledge in the top and front area of the grayed volume, in the context of rate and error control for time-dependent polygonal multiresolution 3-D meshes. One key underlying target is to bring together research in 3-D graphics

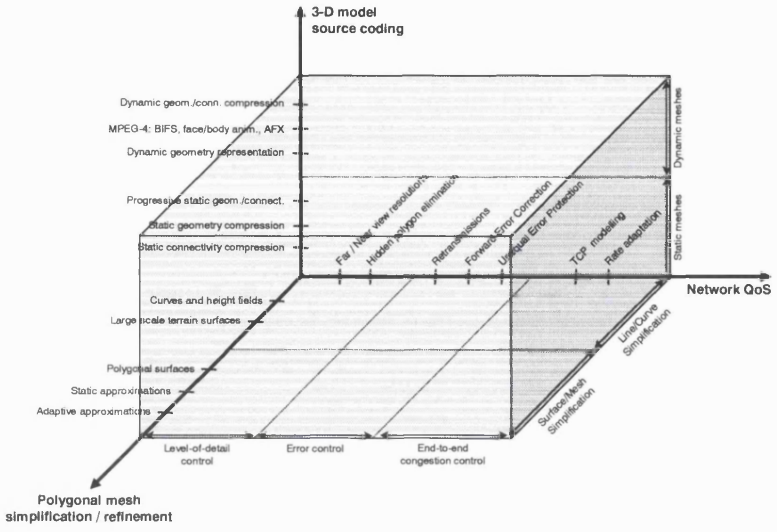


Figure 1.1: Dissertation design space.

coding and end-to-end streaming support over best effort networks. While developments in each area take place individually, this thesis strives to bridge the gap between the two. Arguably, further research in 3-D animation streaming will benefit from, and should develop solutions that build upon this match.

The top level area of the space refers to dynamic animated graphics. The main objective of this level is to express dynamic models in a format which, being aware of the temporal redundancies between successive time representations, develops a baseline *source compression* component for the graphics streaming application. The aim is not to optimally compress the animation signal, but primarily to remove from it redundancy while maintaining the *flexibility* to form: i) *layers*, or ii) *fine-grained batches* of simplification and refinement operations. The latter are denoted by operating at the front side of the volume, which encompasses multi-resolution 3-D representations of polygonal models.

The centre-right subspace represents the context of network quality of service that this thesis is addressing. The form of quality provisioning that is examined is that of visual fidelity and spans the fields of error and rate control; in the error control domain, the amount of unequal error protection provided *proactively* to the animation streams is optimised, and receiver mechanisms are built to partially recover the signal *after* packet loss; in the rate control field, *graceful adaptation* of the animation streams to the available network rates is the primary concern. The approach taken is to investigate the degree to which existing congestion control mechanisms for streaming media can be utilised to support animation streams, by enhancing the rate shaping methods in the encoder component to take advantage of the knowledge of the animation source signal. Adding or removing structural constructs of the models gives better adaptation to the

network while it maintains at least a minimum level of visual quality.

1.3 Contributions

A number of research activities have laid the groundwork for either 3-D geometry and connectivity coding, or quality-aware progressive transmission of 3-D models. However, such research efforts from either side did not take into account the other: 3-D model compression only recently started to exhibit awareness for error control, whereas progressiveness in streaming 3-D models is achieved by compromising data compression. This dissertation leverages existing research in 3-D meshes to streaming animations with optimal smoothness as a form of error control and progressively coded source for rate control.

Ultimately, we believe that transmission of good quality animated 3-D graphics will become commonplace over the Internet for network games, education, e-commerce, and distributed engineering. But before this can happen, we must develop understanding and build knowledge of the technical difficulties and special requirements imposed when 3-D models are transmitted over a best-effort network, before deployment of scalable 3-D graphics streaming engines can start. This dissertation represents one step towards this goal. Its contributions advance the state of the art in both coding and streaming error- and rate-controlled 3-D wireframe animations, as follows:

- **Flexible source coding:** A generic compression scheme for time-dependent 3-D geometric meshes is described, as a baseline source coder. It is shown that, with minimum processing effort over the geometric signal, temporal compression of such geometric sequences performs better than intra- only coding of 3-D models, as can be achieved currently by the state-of-the-art solution afforded by MPEG-4. The flexibility is such that the generic encoder can also produce layers of animated 3-D data.

- **Sender-based redundancy for optimal visual smoothness:** The fidelity of decompressed animated models can be optimally considered when such models are streamed over the best-effort Internet characterised by bursty packet loss. An expression of the reconstructed model's surface smoothness as a quality criterion has been derived. A layered 3-D graphics source supported by forward error control unequally distributed among layers, but with constrained available channel bandwidth, is shown to visually improve transmissions that are unaware of the channel error characteristics.

- **Receiver-based concealment:** Studying the effects of packet loss on 3-D animation reveals that, even after loss, there is significant signal information at the receivers to help

conceal missing animation frames. The approaches taken are much inspired by audio and natural video, but are tailored to the 3-D animation frames produced by the generic encoder, which structures the frames around motion vectors. Receiver-based model concealment is also shown to complement the joint source-channel optimised animation coding.

- **Multi-resolution coding for 3-D stream quality adaptation:** An expression of dynamic multi-resolution meshes is formalised. The formalisation is later projected onto the generic animation codec, which it enhances by allowing it to compress geometry of dynamic meshes in a fine-grained progressive manner.

- **Rate-controlled end-to-end transmission:** A rate-control method for the adaptive transmission of dynamic 3-D meshes is proposed. The method is built around the multi-resolution representation of dynamic meshes and allows streamed 3-D animations to gracefully adapt their rate to available network resources with step quality degradation or refinement, in a way that respects co-existing and well-behaved traffic.

1.4 The Structure of the Dissertation

The dissertation is organised as follows:

Chapter 2 first introduces core terminology used in the field of 3-D geometry. Later, it reviews literature and provides a taxonomy of past work related to 3-D connectivity and geometry coding for static and animated models. The material is presented in a manner that highlights the differences between proposed approaches and previous work. The main approaches of streaming graphics data in a QoS-controlled fashion are also reviewed. Last, a review is presented of the most commonly used metrics for measuring distortion or fidelity in 3-D models.

Having established the background information, chapter 3 outlines a top-level architectural view of 3-D animation streaming applications for the Internet. The two key source components of the design are identified as error and rate control that not only interact with the source encoder, but can also inter-operate in a coherent way to both improve the animation quality of the streamed models and respect network resources. Chapter 3 also argues that the proposed architecture can be used generically in 3-D animation streaming applications by tuning the components' parameters to the appropriate operational space.

In chapter 3, subsection 3.4.1, the differential representation of 3-D animation signals is extended to a multi-resolution expression. Formal notation of such representations is given, as well as a high-level overview of a related encoder and its application to the 3-D animation

streaming architecture. Its interactions with a rate control component, which help to provide quality aware adaptation to the network conditions, are also defined. Evidence of the performance features of the scheme is given by simulation that shows that it outperforms traditional quantisation control schemes.

Chapter 4 studies through extensive simulations the effects of packet loss on differentially encoded 3-D animation streams. It develops and compares three methods for animation stream repair, highlighting possible network state scenarios where these methods can be employed to conceal loss damaged streams.

Chapter 5 takes a layering approach to robustly encode 3-D animations. Using cues from the previous chapter's study of the effects of loss, it models the network's burst loss behaviour to simulate experiments where unequal error protection provides increased resilience to the animation stream. Visual smoothness is introduced as a measure of animation quality, which is optimised by providing error protection information unequally across layers.

Chapter 6 concludes the dissertation and addresses issues appropriate for future work.

To highlight certain design choices, a sequence of appendices is provided, which also contain implementation details of the encoder.

Chapter 2

Related Literature

In the following section, definitions and the basic terminology used in geometry modelling and compression are provided, and the various classification schemes encountered in the literature are identified. A survey of the related literature follows for still 3-D mesh compression based on a comprehensive classification. Literature related to dynamic 3-D meshes then follows.

Before delving into the topics of 3-D modelling animation, compression and simplification, it is appropriate to clarify at this point a basic distinction made in the remainder of this thesis. Although there is a considerable amount of work exhibited in the domains of mesh compression and simplification, the majority of it utilises meshes that show no evolution in time. Meshes of this category will be referred to as *still* or *static* meshes. Time dependency, though, without necessarily implying connectivity changes, refers to meshes whose geometry is time varying. The terms *dynamic*, *time-dependent*, or *animated* meshes will be used for this category. The literature presented in the sections below follows this distinction.

2.1 Basic Terms & Taxonomy

A 3-D mesh consists of three different kinds of mesh elements: *vertices*, *edges* and *faces*. The information describing the mesh elements consists of mesh *connectivity*, mesh *geometry* and mesh *photometry*. Mesh connectivity describes the incidence relations between mesh elements. Mesh geometry specifies the position in space of each vertex. Mesh photometry refers to those attributes of a mesh that describe its appearance and shading, such as colour, texture and normal vectors. In the following definitions, the numbers of faces, vertices, and edges of a mesh are respectively denoted by f, v, and e.

Polygonal vs. triangular meshes. A generic 3-D mesh may consist of faces with an arbitrary number of bounding edges. This mesh is called a *polygonal mesh.* In the special case where the bounding loop of every face in a mesh has exactly three edges (triangular faces) then the mesh is called a *triangular mesh.* Typical triangular meshes have twice as many

triangles as vertices, or $2v \simeq f$.

Degree (valence). The degree or valence of a vertex is the number of edges incident to that vertex. It can be shown that, for triangular meshes, the sum of valences is twice the number of edges, or $\sum_i valence = 2e$. For sufficiently large meshes, the relationship $e \simeq 3v$ also holds.

Regular vs. irregular meshes. A 3-D mesh is *regular* if all its vertices have the same valence otherwise it is *irregular*. Given the previous definitions and equations, in a regular triangular 3-D mesh the average vertex valence is 6.

Triangle strips and fans. A *triangle strip* is a sequence of triangles in which adjacent triangles share an edge. Strips are more efficiently processed by the graphics processor because each successive triangle in the strip is defined by a single additional vertex; the triangle's other two vertices are already known from the preceding triangle in the strip. A strip of f triangles can be drawn with $f+2$ vertices.

A *triangle fan* is a group of triangles sharing the same vertex and forming a circular concatenation.

Figure 2.1 shows a concatenation of a triangle strip and fan.

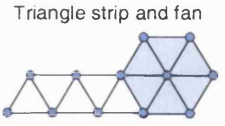


Figure 2.1: Triangle strips and fans (from [KBB+00]).

Manifold vs. non-manifold meshes. A 3-D mesh is called *manifold* if each edge is incident to only one or two faces and the faces incident to a vertex form a closed or an open fan, as shown in figure 2.2. A manifold mesh is *closed* if all edges are shared by two polygons, otherwise it is *open*. *Non-manifold* meshes can be cut into manifold meshes by replicating vertices with more than one fan and edges incident to more than two faces.

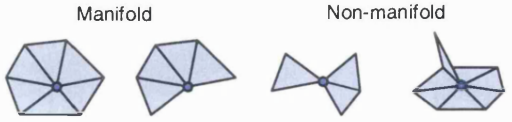


Figure 2.2: Manifold and non-manifold meshes (from [KBB+00]).

n-ring. The n-ring of a vertex v is the set of all vertices at distance n (in edges) to v. The 1-ring of v is the set of immediately adjacent vertices to v. In a regular mesh, each n-ring

has exactly the same number of vertices.

Complexity vs. smoothness. Polygonal mesh representations can be rated according to various criteria. Most notably their *complexity* is measured by the number of vertices or faces. *Smoothness* can be measured in different ways, such as the angle between the normal vectors of adjacent triangles, or other more sophisticated methods.

Single-resolution vs. multi-resolution meshes. Organisation and representation of 3-D mesh spatial data at a single level of accuracy (resolution) forms a *single-resolution mesh.* High-detail single-resolution 3-D models typically involve large amounts of data, which cannot be always efficiently manipulated by the application, e.g., on-line retrieval of a model with real-time rendering, as in flight simulators, navigation in virtual environments, etc.

Multi-resolution models allow for representation, analysis, manipulation, and transmission of spatial data at variable resolutions. Most graphics applications usually require high model resolution only on certain parts of 3-D objects, and the location of such parts may change over time. Thus, the size (and the cost for manipulation) of a 3-D model can be maintained low by locally adapting resolution to the needs of the application. Adjusting the complexity of a mesh introduces scalability to the bitstream representing the coded mesh. It is desirable to have a multi-resolution mesh coded in a progressive manner (see below).

Progressive meshes. A *progressive* representation of a polygonal mesh enables one to define a *base* quality mesh (or *coarse*) representation first and subsequently refine it with details. A refinement scheme inserts vertices or other detailed information to the base mesh. As progressive representations are usually developed to benefit mesh transmission, compression strategies are usually employed to make such transmission compact. Modern mesh compression schemes incorporate progressive representations for the mesh signal, and may trade compression bits to achieve progressive layers.

Refinement vs. decimation. Mesh *decimation* algorithms (or *simplification*) transform a given polygonal mesh into another mesh with fewer faces, edges and vertices. The decimation procedure is controlled by user-defined quality criteria, which preserve specific properties of the original mesh to the extent possible (e.g., geometric distance, such as the Euclidean or Hausdorff distance, or visual appearance - colours, features, etc.). The inverse process, where a polygonal mesh is transformed into another of finer quality, is called *refinement.*

Hierarchical meshes (models). A progressive mesh implies a hierarchy in *level of detail* (LOD) of the mesh representation. Two conceptually different expressions of hierarchy are identified; *topological hierarchies* refer to coarse/fine mesh representations, whereas *geometrical hierarchies* refer to smooth/non-smooth representations. While decimation reduces complexity and hence always removes detail information, the refinement can be used to either re-insert details or to increase smoothness without adding detail.

Homogeneous vs. heterogeneous motion regions. We refer to a subset of a polygonal mesh's vertices as a *mesh region*, if those vertices are neighbours. If each polygon in a mesh region shares at least one edge with another polygon in the same region, the region is said to represent a *connected component*. In an animated polygonal mesh, a *homogeneous motion* region refers to a connected component whose updated coordinates can be expressed as a single transformation (rotation or translation) of its coordinates in the previous instance.

Rigid vs. non-rigid motion. As opposed to free form local mesh deformations, rigid motion is more suited to express human figure and other rigid object animations, where the various body parts or objects exhibit uniform motion as single entities. Rigid body animations can be effectively described by affine transformations or kinematic equations.

Homeomorphic and isomorphic meshes. Two meshes A and B are *homeomorphic*, if A can be stretched or bent to B without tearing. Dynamic meshes that maintain their homeomorphism through time - invariant topology and number of vertices - are termed *isomorphic*.

Sparse vs. dense animations. In a time-dependent mesh, if the number of vertices displaced between successive mesh instances is low, or the displacements of certain moving vertices is small so that the vertices are considered *stationary* by the application, the animation is called *sparse*. Otherwise, if all vertices move by significant (for the application) displacements, the animation is termed *dense*.

The degree of sparsity or density of animations will be mathematically quantified in section 3.3.

2.1.1 Classification of encoding schemes

3-D mesh compression covers the efficient encoding of both *structure data* (topological or connectivity quantities: vertices, edges, faces) and *attribute data* (geometric quantities: vertex positions, face colours and normals), as expressed by Bajaj et al. in [BPZ98].

This initial classification is historically introduced and researched upon by Deering in his pioneering work on geometry compression [Dee95]. Deering introduced the first geometry and connectivity compression scheme to compress the bitstream sent by a CPU to a graphics adapter, generalising the triangle strips and fans. Chow's follow-up work [Cho97] caters for the speed and memory requirements of real-time decompression and rendering of compressed 3-D mesh geometry. (Details of both early schemes are given in the following subsection). The motivation in both these early pieces of work was to meet the stringent real-time requirements of a fast graphics hardware pipeline. Consequently, Deering and Chow built efficient single-rate 3-D mesh compression algorithms where all the connectivity and geometry data are compressed and decompressed as a whole. In contrast, later research focused on efficient mesh transmission over the Internet, thus leading to the more recent progressive compression and transmission schemes. These software-based schemes allow more complicated compression and decompression stages by providing larger memory buffers than an on-chip cache, random-access to an array of vertex data, multiple passes over the vertices, etc. When progressively encoded and transmitted, a 3-D mesh can be reconstructed in continuous levels of detail (LODs), starting from a coarse initial mesh towards a finer maximum detail representation.

Some researchers have attempted a baseline classification of encoding schemes. In related work on single rate connectivity coders, Khodakovsky et al. in [KADS02] determine that there are two major groups of algorithms: those specific to *triangle meshes*, and those for more general *polygon meshes*. In [GBTS99], Guéziec et al. distinguish two categories of meshes, non-manifold and manifold, noting that conversion from the latter to the former type is possible by allowing mesh cuts, and subsequently proposing a compression scheme for manifold meshes. Such low-level categorisations, though, are limited only to the context of the specific encoding methods to which they are bound, and are not suitable for top-level classification and surveying. A handful of other surveys related to 3-D meshes exist [Tau99, Shi00b, GGK02, Gar99a]. These, although more generic, focus only on a certain aspect of 3-D meshes:

- Taubin [Tau99] describes in detail mesh compression and progressive transmission with focus on his topological surgery and progressive forest split as part of the MPEG-4 standard, with other schemes receiving little attention.

- In [Shi00a, Shi00b], Shikare reviews available state-of-the-art compression algorithms for static 3-D meshes, and classifies them according to the type of encoding technique used and the type of bitstream generated: *single* layer (or rate), or *progressively* refining layers.

- Gotsman, Gumhold and Kobbelt [GGK02] reviewed 3-D meshes based on simplification, connectivity and geometry coding approaches. Although their survey is the most comprehensive so far, they do not provide detailed review of progressive coding schemes which have lately gained in importance.

- Garland [Gar99a] surveyed multi-resolution modelling methods with emphasis on iterative edge contraction algorithms for incremental representation (simplification and refinement streams) of 3-D models. This classification, however, does not address compression of 3-D meshes.

All the above literature surveys pay limited attention to 3-D mesh encoding for streaming. They also lack discussion on the emerging research on time-dependent 3-D mesh coding. This chapter provides a detailed classification focusing on both compression and Internet transmission, for static and dynamic 3-D meshes. The classification is expanded in sections 2.2 and 2.3.

In the following section of the chapter, a top-level classification is followed between single-layer (single-rate, or single-resolution) and progressive encoding, as this reflects one of the distinct requirements for transmission of the bitstream over heterogeneous networks. Subsection 2.2.1 discusses single-layer compression. Subsection 2.2.2 presents progressive compression and decimation methods. Each top-level category is then further distinguished between connectivity-based and geometry-based coding. Paragraph 2.2.2.2 in particular, discusses also compression methods that are not based directly on 3-D mesh structure or attribute data, but transform the original mesh signal instead. Finally, subsection 2.2.3 outlines all other 3-D coding efforts that cannot be classified under the previous subsections.

2.2 Static 3-D Mesh Compression

This review of static 3-D mesh coding distinguishes mainly between single-rate and progressive compression, but work in the closely related field of mesh decimation is also presented.

2.2.1 Single-rate compression

There are two different classes of single-rate 3-D mesh coding research: *connectivity* and *geometry* compression.

2.2.1.1 Mesh representation (connectivity compression)

In this subsection, a review is given on how the face-vertex incidence table of a mesh can be encoded efficiently. Different methods are identified depending on whether the mesh is

planar or *non-planar*. The remainder of this section focuses on non-planar meshes as this thesis deals with 3-D meshes and the methods described below represent the most recent connectivity coding attempts. These methods can be further categorised as *face based*, *vertex based* and *edge based* according to the type of mesh element playing the dominant role in the compression scheme.

After Deering's first attempt to code non-planar triangular meshes, Chow's main concern was speed [Cho97] for building cost-effective hardware implementations. He proposes a "meshifying"[1] algorithm that decomposes a given triangular mesh into several generalised triangle meshes. This decomposition allows most vertices to be reused when forming new triangles. He also suggests that different regions of a geometric model should be compressed with variable precision depending on the level of detail present in each region. This lowers the average bits per triangle for a given object. He also presents a method for automating the selection of quantisation levels given a user error threshold. The meshifying algorithm is based on the spiraling traversal of the mesh first introduced by Taubin and Rossignac in [TR98].

This latter research, called *Topological Surgery* (TS), was motivated by Deering's work, but was later optimised for transmission over the Internet. According to this method the encoder first has to construct the vertex spanning tree of the input mesh, based on a *layered decomposition* of the mesh that is similar to the way we peel an orange along a spiral path. Such a spanning tree construction aids the maximisation of the length of the layer and the minimisation of the number of layers generated. The basic idea is to cut a given mesh along a selected set of cut edges to make a planar polygon. Typically, in a single mesh, any vertex-spanning tree can be selected as the set of cut edges. The structures of the cut edges and the planar polygon are then used to represent the connectivity of the mesh.

TS represents the first connectivity preserving single-resolution manifold triangular mesh compression scheme. Follow-up improvements to handle arbitrary polygonal meshes led to its proposal as a compressed file format to encode binary VRML files. And with its subsequent more efficient encoding of general manifold meshes, it is now part of the MPEG-4 standard. It has also inspired most of the 3-D mesh encoding approaches described elsewhere in this section.

Bajaj et al. also suggested *layered decomposition* in their method for compressing large CAD models [BPZ98]. Earlier mesh decomposition work by the same group involved triangle meshes only. They proposed a layering method that first decomposes a triangular mesh into concentric vertex layers and then, within each pair of adjacent vertex layers, it constructs a triangle layer. The connectivity is coded by the total number of vertex layers, the layout of

[1] The term "meshify" was coined by Michael Deering in [Dee95].

vertices in each vertex layer, and the layout of triangles in the triangle layer. Ideally, the triangle layer is a generalised triangle strip. This algorithm was later extended by the researchers to compress quadrilateral and general polygonal models.

Face based. In [GS98], Gumhold & Straßer introduce a compressed representation of the connectivity of a triangle mesh, with a sufficiently fast decoder for real-time applications and hardware implementations. At each step, the *cut-border machine* algorithm inserts a new face (triangle) into the conquered part, closed by the cut-border, using one of the five predefined building operators: *new vertex*, *forward*, *backward*, *split*, and *close*. The sequence of building operators generated at the encoding process is compressed with Huffman codes. Later improvements included an optimised border encoder and adaptive arithmetic coding.

Rossignac, in a nearly equivalent algorithm to the cut-border machine, proposed the *Edgebreaker* [Ros99]. This algorithm defines a slightly different set of growing operations but, as the cut-border machine, it encodes the incidence relation of the current face with the cut-border using a restricted symbol set. Although it achieves good compression performance, it is unsuitable for streaming applications since it requires a two-stage decoding process.

Vertex based. Another popular triangular mesh compression method is given by Touma & Gotsman in [TG98], who pioneered a novel vertex-based traversal scheme that provides a natural adaptation to triangle mesh regularity through entropy coding. The algorithm starts from an arbitrary triangle of the mesh and pushes its three vertices into an *active list*. For each vertex in the active list, its adjacent unprocessed vertices are traversed and pushed on to the end of the list. The algorithm outputs the valence of each processed vertex, or uses special codes to encode a split or a merge operation on the active list. It also employs dummy vertices to close the topology of the boundary loop. This algorithm, followed by an arithmetic coder, achieves very compact encoding of typical meshes with an average vertex valence of 6, and is considered the state of the art.

Alliez & Desbrun [AD01b] improved on Touma & Gotsman's algorithm, by reducing the number of bits required for the encoding of split codes, offsets and dummy vertices. They used heuristic methods selectively to choose the next focus vertex, and employed an efficient adaptive arithmetic encoder.

Edge based. Major efforts in this category come in the form of edge contractions (eliminations from the original mesh) where the main target application is multi-resolution mesh coding. Multi-resolution schemes will be discussed in 2.2.2.

2.2.1.2 Geometry compression and prediction

Although mesh connectivity schemes have reached almost optimality, mesh geometry still dominates mesh storage requirements, and recent 3-D mesh compression efforts increasingly focus on geometry compression. In the previous subsection, lossless schemes were presented that exploit the *discrete* nature of connectivity information. Geometry data are of a *continuous* nature, especially in smooth meshes, and strong correlation exists between the values of neighbouring vertices. Typical geometry encoding schemes consist of the following threes steps:

- **Quantisation**. Data redundancy is first minimised by a lossy technique, such as quantisation. Quantisation is a mature field of data compression that has been studied extensively, and its application to 3-D geometry comes in several forms. In *uniform quantisation* the coordinate vectors are mapped to the discrete values of a uniformly-spaced three-dimensional grid structure. Uniform quantisation was used by Deering in the generalised triangle strips [Dee95], by Taubin in TS [TR98], and by Touma & Gotsman [TG98]. In *non-uniform quantisation*, more quantisation vectors are positioned in the regions where more of the vertices lie, hence reducing the value of the quantisation error. Chow used adaptive non-uniform quantisation in [Cho97], according to local surface curvature and triangle size. Another set of quantisation techniques, known as *vector quantisation* (VQ), take into account the joint distribution of vertex components to exploit the correlation between the three coordinates.

  When quantising 3-D geometry, care should be exercised to quantise all coincident vertices together. In situations where boundaries coincide, with the incident to the boundary vertices subjected to different quantisation levels, or to negligent non-uniform quantisation, *cracks* or *overlaps* may appear on the geometry, as shown in figure 2.3.

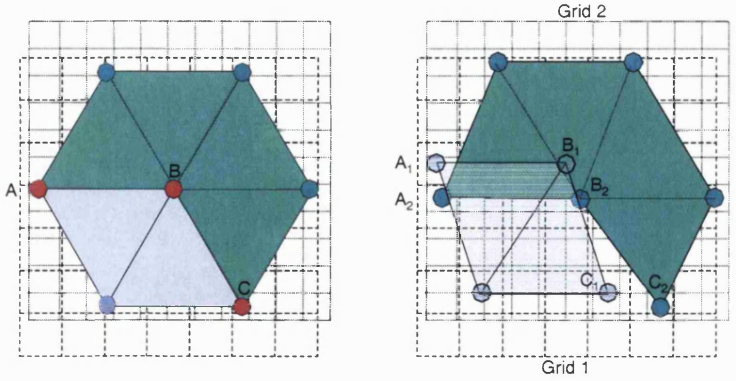


Figure 2.3: Crack and overlap artifact when non-uniformly quantising coincident vertices (from [GGK02]).

- **Prediction**. Predictive coding is extensively employed at the second step, where the

coordinates of a set of previously encoded vertices are used to predict the coordinates of another unprocessed vertex with some prediction error. A good prediction scheme can generate error vectors that have a compact distribution. The order of vertex geometry encoding is usually dictated by the vertex traversal used in connectivity encoding and it drives the prediction process.

- **Entropy coding of residuals**. As a final step, a geometry compression scheme performs entropy coding of residual errors, usually through Huffman or arithmetic coding techniques.

In the following paragraphs, a review of known predictors for 3-D mesh geometry is given as variations of the linear predictor that form the basis of linear predictive coding.

Linear predictive coding (LPC). Although a number of predictors have been proposed in the literature, most of them can be regarded as variations of the linear predictor, with appropriately selected coefficients. In a linear prediction scheme, the geometry of a vertex v_n is predicted from a linear combination of the geometries of k of its ancestors $v_{n-1}, v_{n-2}, \ldots, v_{n-k}$ in the vertex traversal order, which can be expressed in a scalar form as:

$$v_n = \sum_{i=1}^{k} \lambda_i \cdot v_{n-i} + E(v_n) \ . \tag{2.1}$$

In TS [TR98], Taubin and Rossignac first employed such a k-order linear predictor, where the k previous vertices were appropriately selected along the path from the root of the vertex spanning tree to the current vertex, in a way that the mean square $|E(v_n)|^2$ of the prediction error was minimised.

Differential predictive coding. In the special case where $k = 1$ and $\lambda_1 = 1$ in equation (2.1) we obtain the differential predictor, where deltas (δ) between the coordinates of the current vertex position to the previous are usually small, and are preferred for coding over the absolute vertex position, i.e.:

$$\delta = E(v_n) = v_n - v_{n-1} \ . \tag{2.2}$$

The early geometry compression work of Deering [Dee95] and Chow [Cho97] use differential coding of geometry. In the same line of thought, Bajaj et al. [BPZ98] used second-order linear prediction for encoding vertices along the contours. In 2^{nd} order prediction $v_i - v_{i-1}$ is predicted as $v_{i-1} - v_{i-2}$, hence v_i is predicted as $2v_{i-1} - v_{i-2}$, where the prediction coefficients are $k = 2$ and $(\lambda_1, \lambda_2) = (2, -1)$. In higher order prediction, one can compute the optimal set of prediction coefficients λ_i, such that the average prediction error is minimised; however, in practice, the additional benefit of optimal coefficients is small.

Parallelogram prediction (PP). Touma & Gotsman [TG98] better captured the spatial redundancy of vertex positions, relying on the triangular structure of the mesh to apply LPC, so that the geometry of a vertex may be predicted as a linear combination of other vertices in the preceding neighbourhood of the vertex traversal path. They empirically observed that two adjacent co-planar triangles in a triangle mesh tend to form a parallelogram, hence the fourth vertex in such a structure may be predicted from the preceding (neighbouring) three with $k = 3$ and $(\lambda_1, \lambda_2, \lambda_3) = (1, 1, -1)$, or $v_n = v_{n-1} + v_{n-2} - v_{n-3}$. These coefficients yield better prediction over the *x-continuation* $(2, 0, -1)$ and the *y-continuation* $(0, 2, -1)$. Figure 2.4 shows the scheme, widely known as the *parallelogram prediction rule*. In the left hand figure, vertex r is predicted as $p = v + u - w$. In an improved variation of the simple PP, shown in the right hand figure, r is predicted as p' taking into account an estimate of the crease angle α of p around edge (u, v), in the case the adjacent vertices do not lie on the same plane, as in non-smooth meshes.

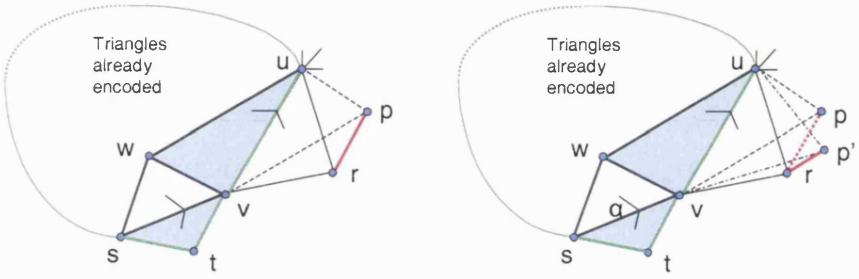


Figure 2.4: Parallelogram prediction (from [TG98]).

Bi-directional prediction (BDP). Sim, Kim and Lee [SKL03], proposed the bi-directional (or dual) parallelogram prediction (BDP). This is a modified PP scheme, based on a triangle fan. As shown in the left of figure 2.5, BDP is applied to the last untraversed vertex V of a triangle fan around a central vertex C. Assuming a counter-clockwise vertex traversal of the fan, the predicted geometry V_{DPP} of vertex V is defined as the average of the forward V_{FPP} and backward V_{BPP} PP predictions of V, or:

$$V_{DPP} = \frac{V_{FPP} + V_{BPP}}{2} \quad . \tag{2.3}$$

The BDP is not a scheme that can be applied alone on all vertices of a 3-D mesh to predictively encode its geometry, but it builds on the PP of Touma & Gotsman.

Multi-way prediction (MWP). Cohen-Or et al. [COI02], improve upon the 1-way PP scheme by using a multiplicity of directions to predict the position of a vertex. They call this scheme k-way, or multi-way prediction (MWP), and its basic principle is depicted on the right in figure 2.5. They define the prediction degree of a vertex as the number of triangles that

can be used to predict its position with the PP rule. For typical meshes, the average prediction degree of a vertex is two. Furthermore, in order to maximise the number of multi-way predicted vertices, they propose a simple heuristic that always tries to select for encoding the vertex with prediction degree two or higher. Therefore, the MWP scheme guarantees lower entropy than the simple PP, or the BDP, as it yields smaller error vectors.

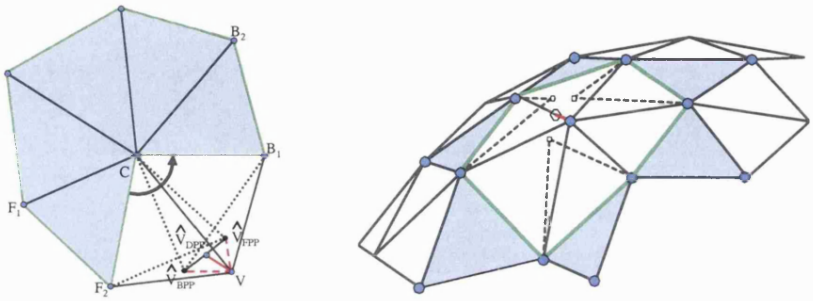


Figure 2.5: Extended parallelogram-based predictions. Left: Dual and, right: Multi-way (from [SKL02, COI02]).

Generalised parallelogram prediction (GPP). Isenburg and Alliez [IA02] presented a generalisation of the PP rule by Touma & Gotsman to polygonal 3-D meshes. They let the polygon information dictate where to apply the PP rule, and made predictions within a polygon rather than across polygons, as shown in figure 2.6.

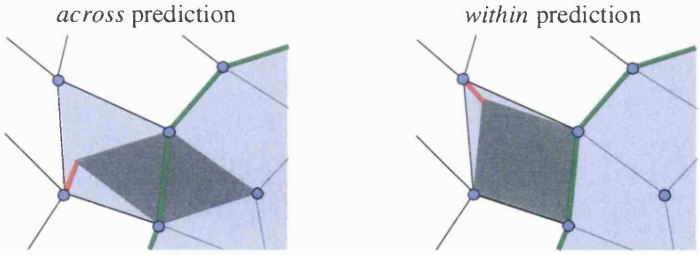


Figure 2.6: Generalised parallelogram-based prediction (from [IA02]).

Butterfly prediction (BP). The progressive coding scheme CPM (presented later in 2.2.2.1) used a geometry prediction method inspired by a butterfly subdivision scheme that inserted a new vertex by splitting an edge, and which calculated the location of the new vertex as a weighted sum of the surrounding vertices. Pajarola and Rossignac in CPM [PR00] extended this idea by approximating the position of a vertex A by a linear combination of its immediate neighbours a_j, with topological distance 1 on the triangulation graph, and the vertices c_j at topological distance 2, as shown in figure 2.7. Equation (2.4) gives the approximation A' of A:

$$A' = \alpha \cdot \frac{\sum_{j=1}^{k} a_j}{k} + (1-\alpha) \cdot \frac{\sum_{j=1}^{k} c_j}{k} \quad . \tag{2.4}$$

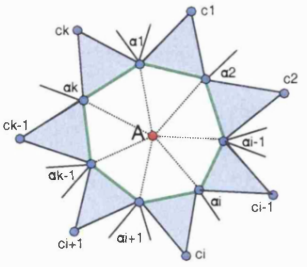


Figure 2.7: Butterfly prediction (from [PR00]).

Specifying a value of less than 1 for the parameter α denotes a weighted averaging between the two crowns a_i and c_i, and a value of more than 1 expresses more of an extrapolation based on the difference between the two crowns for estimating A. The value of α can be adjusted for each model if needed.

2.2.2 Progressive & hierarchical methods

When bandwidth resources are limited, the task of transmitting 3-D meshes needs special consideration. In the single-rate coding schemes described in the previous subsection, a receiver needs to download the whole of mesh connectivity, geometry, and property data before it starts rendering. To overcome such a limitation and allow rendering at different levels of detail (LOD) while still receiving mesh data, progressive compression and transmission schemes have been proposed in the literature [Hop96, PH97, TGHL98, PR00, LLK97, COLR99, AD01a]. These allow the encoding of a base mesh, which they transmit first, followed by refinements on connectivity, geometry, and possibly property data. Decoding can stop at any time, either constrained by network resource unavailability, or interrupted by the end-user.

In theory, building a progressive coder is expected to be less effective in terms of coding gain than a single-rate coder, since it requires the encoder to store additional information that allows the decoder to terminate its process at any instant. Recent mesh coding research [AD01a], however, reports fully progressive encoders with performance as good as the state-of-the art single-rate method [TG98]. Geometry coding in progressive mesh compression is still driven by connectivity coding, but new methods have emerged that either decouple, or reverse, this dependency [KG00, KSS00].

2.2.2.1 Connectivity-based compression

The progressive mesh compression process is closely related to mesh simplification or decimation methods, discussed in more detail below in paragraph 2.2.2.3, and which follow a certain mesh decimation primitive, or topological operation, laid out in figure 2.8. The basic representation of progressive meshes was introduced by Hoppe in *Progressive Meshes* (PM) [Hop96].

Hoppe described the typical progressive mesh encoding scheme, which first gradually simplifies a given orientable manifold 3-D mesh $\hat{M} = M^n$, using n successive edge collapse operations, $ecol_i$, into a base mesh M^0 with considerably reduced number of vertices, edges and faces:

$$\hat{M} = M^n \xrightarrow{ecol_{n-1}} \dots \xrightarrow{ecol_1} M^1 \xrightarrow{ecol_0} M^0 \ . \tag{2.5}$$

By logging the simplification operations and applying the reverse sequence on the base mesh, the original mesh can be reconstructed. It is easily observed that the edge collapse operations are invertible into vertex splits, $vsplit_i$, therefore the arbitrary triangle mesh $\hat{M}$ can be represented as a simple mesh M^0 followed by a sequence of n vertex splits:

$$M^0 \xrightarrow{vsplit_0} M^1 \xrightarrow{vsplit_1} \dots \xrightarrow{vsplit_{n-1}} \hat{M} = M^n \ . \tag{2.6}$$

A progressive mesh is then represented as follows:

$$(M_0, \{vsplit_0, vsplit_1, \dots, vsplit_{n-1}\}) \ . \tag{2.7}$$

Hoppe also suggested a method to select the proper edge to be collapsed at each step. The method he proposed sorts all candidate edge collapse operations into a priority queue, according to an estimated energy cost ΔE, which measures distance accuracy, scalar attributes accuracy, and the accuracy of the discontinuity curves of the mesh M^i. In each iteration, the algorithm applies the edge collapse operation of the lowest ΔE and re-computes the priorities of edges in the neighbourhood of the transformation.

Popović and Hoppe proposed *Progressive Simplicial Complexes* (PSC) [PH97] as an improvement over PM. PSC is not only able to handle non-manifolds, but allows topological changes during simplification and refinement, at the expense of compression efficiency. They encoded changes in both connectivity and geometry by introducing a generalisation of the vertex split operation. In the new progressive scheme, the mesh consists of a single-vertex base model and a sequence of generalised vertex splits. The PSC compression process records a code that indicates one of four possible configurations that result from a generalised vertex split operation.

Taubin et al. also proposed a progressive compression scheme called *Progressive Forest Split* (PFS) [TGHL98]. Similarly to the PM scheme, Taubin's PFS encodes a base mesh followed by a sequence of refinements. The vertex split operation in PFS is substituted by the *forest split*. This process cuts a mesh along the edges belonging to a forest, and re-triangulates the generated gap space. With the forest split operation, the PFS scheme can achieve better compression performance at the expense of reduced granularity.

Pajarola & Rossignac proposed *Compressed Progressive Meshes* (CPM) in [PR00], which, like PFS, trades granularity for improved compression gain. To encode connectivity, CPM groups vertex splits into groups called batches and uses a sequence of marking bits to index the vertex to be split within a batch. For geometry coding, CPM originally predicted a fully collapsed edge $e = (v_1, v_2)$ to its middle point $v = (v_1 + v_2)/2$, using a butterfly prediction scheme within the 1-ring and 2-ring from v. It was later improved by employing the half edge collapse of edge e to one of its endpoints v_1 or v_2 and predicting the new endpoint vertex position within a 1-ring only of the half-collapsed edge. Huffman coding was used to encode the prediction errors.

Although all previous methods discuss compression of geometry, with or without connectivity, in a progressive manner, they do not distinguish the encoding of different attributes but rather treat these in the traditional run-length single solution manner. Second, they do not integrate the coding of the structure data with the coding of the attribute data. Yet another truly progressive approach was suggested by Li and Kuo in [LLK97]. According to this coding method, both structure and attribute data are compressed progressively. During the encoding process, every output bit contributes to reduce the distortion, and the contribution of bits decreases according to their order of position in the bitstream. In this way, a series of models of continuous resolution can be decoded from a single bitstream, which is called an *embedded bitstream*. Such a coding scheme, however, does not exploit the fact that only the front parts of a model are visible to the viewer at any time and transmission of the invisible parts is simply a waste of bandwidth.

The encoding algorithm proposed by Li and Kuo operates as follows: A hierarchical structure for a 3-D graphic model is first built by making multiple passes through the vertices of the original mesh in order to organise the vertices into a layered fashion. Then, a mesh simplification step is applied, where vertices are classified as simple, complex, or boundary. Simple or boundary vertices can then be removed from the mesh, along with the triangles depending on this vertex, since these elements do not alter topology. The vertex and triangle removal process may result in a hole in the mesh. This hole is then filled by performing local re-triangulation. The delete-fill operation is repeated until further removal of any vertex would result in topological violation. The resulting mesh serves as the base mesh that is the coarsest approximation of the original mesh. The first refinement layer adds back deleted vertices to the base mesh by predicting their attribute data from the average of the local vertex neighbourhood.

This method also improves on the entropy encoding of attribute data residual prediction errors while encoding the 3-D model in an embedded bitstream. This is due to the use of a

successive quantiser with gradually refined step size. This last feature of this encoding scheme has been integrated by the MPEG-4 Synthetic Natural Hybrid Coding (SNHC) group into its state-of-the-art 3DMC coding scheme.

Cohen-Or et al. proposed *Patch Colouring* (PC) [COLR99] for progressive encoding of 3-D meshes. Their method removes an independent set of vertices at each iteration, where there are no adjacencies between vertices of the set in the original mesh. The vertex removal generates holes that are re-triangulated. The set of new triangles is called a *patch*, which is coloured according to one of two proposed techniques: 2- and 4-colouring. The colouring method to be chosen depends on the distribution of vertex valences at encoding time. The geometry is encoded by predicting each vertex position from the average of its neighbouring vertices.

Alliez and Desbrun proposed a valence-driven progressive coding method [AD01a] after observing that when encoding the connectivity of a mesh, the entropy depends on the distribution of vertex valences. They suggested a breadth-first degree-n patch conquest over the mesh surface, where a degree-n patch is defined as a set of faces incident on a common vertex of valence n. The patch is conquered by decimating one of the boundary edges and re-tiling the remaining polygon. As this process may not fully cover the surface of a 3-D mesh, null patches are generated to cover the unconquered triangles. The encoder then outputs a sequence of patch degrees and null patch codes. Additionally, a 'cleaning conquest' may improve the regularity of the resulting mesh. For geometry coding, they used the typical parallelogram rule prediction.

2.2.2.2 Geometry-based compression

Progressive methods of compressing geometry of 3-D meshes bear great similarity to transform coding used widely for still image, audio, and video compression. For 3-D meshes, such transform coding techniques are based on the construction of a set of orthogonal basis functions for decomposition of a polygonal mesh into signals. The signals have components across a spectrum of frequencies. The low frequency components are regarded as the foundation data (in 3-D graphics these signals usually correspond to smooth features on a mesh), whereas high frequency components are considered noise to be removed (signals corresponding to mesh discontinuities such as creases, folds and corners). In the case of compression of 3-D mesh signals, the underlying assumption is that a relatively good approximation of the original mesh can be obtained by using only a small number of the low-frequency basis functions, whereas further quality refinements can be obtained by decoding higher frequency component coefficients. In the following paragraphs, three significant progressive geometry compression methods are described where the signal decomposition functions are based on mesh vertex adjacency, wavelet

functions, or Fourier transforms. In general, we refer to such methods of 3-D mesh compression as *signal processing based* methods.

Spectral methods. Karni & Gotsman proposed a spectral method for progressive geometry compression [KG00] of triangular meshes. The formulation of the geometry signal decomposition is as follows. For a 3-D mesh consisting of n vertices the *Laplacian operator* L is constructed as the symmetric $n \times n$ matrix, $L = I - DA$. The adjacency matrix A is populated by assigning $A_{ij} = 1$ if vertices i and j are incident, 0 otherwise. I is the identity matrix and D is a diagonal matrix with d_{ii} set to the degree of the i-th vertex. The eigenvectors of L give an orthogonal basis of an n-dimensional space. The spectral analysis is achieved by obtaining the inner products of each of the x, y, z vertex coordinate vectors with the basis vectors. Obviously, there are three different spectra, one per dimension x, y, z of the geometry, which exhibit different frequencies depending on the directional geometric properties of the mesh (e.g., curvature). High frequency components of smooth mesh spectra have insignificant values, which can be truncated or quantised to a small number of discrete values. This scheme naturally enables progressive transmission by encoding the low-frequency spectral coefficients before those of high-frequency. The connectivity of the mesh can be compressed using any best method described earlier in paragraphs 2.2.1.1 and 2.2.2.1.

The disadvantage of spectral techniques lie in the calculation complexity of the eigenvectors of the Laplacian operator L. This limitation makes spectral methods impracticable for compressing large still models, let alone dynamic mesh compression in real-time. Karni and Gotsman proposed efficient mesh partitioning to alleviate computation complexity. The partitions need to be small enough to reduce computation time, but small partition sizes reduce compression efficiency. Spectral methods are most suitable for encoding smooth meshes, for which the bulk of the transformed geometries are concentrated in the low-frequency area of the spectrum.

Fourier transform methods. Karni and Gotsman's spectral compression method described in the previous paragraph can be seen as an extension of Fourier analysis on 3-D mesh compression. Past literature also proposed the use of Fourier transforms on 3-D meshes, where the main aim was not geometry compression, but geometric detail suppression. For example Lee and Lee [LL98] proposed a Fourier transform method for reducing the geometric details of polygonal mesh models representing solids.

Wavelet transform methods. Khodakovsky, Schröder and Sweldens introduced a wavelet-based technique for progressively compressing geometries [KSS00]. They convert an arbitrary manifold mesh into a semi-regular mesh with vertex valence 6 using the MAPS algo-

rithm, which finds a coarse base approximation of the arbitrary mesh and iteratively subdivides each triangle by four at each refinement step. Then, with a wavelet transform, they replace the original mesh with the coarse base mesh and the sequence of wavelet coefficients, which expresses the difference between successive levels. The distribution of wavelet coefficients is centred around zero, and these are entropy coded with a modified zerotree[2] algorithm. This geometry compression technique, although powerful, was only shown upon manifold meshes and requires the source mesh to have a strongly regular structure. If applied to an arbitrary mesh, the original connectivity cannot be preserved.

2.2.2.3 Mesh decimation methods

Progressive coding methods of connectivity and geometry are used to compactly represent 3-D mesh hierarchies *bottom-up*, by refining a coarse base mesh to a specified level of detail. Following the inverse *top-down* hierarchy build, a detailed mesh can be successively decimated to a coarse base mesh by iteratively removing information. This paragraph surveys the distinguished literature that addresses the problem of reducing the *complexity* of a 3-D mesh through decimation or simplification techniques. Although mesh decimation methods can be seen as contrasting to progressive encoding, they are included under the progressive compression subsection in this chapter since they are usually complementary to progressive schemes, and most state-of-the-art progressive methods have been derived by applying the inverse sequence of simplification operations on the coarse base mesh that resulted from the decimation.

Surveys by Garland [Gar99a] and Kobbelt et al. [KBB+00] discuss typical application areas where mesh simplification can be used. These include: (i) overly sampled models, which are usually uniformly sampled (such as those generated by 3-D scanners), (ii) overly tessellated surfaces, (iii) level-of-detail rendering, either to speed-up rendering of a dense mesh on a high-end processor, or to render a lower quality of the same mesh on a low-end device such as a PDA, (iv) progressive transmission, where a 3-D mesh can be transmitted over a low-bandwidth connection.

Garland [Gar99a] and Cignoni et al. [CMS98] provide comprehensive listings of available decimation methods. All above surveys build their classifications by identifying three characteristic key elements of simplification algorithms:

1. the *properties* that drive the simplification process,

2. the *error metric* that evaluates the approximation process, and,

[2]The embedded zerotree wavelet (EZW) coding algorithm, pioneered by Shapiro [Sha93], is an efficient technique based on quad-trees that encodes the coefficients of a wavelet transform.

3. the *quality criteria* that the simplification process strives to achieve.

Properties that drive the simplification process can be any geometric and topological entities, or attributes. Typically, most decimation algorithms apply atomic changes to the mesh, affecting its connectivity. Such possible decimation primitives are depicted in figure 2.8 [KBB+00] and are called *topological operations*. It is possible to introduce a global *ranking* of candidate topological operations on a mesh, by assigning a cost value to each operation and pushing them to a priority queue, thus constructing *incremental decimation methods* that dominate the field of mesh simplification [SZL92, RB93, CVM+96, RR96, Hop96, GH97, CCMS97].

The error metric is where incremental decimation algorithms differ the most, and it may evaluate different properties of the mesh, ranging from textural and colour attributes to geometric distances between various topological constructs (edges, faces, vertices). Error evaluations between successive steps of the decimation process are called *local*, as opposed to *global* evaluations between the current decimation step and the original mesh. Error evaluation is usually intertwined with the topological operation (simplification) it evaluates. Section 2.5 discusses various error metrics, measuring geometric distance, attribute deviation, or energy optimisation, introduced by known simplification methods. The extension of some error expressions to animated meshes is further outlined in subsection 2.5.3.

The quality criteria conceptually represent a separate entity, but in practice they are closely connected to the error metric, and serve to define the order in which the topological operators can be applied to optimise geometric criteria (surface smoothness, or 'roundness', etc.), topological criteria (achieve mesh regularity, control the number of separate mesh components, etc.), or attribute criteria (minimise texture distortion, minimise surface normal deviation, etc.).

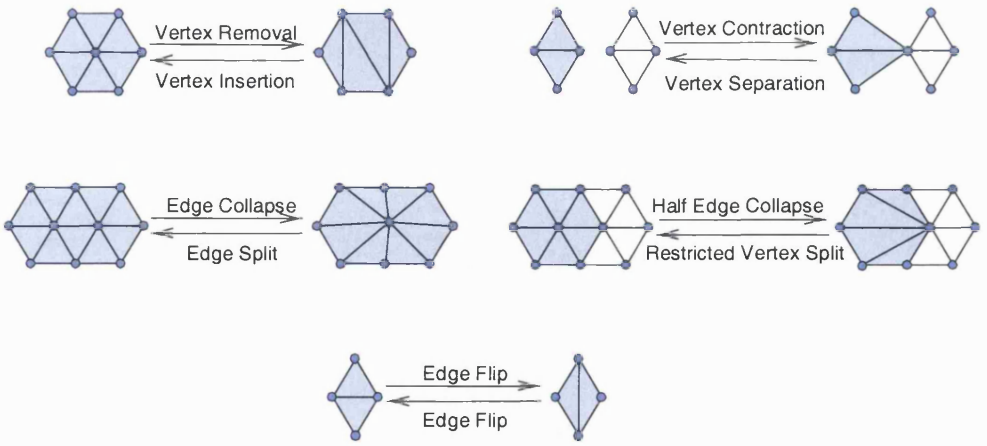


Figure 2.8: Topological operators (from [KBB+00]).

In the remainder of this subsection, an overview of key simplification methods is presented, based on the topological operator that influences the decimation process.

Vertex removal. Pioneering mesh simplification work that introduced the term "mesh decimation" was done by Schroeder et al. in [SZL92]. They proposed the removal of a vertex and re-triangulation of the resulting hole. The vertex to be removed at each iteration was the one whose distance from the average plane of the re-triangulation fell below a user-specified threshold. This method was later included in the popular visualisation toolkit VTK.

Cohen et al. [CVM+96] proposed simplification envelopes as another decimation process, according to which two offset surfaces, the outer envelope and the inner envelope, were generated as copies of the vertices of the original surface at bounded distances $+\varepsilon$ and $-\varepsilon$ along their normals respectively. The envelopes were not allowed to self-intersect; where the curvature would create such self-intersection, ε was locally decreased. These envelopes guided the simplification process by allowing iterative vertex and face removals in a way that the simplified surface remained within the envelopes, thus guaranteeing a deviation of no more than ε from the original surface and resulting in very good fidelity. For the same reason, though, this algorithm was also self-limiting in the sense that it prevented drastic mesh reduction in order to preserve topology and to guarantee error bound of ε.

Edge collapse. Ronfard and Rossignac [RR96] observed that each vertex in the original model lies at the intersection of a set of planes, those defined by the faces adjoined to the vertex. This observation underlies their simplification algorithm, which associates each vertex with a *zone* of incident planes. The error metric, or cost, is defined as the maximum distance between the vertex resulting from an edge collapse and the planes in its zone. When collapsing edges, the zone of the new vertex is the union of the zones of the edge endpoints. The algorithm proceeds by iteratively collapsing the edge associated with the lowest cost.

Hoppe's method [Hop96] for constructing progressive meshes was discussed in paragraph 2.2.2.1 and is also based on edge collapse sequences that decimate a given mesh based on minimisation of an energy function.

Vertex clustering. Rossignac and Borrel [RB93] described a very general technique for simplifying triangulated 3-D meshes, called vertex clustering. They subdivided the model's bounding volume into a regular grid of boxes of user-specified size and suggested that all vertices within each box be merged together into a new representative vertex. The new set of representative vertices is triangulated by preserving as much as possible of the original topology and forms the simplified mesh. A conceptually similar approach was proposed by Reddy [Red96], where clustering is performed based on perceptual properties of the mesh.

Garland and Heckbert [GH97], in a similar approach to Ronfard and Rossignac's [RR96], proposed a simplification method whose error metric is based on the calculation of the distance of a vertex to a set of planes. The method operates by associating each vertex with a 4×4 symmetric matrix and generalising the edge collapse operation as a vertex-pair contraction where the vertex pair need not necessarily be connected by an edge. The method is called Quadric Error Metric simplification since the symmetric matrices are constructed so that their level surfaces are almost always ellipsoids. This method, although very similar to [RR96], is more general as it can simplify non-manifolds with disconnected components.

Fitting. Although topological operations discussed so far aim at reducing mesh complexity, edge flip can be used to adjust a mesh to help preserve some features, such as volume, creases, and folds, or to optimise the error metric by adjusting the geometric position of the vertices through vertex displacement. Such operations have been used by Hoppe [Hop96] and Ciampalini [CCMS97] and, in general, they assist the leading simplification methods to achieve global mesh geometry (vertex displacement) and connectivity (edge-flip) optimisations.

2.2.3 Other approaches

2.2.3.1 Compression of large 3-D models

Shikhare discussed compression literature for large-scale 3-D data models and proposed a custom scheme for 3-D model compression [SBM02, SBM01]. The technique he proposed automatically detects repetition of component shapes, and repetition of groups of component features, and then compactly encodes the geometry using a 'master geometry - instant transform' hierarchy. The master geometry can be encoded using well known suitable geometry compression algorithms in the literature.

Bajaj et al. [BPZ98] proposed a system for compression and transmission of very large CAD models. Their system extended existing compression schemes to handle quadrilateral and general polygonal models, as well as CAD models with smooth NURBS patches. The resulting compressed bitstream is partitioned into variable length blocks which can be transmitted independently. They also proposed a novel layering structure to organise the redundant information of the CAD model.

2.2.3.2 Resilient streaming of 3-D models

Error resilient transmission of static 3-D meshes is a field closely related to the topic of 3-D mesh compression, on which it often passively relies, or actively co-operates with, and recently received increased attention.

Most notably, Bajaj et al. [BCPZ98], designed a mesh layering algorithm that first par-

titions a mesh into classic triangle strips and fans, and then forms vertex layers based on a contouring scheme. The latter is used to improve the reconstruction process by speeding up searches for common triangle edges in both the triangle layers and the vertex layers. Due to the small size of the layers this search is performed fast. By construction, these layers exhibit strong locality property, which is exploited to achieve passive error resilient transmission. The two types of layers are interlaced during transmission and loss of one such layer only affects the rendered model locally.

Taking a proactive approach, Al-Regib [RA02, RA01] designed an error resilience method based on forward error correction codes (FEC), which are transmitted as redundant information, by simultaneous reduction of the source rate. They selectively provided higher redundancy to certain parts of a 3-D mesh that are more likely to lead to higher distortion of the model. The same team also took another step forward by extending the principle of error resilience into 3-D animations [RARM02], through redundancy and re-transmission of the most sensitive parts of a dynamic model.

Bischoff & Kobbelt [BK02b, BK02a] proposed a progressive mesh coding algorithm which, based on the construction of overlapping ellipsoids covering the mesh's interior and a set of refinement details, transmits the ellipsoids with increased reliability than refinement data.

To better organise the literature material of this thesis, the most relevant error resilient approaches are critically reviewed in subsection 2.4.4.

2.2.4 Summary of static 3-D mesh compression

Table 2.1 summarises the main literature in statically encoding a 3-D mesh in single-resolution or in progressive manner. Table 2.2 summarises the decimation methods.

2.3 3-D Animation Coding and Compression

In the 3-D coding literature discussed above, all of the work focused on encoding and compression of connectivity and geometry of static meshes. As modelling packages become more sophisticated, complex animations can be generated from static meshes that need to be transmitted fast and rendered in real-time. Animation data should effectively be seen as another streaming media type and no longer as a plain 'bag of polygons', with little change between frames, to be passed down the hardware rendering pipeline. Animated geometry, in particular, should be seen as a first-class media stream that, if coupled with compressed texture and connectivity data, can produce realistic 3-D simulations streamed over wide area networks. In order

Table 2.1: Summary of single-rate and progressive encoding methods for static 3-D meshes.

CATEGORY		SINGLE-RATE		PROGRESSIVE	
CONNECTIVITY		Triangle strips	[Dee95]	Progressive Meshes	[Hop96]
		Generalised strips & fans	[Cho97]	Progr. Simplicial Complexes	[PH97]
		Topological Surgery	[TR98]	Progr. Forest Split	[TGHL98]
		Layered Decomposition	[BPZ98]	Compr. Progr. Meshes	[PR00]
	VERTEX	Touma & Gotsman	[TG98]	Embedded coding	[LLK97]
	VERTEX	Alliez & Desbrun	[AD01b]	Patch Colouring	[COLR99]
	VERTEX	Guéziec	[GBTS99]	Valence-driven	[AD01a]
	FACE	Cut-border Machine	[GS98]		
	FACE	Edgebreaker	[Ros99]		
GEOMETRY		Linear	[TR98]	SIGNAL: Spectral compression	[KG00]
		Delta	[Dee95, Cho97]	SIGNAL: Wavelet compression	[KSS00]
	PARALL/GRAM	Simple	[TG98]		
	PARALL/GRAM	Bi-directional (Dual)	[SKL03]		
	PARALL/GRAM	Generalised	[IA02]		
	PARALL/GRAM	Multi-way	[COI02]		
		Butterfly	[PR00]		

Table 2.2: Summary of decimation methods for static 3-D meshes.

CATEGORY		METHOD NAME	
DECIMATION	VERTEX REMOVAL	Simplification Envelopes	[CVM+96]
DECIMATION	VERTEX REMOVAL	Mesh Decimation	[SZL92]
DECIMATION	EDGE COLLAPSE	Full-range approximation	[RR96]
DECIMATION	EDGE COLLAPSE	Progressive Meshes	[Hop96]
DECIMATION	VERTEX CLUSTERING	Vertex clustering	[RB93]
DECIMATION	VERTEX CLUSTERING	Perceptual clustering	[Red96]
DECIMATION	VERTEX CLUSTERING	Quadric Error Metrics	[GH97]
FITTING	EDGE FLIP	Global Error	[CCMS97]
FITTING	EDGE FLIP	Progressive Meshes	[Hop96]
FITTING	VERTEX DISPLACEMENT	<many>	

to prevent the modelling tools from swamping the run-time rendering engines and congesting networks, compression of the animation data is paramount.

Research in the area of generic time-dependent geometry compression has been limited. From its early stages, representative work was motivated either by compression as storage space savings [Len99, GSK02, AKKH01, AKKH02, YKL01, YKL02], or just efficient dynamic mesh representations [SPB00], possibly combined with spatio-temporal progressive representation or levels of detail [AM00, SP01]. The approaches described below in detail go beyond plain rigid object transformations, and study animation as composed of a coarse mesh representation

(or mesh components) for the gross movement and residual predictive coding (or differential coding to average components respectively), for the details. None of the work presented below, however, considers the network from any viewpoint.

From a data-encoding standpoint for visual data, one could separate the following work into: i) *coding* literature only [SPB00, AM00, SP01], ii) *geometry compression* literature [Len99, GSK02], and, iii) *motion-compensated compression* literature [AKKH02, YKL02]. The following subsections reflect this classification. All literature presented below considers isomorphic dynamic meshes, except from Gupta's [GSK02] dynamic geometry compression algorithm.

2.3.1 Dynamic 3-D mesh coding

Alexa and Müller [AM00] described an approach for representing dynamic mesh sequences based on *Principal Component Analysis* (PCA). They restricted their work to scenes comprising animated polyhedral shapes described by key-frame geometries. They elaborated on the idea of *shape space*, which was originally used for interpolating between implicit descriptions of shapes. Alexa and Müller attempted to structure the elements of an animated scene in terms of a linear shape space. They derived linear spaces by performing a basis transformation on appropriately represented animated polyhedral models. Using PCA, they first determined the average shape that contains the common properties of the shapes in all key-frames. Other components represent differences to this basic average shape. Their animation method is based on the computation and off-line transmission of the set of basic shapes as determined by the PCA analysis.

This work differs from others discussed later in this section in the fact that the majority of other dynamic geometry compression schemes are based on prediction that they use to remove temporal redundancies from a specific animation sequence. Alexa's work based on principal animation components led to the abstraction of linear shape spaces that decouples animation from geometry and can be used to drive the same animations over different geometries. Their approach also overcame other problems related to key-frame based animations; it avoids complete object descriptions for each key-frame, and introduces the LOD concept in dynamic meshes, which would not be possible to exploit otherwise in key-frame animations. Nevertheless, the method cannot easily be applied to streaming interactive animations. It requires a considerable amount of pre-processing during which the PCA analysis determines the base shapes. Although the animation representation is inherently progressive, the authors do not specify any method to address loss of a basic shape matrix. This would inevitably result in the animation reaching

terminal resolution equal to the best model quality experienced just before the loss, unless some mechanism is devised to account for the missing shape. Animation frame sizes, though, could be decreased by prediction and entropy compression of the basic shape matrices.

Shamir et al. [SPB00] introduce T-DAG, an adaptive multi-resolution representation for dynamic meshes with arbitrary deformations including attribute, position, connectivity and topology changes. They also provide an on-line algorithm for constructing the T-DAG, enabling the traversal and use of the multi-resolution model for partial playback while still constructing it. Therefore, their flexible mesh representation scheme can be used for compressing 3-D animated meshes. However, such a direction is not being followed in their research, and the T-DAG structure does not refer to compression, but only to coding for efficient representation of dynamic meshes. In a companion work [SP01], Shamir further exploits T-DAGs to represent spatial and temporal levels of details for time-dependent deformable meshes. This use of T-DAG is further discussed in subsection 2.4.3.

2.3.2 Dynamic 3-D mesh compression

The idea of exploiting temporal redundancy in a series of static meshes or a dynamic mesh by differentially compressing them was first introduced in MPEG-4 SNHC ad-hoc group in 1997. The ISO output document [ISO98] outlined the core experiment plan for coding and compression of connectivity, geometry and photometry of static 3-D models and, optionally, for dynamic meshes. Although dynamic mesh compression was mentioned, the group did not show any related output, and the whole specification relies on BIFS-Anim to handle animation.

BIFS-Anim is the animation coding protocol made available for standardisation by MPEG-4 [ISO01a]. Its advantages over 3DMC are obvious; BIFS-Anim follows a temporal prediction scheme for dynamic fields of MPEG-4 nodes and, therefore, exploits the temporal redundancy inherent in time-dependent meshes to achieve rate reduction. However, in order to be fully integrated into the standard as an operational protocol, BIFS-Anim suffers from overheads associated with the system-level headers (SL headers). For example, timestamps and sequence number information can be provided by RTP [SCFJ03] in a streaming architecture and duplication of the corresponding fields in the SL headers is unnecessary. SL packets are also strictly defined to be 256 bytes long. This would require payload fragmentation that does not respect common rules, such as the Application Level Framing (ALF) rule [CT90]. Furthermore, although theoretically BIFS-Anim can express any mesh deformation using animation bitmasks to signal the dynamic fields, practically the protocol requires definition of such masks prior to the animation, thus making real-time animation compression in MPEG-4 a more difficult task.

Last, the standardisation body made no provisions for adaptation of BIFS-Anim streams to network conditions. The visual result of a high-resolution model, or an intensely animated model, under network congestion is unpredictable, with poor quality or discontinued animation most likely to occur.

Also in the context of MPEG-4, it has been noted that up to 80% of a BIFS scene's file size may be occupied by data representing geometry and animation [BS02], therefore the SNHC, and more recently the Animation Effects (AFX), groups adopted a number of technologies that allow the expression of realistic 3-D model and animation compression features. In the past, geometry size had been addressed by 3-D model compression (3DMC), whereas interpolator-based animation technologies were later developed (Coordinate, Position, and Orientation compressed node interpolators). The latter provide reported gain between 30% and 100% over the previous compression method of `PredictiveMFField` in MPEG-4 BIFS [KJH+02].

In [Len99], Lengyel described a compression method for time-dependent geometry. He first introduced the distinction between static and dynamic mesh coding and made a pioneering effort to express dynamic mesh deformations generically, by placing vertices of a dynamic mesh in the rows and their sampled motion trajectories in the columns of a matrix V. He followed this input vertex matrix placement by appropriate segmentation that greedily clustered columns together in order to enable their coding in local coordinate systems. The encoder was designed to calculate series of affine transformations that best matched the base mesh to the current, using least-squares methods. Segmented clusters of vertices were considered independently. The encoder could also use a feedback-loop with the inverse mesh transformations, using the previous frame's mesh as the base mesh against which to match. This is analogous to the I frames of MPEG and other natural video encodings. After removing as much coherence as practically possible through the previous mesh transform coding techniques, the best case would be for the residuals to be negligible. However, typical non-rigidly moving objects decomposed into static base shapes may result in non-zero residuals. These were handled with row difference prediction on the input vertex matrix.

The method described above by Lengyel, removes considerable temporal redundancy from a dynamic sequence, as reported in the experiments. Results, though, show good compression only when the sequence is close to an affinely transformed hierarchy. If not, compression ratios are compromised. In addition, calculating the series of affine transformations can be a very expensive operation for large input vertex matrices. Processing within a frame may incur considerable overhead too. For example, column sorting and prediction may be desirable on such general vertex matrices, where measures of column similarity are required. On large

sequences this may be inefficient, or an effective column similarity may not be found at all, as happens with the test model CHICKEN. (Details of this sequence can be found in appendix A.) Furthermore, Lengyel reports unpleasant visual artifacts resulting from vertex clusters jittering together, if affine transformation matrices are quantised with fewer than 16 bits. The human visual system is much more tolerant of per-vertex jitter and work will be needed to determine an error measure with more correlation to perceived quality to drive the proposed encoding method.

Gupta et al. [GSK02] presented another coding algorithm for time dependent meshes based on image registration techniques. The proposed encoding employed the Iterative Closest Point (ICP) algorithm for motion prediction between two successive 3-D frames, as well as for connectivity registration. Motion segmentation was performed in a way that separated the vertices into two sets; the first set containing those vertices whose motion could be encoded with a few affine parameters, whereas the second set contained those vertices needing further encoding of residual errors along with the estimates of affine motion parameters. Also, within the second set, those vertices associated with large residual errors under affine mapping were submitted to further motion coding with Newtonian (non-linear) motion estimates. In their follow-up research [GSK03], Gupta et al. studied the insertion of I-frames in the encoded sequence based on SNR objective performance criteria. The SNR metric was also used to derive rate-distortion curves that allowed them to appropriately determine the number of bits for encoding the error residuals.

Gupta's ICP-based algorithm is novel in the sense that it expands 3-D dynamic geometry coding with dynamic connectivity. However, this advantage comes at the expense of considerable computation time due to the operation of the ICP algorithm on a per-frame basis. This restricts the potential use of the proposed encoding scheme to storing animations only and renders it unsuitable for streaming. Moreover, the encoding process does not take the network into account, and losing vital affine mapping motion parameters would result in great distortion. Another interesting aspect of Gupta's work [GSK02, GSK03] is the discussion on error measurement in 3-D sequences, and the introduction of an SNR variant for 3-D signals, based on signal variance. Subsection 2.5.3 discusses this expression of error along with other error metrics. In a related area, chapter 4, following similar to Gupta's motion estimation methods, shows receiver-based concealment of 3-D dynamic sequences for missing frames.

2.3.3 Motion compensated compression

Object-based compression and composition of 2-D synthetic video coding emerged early in the course of MPEG-4 standardisation. In particular, motion compensated 2-D triangular geometry compression was reported by van Beek, Tekalp and Puri [BTP97]. They proposed an integrated approach to compressing 2-D video objects using mesh-based motion estimation and compensation to replace the block-based motion compensation part of a standard natural video codec. Their method comprised two ways of coding the 2-D mesh motion vectors; the first way is based on using motion vectors of surrounding vertices as predictors; the second way is based on using motion vectors of a block-based motion video coder as predictors. The spatial and block-based motion vector prediction forms the basis for the 3-D motion-compensated compression methods described below.

In [AKKH01, AKKH02] Ahn et al. described a motion-compensated 3-D animation coding algorithm in analogy to the popular video coding standards of H.26x and MPEG. Their codec was based on the following steps; mesh decomposition to generate triangle strip segments equivalent to the macro-blocks of the 2-D video counterparts; motion compensated prediction for each segmented block; and 1-D discrete cosine transform (DCT) coding for the residuals. The DCT coefficients are quantised, DPCM coded, entropy coded and multiplexed. As in the 2-D video world, the authors distinguish between reference (intra), predictive, and bi-directional meshes, or frames, which they encode with the same repeating sequence patterns as H.26x and MPEG. Encoding of the reference mesh is performed with the parallelogram prediction rule by Touma and Gotsman [TG98]. For the predictive frames, encoding involves subtracting the real motion vector of each vertex from the average motion vector of all vertices of the same segment. Bi-directional frame encoding is achieved by linearly interpolating vertex positions from previous and subsequent time instances. Compression ratios are only reported in comparison to MPEG-4's SNHC 3DMC codec for static meshes, which they outperform. However, they only use a point set measure, the Hausdorff distance, and they restrict the comparison only to the starting 28 frames of the CHICKEN sequence, which are not representative of high motion that their codec could exploit.

The above encoder proposed by Ahn and colleagues represents an elegant translocation of known coding methods from motion compensated 2-D video coding to the 3-D space, and may indeed give good compression ratios for certain sequences. There is, however, a substantial difference between the 2-D and 3-D encodings: 2-D video is a bounded rectangular area, coding is performed at pre-specified resolutions, and the signal (luminance and chrominance) is of bounded range. 3-D meshes can be of arbitrary shape, resolution, size, and motion. This may

incur the following trade-off: very dense triangular meshes will indeed result in small residuals by the parallelogram rule because each segment's vertices will carry high spatial redundancy. But a dense mesh will also generate a very high number of such triangle strips, thus increasing processing time and memory. On the other hand, in medium to low detail meshes, the segments do not necessarily exhibit high coherence as their vertices typically lie in a sparser area. Therefore, the average motion vector used to compute the residuals may not be appropriate to reduce the dynamic range of the signal and achieve high compression.

Further to the above discussion, Ahn's work implicitly assumes triangular meshes only. Other types of polygonal meshes are not considered and, in general, efficient triangulation of a generic polygonal mesh may be an expensive pre-processing step for compression. Finally, such coding does not naturally adapt to network congestion to achieve graceful degradation of the represented animation. Any attempt to provide fine-grained control over the output bitrate would typically involve quantiser step adaptation. However, using coarser quantisers unevenly over mesh segments, or greater clusters of those segments (components), will lead to the generation of surface cracks.

Yang et al. [YKL01] presented another motion-compensated 3-D coding scheme for animated meshes, based on vertex-wise motion vector prediction. Intra-frame coding was applied to the first frame of the sequence. It was achieved by employing Touma and Gotsman's encoder [TG98] for the connectivity data and the dual parallelogram prediction method by Yang, Kim and Kuo [YKK01], for the geometry data. Subsequently, the prediction errors were quantised and arithmetically coded. In inter-frame mode they suggested motion vector prediction, which was done upon neighbouring vertices' motion vectors. The neighbourhood of a vertex was defined not as the set of geometrically nearest vertices, but as those vertices already encoded and located within a pre-specified distance from the reference vertex. The distance between two vertices, in their case, referred to the number of edges along the shortest path connecting them. The encoding process involved solution of an optimisation problem that minimised the magnitude of the error motion vector. To reach a solution, the encoder had to keep track of all motion vectors in all frames, thus the minimised error vector removed redundancy incurred by global deformation of the 3-D mesh. Yang and his colleagues made the observation that the error vector reflected the local deformation and might still have redundancy. To exploit this redundancy, the error vector was further predicted spatially or temporally. By monitoring the temporal correlation of the error vectors during encoding, they could predict the error vector from its corresponding one in the previous frame in case of high correlation, or by averaging the spatially adjacent error vectors otherwise.

The previously described method is a fine approach to encoding 3-D animation sequences. However, to optimise the error vectors on a per frame basis is a tedious and costly task with possible impact on the performance of the encoder for large meshes. Further to this, the encoding process implicitly assumes that for any generic 3-D mesh, an appropriate traversal of the vertices can be built, as is done in intra-mode by Touma and Gotsman's connectivity encoder. If no such traversal exists that maximises the number of prediction coefficients, we may face a situation of having only a few or no predictors for some vertices. In the reported results for the 3-D sequence BOUNCEBALL, for example, it is observed that the error vector has low correlation between frames, and is predicted with the spatially adjacent error vectors. (Details of this sequence can be found in appendix A.) Luckily, the sequence represents a regular mesh, therefore spatial prediction of error vector is effective. Had the mesh been non-regular, it is questionable whether a good compression ratio would have been achieved. Finally, in Yang's work there is no consideration of coding for network QoS. In chapter 4, among decoding options that provide robustness against packet loss, a method based on motion vectors is presented.

2.3.4 Summary of dynamic 3-D mesh coding

Table 2.3 summarises the reviewed methods for coding and compressing time-dependent 3-D meshes.

Table 2.3: Summary of dynamic 3-D mesh coding methods.

CATEGORY	METHOD NAME		SUMMARY NOTE
CODING	Alexa & Müller	[AM00]	Shape space, PCA analysis for key-frame animated polyhedral shapes
	Shamir et al.	[SPB00]	Dynamic mesh encoding with T-DAG representation
	Shamir et al.	[SP01]	Spatio-temporal LODs with T-DAG
GEOMETRY COMPRESSION	Lengyel	[Len99]	First compression effort, segmentation, affine transforms and residual prediction
	Gupta et al.	[GSK02]	ICP model registration, non-isomorphic meshes, affine and non-linear motion parameters, I-frame optimised positioning
MOTION-COMPENSATED	Ahn et al.	[AKKH02]	Triangle strip decomposition, segment prediction, residual DCT, DPCM, entropy coding
	Yang et al.	[YKL02]	Vertex-based motion vectors, temporal & spatial 2-stage optimised residual vector prediction

2.4 Quality of Service for 3-D Mesh Transmission

In the previous sections, 3-D model simplification and encoding were presented from a signal processing perspective. Most literature presented therein dealt with mesh size reduction, either in some resolution mode similar to the original mesh through signal compression, or in a lower resolution mode dictated by some mesh decimation technique, or both of the above. The signal undergoing processing represented one of mesh geometry, connectivity, photometry attributes, or combinations of those.

In modern applications related to 3-D graphics and virtual environments, where interactivity is paramount, new parameters are introduced when compressing a still or a dynamic 3-D mesh. Clearly, state-of-the-art applications imply that a key requirement is the ability to transmit meshes to a remote terminal, or rendering client, at a potentially reduced or varying rate.

In this section, a deeper exploration of the literature is presented to highlight existing research efforts dealing with transmission of 3-D models over packet networks and issues associated with quality of service. A taxonomy of such literature is built from a network and application QoS perspective.

2.4.1 Expressions of QoS in the 3-D world

Despite the increasing number of applications using 3-D graphics to provide visual information, a number of challenging issues remain to be addressed as far as quality of service (QoS) for 3-D media delivery and 3-D content adaptation is concerned. In an attempt to identify these issues, one often comes across the difficulty of defining what the terms 'QoS' and 'adaptation' themselves mean in the 3-D world.

There seems to be a diverse expression of QoS in the area of 3-D graphics transmission in existing literature. This is mainly due to the fact that in order to fully realise systems capable of delivering and rendering 3-D content efficiently to potentially large number of heterogeneous receivers, one has to encompass at least two scientific disciplines: *computer graphics* and *networked systems*.

Given this diversity and the breadth of the topic, it is wise at this stage to list how QoS is perceived in the literature of these disciplines. A closed definition of QoS in 3-D is not yet provided at this stage as this would restrict its meaning to past and current developments in applications from either graphics or networking worlds. Instead, the remainder of this section shapes the term by delving into literature related to 3-D graphics processing for rendering and network delivery. Then, after identifying the different forms QoS takes in the 3-D world, critical

review of key literature for each form of QoS will be presented.

The following three different flavours of QoS in 3-D are identified in the literature:

Terminal QoS: Van Raemdonck, Lafruit, and Pham Ngoc, discussed *architectural* [NRL+02] and *rendering performance optimisation* [RLS+02, LNDB00] issues for streaming 3-D content to client terminals.

They used the term "3-D computational graceful degradation" and "scalable 3-D graphics" to introduce features of their proposed 3-D scene rendering technology. According to it, the decoding and rendering time of 3-D graphics content can be predicted using complexity estimation functions, so that the rendering process can be adapted to the terminal's processing capabilities. Their work shows early awareness of the need for adaptation of 3-D graphics content through: i) a heterogeneous set of terminals, ii) LOD management. It forms a statement of QoS in 3-D from a terminal processing workload viewpoint; therefore one can refer to this expression of QoS as *Terminal QoS*.

Application/System QoS: In a slightly different context, Martin has shaped the term "adaptive graphics" [BM03] around the generic notion of *transcoding*, where conversions of the same 3-D object to different modalities (resolutions, or other forms of multiple representations) are used for transmission to clients with different rendering and network capabilities. Such modalities are either predefined, or applied on-the-fly, and optimal combinations of those in distributed 3-D visualisation applications are orchestrated by a single system, a prototype of which, ARTE, has been developed and demonstrated (the system is reviewed further in subsection 2.4.4).

In the context of ARTE, Martin recognises 3-D model compression, progressive coding, model simplification, LOD management, and image-based coding and rendering of 3-D objects, as optimisation methods for transmission and rendering of 3-D models. However, the notion of QoS that is being achieved through the term "graphics adaptation" is that of different descriptions of 3-D media, or resolutions, from a visualisation system's point of view. This expression of QoS is termed *System QoS*, or *Application QoS*.

Monitoring and recording system resources in 3 layers, namely OS workload, application performance (e.g., frame rate), and network (latency and bandwidth), is also a key feature of ARTE in providing QoS, but it is not specified what timescales are met for adaptation. From a 3-D visualisation perspective, recording such resource information is *desirable* to predict future QoS levels and adapt to them. From a networking application's point of view, regular and frequent network feedback is *mandatory* to allow the 3-D source to react promptly to resource variations.

Network QoS: Olbrich and Pralle [OP99] describe a distributed system for three-dimensional exploration in the context of high-performance networked computing, which supports 3-D scene processing, on-line visualisation, and interactive steering. Despite the fact that their system is essentially just a high-performance visualisation application, the terms "continuous 3-D media streams" and "streaming of 3-D scene sequences" are used, denoting that streams representing possibly dynamic 3-D content will be transported over a network in a similar way to other continuous streams, such as video and audio. In addition, geometry compression is mentioned as a means of reducing the size of the source representation of 3-D scenes before streaming, but no such method is used in the actual system.

In this work, among other architectural considerations, the authors identify and address discrete 3-D QoS requirements for their network platform. In particular, the network architecture of the proposed system allows the following novel approaches to 3-D media streaming: i) controlled streaming of 3-D scene media with RTSP, ii) an optimised transport protocol for 3-D scenes, called the DVR protocol, iii) buffering of scenes at the client before viewing. In all three such elements of the network module design, there is a discernible awareness and similarity to designing core elements of a network streaming application.

Olbrich and Pralle also recognised the need of reliably transmitting their proprietary binary representation of 3-D scenes with TCP. This design choice reveals two facts. First, it demonstrates that 3-D system designers were aware that data losses and out-of-order 3-D data delivery was a known network-imposed problem. Second, it indicates that due to this lack of explicit expression of QoS in 3-D graphics systems, data loss or out-of-order delivery could not be tolerated. Expressions of QoS in 3-D systems related to network problems are termed *Network QoS* in this thesis.

2.4.2 Defining QoS in 3-D

Given the previous discussion on expressions of QoS in the 3-D graphics world, one realises that it is not possible to have a strict definition of the term QoS in 3-D graphics. But, we can shape the term and derive the following summary:

QoS for 3-D graphics encompasses terminal processing heterogeneity, visualisation system resolution capabilities, and network-imposed constraints. Adaptivity refers to the ability of visually interacting with still or animated models regardless of rendering, resolution, and transmission capabilities of a networked 3-D graphics environment. A QoS-enabled 3-D streaming architecture should support, at least, multiple or hierarchical resolutions, rate adaptation, dejittering, and repair mechanisms, like conventional audio-visual media adaptive systems. And

at best, simplification methods, LOD management, view-dependent resolution, time-critical rendering, and interactivity. To this end, congestion management modules, the operating system, and the graphics application must continuously synergise in order to respect bounded end-to-end transmission latencies, and to achieve optimised visual quality within the networked 3-D environment's resource constraints.

The approach taken in defining QoS for 3-D in this thesis is to relax the boundaries between the graphics and network worlds, by defining two sets of QoS elements identified in their respective literature. 3-D QoS can then be perceived as providing care in modern 3-D applications for any combination of elements taken from these sets, which are defined in table 2.4 as follows.

Table 2.4: QoS element sets for 3-D applications.

Networks QoS set	
Low delay/latency	[BM03, Mar02]
Packet loss	[VOH01, VHO02]
Network available bitrate and rate variation	[JMT02] and [CN02]
Applic.-layer or hybrid protocols for 3-D model transmission	[RA03, RARM02]
Network scalability	[W3C03, Kri03, NCO03]

Graphics QoS set	
Client, or terminal, heterogeneity	[BM03, RLS+02, NRL+02]
Model resolution	[TLG99, FCOIZ01]
Model simplification and LOD management	[LGTW98, SP01, GTHL99]
3-D model error resilience through redundancy	[RA01, RA02]
Robust and selective broadcasting of 3-D geom.	[BK02a, BK02b, LKSS00, RARM02]
View-dependent transmission of 3-D model	[YKK01, KH03]
Time critical rendering and interactivity	[GB99] and [Kri03]

The contribution of this thesis in 3-D graphics QoS lies at the intersection of three main elements from the tables above: packet loss repair, redundancy control, and 3-D model resolution adaptation to network available bandwidth.

The reviewed literature in the following subsections, focuses only on a subset of issues in QoS-controlled 3-D mesh streaming. The classification applied to this literature reflects this fact, and some overlap may exist between the following categories. In particular, the material is organised into three main tracks:

- Research that has dealt with compressed or uncompressed geometry transmission either as a level-of-detail, or view-dependent encoding problem [TLG99, YKK01, SP01, JMT02, CN02, WL00].

- Work using network protocols not originally designed for streaming [RA03, Mar02, CN02]. Or, research dealing with robust and error-resilient broadcasting of 3-D geometry, focusing on bandwidth optimisation, in conjunction with, or in separation from compression [BK02a, BK02b, RARM02, LKSS00].

- Or, finally, by placing the work in some standardisation framework, or by investigating and optimising time constraints that enhance interactivity. The latter is a feature of potential commercial benefit, as a number of interactive services can be built over developing standards [GTHL99, FCOIZ01, COMF99, NCO03, W3C03, SMG+02].

The notable contributions of these works are critically analysed below. It has to be noted that the following subsections do not only review animated 3-D meshes. Since the vast majority of existing work in 3-D graphics adaptation is concerned with static meshes and virtual environments, some related efforts are included. Where the literature deals with any form of support for 3-D animation, this will be clearly stated.

2.4.3 QoS through view-dependent and progressive mesh coding

In [TLG99], To, Lau and Green, presented a method that allows progressive and selective transmission of multi-resolution models. Their development effort stemmed from the fact that other existing work on adaptive (or view-dependent) model transmission did not address support for progressive transmission and reconstruction. The major reason for this is that most other methods to date required large portions of the hierarchical data structures to be available at the client before rendering started, mainly due to neighbouring vertex or face dependency constraints. With their approach, To and colleagues reduce these dependencies and allow the visually important parts of an object to be transmitted to the client at higher priority than the less important parts.

This work represents a credible early effort to put 3-D data onto the network; however, its value is undermined by a lack of important capabilities: not only do they transmit uncompressed geometric models, they do not consider animated models either. Likewise, neither texture nor other attribute information are transmitted.

In a companion paper [LGTW98], Lau et al. used simplification streams to provide incremental representation of meshes for transmission. They used simplification streams of bounded

size as a cache of recently generated approximations for run-time simplification. Recording the last k contractions allowed their system to refine the current approximation by at most k steps. However, if further refinement is required, for example as a result of reduced congestion levels, simplification has to begin again from the original model, thus limiting the use of their system to applications requiring a small range of approximations.

Yang and Kuo [YKK01] considered a view-dependent progressive mesh coding scheme (VDPM) for graphics streaming. The proposed algorithm first splits the input mesh into several partitions, and compresses each partition progressively. A half-edge collapse multi-resolution method is used for progressive coding of the partitions. (In the same research, an interesting expression measuring visual quality is also expressed, which is further discussed in paragraph 2.5.3.4.) The algorithm also suggests that RTSP-based streaming control allows the client to provide clues to the server beforehand for the suitable viewing parameters during transmission. According to these client-negotiated viewing parameters, the server assigns appropriate resolutions to each partition. The server then reorganises and prioritises the transmission of topological and geometrical data that maximise the quality of the visible parts subject to the bandwidth constraint. To determine the visual quality (visibility) of vertices, a novel criterion based only on normal vectors is used.

Although computationally simple, the drawback of using this criterion is that it may mark hidden or shadowed vertices for coding as if they were visible. This case may occur in multi-object environments. If the visibility criterion were to be improved, transparent faces (and possibly reflective surfaces) should also be considered, since transparency may re-validate previously invisible vertices as candidates for coding. Another shortcoming in Yang's work is that the entire topological data needs to be transmitted with the base mesh, thus considerably increasing start-up latency. It is, however, a notable foresight that RTSP has been considered as the media control protocol for the first time in the graphics literature, if only to aid the set-up phase.

Chen and Nishita [CN02] recognised the need for QoS control in streaming 3-D meshes. They point out that QoS for 3-D data over the Internet should take into account the nature of Internet bandwidth as a resource that is unknown when streaming starts, and unstable for the duration of the mesh transmission. Their proposed algorithm suggested that the original mesh is decimated using some form of progressive coding, the resulting base mesh is transmitted as a *coarse patch*, and the time of coarse mesh transmission is recorded. In the event a client wishes to receive refinement information, this is transmitted as a *refinement patch*. Multiple refinement patches can be sent sequentially, until the client is satisfied. Continuous evaluation of

the network conditions is performed by re-calculating the average bandwidth usage by recording the transmission time of each patch. The algorithm places emphasis on this re-calculation of bandwidth from the last *experienced* time for a patch transmission, and the updated value is used to estimate the refinement level to be achieved in the subsequent patch.

The value of Chen and Nishita's scheme is undermined by the use of the HyperText Transfer Protocol (HTTP) as the basic transport protocol. Each patch request to the server suggests extra time is spent for the new connection set-up; server response must also be taken into account. An HTTP-based streaming system could become a bottleneck if it serves multiple other HTTP requests simultaneously. Using HTTP in order to achieve firewall traversal is unconvincing as an argument that Chen's solution is suitable for real-time 3-D streaming or dynamic mesh transmission. The work does, however, capture the principle of *adaptive* Internet multimedia applications in a rather synchronous manner. Section 3.4 proposes a more elegant approach to QoS preserving in 3-D mesh streaming, where a 3-D model is made adaptive to network conditions.

Kim and Ho [KH03], in recent work, have taken 3-D QoS a step closer to the analysis of the term provided in the previous subsection, by combining view-dependent representations with hierarchical coding. According to their approach, the server-stored *hierarchical* 3-D mesh consists of sequentially and progressively coded submeshes that can be stitched together, or split, to achieve a desirable resolution for transmission. For the transmitted resolution, the scheme can take into account viewpoint information of a client. The merging and splitting operations can achieve reasonable viewpoint and bandwidth adaptivity. However, animated meshes are not considered, nor are the 3-D submeshes compressed.

Guéziec et al. [GTHL99] proposed a complete framework that allowed effective visualisation of 3-D models over the Web. Their efforts concentrated on coupling together existing state-of-the-art work on static 3-D mesh compression [Dee95, Hop96, TGHL98, GS98] with a process that generates multiple LODs through edge-collapse operations [RR96, XV96, Hop97], and a novel and flexible LOD storage scheme called a *progressive multi-level mesh*, for streaming geometry in VRML worlds. The main motivation of their work had been to eliminate the need to perform complete downloads of geometric models before starting to display them on a local terminal. The proposed framework highlights the benefits of using the novel LOD storage scheme as its low memory footprint only requires a 10% overhead to the full detail mesh. The LODs, however, need to be pre-determined at encoding time, thus decreasing the framework's flexibility for real-time streaming.

The work shows strong focus on integration of the compression, LOD generation, and

LOD storage processes, while it quietly disregards QoS issues related to network bandwidth limitations, loss and delay. From the demonstrated design of the `IndexedFaceSet` node extensions, it is hard to realise how the mesh quality will be preserved and enhanced in a hostile Internet environment when employing this complete framework. Even at low packet loss rates, and particularly when they occur as bursts, losing a fraction of the progressive multi-level mesh structure may totally invalidate a whole LOD level and force it to be discarded. Furthermore, the proposed VRML node extension constructs cannot accommodate time-dependent geometries since the framework is strictly coupled to static mesh compression schemes.

Shamir and Pascucci's [SP01] recent work on separating spatial and temporal LODs for dynamic meshes could potentially provide insights into how to extend Guéziec's VRML geometry streaming framework and pave the way towards supporting wireframe animations in a networked visualisation environment. The authors make use of the Temporal Directed Acyclic Graph (T-DAG) data structure, introduced by the same team [SPB00], to facilitate storage of dynamic multi-resolution 3-D mesh geometric and attribute data. This use of T-DAG was presented in this thesis in subsection 2.3.1. Shamir and Pascucci adapted the use of T-DAG in such a way that, instead of encoding the original series of time dependent meshes in the data structure, they first extracted the low frequency deformations of the meshes, and encoded only the resulting residual meshes as refinements to the originals. The low frequency information, expressed as global affine transformations on the original meshes, captures the most visually significant temporal displacements using a coarse and inexpensive approximation. The high frequency information adaptively refines the coarse meshes to achieve higher resolutions and greater details in both time and space dimensions.

The T-DAG, as an underlying data structure for dynamic multi-resolution 3-D meshes, can potentially provide the flexibility to enable a packet-based networked visualisation system to cope with latencies. It does not, however, address dynamic mesh compression, resulting in huge demands on storage space. Furthermore, the data structure is inherently too complex to be suited to real-time streaming. In their experiments, Shamir and Pascucci report T-DAG update times of 30 sec to merge a new mesh into the data structure with medium-end computers, and storage demands of twice the size of the binary uncompressed original mesh representation. Finally, the effect of packet loss on the quality of the dynamic mesh was not studied. In the event of missing affine maps, as defined in T-DAG, even the low frequency components may not be rendered, resulting in a poorer than coarse quality animation.

Noimark and Cohen-Or [NCO03] recently proposed a method for streaming remote walk-through scenes to MPEG-4 enabled thin clients. The method takes advantage of a thin client's

embedded ability to decode MPEG-4 video sequences with the help of a video decoding chip. This embedded hardware acceleration is exploited by a server-based system that streams computer-generated video rendered from an interactive remote walkthrough, thus releasing the post-processing burden from the client. The server renders the 3-D model sequence and encodes the frames into a video stream, which has been optimised with a dynamic I-frame encoding pattern, derived in real-time by monitoring network conditions and scene content. The system adapts to higher frequency of I-frame encoding in cases with increased frame losses, or scene cuts that require walkthrough background refresh. The system also suggests the use of 3-D object-based frame layering (similar to video segmentation techniques), which comes at low or no cost for computer-generated 3-D models, and proposes a per-layer quantisation scheme, which is expected though to cause cracks and other artifacts. These artifacts are overcome with gradual change of the quantisation values over time between layers, so that the edge transitions are not abrupt and noticeable.

The dynamic I-frame adaptation scheme of the proposed system is an elegant approach to providing QoS in a streamed 3-D walkthrough environment. However, in Noimark's literature, the effects of frame loss to the delivered 3-D mesh quality have not been investigated. Chapter 4 studies the impact of frame rate reduction due to packet loss on the perceived animation quality in 3-D mesh based wireframe animation streaming scenarios. In addition, the proposed per-layer quantisation method in [NCO03] can lead to noticeable artifacts should Noimark's system be extended with rate adaptation for network adaptive streaming of the server-based video walkthrough. The alternative, decimation-based 3-D wireframe coding method described in section 3.4 represents a more suitable candidate for rate-adaptive layered 3-D systems.

2.4.4 QoS through bandwidth heterogeneity and error resilience

Joslin and Magnenat-Thalmann [JMT02] tackled a related subject in the context of MPEG-4, that of preserving bandwidth resources while transmitting face and body animation. Being aware of the vast bandwidth consumption that 3-D animation implies if many elementary animation streams are involved in a scene, they designed a filtering method based on statistical analysis that *removes* ineffective information from the data streams, as opposed to conventional approaches that *reduce* the resolution of animation data based on encoding and quantisation methods.

Although novel, this approach suffers from a number of limitations. First, it only addresses MPEG-4 animation parameters, namely face and body animation. Second, the statistical analysis upon which they evaluate the success of their proposed scheme is performed on results

collected from human observers. It is unclear how this method can be applied effectively in transmitting interactive graphics. Third, their method uses the distance between the observer and the animated model as a parameter that determines where the transition from data removal to data reduction should occur or vice versa. Driving an interactive networked graphics system with such a subjective threshold would not guarantee consistently stable performance for a variety of model resolutions and users (or observers). Worse, in scenes where multiple models are represented simultaneously at different resolutions it is hard to set the transition threshold, and a higher resolution model may drive the animation data reduction, wasting bandwidth unnecessarily.

Martin [Mar02] recognised the need of being able to stream 3-D graphics digital content in different resolutions and described ARTE, an adaptive environment for the transmission of 3-D models to clients with different graphics capabilities and connection types. ARTE's success is built around two of its key components: *environment monitoring* and *transcoding*. The monitoring tool employed by the system quantifies information about the rendering performance and processing load of the client and server, as well as the performance of the network link connecting them, which is purely evaluated upon delay (round-trip latencies) and bandwidth (achieved frame rate) measurements. The transcoding engine supports either on-the-fly or stored processing of models. Transcoding is done according to predefined performance parameters and an estimation of the perceptual importance of a model's component over its final rendering.

Although ARTE exhibits good adaptation to different clients over other level-of-detail based algorithms, it does not consider animated models. Furthermore, its communications architecture does not address QoS as a function of packet loss on the Internet, making it unsuitable to stream any sort of model in the wide area where discernible packet loss rates occur. The system transmits packets unreliably over UDP and, in the case of a single packet loss, it would discard a whole application frame that could potentially contain a significant amount of model data.

Bischoff and Kobbelt [BK02a] devised a robust geometry broadcasting scheme with respect to packet loss. They recognise that robust transmission schemes should be able to handle data loss, duplication and misordering. Hence they propose a novel progressive mesh coding algorithm which, during a pre-processing step, decomposes a given polygonal mesh into a set of overlapping ellipsoids that cover the mesh's interior and that represent the coarse shape of the model, and an independent stream of sample points that constitute the refinement details. Due to the independence of these two sets of geometric data, partial data loss can be tolerated on the refinement stream, which eventually will only lead to gradual and controlled degradation of the

original mesh. Explicit connectivity information is deliberately not attached to the ellipsoids nor to the sample points in order to achieve increased robustness to packet loss.

Such a design leads to reconstructed 3-D models where the connectivity is not the same as that of the original model. This makes the encoding scheme potentially inefficient for coding 3-D animation as the latter's temporal redundancy is not explicitly exploited. Besides, the heavy pre-processing step and lack of compression shows no consideration of real-time transmission requirements. The published work only visually evaluates the efficacy of the proposed solution, without particularly studying the dynamics of a networked environment, where packet loss and delays due to congestion are common. Section 3.4 of this thesis provides such insights.

In other work, Bischoff and Kobbelt [BK02b] presented algorithms for robust transmission of geometric data sets. In this, the authors assert that one of the main problems that needs to be addressed when transmitting 3-D geometry over the Internet is packet loss. They suggest that in order to achieve robust transmission of 3-D models, one has the option to exploit the geometric coherence of the data instead of providing redundant information with a lower layer protocol. Following this principle, they construct a coarse base mesh with methods such as the one described in the above paragraphs, and attempt to reconstruct the original mesh with increased resolution in the receiver, by progressively transmitting a triangulation of *point clouds* (i.e., refinement vertices) that has a certain quality property. To reduce the complexity of the surface reconstruction from a point cloud, they take advantage of knowledge of the model's topology prior to transmitting the independent refinement vertices. They rely, though, on redundant retransmissions of the base mesh through some reliable transmission protocol, and their scheme caters for reducing the reconstruction latencies if the receiver misses a prefix of the data stream, by simply sending the base mesh more frequently than the other parts of the geometric data.

The same approach is adopted in related work by the same research group using progressively transmitted *subdivision surfaces*. Labsik et al. in [LKSS00] propose algorithms for the decomposition and reconstruction of surfaces with subdivision connectivity, and use these to introduce a new progressive 3-D model representation that allows random access to the detail information. Any given mesh with arbitrary connectivity can be converted to a mesh with subdivision connectivity within some error tolerance. Meshes with subdivision connectivity can be generated by many different algorithms where a uniform splitting operator can be used to refine the initial control mesh. With this splitting operator, a single triangle can be divided into four by inserting a new vertex per edge. Although Labsik's work is strongly related to the wavelet-based hierarchical decompositions introduced by Lounsbery et al. [LDW97], the multiresolution analysis used for the above subdivision-based progressive transmission scheme is not

explicitly based on wavelet functions. (Wavelet-based coding of subdivision surfaces in MPEG-4 is described by Morán et al. [MGS+02].)

The novelty of the subdivision-based scheme is that it achieves smooth approximations of original surfaces from a rather small amount of received data. In regions where no detail information is available, vertex positions can be predicted using the subdivision operator. However, the mutual dependencies generated during the construction of the sequence of detail coefficients, suggest that the ordering of the detail coefficients is very critical to the reconstruction process. Thus, such schemes are very sensitive to network latencies, packet losses, and reordering, which commonly occur in the Internet.

Al-Regib et al. [RARM02] recently studied the subject of streaming compressed 3-D animation over lossy channels. They proposed a joint source-channel bit-budget optimised protocol that first transmits a crude representation of the 3-D mesh, accompanied by its texture and immediate time evolution information (animation data). This first-round data is endowed with significant error protection, in the form of a forward error correction code (FEC), to protect the vital initial information from packet loss. Then, the protocol transmits a series of upgrades that refine the accuracy of the mesh and the animation. They have designed the upgrade chunks to contain either *selections* of vertex subsets, or *adjustments* to vertex positions, motion vectors, and texture coordinates, or connectivity *refinements*. The proposed scheme caters for the resilience of the second-round data chunks against transmission errors at a lower level than it caters for the initial data. The protection level is chosen to be proportional to the impact of the upgrade data on animation quality.

The scheme shows thorough consideration of compression issues by categorising and transmitting chunks of data covering both geometric and photometric attributes of a mesh, along with their time evolutions for animation. The work also places dynamic 3-D model streaming within a realistic Internet scenario where burst packet losses occur. However, the experimental evaluation of the protocol solely uses the Hausdorff distance as quality metric. Being a measure of geometric distance only, such a metric partially highlights the efficiency of the proposed protocol, and ignores the photometric data transmitted. As this data may represent a considerable portion of the available bit budget, the evaluation of the protocol at medium and high packet loss rates may be slightly skewed. At these loss rates, their results show no significant effect of the optimisation operation, which is attributed to loss of vital connectivity and geometry data of the coarse mesh in the first place.

A potentially more versatile bit budget optimiser in this case, not solely relying on a distance metric, would allocate most bandwidth resources to geometry and exclude texture data in

a first phase. The idea of the visual smoothness metric presented in section 5.5 is a good such candidate. In addition, the compressed progressive mesh [PR00] algorithm used as the source encoder and the iterative post-processing optimisation steps provide some sort of quantisation-level rate control which may not allow fast adaptation to the available bandwidth. In section 3.4 a more efficient and fully rate adaptive scheme based on multi-resolution source coder and receiver-based predictive geometry coding is proposed.

Finally, in related work Al-Regib et al. [RA03] describe 3TP, an application-layer protocol that uses a hybrid approach to download 3-D models to users signing into a multi-user game. According to the hybrid TCP/UDP design, a subset of packets is transmitted using TCP in order to meet a distortion constraint, while the remaining packets are transmitted using UDP to minimise the end-to-end delay.

Although this combined scheme has been selected to respect the distortion-delay trade-off, it may not provide a viable solution in streaming continuous animation frames; in such scenarios, a server would need receiver feedback to monitor the streaming process and adapt to variable network conditions. There, the interaction between the reliable part (TCP) and real-timeness is unspecified, if not inefficient. 3TP, however, is only a fragment of a wider design approach also involving players' action and state information streaming, as well as a multicast-based protocol to accommodate large number of participants, and should only be considered as such.

2.4.5 QoS through standards frameworks and interactivity

In other network-aware research about 3-D mesh transmission, some less QoS-controlled solutions have been proposed. Although they are based on non-streaming provisioned protocols such as HTTP, their value cannot be underestimated as they conform to a different networked multimedia model - Extensible 3-D (X3D) - supported by the international body of Web3D. Its model is purely based on HTTP and the Web to deliver high-quality 3-D content quickly and reliably.

Earlier in this section, Chen's and Nishita's [CN02] work was highlighted as an early contributor in identifying QoS needs for streaming 3-D meshes. Their work has been critically reviewed for employing HTTP, following the X3D streaming pipeline.

A similar approach in the context of X3D was proposed by Fogel et al. [FCOIZ01]. This work focuses on engineering the X3D model to deliver 3-D content asynchronously, according to user-specified 'best' levels-of-detail. It places emphasis on appropriately adapting a proprietary progressive mesh coding method developed by the same research team [COLR99], which

is based on triangle patch colouring, without paying particular attention to Internet QoS. The HTTP protocol is adopted, thus it suffers from the same drawbacks discussed earlier while reviewing Chen's work. The authors do hint, however, at the need for optimisation in visual distortion while streaming 3-D content.

Where interactive browsing is the key requirement, Wang and Li [WL00] developed `OctMesh`, an elegant large-scale 3-D mesh browsing scheme that only streams localised viewable portions of a large mesh at desired resolutions, thus also achieving bandwidth savings. `OctMesh` achieves this by a mechanism that clusters the vertices of the original mesh and builds a hierarchical space-partitioning oct-tree. Based on this structure, Wang and Li devised a compressed bitstream that can be accessed interactively by a browser at random entry points. Network aspects were not addressed in `OctMesh`.

In the work discussed previously in this section, the motivation was to be able to transmit 3-D meshes reliably, at high quality, and in a way that facilitates mesh browsing and user interactivity. Nonetheless, all this literature only addressed static meshes, leaving out time-dependent ones. Cohen-Or et al. [COMF99] came to terms with this problem by proposing a framework suitable for streaming texture intensive animation. They recognised the existence of strong temporal coherency between successive frames of such geometry and texture synthetic animations. Their solution, although aware of network bandwidth constraints occurring in the Internet, only covers a subset of potential applications, namely those with ample synthetic texture information. No reference to network-based incentives is given in this work either.

For visualisations sensitive to interactivity, Gobbetti and Bouvier [GB99] have formalised the issue of scene rendering as an optimisation problem, and proposed a solution for time-critical transmission of graphics objects making up a 3-D scene. The problem is that of selecting appropriate resolutions for a set of visible objects that maximise the visual quality within predefined timing constraints. The proposed solution assumes support of multi-resolution coding for the 3-D objects. It achieves smooth LOD control and guarantees a uniform, bounded frame rate even for widely changing viewing conditions. The solution is shaped around two heuristics. A cost heuristic, estimating the time required to render a scene containing the visible objects at a certain resolution for each object. A benefit heuristic, estimating the quality of the rendered image containing the set of objects at the optimised resolutions. Although the proposed optimisation method is independent of the data structure used to represent the multi-resolution representation of the objects, the authors also propose Totally Ordered Mesh (TOM), a mesh packing data structure that facilitates mesh traversal at an arbitrary desired resolution within short predictable time.

The MPEG-4 Animation Framework Extensions (AFX) is a proponent of subdivision surfaces for coding and transmission of 3-D surfaces. Morán et al. [MGS+02] describe the framework's `WaveletSubdivisionSurface` node supporting wavelet hierarchical coding of 3-D details in a SPIHT-like embedded bitstream. The AFX framework through this node achieves three kinds of adaptive mesh refinements: view-dependency, sub-band decomposition, and bitstream embedding.

Within the context of AFX in MPEG-4, Salomie et al. proposed MeshGrid [SMG+02], a compact, multi-scalable, animation-friendly, surface representation method. MeshGrid describes the global vertex connectivity of a 3-D object's surface with the help of a regular 3-D reference grid of points, and exploits further this regularity of the grid with wavelet decompositions, in order to achieve compact lossless mesh representations of up to 4 bits/vertex. This kind of representation lies between the classic approaches for compact surface encoding techniques: vertex and connectivity *IndexFaceSet-based* encodings, and alternative *surface representation* methods. It maintains features common to the first approach, but exploits wavelet multi-resolution analysis techniques, typical of the second category, to refine a 3-D shape. Using the latter techniques, MeshGrid achieves three simultaneous scalability modes: i) mesh resolution scalability (i.e., adapting the number of transmitted vertices), ii) global shape precision scalability (i.e., adaptive reconstruction of the reference grid positions), and, iii) local vertex position scalability (i.e., increasing the precision of the known vertex positions in relation to the reference grid). It also permits topological changes between successive resolutions of the mesh.

The MeshGrid approach has been recently integrated to the AFX framework of MPEG-4, and represents an expression of the state of the art in multi-resolution based, view-dependent 3-D object coding with animation capabilities. A unique feature of the MeshGrid technology is its ability to express various surface representations, or 3-D meshes (triangular, quadrilateral, etc.); however, arbitrary IndexFaceSet models need to be re-triangulated in order to be brought to the MeshGrid format. This is imposing an extra pre-processing step to the large existing base of triangulated 3-D models that are potentially currently being used in other vertex-based animations.

In MeshGrid, as in the whole AFX framework, however, the exploitation of compact mesh representations for efficient, adaptive (or QoS-based) transmission over the Internet has not been defined. Therefore, animation streaming systems will require customised application-level protocols to achieve network QoS for 3-D graphics in the context expressed earlier in this section. Ideally, these protocols would interact with congestion management modules in the

network or application layer. Such an approach is taken with the end-to-end 3-D streaming architecture described in this thesis, with details on the interaction with the network protocol described in section 3.5.

Finally, Scalable Vector Graphics (SVG) [W3C03] is a newly finalised standard by the World Wide Web consortium, based on XML grammar, for defining vector-based cross-platform graphics over the Web. Although it covers 2-D only graphics, its inherent design scalability due to the textual XML-based description of the graphics content, makes this approach a candidate for high-quality uncompressed 2-D graphical content *dissemination* to a large number of heterogeneous clients. Of particular interest is SVG's support for applying transformations to the positional matrices of the graphics objects at runtime, thus enabling limited animation capabilities through coordinate translation, rotation, and scaling. The standard does not specify any form of network adaptation since it does not deal with continuous media. But, through its various 'profile' specifications, it can be seen as an immediately available technology for scriptable, cross-platform, 2-D animation, dealing with the variety of target devices with different processing and rendering capabilities.

2.4.6 Summary of QoS-based 3-D mesh transmission

Table 2.5 summarises the reviewed methods for QoS-based transmission of 3-D meshes.

Table 2.5: Summary of QoS-aware 3-D mesh transmission.

CATEGORY	METHOD NAME	
VIEW-DEPENDENT & PROGRESSIVE	To et al.	[TLG99]
	Lau et al.	[LGTW98]
	Yang & Kuo	[YKK01]
	Chen & Nishita	[CN02]
	Guéziec et al.	[GTHL99]
	Shamir and Pascucci	[SP01]
B/W AWARE & ERROR RESILIENT	Joslin & Thalmann	[JMT02]
	Martin	[Mar02]
	Bischoff & Kobbelt	[BK02b, BK02a]
	Labsik et al.	[LKSS00]
	Al-Regib et al.	[RARM02]
STANDARDS BASED	Fogel et al.	[FCOIZ01]
	Wang & Li	[WL00]

2.5 Error Measurement and Quality Evaluation

Polygonal 3-D models are growing steadily in detail and complexity for most modern applications. This growth is mainly reflected in their correspondingly increased file sizes. Because encoding techniques, and simplification and compression algorithms, strive to preserve credible model appearance, a measure of visual quality is necessary. Ultimately, visual quality can only be assessed by human observers. However, interactive demands imposed by the majority of modern applications requiring control of quality of 3-D models do not allow continuous experimentation. Furthermore, subjective quality assessment, although effective, is costly and resource consuming.

Watson et al. [WFM01] used the term *visual fidelity* to express that visual quality is a process with a wide range of meanings, such as similarity of a model to the original, or successful perception and communication of the original concept in a modelled environment. They distinguished between *experimental* and *automatic* measures of fidelity of 3-D graphics models. The first category refers to those measures that involve human participants assessing experimental stimuli through well-defined scientific processes in order to obtain their relative judgements. The second category refers to all other measures that are generated purely automatically.

Clearly, the wider notion of fidelity as expressed by Watson et al. may extend to 3-D animation. A quality measure of 3-D animation should be able to express the notion of fidelity most relevant to this type of application, that is smooth vertex trajectories and smooth animated surfaces, without cracks or bumps. In this thesis, fidelity, visual quality, error, and distortion, are terms that will be used to refer to the same process: measurement of the deviation of a model, or model sequence, from another model or sequence, that is referred to as the *original.* When deviation is defined on a basis that expresses a purely physical quantity (distance, volume, etc.), the measure is automatic. When deviation refers to more subtle properties that the human observer appreciates more, the measure is experimental, following Watson's terms. Furthermore, the function, algorithm, or other method upon which similarity is measured, will be called a *metric* throughout this thesis.

This section clarifies the distinction between two large categories of distortion measurement, geo-metrics and photo-metrics, in the wider area of 3-D mesh processing. Following this distinction, a review of some of the most common metrics, encountered in particular in the literature of 3-D mesh coding and animation areas, is given.

2.5.1 Geo-metrics

Model simplification researchers have long used three-dimensional measures of distance [SZL92, RR96, Ros97, KLS96], curvature [Red96], volume [LT98, Gué95, ALSS99], or their combinations [CRS98] to perform various mesh processing operations since one typically does not know what part of the model a user may be observing. The term *geo*-metrics is introduced to encompass all automatic measures of distortion that perform calculations of distance, curvature, or volume, on 3-D surfaces and models.

Ronfard and Rossignac [RR96, Ros97] used an error bound metric that calculates distances to supporting planes. Their error estimator measured the displacement of vertices in the direction orthogonal to their incident faces. Their work defined two energy functions: *local tessellation error*, and *local geometric error*. The local geometric error computes the maximum distance from the new location P of a vertex to all the supporting planes of the vertices that have collapsed into P. This metric works well for flat, or nearly flat, surfaces. Similarly, Schroeder [SZL92] in his mesh decimation method had used a vertex-to-plane distance as decimation criterion.

Klein et al. [KLS96] used an error metric based on the Hausdorff distance. This work differs from the previous in the fact that the error metric used measures distance between the reduced and the original meshes, rather than between Rossignac's reduced mesh and a plane or Schroeder's average surface.

Reddy [Red96] used an 'error' function based on curvature to guide a mesh simplification process. This follows previous remarks that human vision acuity varies with spatial frequency. In his simplification algorithm, he limited the spatial frequencies that could be impacted due to simplification operations.

Lindstrom [LT98] further elaborated on simplification using area- and volume-based calculations that relate to expressions of local error. Guéziec [Gué95] used a tolerance volume as an error bound measure for mesh simplification. His tolerances and errors are defined with respect to the simplified surface, as opposed to the original surface. Alliez et el. [ALSS99], not initially driven by compression or simplification oriented motives, contributed a mesh approximation method that minimises the volume between two meshes. Clearly targeting mesh complexity and visually attractive rendering, they allow vertices to move while minimising the geometric deviation. They later extended their technique to address geometric compression, leading to multi-resolution meshes with minimal visual losses.

Cignoni et al. [CRS98] created the `Metro` tool that integrates the calculation of distance and volume distortion between two 3-D models. In a companion survey [CMS98] on public

domain simplification codes, the same authors discussed empirical results they obtained by using `Metro` and presented a thorough classification of error evaluation criteria. `Metro` achieves an approximation of error based on surface sampling (integration process over a surface) and the computation of point-to-surface distances. Aspert et al. [ASCE02] built another tool called `Mesh` that simplifies the calculation of the Hausdorff distance between 3-D models. It follows a similar approximation principle to `Metro`, by sampling the original mesh surface, and calculating the Hausdorff distance in a more speed- and memory-efficient manner than the distance's full algebraic expression. Both tools return numerical and visual evaluations and histogram reports of error distributions.

All the above literature introduced measures of geometric error to simplify geometry and ignored distortion due to surface or appearance attributes (such as colours, textures, and normals). There are significant works reported where metrics cater for more than the pure geometric features of a mesh. For example, Cohen et al. [CVM+96] did not directly use any error measure, but devised a geometric construct, called simplification envelopes, to minimise surface deviation. Garland [GH97] measured vertex-to-plane distances, called Quadric Error Metrics, which were also suitably extended for meshes with attributes [GH98]. Hoppe [Hop96] initially introduced the energy functions, used to preserve surface geometry, scalar attributes and discontinuity curves. Later [Hop99, HM00], he improved upon Garland's Quadric Error Metric for attributes [GH98].

2.5.2 Photo-metrics

As mentioned in the previous subsection, many existing techniques for still or dynamic 3-D model coding, compression, or simplification, rely on some measure of geometric distortion between two surfaces, models, or sequences of models, with the same or different number of polygons, vertices, or edges between them. For some graphics applications, however, these surfaces or models are ultimately used to produce raster images. If one is interested in the visual quality of a model, then a metric based on properties of comparative photometry of the two models should be considered. Metrics that account for the photorealism in computer generated images, are termed *photo*-metrics.

In the field of 2-D image and video processing, research mainly focused on automatic fidelity measures that compare the quality between images or sequences of video frames. The mean squared error (MSE), for example, simply calculates the mean of the squared pixel-by-pixel differences between the original and approximate images. More complex error measures have been devised based on numerical models of the human visual system (HVS)

[Dal93, BM98, Lub95].

Lindstrom [LT00], in a photo-metric approach, measured fidelity of simplified models using MSE by taking virtual snapshots of the model before and after the simplification process. This approach had been proposed in a similar way by Garland in his thesis [Gar99b], but no experimental evidence had been given. Lindstrom suggested that evaluation of mesh quality could be possibly done using the root mean square metric over a collection of images from different viewpoints. Roy et al. [RFT02] built a tool that assesses mesh simplification quality, based on appearance attributes, regardless of viewpoint. In their work, they formally defined *attribute deviation* as the local difference of the attributes between the original and a simplifies mesh. (This definition is in equivalence to geometric deviation in geo-metrics.) They also suggested a fast algorithm to sample the surface of a triangular mesh, which increases assessment resolution and statistic precision.

Other very common distortion metrics on the 2-D plane, based on the MSE, include the signal-to-noise ratio (SNR), and peak signal-to-noise ratio (PSNR). By equivalence to these common 2-D error metrics, some corresponding 3-D metrics have been expressed. Their expression either directly considers the error on the 3^{rd} dimension [CKL+00], thus only numerically expanding their 2-D equivalents, or it is more carefully tailored to the animation signal they evaluate [GSK02, VOH01].

The following subsection 2.5.3 discusses the above automatic measures of fidelity, and in particular their expressions adapted to 3-D animation.

2.5.3 Automatic error metrics for mesh animation coding evaluation

In order to measure the visual loss resulting from a non-perfect reconstruction of the animated mesh at the receiver, a metric is required that is able to capture the visual fidelity of a decoded mesh $\hat{M}$ to its original equivalent M during animation. Any differences can be expressed either in a quantitative manner, by providing some deterministic measurement of geometric error, or in a way that the human visual system appreciates that is subjective and possibly non-deterministic. Only calculating the geometric deviation between two 3-D models is better suited to still models or frame-by-frame model comparisons in an animation sequence and we can refer to such evaluation methods as spatial methods.

As animated wireframes introduce the time dimension, spatial methods alone may not suffice to provide adequate measure of distortion. In the developing area of 3-D animation coding, there is a trend to design error metrics in such a way so that they capture artifacts of a more subjective nature, such as mesh surface smoothness. The remainder of this section reviews

some known spatial metrics commonly used in 3-D mesh compression and further discusses how these can be extended to evaluate animation distortion due to compression. These metrics lay the foundation for the design of the visual smoothness metric, proposed in section 5.5, that quantifies and considers surface smoothness in an attempt to correlate geometric error to geometry distortion as perceived by the human eye.

2.5.3.1 Euclidean distance

The simplest measure is the RMS geometric distance, or Euclidean Distance (ED), between corresponding vertices of the original and decoded models. The ED between two vertices v_i and v_j is denoted by the $\|\cdot\|_2$ norm, or l_{ij}, and defined as in equation (2.8).

$$l_{ij} = \|v_i - v_j\| = \sum_{k=x,y,z} (v_{ik} - v_{jk})^2 \ . \tag{2.8}$$

2.5.3.2 Mean geometric error — Metro

The approximation error between two meshes may be defined as the distance between corresponding sections of the meshes. Given a point p and a surface S, the distance $e(p, S)$ can be defined as:

$$e(p, S) = \min_{p' \in S} d(p, p') \ , \tag{2.9}$$

where $d()$ is the ED between two points in E^3. The one-sided distance between two surfaces S_1, S_2 is then defined as:

$$E(S_1, S_2) = \max_{p \in S_1} e(p, S_2) \ . \tag{2.10}$$

Given a set of uniformly sampled distances, we denote the mean distance E_m between two surfaces as the surface integral of the distance divided by the area of S_1:

$$E_m(S_1, S_2) = \frac{1}{|S_1|} \int_{S_1} e(p, S_2) \ ds \ . \tag{2.11}$$

The latter equation (2.11) is the *mean geometric error*, the quantity evaluated by the `Metro` tool [CRS98] when calculating the distance between 3-D meshes.

2.5.3.3 Hausdorff distance

The Hausdorff distance has been commonly used in the literature [ASCE02, Cru03, RA02] as an error metric. It is defined as the maximum minimum distance between the vertices of two sets, M_t and $\hat{M}_t$, so that every point in M_t lies within the distance $h(M_t, \hat{M}_t)$ of every point in $\hat{M}_t$. This can be expressed as:

$$h(M_t, \hat{M}_t) = \max_{m_t \in M_t} \min_{\hat{m}_t \in \hat{M}_t} \|m_t - \hat{m}_t\| \ , \tag{2.12}$$

where $\|\cdot\|$ denotes the Euclidean distance between the two vertices, m_t and $\hat{m}_t$ [CKL+00].

It is important to note that this distance, as the one-sided distance of equation (2.10) between two surfaces, is in general not symmetrical, i.e.:

$$h(M_t, \hat{M}_t) \neq h(\hat{M}_t, M_t) \ . \tag{2.13}$$

We refer to $h(M_t, \hat{M}_t)$ as the *forward distance*, and to $h(\hat{M}_t, M_t)$ as the *backward distance*. It is then convenient to introduce the *symmetrical* Hausdorff distance $H(M_t, \hat{M}_t)$, defined as follows:

$$H(M_t, \hat{M}_t) = max\{\ h(M_t, \hat{M}_t),\ h(\hat{M}_t, M_t)\ \} \ . \tag{2.14}$$

The symmetrical distance provides a more accurate measurement of the error between two surfaces, since the computation of a "onesided" error can lead to significantly underestimated distance values. When $H(M_t, \hat{M}_t) = 0$ the two sets are identical and the implication is that a lossless compression scheme has been used. However, a small value of $H(M_t, \hat{M}_t)$ does not mean that the models, M_t and $\hat{M}_t$, are similar. In fact, their shapes could differ significantly.

Calculating the Hausdorff distance may take a considerable amount of time for large models because the time complexity of the algorithm is $O(n^2)$. Recently, however, Aspert et al. [ASCE02, Cru03] suggested a more efficient and less memory intensive calculation, demonstrated with their `Mesh` tool. Their algorithm achieves a close approximation to the Hausdorff distance between two 3-D models by selectively reducing the number of point-triangle distances that the original algorithm needs to calculate between the original model (points) and the reconstructed one (triangles). This reduction can be achieved using a uniform point sampling grid as used in the `Metro` tool [CRS98].

2.5.3.4 SNHC model distance

For two isomorphic models, the SNHC distance metric, D_{SNHC}, as defined in [ISO98], is an approximation of the volume spanned by the surface of the original model while it is continuously deformed into the decoded model, or vice versa.

Analytically, this metric can be defined as the sum of two asymmetric terms. The first term is the average of the distances (D) from a number of sample points (N) chosen on the surface of the original model (M_t) according to the uniform distribution (constant number of sample points per surface area) to the surface of the distorted model ($\hat{M}_t$). The second term is computed as the first one, with the same number of samples, but with the two models switched. The result is a symmetric measure, D_{SNHC}, which can be expressed as follows:

$$D_{\mathrm{SNHC}} = \overline{D^N_{M_t, \hat{M}_t}} + \overline{D^N_{\hat{M}_t, M_t}} \ . \tag{2.15}$$

Computation of the SNHC model distance is simple and straightforward once the N sample points have been selected. Obviously, its complexity is bounded by $O(n)$ if $N \leq n$, otherwise it is in the order of $O(N)$.

2.5.3.5 3-D PSNR

While adapting the peak Signal-To-Noise Ratio (PSNR) from its 2-D expression (i.e., when used to measure natural video) to 3-D animation streams seems to be a straightforward operation, in reality one needs to be able to interpret the adaptation of this metric in the 3-D domain. For 3-D animation, being a time-based signal, PSNR does not simply relate to the quality of the still picture as a spatial metric would do in natural video. Rather, it provides a quantitative expression of animation smoothness or jerkiness in time, thus representing a temporal metric.

As shown in equation (2.16), frame PSNR is calculated based on the peak Mean Square Error (PMSE) per vertex and per node. The expression captures animated geometry samples that have been predictively encoded (in what is the equivalent to a P-frame in natural video) by considering only those vertices and nodes that have changed since the previous frame. The notation is explained below.

$$\begin{aligned} \mathrm{PMSE} &= \frac{1}{R^2} \cdot \sum_{n=1}^{N} \frac{1}{N_n} \sum_{v=1}^{P} \frac{1}{P_v} \cdot \frac{1}{3} l_{s(n,v),\hat{s}'(n,v)} \ , \\ \mathrm{PSNR} &= -10 \log_{10} \mathrm{PMSE} \ . \end{aligned} \tag{2.16}$$

Symbol	Meaning
$s(n,v)$	original sample of vertex v in node n
$\hat{s}'(n,v)$	decoded sample of $s(n,v)$
R	maximum displacement of $s(n,v)$
N_n	number of changed nodes in a D-frame
P_v	number of changed vertices in a node of a D-frame

2.5.3.6 An alternative SNR

An alternative error metric in 3-D wireframe animations, provided by Gupta et al. [GSK02], uses the Signal-To-Noise ratio (SNR). According to the formulation shown in equation (2.17), the signal is defined as the variance of the absolute error in positional matrices between two consecutive frames (let ΔM be this quantity). The error term is defined as the difference between a reconstructed positional matrix and its corresponding original (let ΔE be this quantity).

Then, the SNR ratio can be expressed as:

$$\begin{aligned} \sigma^2 &= \frac{1}{num}\sum_{j=1}^{num}\sqrt{\sum_{i=x,y,z}((p_{j,k}-p_{j,k-1})_i-\mu_i)^2}\ , \\ \sigma_e^2 &= \frac{1}{num}\sum_{j=1}^{num}\sqrt{\sum_{i=x,y,z}((p_{j,k}-\widetilde{p_{j,k}})_i-\mu_{ei})^2}\ , \\ SNR &= 10\log_{10}\frac{\sigma^2}{\sigma_e^2}\ , \end{aligned} \tag{2.17}$$

where the notation is:

Symbol	Meaning
num	total number of vertices
$p_{j,k}$	the j^{th} vertex of the k^{th} frame
μ_i	mean value of the frame differcnce signal $(p_{j,k}-p_{j,k-1})_i$
μ_{ei}	mean value of the estimation error $(p_{j,k}-\widetilde{p_{j,k}})_i$
σ^2	variance of the frame difference signal $(p_{j,k}-p_{j,k-1})_i$
σ_e^2	variance of the estimation error $(p_{j,k}-\widetilde{p_{j,k}})_i$

Equation set (2.17) can be significantly simplified to the closed form:

$$SNR = 10\log_{10}\frac{\underset{j\le n}{avg}\|\Delta M_j-\mu\|}{\underset{j\le n}{avg}\|\Delta E_j-\mu_e\|}\ . \tag{2.18}$$

2.5.4 Summary of distortion measures for 3-D meshes

Table 2.6 summarises the reviewed distortion metrics used in 3-D mesh coding and animation.

Table 2.6: Summary of common 3-D mesh distortion metrics.

METRIC TYPE	METRIC NAME	
STATIC MODEL DISTANCE	Euclidean Distance	[CKL+00]
	Mean Geometric Error	[CRS98]
	Hausdorff Distance	[Cru03, RA02]
	SNHC Model Distance	[ISO98]
ANIMATION ERROR METRIC	3-D PSNR	[VOH01]
	Alternative SNR	[GSK02, GSK03]
	Visual Smoothness	[VHO03]

Chapter 3

3-D Animation Coding

Data compression in the area of 3-D graphics has focussed on reducing spatial redundancy from 3-D models, by compactly representing their geometric and attribute information. Existing algorithms are considered static, in the sense that they can only operate in intra mode, and do not exploit temporal coherence of model sequences, if applied to compression of time-dependent 3-D models. Furthermore, no known compression method exists that can operate synergistically with network feedback in order to adapt its bitrate to degrading network conditions expressed as variable available bandwidth.

This chapter proposes and evaluates a coding method for dynamic 3-D wireframe models, suitable for Internet streaming. It concludes that dynamic coding methods lead to significant bandwidth savings in comparison to static methods. Variations of the proposed method also enable adaptive bitrates, which can be exploited by a prototype 3-D streaming architecture to provide certain level of QoS to dynamic wireframes.

3.1 Background and Motivation

Image, video, audio and computer graphics represent a major source of multimedia in our time. Increasing demand for modern applications such as audio and video conferencing or IP telephony has led audio, video and still image media to become particularly popular and fuelled further research and development in the processing of their signal, as well as in multimedia communications. As a result, a large international standardisation effort has taken place. Applications such as digital TV broadcasting, interactive 3-D games and e-shopping, combined with the high popularity of the Internet, are rapidly changing the scenery demanding richer interactive multimedia services. Existing media gain new importance: 3-D graphics and animation are among them. Despite the existence of a plethora of file formats and encodings for 3-D data (i.e., VRML 2.0), an efficient compression and animation framework is sought [Jan00] in the context of MPEG-4.

MPEG-4 attempts to provide a state-of-the-art standard that covers among others the aforementioned areas of 3-D scene compression and delivery through a tool set called Binary Format for Scene (BIFS) [SFE00]. BIFS is a compressed binary format derived from VRML 2.0, which it extends by preserving backwards compatibility. Apart from 3-D scene definition, BIFS also allows scene modification, through BIFS-Command, and limited scene animation, through BIFS-Anim [Sig00].

BIFS tools (or components) are, like in VRML 2.0, organised in a scene graph. The scene graph is a tree structure in which each node is a component and branches are its properties. As each component represents one aspect of functionality, the early components that appeared in VRML and BIFS were low-level and covered the basic graphics functionality (lines, shapes, colours, etc.). In this sense they were very close to the graphics API such as OpenGL. In later extensions of the BIFS specification, and recently in the AFX amendment [ISO02], higher-level components were needed and were finally adopted.

For example, the MPEG-4 SNHC tools comprise few components that yield reasonably high compression for still textures, face and body animation, and 3-D mesh coding. In MPEG-4, the two early animation tools, BIFS-Anim and 'faceanim' were based on a DPCM system and arithmetic encoding that allowed for low delay coding. The coding scheme described in subsection 3.2.2 follows a similar block structure.

Despite the capability of VRML 2.0 and BIFS to provide animated graphics, it was soon felt that such support was limited due to a number of factors. VRML, for one, follows a download-and-play philosophy, which is inadequate for streaming synthetic 3-D content. Likewise, the interpolator-based animation capabilities of this standard not only require a great amount of stored data to describe the animation, but also demand the existence of modelling and authoring tools to automate the generation of realistic animation interpolators. It has been reported that in a VRML/BIFS scene with 2-D and 3-D animation content, 80% or more of the file often contains animation and geometry data [BS02]. MPEG-4 promised to reduce the storage requirements of 2-D and 3-D graphic scenes by calling for suitable technologies that could efficiently compress scene geometry. This led to the development of the 3-D mesh coding component (3DMC [ISO99]). And, more recently, MPEG-4 adopted methods of compressing interpolators [KJH+02] within the BIFS tool in a step towards efficient representation of rich synthetic scenes containing animations.

These source coding tools alone do not allow the robust transmission of the scenes in error-prone environments, nor could they scale to accommodate heterogeneous receiver capabilities. Moreover, coding and transmitting rich synthetic scenes that contain animated objects is a cum-

bersome task, mainly because there is no unified solution that could take into account efficient source coding of animation, loss robustness, and scalability.

Designing a 3-D animation codec suitable for the best-effort Internet would require, besides the signal-processing domain, special consideration of the channel characteristics. These refer to packet loss, reordering and duplication, delay, delay variation (jitter), and even fragmentation. Traditional packet audio/video tools, such as rat and vic [uu00] use RTP [SCFJ03] with adaptive playout buffering algorithms to cope with variable delay. Loss is unpredictable, whether it is expressed in the short-term as some packets being independently dropped, or in the longer-term in the form of loss bursts or network outages. In such cases, an error resilience scheme is desirable along with a good payload format design.

This chapter lays the foundations of a coding scheme, called 3D-Anim, suitable for the compression and robust transmission of animated 3-D models. Also included is its RTP payload format, which draws heavily on the best common practice guidelines for writers of RTP payload formats in [HP03]. 3D-Anim represents the source compression component of a generic architecture, suitable for streaming 3-D animated content with some degree of QoS. This streaming architecture is also presented, and the interaction between its source and network components is described. Further description of its capabilities for error resilience and data repair is provided in the following chapters.

The remainder of this chapter is organised as follows. Section 3.2 formally introduces the area of dynamic 3-D animation coding and describes 3D-Anim, a basic single-resolution animation coding method. Section 3.3 presents enhancements and optimisations to basic wireframe animation coding. Section 3.4 formally introduces multi-resolution wireframe animation coding and argues that such codecs can form the basis of network adaptive 3-D animation streams. The Internet architecture is then presented in section 3.5. It is designed for streaming dynamic 3-D animation content adaptively, and was used to evaluate the coding methods of the previous sections. Section 3.6 concludes this foundation chapter with reference to the error resilience capabilities of the streaming architecture that will be described in the following chapters.

3.2 Dynamic 3-D Wireframe Coding

3.2.1 Formal definition of wireframe animation

Assume that a static 3-D model, $\mathcal{M}$, exists whose geometry and connectivity are given. In popular 3-D graphics standards, this information is provided either in a simple structured uncompressed matrix form (i.e., in VRML 2.0) called the `IndexedFaceSet`, or in an enhanced, more compact, and yet structured form (i.e., in BIFS) called `IndexedFaceSet3D`. In both

representations, the underlying data structure is an ordered set of indexed vertex positions that stores the geometry, along with an unordered set of polygon descriptions (where a polygon is a list of edges followed by a delimiter) that stores the connectivity of the given model $\mathcal{M}$. However, none of these `IndexedFaceSet` structures can store time-varying geometry or connectivity.

The vertices m_j of a time-dependent isomorphic 3-D mesh, corresponding to model $\mathcal{M}$ at time t, form the *indexed set* $M_t = \{m_{jt},\ j{=}1, 2, \ldots, n\}$, where n is the total number of vertices in the mesh. Since a vertex has three spatial components (x_j, y_j, z_j), and assuming that no connectivity changes occur in time (isomorphic mesh with constant n), the indexed set's geometric data at time t can be represented by the *position matrix* M_t, as:

$$M_t = \begin{bmatrix} x_{1,t} & x_{2,t} & \ldots & x_{n,t} \\ y_{1,t} & y_{2,t} & \ldots & y_{n,t} \\ z_{1,t} & z_{2,t} & \ldots & z_{n,t} \end{bmatrix}.$$

The indexed set of vertices may be partitioned into intuitively natural partitions, called *nodes*[1]. For example, in wireframe model FRED shown in Figure 3.1, different nodes have been coloured in the rendered model on the right to be easily identified.

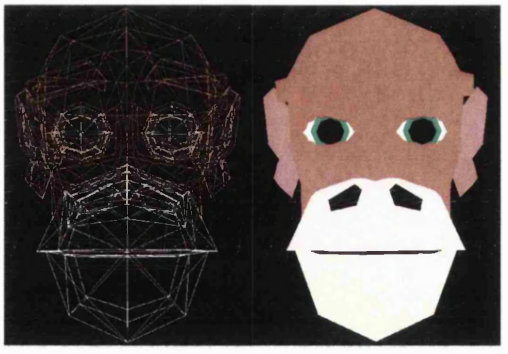


Figure 3.1: Nodes of model FRED.

Then, the position matrix of the i^{th} node is denoted by $N_{i,t}$. Note that, without loss of generality, the position matrix M_t for k such nodes can now be expressed as:

$$M_t = \begin{bmatrix} N_{1,t} & N_{2,t} & \ldots & N_{k,t} \end{bmatrix}.$$

For notational convenience, $N_{i,t}$ $(i{=}1, 2, \ldots, k)$ is used for representing both a node's position matrix and to refer to the i^{th} node $\mathcal{N}_i$ of the model as well. Similarly, the notation M_t refers both to the model's positional matrix at time t and to denote the model $\mathcal{M}$ at time instant t.

[1]The term is chosen to highlight the correspondence of such nodes with the nodes as defined in VRML and BIFS. The usefulness of nodes will be made clearer after a related discussion on model layering, in section 5.3.

Free-form animations of a 3-D mesh may result in position matrices that exhibit high variability in the time domain, which makes them unsuitable for applications where transmission of the animation data over a communications channel is required. In such applications, it is desirable for a compressed representation to be as compact as possible. The objective of a time-dependent 3-D model compression algorithm is to compactly represent the sequence of position matrices M_t, as defined above, that form the synthetic animation.

3.2.2 The 3D-Anim codec

The 3D-Anim input signal is defined as the set of *non-zero displacements* of all vertices in all nodes at time t:

$$D_t = \{\ d_{it}=m_{it} - m_{n0},\ i=1,2,\ldots,p\ (p \leq n) : d_{it} \neq 0\ \}. \tag{3.1}$$

Following the earlier notation, the input signal can be expressed with a displacement matrix, D_t, as:

$$D_t = M_t - M_{t-1} \Leftrightarrow$$

$$D_t = \begin{bmatrix} x_{1,t} - x_{1,t-1} & x_{2,t} - x_{2,t-1} & \ldots & x_{p,t} - x_{p,t-1} \\ y_{1,t} - y_{1,t-1} & y_{2,t} - y_{2,t-1} & \ldots & y_{p,t} - y_{p,t-1} \\ z_{1,t} - z_{1,t-1} & z_{2,t} - z_{2,t-1} & \ldots & z_{p,t} - z_{p,t-1} \end{bmatrix}, \tag{3.2}$$

or, equivalently, using node matrices:

$$D_t = \begin{bmatrix} F_{1,t} & F_{2,t} & \ldots & F_{l,t} \end{bmatrix}, \tag{3.3}$$

where $F_{i,t}$ the displacement matrix of node i, for l such nodes ($i=1,2,\ldots,l$). Note that D_t's dimension may be reduced to p (where $p \leq n$) compared to M_t, since D_t by the above definition does not contain vertices for which the displacement on all axes is zero. The latter vertices are called *stationary vertices*. Note, too, that $l \leq k$ holds, in the event that no vertices in a node are displaced, resulting in *stationary nodes* (or, $F_{i,t}=\mathbf{0}$).

Figure 3.2 shows the effect of differential coding of geometry coordinates. The original range of the distribution of geometric coordinates of model FRED, shown in the upper three plots of the figure (one plot for each component coordinate), is largely reduced to the compact distribution of the lower plots. Similar distributions for all other sequences used in this dissertation are given in Appendix A.2.

The scheme described and summarised in equation (3.2), is suited to a DPCM coder, as described in paragraph 3.2.2.2. The coding process assumes that the initial wireframe model M_0, here termed as the *reference model*, is already present at the receiver. The reference model

can be compressed and streamed with an existing method for static 3-D mesh transmission, such as the robust method proposed in [RA02], along with error protection if it is assumed the transmission is done over the same lossy channel as the time-dependent mesh. The proposed 3D-Anim coding method may interoperate with such static mesh transmissions and these will not be discussed further here.

3.2.2.1 Frame types

In the 3D-Anim codec's context, a *P-frame* describes changes from the reference model M_0 to the model at the current time instant t. A *D-frame*, describes the changes of a model from the previous time instant $t-1$ to the current time instant t. The corresponding position and displacement matrices for P and D frames are denoted respectively by M_t^P, M_t^D, D_t^P, D_t^D.

In P-frames, the residual error vectors generated by differentially encoding a 3-D model against a reference model are larger than the corresponding error vectors in D-frames. The magnitude, however, of this difference is not large and usually depends on the amount of motion present in the frames being coded. The size of a P-frame is, on average, larger than the preceding or following D-frame by less than 6% in 4 out of the 5 models used to evaluate 3D-Anim (models listed in appendix A).

The way these two types of frames are placed in the sequence bitstream is a design decision, which can be based on source coding criteria (i.e., signal attenuation), or on feedback received from the network (i.e., random unidentified frame loss). For example, a simple placement policy would introduce P-frames at constant regular intervals in a stream of D-frames, if the content motion pattern of the sequence does not exhibit sudden peaks or falls (smooth mo-

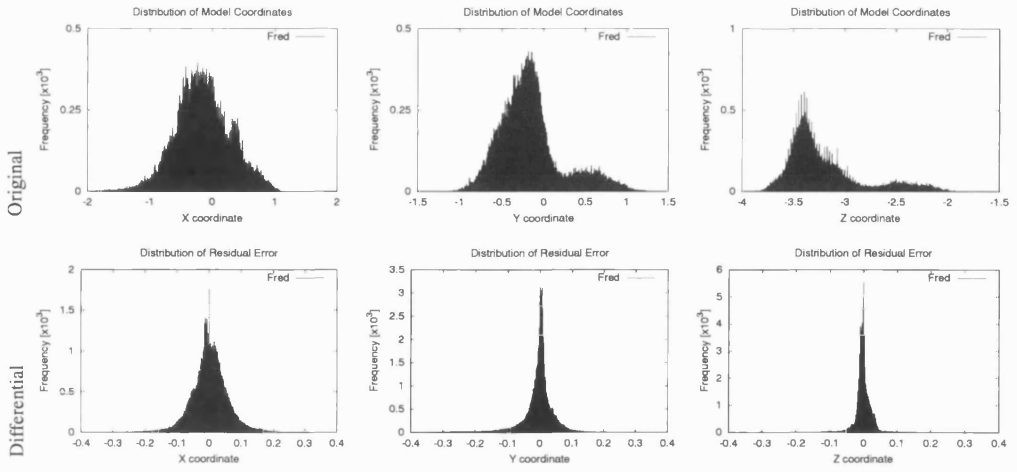


Figure 3.2: Distributions of absolute position matrix coordinates (upper row) and differential coordinates (lower row) for sequence FRED.

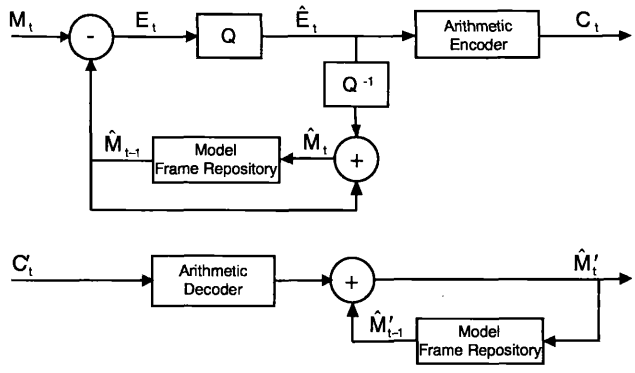


Figure 3.3: Block diagram of the 3D-Anim codec. Top: Encoder, Bottom: Decoder.

tion). In parallel, when loss is reported by receivers, P-frame frequency in the sequence can be increased to prevent prolonged deterioration of the animated model as a result of the prediction step in the decoder.

Due to the small variance of P-frame sizes in comparison to the size of D-frames, the simple regular frame placement policy is used in the experiments of section 4.4.

3.2.2.2 Encoding and decoding process

Figure 3.3 shows a block diagram of the coding scheme. The top figure 3.3 diagram depicts a DPCM encoder that takes advantage of the temporal correlation of the displacement of each vertex along every axis in the 3-D space. To encode a D-frame, the decoded set (animation frame or displacement matrix) of the previous instance is used as the predicted value, $\hat{M}^D_{t-1}$. (Equivalently, for encoding an P-frame, the predicted matrix is $\hat{M}^P_{t-1}$, where $\hat{M}^P_{t-1} = \mathbf{0}$ at $t = 0$ is the displacement matrix for the reference model.) Then, the prediction error E_t, i.e., the difference between the current displacement matrix and the predicted one, is computed and quantised ($\hat{E}_t$). Finally, the quantised samples are entropy coded (C_t) using arithmetic coding [MNW98], which adapts to the distribution of the geometric coordinate components. The predictive coding scheme of 3D-Anim prevents accumulation of quantisation error.

A DPCM decoder (bottom of figure 3.3) first decodes arithmetically the received samples (C'_t) and then computes the decoded samples ($\hat{D}'_t$).

Similar coding schemes have been used in MPEG-4 for the compression of facial animation parameters [TO00] and for BIFS-Anim [Sig00].

3.2.2.3 Quantisation

In previously proposed methods for still geometric data compression, scalar quantisation individually for each component coordinate has been followed. The approaches taken for example in Deering's triangle strips method [Dee95], or in Topological Surgery [TR98], Edgebreaker

[Ros99], Touma-Gotsman's method [TG98] and others, were to first normalise the coordinate values in a unit cube and then to round off the normalised coordinates to an integer of fixed length k. Depending on how the rounding operation was performed, this process might have led to uniform or non-uniform quantisation. Clearly, the number of quantiser levels is restricted to be $K=2^k$ in this approach.

In 3D-Anim, this restriction has been lifted, and the quantiser's levels, K, can take values that are not necessarily a power of 2. This is to benefit the entropy encoding process that follows quantisation, which may be able to better allocate variable length codes to the most frequently used quantiser indices. A *uniform scalar quantiser* was designed, where the K quantisation levels for each component coordinate are of equal step size q, given for the x component by:

$$q_x = \frac{x_{max} - x_{min}}{K} \ . \tag{3.4}$$

Then, the quantiser is designed as:

$$Q(x) = \lfloor \frac{x - x_{min}}{q_x} \rfloor \ , \tag{3.5}$$

leading to the quantised indices:

$$0, 1, 2, \ \ldots \ , K-1 \ . \tag{3.6}$$

The corresponding inverse quantiser is:

$$\hat{x} \ = \ Q_x^{-1}(i_x) \ = \ i_x \cdot q_x + \frac{q_x}{2} + x_{min} \ . \tag{3.7}$$

The uniform scalar quantiser designed above is suboptimal. However, optimal quantisers such as those based on the iterative Lloyd-Max algorithm can be computationally expensive, and their implementations require maintenance and searching of look-up tables. This fact, combined with the sheer volume of vertex coordinates that may be involved in free-form animation scenarios of rich models, makes the implementation of optimal quantisers rather impractical for on-the-fly encoding and streaming. In addition, it has been reported that low-frequency errors, such as small vertex displacements due to uniform quantisation, are less noticeable to the human visual system, which is more sensitive to normals and lighting errors than geometry [SCOT03]. Furthermore, the inverse uniform scalar quantiser of equation (3.7), as designed above for 3-D models, is amenable to hardware-based decoding. Such a technique is described by Fowler et al. [FvdZT+00] in a terrain rendering system using 3-D geometric position data similar to those of the 3D-Anim codec. For these reasons, the simple uniform scalar quantiser was preferred over an optimal scalar quantiser.

The quantisation step size q was assumed to be invariant among different nodes of the same model. Allowing different quantisation step sizes for different nodes may result in artifacts, such as mesh cracks and overlaps, especially in the edge and vertex boundaries between nodes, as was discussed in the literature (paragraph 2.2.1.2) and experienced by Chow [Cho97] in his adaptive non-uniform quantisation for still 3-D mesh coding. Such cracks can be repaired with *stitching* operations; however, these add complexity and may require modifications to the surface mesh in order to achieve good visual results. They have not, therefore, been used in 3D-Anim.

3.2.2.4 Entropy coding

In figure 3.3, the quantised prediction error $\hat{E}_t$ of the geometric coordinate samples is entropy coded without loss, using an adaptive arithmetic coder. Arithmetic coders are known to be asymptotically optimal, whereas Huffman coding performance is within 1 bit of the entropy. This means that arithmetic coding leads to more compact symbol representations on average [MNW98].

The input string of symbols ($\hat{E}_t$) is compacted into an encoded string sequence of bits with the aid of an adaptive model. The model is a way of calculating, in the context of 3D-Anim, the distribution of probabilities for the next input quantised geometric coordinate component. Being adaptive, the model recalculates and assigns different such probabilities for each symbol encountered so far in the message based on its frequency of appearance. This process leads to the assignment of variable length codes to each input symbol in the message, with the most frequent symbols being finally represented with shorter codes in the encoded bit sequence[2].

3.2.2.5 Animation masks

The 3D-Anim scheme defines one table and at least two bitmask structures for each dynamic model in order to convey which vertices and nodes of the model are non-stationary, and therefore encoded. These structures are the *Node Table*, the *Node Mask* and one or more *Vertex Masks*.

The Node Table is an indexed list of all `IndexedFaceSet` nodes in the dynamic model, and is assumed to be known *a priori* at the decoder. This fact follows the earlier assumption that the wireframe model already exists at the decoder, or is downloaded through other means. The Node Table helps the decoder identify and refer to each of the model's nodes by their index in the table.

[2]The implementation of the arithmetic codec for 3D-Anim follows that of Moffat et al. [MNW98] with low bound `mLow` = 0 and high bound `mHigh` = 2^{16}.

The Node Mask is a bit sequence where a bit, if set, denotes that the corresponding `IndexedFaceSet` node in the Node Table is non-stationary.

Vertex Masks follow a similar definition, where set bits correspond to non-stationary vertices to be coded. There is one Vertex Mask per coordinate component.

From the previous definitions it follows that a model may have as many sets of Vertex Masks as its number of non-stationary nodes. For stationary nodes, no Vertex Masks are defined. Finally, in animations where all vertices and all nodes are non-stationary, definition of the previous structures is redundant.

The bitmask structures as defined previously are not compact. The 3D-Anim scheme can be extended at this point to cover lossless compression of the Node and Vertex Masks. In the bitmask case, a run-length encoder (RLE) would efficiently compact a stream of consecutive occurrences of the same bit symbol into a single symbol preceded by a number signifying the length of the symbol run. The RLE steps for the mask data of 3D-Anim can be summarised as:

1. Measuring separate histograms of run-lengths for 0's and 1's.
2. Smoothing ragged histograms by averaging neighbouring bin heights.
3. Creating Huffman codes, one for the 0's histogram and one for the 1's histogram.

Uncompressed animation masks are defined in BIFS-Anim [Sig00] in a compatible way and this allows the 3D-Anim scheme to be directly integrated with an MPEG-4 terminal.

3.2.2.6 Coding performance

In this paragraph, the performance of the simple 3D-Anim codec is compared against MPEG-4's 3-D Mesh Coding scheme (3DMC) [ISO99]. Additional comparison results are provided against MPEG-4's `predictiveMFField`, the predictive coding method used for compressing each *multiple field* structure (MFField, i.e., indexed face sets) in each frame individually.

The basic functionality of 3DMC as a progressive single-resolution encoder relies on the TS [TR98] and PFS [TGHL98] schemes presented in the literature, and its subsequent improvements during the course of development of the standard. Namely, Touma and Gotsman proposed the use of parallelogram prediction [TG98], while Li and Kuo proposed an improved error coding scheme [LLK97]. Later, the connectivity encoding for progressive transmission was enhanced by Bossen [Bos99]. Finally, Guéziec et al. contributed the non-manifold encoding [GBTS99] based on cutting and stitching. For these technologies, the 3DMC coding scheme represents the state of the art as shown by extensive experimentation performed during the course of the MPEG-4 process.

The block diagram for the predictive coding scheme of MFField is shown in figure 3.4. This coding method exploits the redundancy that exists between vertices lying in the same 'neighbourhood' in the 3-D space, and has an effect on sequences of statically encoded `IndexedFaceSets` (encoding of a model on a frame-by-frames basis).

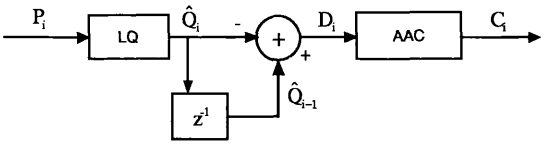


Figure 3.4: Block diagram of the `predictiveMFField` coder.

Results of the compression comparison are plotted in figure 3.5 for models BOUNCEBALL, FRED, CHICKEN and TELLY, encoded with P-frames inserted every 15 frames (denoted by P/15). All graphs show that by removing temporal redundancy with the basic 3D-Anim (blue plot) one can achieve deeper compression of dynamic wireframe sequences as opposed to compressing the sequence's frames spatially only with 3DMC (red plot). The green, grey and black plots correspond to the compression performance of textual descriptions of the frames in either differential geometric coordinates (removing temporal redundancy), statically compressed 3-D coordinates (removing spatial redundancy), or in VRML mode, respectively (labelled as 'DiffTab.gz', 'Tab.gz', 'VRML.gz'). In all these cases, the quantised information has also been adaptively arithmetically coded for fair comparison against 3D-Anim.

The red and blue plots in figure 3.5 also reveal the effects of P-frames on the coding performance of 3D-Anim. The periodic bitrate rises due to D-frame coding in sequences FRED and CHICKEN are apparent, and are expressed as small spikes in simple animations, or in dynamic models with few to medium number of vertices (FRED), or as sharp increases of the bitrate in more detailed models (like CHICKEN).

Another remark worth mentioning can be made on the smooth and straight shape of the 3DMC plots (red graphs). 3DMC is tailored to compression of static 3-D models and, having assumed isomorphic meshes whose connectivity and resolution does not change between successive frames, it results in almost the same bitrate per frame, hence a smooth line.

In predictive multiple field mode, the performance of compressed `IndexedFaceSet` structures for models BOUNCEBALL and TELLY are also plotted as a purple line. Conceptually, the `predictiveMFField` method can be seen as the binary form of the textual tabulated descriptions and their performance is expected to be rather similar. This is, indeed, confirmed by the purple plots on models BOUNCEBALL and TELLY (labelled as 'predMF').

With the simple 3D-Anim codec described in subsection 3.2.2, the average compression

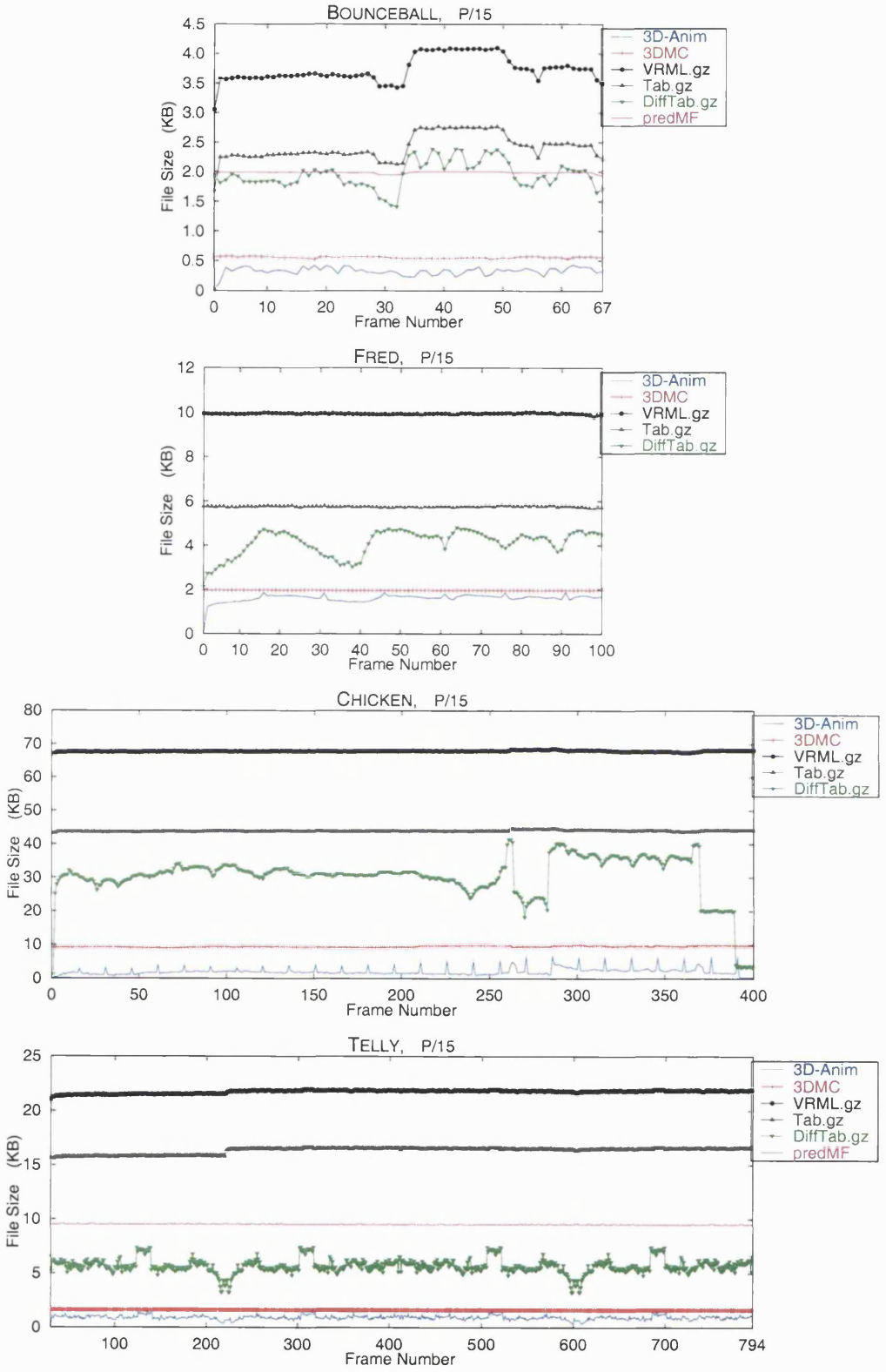


Figure 3.5: Comparison of 3D-Anim with 3DMC and other statically, or differentially, encoded sequences.

efficiency achieved per vertex component x, y, z, is listed in table 3.1 for wireframe TELLY and for different P-frame frequencies. Quantisation step size was assumed to be the same for all nodes of TELLY to avoid surface cracks.

Table 3.1 also reveals that the bitrate overhead due to introduction of P-frames in the sequence is small compared to D-frame bitrates. This can be derived by examining the compression efficiencies reported in the second and third large columns of table 3.1, corresponding to increased frequencies of P-frames in the animation sequence, against the first large column (where only one P-frame is used). Namely, by introducing P-frames regularly at intervals of 20 frames the bitrate overhead of sequence TELLY is 2.2% at worst, calculated as the ratio of the total bits per vertex of the second over the first large column of table 3.1. Similarly, for P-frames inserted every 8 frames the corresponding overhead is calculated to be 3.7%. It is also notable that these overheads show little variation when modifying the quantisation step sizes. The previous outcome is also noticeable in all other sparse or medium detail sequences, such as FRED and BOUNCEBALL (5% and 3.1% at worst).

In CHICKEN, however, P-frame bitrate is significantly higher than that of a D-frame, with the overhead reaching 35%. This is not surprising as motion in CHICKEN is intense and complete in all three geometric components, thus resulting in the model being considerably displaced with regards to the reference. Recall that, in paragraph 3.2.2.1, 3D-Anim P-frames have been defined as displacements relative to the reference model. The latter remark about CHICKEN is also confirmed by the 3D-Anim plots in the graphs of figure 3.5.

In table 3.2, the average bitrate of TELLY is listed for various streaming experiments as calculated by experiment logs. There are two interesting aspects worth noting: a) TELLY's bitrate varies significantly over time as its standard deviation suggests and, b) TELLY's bitrate varies little with increasing P-frame frequency. The latter point reconfirms the discussion of the previous paragraph. The former point implies that wireframes of sparse animation content and up to medium resolution (fewer than thousands of vertices) may result in rather higher variability rates than more intense animations of high detail (resolution) models. CHICKEN's plot in figure 3.5 confirms this remark by exhibiting very smooth D-frame bitrates, but further validation is necessary to generalise this statement.

Generally, when compressing dynamic 3-D wireframes with 3D-Anim, the previous results represent useful knowledge about the behaviour of the codec, and may lead to the development of variable P-frame placement policies according to network reported feedback and animation content. Such policies, however, are not examined in this thesis.

Table 3.1: TELLY: Compression efficiency (bits/sample) per component x, y, z, for various quantiser step sizes and P-frame positions.

		P-FRAME POSITION								
		Only frame 1			Every 20 frames			Every 8 frames		
QUANT. STEP SIZE		x	y	z	x	y	z	x	y	z
(2^5)	32	3.39	4.28	4.23	3.41	4.33	4.25	3.46	4.42	4.29
(2^6)	64	3.84	4.75	4.70	3.88	4.83	4.74	3.94	4.95	4.81
	96	4.12	5.01	5.05	4.18	5.11	5.07	4.26	5.23	5.13
(2^7)	128	4.37	5.21	5.27	4.41	5.32	5.31	4.47	5.45	5.39
	160	4.47	5.35	5.47	4.51	5.48	5.49	4.60	5.61	5.57
	192	4.58	5.46	5.58	4.67	5.60	5.64	4.72	5.74	5.71
	224	4.69	5.55	5.70	4.78	5.68	5.75	4.82	5.87	5.85
(2^8)	256	4.80	5.67	5.85	4.84	5.79	5.89	4.91	5.95	5.97
	288	4.89	5.72	5.94	4.94	5.87	5.97	5.03	6.02	6.03
	320	4.94	5.81	6.00	5.03	5.94	6.02	5.08	6.13	6.13
	352	5.04	5.84	6.07	5.06	6.01	6.14	5.17	6.19	6.19
	384	5.12	5.96	6.12	5.12	6.05	6.21	5.20	6.23	6.27
	416	5.17	6.02	6.24	5.19	6.15	6.26	5.28	6.32	6.34
	448	5.24	6.07	6.27	5.24	6.19	6.32	5.34	6.34	6.39
	480	5.26	6.09	6.35	5.27	6.23	6.36	5.38	6.42	6.45
(2^9)	512	5.28	6.12	6.37	5.33	6.28	6.43	5.39	6.46	6.54

Table 3.2: TELLY: Average output bitrate for various P-frame positions.

P-FRAME POSITION	AVG. ENCODER BITRATE (kbps)	STD. DEV.
None	232.5	38.0
30	234.8	39.7
15	236.7	40.6
8	240.1	42.3

3.2.3 The 2D-Anim codec

The same definition of 3-D model data given earlier in subsection 3.2.1 can cover the reduced case of two-dimensional graphics. In particular, the 3D-Anim encoder can be seen as a superset of 2D-Anim, the corresponding temporally predictive wireframe mesh coder for 2-D geometry data.

Streaming applications and architectures can take advantage of the reduced amount of data required to achieve realistic graphics models in 2-D . The geometric data can be either stored (pre-designed animations or cartoons) or, most intriguingly, generated on-the-fly by a 3-D to 2-D projection engine, if the corresponding 3-D models are already available. In such scenarios, graphics streaming architectures can accommodate heterogeneous sets of clients with

reduced processing, rendering, or networking capabilities. Such an approach is absent from other graphics streaming systems [Mar02] targeting heterogeneous client populations, which are based on transcoding.

Concolato et al. [ISO01c, CDM03] have identified methods of adapting BIFS to encoding cartoons. However, all proposed BIFS-based methods suffer from those inadequacies associated with BIFS itself: lack of robustness in streaming, and inflexible decoding. Missing BIFS data due to network packet loss, or lack of repair mechanisms at receivers, will force BIFS to quit decoding in the event of packet loss. A scheme based on 2D-Anim and the stream robustness provisions proposed in the following chapters are complementary solutions.

Finally, the 2D-Anim coder can take advantage of any enhanced features available to 3D-Anim, such as distortion optimisations, or rate adaptation, as described in section 3.3.

3.3 Enhanced Wireframe Animation Codecs

In 3D-Anim terminology, *dense* animations are called those sequences with a large number of non-stationary vertices. Following the notation introduced in subsection 3.2.2, this notion of density (or sparsity) of an animation sequence can be expressed through the number of non-stationary vertices p ($p < n$) and the number of non-stationary nodes l ($l < k$). The density (or sparsity) of the animation can be formally quantified by the *average density ratio*, df_{avg}, defined as:

$$df_{avg} = \frac{1}{N}\frac{1}{k}\sum_{f=1}^{N}\sum_{j=1}^{l}\frac{p_{jf}}{n_j}, \text{ with } l \leq k \text{ and } |p| \leq |n|, \tag{3.8}$$

in the range [0..1], where N is the number of animation frames and k the number of nodes in the reference model. When $p \rightarrow n$ and $l \rightarrow k$ equation (3.8) gives $df \rightarrow 1$, thus describing a *dense* animation. For $p = n$ and $l = k$ the above equation results in $df = 1$ and the animation is termed *complete*.

By controlling parameters p and l, the encoder can generate layered bitstreams (through parameter l), where each layer L can be scalable (by adjustment of parameter p). Different options for organising 3-D animation data into layers, and thus how parameter l can be adjusted in this case, are proposed and discussed further in chapter 5.

3.3.1 The enhanced 3D-Anim (E3D-Anim)

Controlling the number of non-stationary vertices through parameter p leads to introduction of enhancement features to 3D-Anim. When the displacement of a non-stationary vertex is below a threshold, ϵ, the coding distortion associated with this vertex is expected to be low compared to the distortion introduced by other non-stationary vertices with larger displacements. In fact,

animated vertices with large displacements tend to dominate in all formal expressions that define error metrics, such as those presented in subsection 2.5.3. Thus, vertices with very small displacements can be treated as stationary and be excluded from coding. The Vertex Mask structure facilitates this by resetting the vertex's corresponding bit.

Let d_{it} denote the displacement of the i-th vertex v_{it} of a dynamic 3-D model $\mathcal{M}$ with n vertices between time instants $t-1$ and t, under the simplifying assumption that $\mathcal{M}$ is structured as a single node ($k=1$). Let b_{it} be the bit corresponding to v_{it} in the Vertex Mask. Then,

$$b_{it} = \begin{cases} 0, & |d_{it}| < \epsilon \ , \\ 1, & otherwise \ . \end{cases} \tag{3.9}$$

For the displacement vector $d_t = [d_{1t}, d_{2t}, \ldots, d_{nt}]$, a corresponding 'profit' vector can be defined as $P_t = [P_{1t}, P_{2t}, \ldots, P_{nt}]$, with $P_{it} = \frac{1}{D_{it}}$, where D_{it} is the distortion introduced to the animation frame t of model $\mathcal{M}$ due to treating vertex v_{it} as stationary. Similarly, the cost vector $R_t = [R_{1t}, R_{2t}, \ldots, R_{nt}]$ is defined to represent the *rate* consumed by vertices $[v_{1t}, v_{2t}, \ldots, v_{nt}]$. With this notation, R_{it} represents the current measurement of bits consumed by vertex v_{it} in frame t during the encoding phase.

Then, maximisation of the profit vector P_t is sought (or, minimisation of the distortion of the model) subject to a cost (rate) constraint, as:

$$P_t(R_t) = max \sum_{i=1}^{n} b_{it} \cdot P_{it} \ , \tag{3.10}$$

subject to:

$$\sum_{i=1}^{n} b_{it} \cdot R_{it} \ \leq \ R_t \ . \tag{3.11}$$

Equations (3.9)-(3.11) formally define the constrained optimisation problem known as the *0-1 Knapsack Problem* (also known as the Capital Budgeting Problem), which forms the basis of the enhanced 3D-Anim coder. In this mode, 3D-Anim operates as a sieve, by excluding from coding stationary, as well as nearly stationary, vertices. The E3D-Anim problem formulation, despite being NP-hard, can be significantly reduced to yield a fast and optimal solution by appropriately selecting the cost vector to denote average vertex rates in the sequence.

The decoder, if necessary, may apply post-processing mesh optimisation or fairing techniques [HDD+93, Kob97] to improve the visual quality of the rendered model.

3.4 Multi-resolution Wireframe Animation Codecs

In the previous sections, the simple and enhanced 3D-Anim codecs assumed single resolution models and coding methods, thus yielding bitstreams that may not easily adapt their rates to

varying network conditions. Any attempt to shape the outgoing rate would require the use of coarser quantisation which, in the case of model-based coding, is known to result in mesh cracks. A further enhancement to 3D-Anim is possible whereby the number of faces of model $\mathcal{M}$ and, therefore, the corresponding transmission rate of the encoder, are adapted to the available network bandwidth constraints. This approach has the advantage of trading a wireframe mesh's geometric constructs (face resolution) for crack avoidance over the classic quantisation-based adaptation.

The following subsection introduces the general concept of multi-resolution meshes, which is then extended in subsection 3.4.2 to represent dynamic models.

3.4.1 Multi-resolution meshes

A multi-resolution mesh representation of a geometric object $\mathcal{M}$ is a representation that embodies a set of meshes $\{\mathcal{M}^k, \mathcal{M}^{k-1}, \ldots, \mathcal{M}^1\}$, each of which is in turn a representation of $\mathcal{M}$. These representations can be seen as different approximations of the original object subject to some tolerance. The set of approximations is considered ordered with $\mathcal{M}^i$ denoting a representation 'closer' to $\mathcal{M}$ than $\mathcal{M}^j$ for $i > j$. This ordering implies that $\mathcal{M}^1$ represents the coarsest approximation to $\mathcal{M}$ and $\mathcal{M}^k$ the finest. Following this notation, the original model $\mathcal{M}$ can also be referred to as $\mathcal{M}^k$.

In general, creating the multi-resolution representations involves the following primary decisions:

- Selection of a decimation primitive.
- Quantitative definition of the tolerance (error) to prioritise the decimation elements.

Once these parameters are established, a multi-resolution representation of a model can be built as a sequential process, where the sequence of discrete modifications (decimation operations) is recorded in a linear data structure such as a priority queue. According to the direction of the traversal on this structure and the current approximation, one can apply a series of decimation operations from the sequence on the existing mesh either from coarse to fine or from fine to coarse. For simplicity, and for other reasons explained in paragraph 3.4.4.3 of this chapter, the mesh simplification process assumes only independent decimation operations, i.e., such that two sequential decimation operations do not share any geometric elements (vertices, edges, faces).

Mesh simplification methods like those described in paragraph 2.2.2.3 of the literature, construct a coarse base representation $\mathcal{M}^0$ of a detailed initial mesh $\mathcal{M}^k$, by iteratively applying a contraction operation ϕ^i, such as edge collapse. The recorded sequence of simplification

operations forms a *simplification stream*, whereas the ordered set of intermediate mesh representations $\mathcal{M}^i$ forms an *incremental representation*, as:

$$\mathcal{M}^k \xrightarrow{\phi^k} \mathcal{M}^{k-1} \xrightarrow{\phi^{k-1}} \ldots \xrightarrow{\phi^1} \mathcal{M}^0 .$$

Hoppe [Hop96] introduced the progressive mesh structure, by exploiting the fact that the edge collapse operation is invertible as a vertex split operation. Thus, for each contraction ϕ^i, Hoppe defined the corresponding inverse split, ψ^i, and proposed the *progressive representation* as a base mesh $\mathcal{M}^0$ followed by a refinement sequence ψ^i :

$$\mathcal{M}^0 \xrightarrow{\psi^1} \ldots \xrightarrow{\psi^{k-1}} \mathcal{M}^{k-1} \xrightarrow{\psi^k} \mathcal{M}^k .$$

(The formal notation here is similar to that of paragraph 2.2.2.1 where the edge collapse operation is covered by ϕ and the vertex split by ψ.)

3.4.2 Formal definition of dynamic multi-resolution meshes

In mesh decimation literature, simplification techniques applied solely to static meshes are encountered. Similarly, in the emerging field of dynamic mesh geometry coding, only single resolution approaches have been proposed. In this subsection, the *time-dependent multi-resolution geometry representation* is introduced, by extending Hoppe's original progressive mesh representation to the time domain. This concept can be formally expressed as follows:

$$\begin{array}{c|ccccccccc}
\text{TIME} & & & \text{RESOLUTION} & & & & & & \\
 & \mathcal{M}^{k,0} & \xrightarrow{\phi^{k,0}} & \ldots & \xrightarrow{\phi^{i+1,0}} & \mathcal{M}^{i,0} & \xrightarrow{\phi^{i,0}} & \ldots & \xrightarrow{\phi^{1,0}} & \mathcal{M}^{0,0} \\
 & \downarrow \omega^{k,1} & & \vdots & & \downarrow \omega^{i,1} & & \vdots & & \downarrow \omega^{0,1} \\
 & \ldots & & & & \ldots & & & & \ldots \\
 & \mathcal{M}^{k,j} & \xrightarrow{\phi^{k,j}} & \ldots & \xrightarrow{\phi^{i+1,j}} & \mathcal{M}^{i,j} & \xrightarrow{\phi^{i,j}} & \ldots & \xrightarrow{\phi^{1,j}} & \mathcal{M}^{0,j} \\
 & \downarrow \omega^{k,j+1} & & \vdots & & \downarrow \omega^{i,j+1} & & \vdots & & \downarrow \omega^{0,j+1} \\
 & \ldots & & & & \ldots & & & & \ldots \\
 & \mathcal{M}^{k,t} & \xrightarrow{\phi^{k,t}} & \ldots & \xrightarrow{\phi^{i+1,t}} & \mathcal{M}^{i,t} & \xrightarrow{\phi^{i,t}} & \ldots & \xrightarrow{\phi^{1,t}} & \mathcal{M}^{0,t}.
\end{array}$$

In the notation tabulated above, each row represents a incremental representation and each column a dynamic mesh evolving downwards. Starting from the top-left representation of a detailed mesh at time 0 ($\mathcal{M}^{k,0}$) the proposed representation of the mesh at time j ($\mathcal{M}^{i,j}$) can either be transformed horizontally to $\phi^{i,j} * (\mathcal{M}^{i,j}) = \mathcal{M}^{i-1,j}$ following Hoppe's progressive decimation concept, or vertically as $\omega^{i,j+1} \circ (\mathcal{M}^{i,j}) = \mathcal{M}^{i,j+1}$ following a dynamic evolution

of the geometry. The inverse operation of a contraction can be similarly expressed with this notation as $\psi^{i+1,j} * (\mathcal{M}^{i,j}) = \mathcal{M}^{i+1,j}$ denoting a refinement (for simplicity, this is not shown in the tabulated notation).

Using the previous notation, an accumulation of elementary topological operations can also be expressed to describe transitions between any two models, $\mathcal{M}^{p,t}$, $\mathcal{M}^{q,t}$, in the incremental representation. A batch $\Phi_t^{p,q}$ of such simplification operations, and the inverse batch of refinements $\Psi_t^{q,p}$, is defined as:

$$\mathcal{M}^{p,t} = [\Phi_t^{p,q}] * \mathcal{M}^{q,t} = (\phi^{p,t} * \cdots * \phi^{q,t}) * \mathcal{M}^{q,t} \ ,$$
$$\mathcal{M}^{q,t} = [\Psi_t^{q,p}] \circ \mathcal{M}^{p,t} = (\psi^{q,t} \circ \cdots \circ \psi^{p,t}) \circ \mathcal{M}^{p,t} \ .$$

The increment (or decrement) $K = |p-q|$ with the batch operations $\Phi_t^{p,q}$, $\Psi_t^{q,p}$ can express any forward or reverse *fine-grain adaptation* of a model $\mathcal{M}^{p,t}$ to another $\mathcal{M}^{q,t}$.

3.4.3 Why use a multi-resolution model?

The relative merits of transmitting a multi-resolution 3-D mesh over a single-resolution mesh are numerous. Below, two key merits are described and argued for:

1. To avoid mesh cracks while *adapting* a model's resolution to network conditions.

2. The ability to transmit animation *while* a model is being streamed.

Traditionally, media rate control is based on increasing or reducing the quantiser's step size in order to respectively reduce or increase a codec's output bitrate. This approach, known as adaptive quantisation, if applied to 3-D animated meshes, may result in successive transitions (i.e., frames) between coarse and fine encodings of a 3-D model with potentially visible undesirable effects, such as cracks or rough model surfaces, during reconstruction. If, instead of variable quantisation, the encoder uses a reduced resolution model (by vertex elimination, or any other decimation rule), then the crack effect gives way to smoother mesh resolution transitions with much less noticeable surface distortions.

On the other hand, single-resolution meshes (i.e., progressive meshes) are *static* in the sense that once a sequence of detail coefficients is constructed, its ordering is rather inflexible and a fixed number of detail removal steps when applied leads to ordered and deterministic coarser resolution levels. Hence, transmission of progressive models is also static, in the sense that streaming is only permitted for rigid models. Without redundancy, or under variable network conditions, progressive meshes cannot be used to achieve adaptation and animation.

Multi-resolution animated 3-D models can be efficiently streamed over packet networks by allowing coarser models to adapt to reduced available bandwidth. The term "coarser models"

here reflects a state of reduced resolution through a decimation rule of choice (vertex, edge, face elimination, or other rule). Similarly, during refinement, the multi-resolution approach proposed in this thesis refers to the addition of geometric detail to the streamed model without considering other mesh processing techniques, such as smoothing or fairing.

Other mesh coding techniques that could allow network adaptivity are those based around the notion of subdivision surfaces. For example, the work of Labsik et al. [LKSS00] introduces the term adaptive subdivision for the application of progressive broadcasting of 3-D models. However, an adaptive subdivision streaming method only adds smoothness to the original model, not geometric detail through wireframe geometry refinement, as proposed in this thesis.

3.4.4 The progressive 3D-Anim (P3D-Anim)

3.4.4.1 Source encoding

The P3D-Anim encoder shown in figure 3.6 compresses the dynamic geometric positions of the vertices of the 3-D model while successive simplification or refinement operations take place on its structure [VHO04]. The encoder builds upon the simple 3D-Anim geometry coding scheme of subsection 3.2.2, in which each vertex geometry component is quantised and differentially predicted from its previous instance, after decimation analysis on the current mesh. The decimation analysis step identifies candidate vertices to split or edges to collapse (or any other topological operation) according to some criteria, usually minimisation of some form of error metric.

To encode a frame, the Vertex Mask structure of 3D-Anim is re-employed with slightly modified semantics. Setting a bit denotes that the corresponding vertex has been predictively coded, as in 3D-Anim. Resetting a bit now signifies that the corresponding vertex has been *contracted*. Each component stream of such predicted geometries is further arithmetically encoded in an adaptive manner separate from other components. To reduce the number of predicted geometry vectors, vertex pair contraction is utilised as decimation scheme [GH97]:

$$(v_i, v_j) \rightarrow \overline{v} \, . \tag{3.12}$$

The contracted vertex $\overline{v}$ is selected as one of the endpoints, v_i or v_j, and further geometric adjustment of the edges incident to the vertex pairs does not take place. This is to allow the decoder to reconstruct the animated model frame with the geometry of the 'decimated' (non-coded) vertices predicted by their neighbours. Having assumed isomorphic dynamic meshes means that the same edge indexing information can be reused at the decoder.

3D-Anim distinguished encoded vertices into P-frames and D-frames, by coding vertex positions relative to the geometry of the reference model ($\mathcal{M}^{k,0}$), or a previously encoded frame

($\mathcal{M}^{i,j-1}$ or $\mathcal{M}^{i-1,j-1}$) respectively. A P-frame in P3D-Anim enables the correction of error vectors of the 'decimated' vertices. This is exploited further in paragraph 3.4.4.4 to provide rate control.

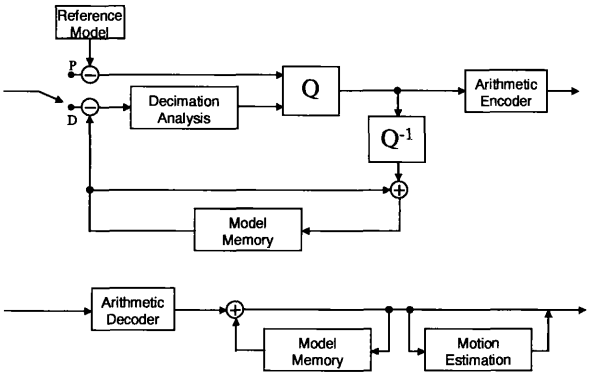


Figure 3.6: P3D-Anim encoder and decoder.

3.4.4.2 Motion-compensated decoding

The previous paragraph 3.4.4.1 described how the dynamic 3-D mesh can use Vertex Masks to register the vertices that have been differentially compressed, and also to indicate which vertices have remained un-coded. The decoder should be able accurately to predict geometry of both types of vertices.

To achieve this, a butterfly motion vector prediction scheme is employed, as described in paragraph 2.2.1.2 and shown in figure 2.7. For the un-coded vertices, a simple 1-ring butterfly predictor is used, obtained from equation (2.4) by setting $\alpha = 1$, if each un-coded vertex v_d has a sufficient number of neighbouring vectors. If insufficient 1-ring neighbours exist to substantiate the value of k (i.e., when $k = 0$), then vertex v_d is decimated by applying the vertex pair contraction operation with the nearest vertex v_n, as:

$$(v_d, v_n) \rightarrow \overline{v_{dn}} \,, \tag{3.13}$$

where $\overline{v_{dn}}$ is the optimised position of the endpoints v_d and v_n that minimises the error metric.

3.4.4.3 Choosing the optimal predictor

For the mesh decimation strategy, computation of an efficient predictor is needed for the geometry of the decimated vertices. This paragraph evaluates prediction from $1..N$ coefficients, where N is the valence of a vertex v before decimation.

The evaluation is based on the MSE of the geometry of the predicted vertices v_i. Intuitively, when evaluating prediction from a small number of coefficients (smaller than the va-

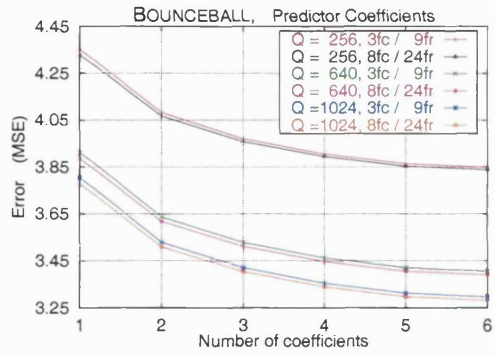


Figure 3.7: Average MSE of sequence BOUNCEBALL in multi-resolution format for varying number of displacement vector predictor coefficients. Graph plotted for varying quantisation levels and decimation step policies.

lence of v_i), one should consider those displacement vectors with the smallest distance from v_i as coefficients. In a similar situation, for their mesh broadcasting algorithm, Bischoff and Kobbelt [BK02b] considered all those vertices lying within a sphere of radius d centred at v_i.

For the evaluation, $\mathcal{M}^0$ has been considered as the coarser mesh such that any further edge contraction would result in at least one vertex within a *decimated hole* (i.e., that cannot be predicted since all its adjacent displacement vectors have been decimated). This rule can be relaxed in order to reach an even coarser mesh than $\mathcal{M}^0$ by considering prediction of vertex v_i from its higher distance neighbours that are not immediately adjacent to v_i. This approach, however, not only increases complexity, but is also likely to lead to volume shrinkage of low resolution models. Such distortions, however, may be less noticeable on rich models.

Figure 3.7 shows the prediction accuracy as the average MSE of sequence BOUNCEBALL for varying numbers of prediction coefficients. As can be seen, the average distortion decreases consistently while increasing the number of coefficients. The error is minimised when the number of prediction coefficients for each vertex reaches the maximum valence of the mesh. (Sequence BOUNCEBALL is semi-regular with more than 95% of its vertices having valence 6.) It is also shown that this relationship holds regardless of the resolution of the original model (achieved by varying quantisation levels $Q \in [256..1024]$), or the decimation step policy (consisting of either smooth but frequent steps of 3 decimated faces every 9 frames, or slow, but more aggressive, decimation of 8 faces every 24 frames).

3.4.4.4 Rate control

P3D-Anim operates a rate-control encoding algorithm by iteratively identifying candidate vertex pairs for contraction such that the output bitrate does not exceed the instantaneous available rate of the network R. The algorithms for the encoder and decoder are shown in figure 3.8.

Assuming that the network provides at least binary feedback of experienced rate increase or decrease along a sender-receiver path, the algorithm constructs appropriate batches $\Phi_t^{p,q}$ and $\Psi_t^{q,p}$ between detected bandwidth transitions.

Formally, this is expressed as:

$$Estimate\{p,q\}\ , \tag{3.14}$$

such that batches $\Phi_t^{p,q}$ and $\Psi_t^{q,p}$ satisfy the rate constraint:

$$R_K < R\ , \tag{3.15}$$

where R_K is the rate achieved after a batch of $K=|p-q|$ transitions.

As feedback packets in modern congestion control protocols, such as the equation-based TFRC [FHPW00], come at a frequency of one per round trip time T_{rtt}, fine adaptation is achieved by spreading the elementary operations contained in the batch simplifications and refinements over intermediate steps of the T_{rtt} interval. The number of steps within a round trip time interval can be controlled by appropriately selecting the number of elementary operations performed per step, thereby adjusting the *smoothness* of rate change between instances of feedback. In the P3D-Anim case, linear rate control in an additive fashion is used, in the belief that providing more aggressive adaptation by reducing the number of steps was likely to result in visible coding artifacts.

3.4.5 Evaluation

The progressive dynamic mesh codec P3D-Anim was used to evaluate the adaptation advantages of multi-resolution dynamic codecs over quantisation-based schemes.

For the experiments described in this subsection, the animations represented by the semi-regular model BOUNCEBALL and the non-regular CHICKEN were used. The original sequences have 66 and 400 frames, coded at 24 Hz and 15 Hz, and represented with 160 and 3030 vertices (320 and 5664 triangles) respectively, as described in appendix A. Both sequences have been coded with an P-frame inserted every 30 frames. To produce a substantial number of frames from these short sequences, each sequence was repeatedly concatenated with its reversed dual, resulting in a total of 528 (66 x 8) frames for BOUNCEBALL and 800 (2 x 400) frames for CHICKEN. Sequence mirroring also ensured smooth animation at the concatenation points, avoiding mesh jumps similar to scene changes in natural video. Simple concatenation of forward animations could have introduced bias in favour of the multi-resolution scheme.

The rate-controlled scheme proposed in paragraph 3.4.4.4 was compared to a typical scheme based on quantiser level adjustment that trades bitrate for geometric precision (here-

ENCODER

```
1.   R := rate budget;
2.   Repeat:
3.      Identify vertex pairs for contraction;
4.      Encode one endpoint and mark 'coded';
5.      Mark other endpoint 'decimated';
6.      R_K:= bitrate after batch Φ^{p,q};
7.   Until R_K < R;
```

DECODER

```
1.   V_c := vertex marked 'coded';
2.   V_d := vertex marked 'decimated';
3.   Repeat:
4.      Decode V_c;
5.      If enough neighbours
6.         Predict V_d;
7.      else
8.         Contract V_d with nearest vertex;
9.   Until no more vertices received;
```

Figure 3.8: Rate control algorithm of the P3D-Anim codec.

after called QUANT). Figure 3.9 (left plot) shows rate-distortion curves obtained for sequence BOUNCEBALL whilst streaming with quantisation-based rate control and the P3D-Anim codec. Distortion, measured as the average mean square error between the original and the decoded frames for the whole sequence, is constantly lower in the P3D-Anim scheme, for a range of bitrates in [32..62] kbps. In the P3D-Anim case, the animation vectors were quantised constantly at 1024 quantisation levels, for comparison with the QUANT method. The labels on each curve represent, for the P3D-Anim scheme, intermediate values of removed triangles and, for the reference QUANT scheme, the corresponding quantiser levels.

It is interesting to note that at a bitrate approximately half that of the original, the distortion of the reference model is 48% higher than base, whereas the corresponding figure for P3D-Anim is only 27% higher, despite having removed over half of the triangles (180/320). This

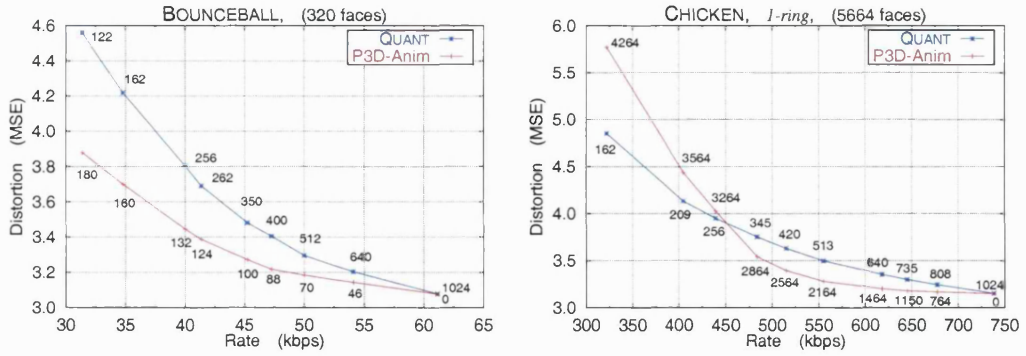


Figure 3.9: Rate-Distortion curves of quantisation-based against multiresolution-based rate control for the sequences BOUNCEBALL (left) and CHICKEN (right). The sharp deterioration of the P3D-Anim method in CHICKEN is due to insufficient prediction of un-coded vertices, which eventually get contracted.

represents an improvement for P3D-Anim of 45% over the reference.

The same simulation was conducted for sequence CHICKEN, with the results shown in the right of figure 3.9. This is a large mesh, allowing many more vertex pair contractions, and exhibiting a striking difference to BOUNCEBALL. Here, the P3D-Anim scheme performs better only down to a certain bitrate ($\approx$ 450 kbps), beyond which it sharply deteriorates. The reason is that the model of this sequence is not semi-regular with its vertices exhibiting various degrees, mainly between 4 and 13. After those vertices with the least degree become 'decimated' (un-coded) there is a sharp error increase since it is not possible to predict their geometry from their 1-ring neighbourhood, and these vertices are contracted with their nearest connected neighbour within a 2-ring. On the one hand, this contraction offers visual adaptation of the model's surface to a reduced number of vertices. P3D-Anim achieves the contraction of the un-coded vertices at the expense of geometric precision of their nearest neighbours, which are now fitted to a position mid-way between the endpoints.

Figures 3.10 and 3.11 subjectively compare frames from both the quantisation-based and the multi-resolution decoded sequence to the original sequence. The top row of each figure shows original frames of the sequences with the frame number below them. The middle row shows the corresponding decoded frames of the same rate, achieved by adapting the quantisation level. The quantisation level figure is also shown below the frames in the middle row. The bottom row shows the corresponding decoded frames, which have been received at controlled rates with the P3D-Anim scheme. These frames contain reduced geometry data, as a subset of the vertices has been eliminated from coding, or 'decimated'. The numbers below the P3D-Anim frames represent the number of faces that have been eliminated from the original model as a result of vertex decimation. Subjective evaluation suggests that coding artifacts due to

coarse quantisation are much more disturbing visually than the corresponding rendered models (frames) of the P3D-Anim adaptive sequences. The latter maintain the shape and volume of the original model, albeit at reduced resolution. This result holds for sequence CHICKEN regardless of the poorer P3D-Anim performance in objective terms, as pointed out by the average MSE in figure 3.9. The deteriorating numerical performance at reduced bitrates, exhibited by P3D-Anim in CHICKEN, is exaggerated by the large number of un-coded vertices, which amount to 2200 in the decoded frame 300 (corresponding to roughly 3/4 of the original sequence's vertices). The un-coded vertices contribute towards a larger average MSE, but this has hardly any visual effect, as the model maintains its shape and volume.

Motivated by the P3D-Anim results of sequence CHICKEN in figure 3.9, one would seek

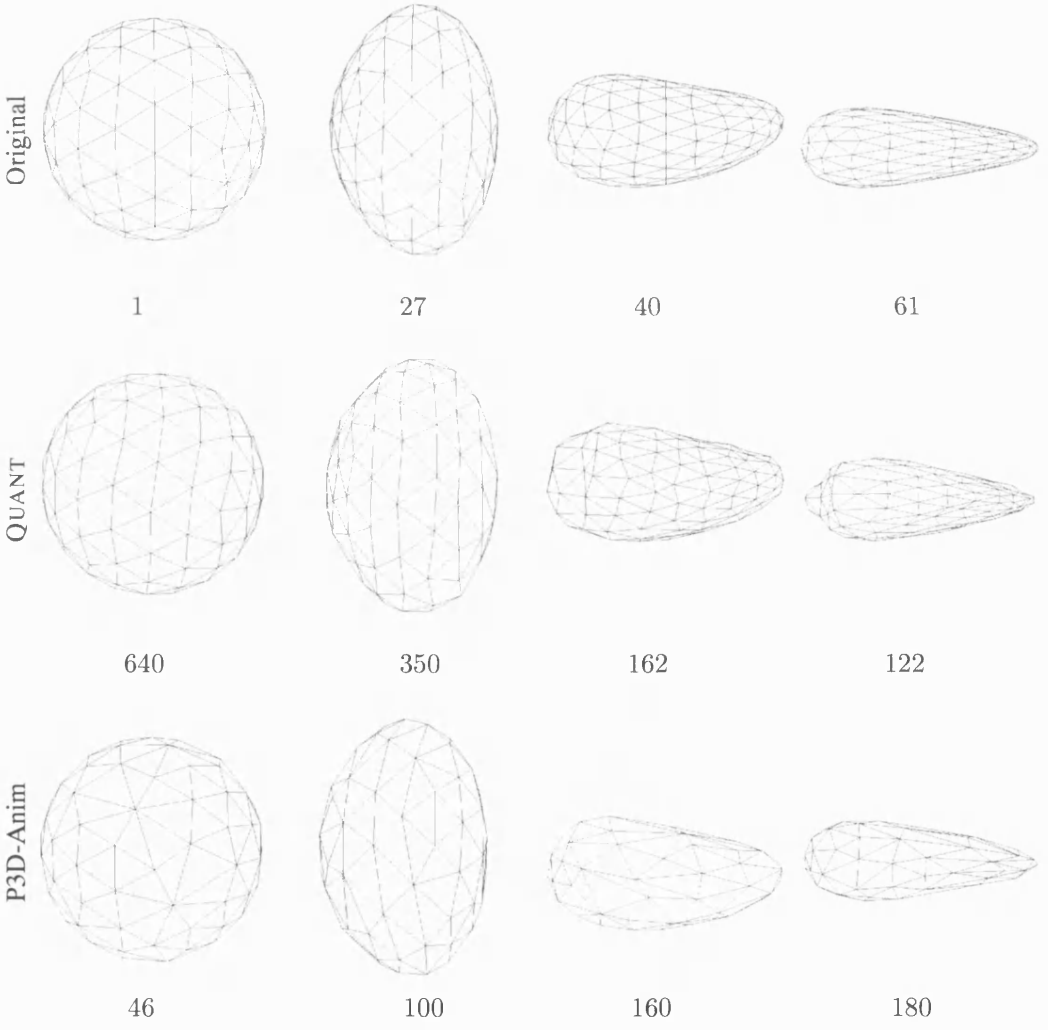


Figure 3.10: Snapshots from the dynamic sequence BOUNCEBALL as decoded by QUANT and P3D-Anim.

to improve the decoder's performance by introducing potentially more efficient motion estimation schemes. Based on geometric neighbourhood approaches, the 1-ring butterfly estimation scheme used in P3D-Anim can be extended to 2-ring estimation with the expectation of providing the sought performance improvement over the 1-ring. This setup has also been evaluated with P3D-Anim.

2-ring motion estimation comes into effect only after all 1-ring neighbouring vertices of a vertex v_i have been exhausted. Knowing that the vertex valence of sequence CHICKEN ranges mainly between 4 and 13, it is expected that the P3D-Anim decoder may resort to 2-ring estimation mode sooner for vertices with small valence than other vertices with higher valence. However, low-valence vertices for this particular sequence are few and are located in those areas of the mesh with higher vertex density (i.e., details of the head, legs). In these areas, using

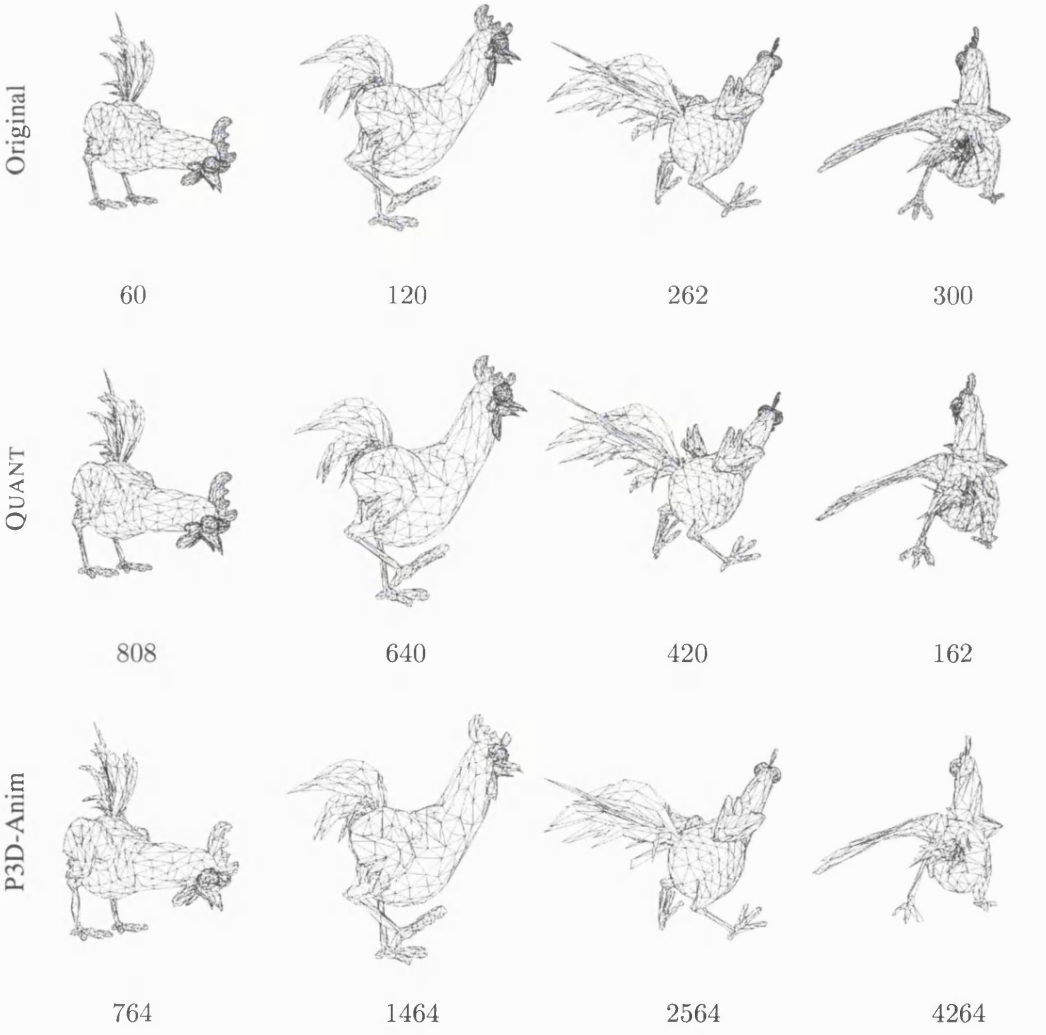


Figure 3.11: Snapshots from the dynamic sequence CHICKEN as decoded by QUANT and P3D-Anim.

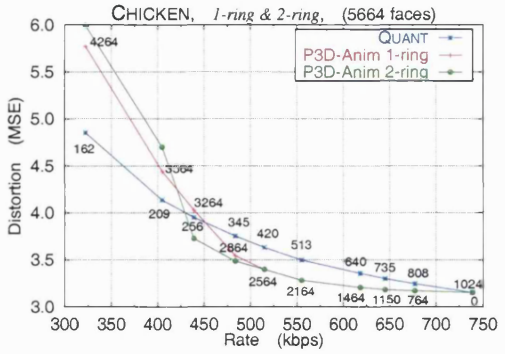


Figure 3.12: Rate-Distortion curves of quantisation-based against multiresolution-based rate control for the sequence CHICKEN with 2-ring motion estimation.

2-ring estimation does not noticeably distort the model's volume.

As can be seen by the evaluation in figure 3.12 the 2-ring method improves upon the 1-ring mode for a prolonged range of rates from 520 kbps down to about 430 kbps by providing better estimation for some motion vectors that would have otherwise been decimated. However, this improvement is also expected to be limited. The reason for this performance bound is that gradual geometry reduction (through vertex decimation in this case) will eventually lead to progressively lower model resolutions and rather rough approximations of the original mesh. Sequence CHICKEN is not manifold and 2-ring vector estimations may reach the model's boundaries where the highest distortion is likely to occur. This is expressed in CHICKEN's lower rates by worse 2-ring performance over the 1-ring mode. Nevertheless, P3D-Anim's 2-ring performance is near that of 1-ring and, at least for bitrates down to 430 kbps, subjective assessment would show equivalent visual performances.

As a final remark, the result of figure 3.12 suggests that the overall distortion D_{mr} in multi-resolution wireframe animation codecs, like P3D-Anim, can be expressed as a composite distortion measure of geometric prediction, D_g, plus distortion due to vertex decimation, D_d:

$$D_{mr} = D_g + D_d \ . \tag{3.16}$$

In equation (3.16), the term D_g is initially stronger at higher bitrates than D_d but, while progressing to low rates, D_d dominates, thus leading to high mesh volume distortions.

3.5 Internet Architecture for Wireframe Streaming

The importance of integrating QoS control into streaming applications has been previously recognised and justified for a variety of streaming media architectures. Most notably, existing architectures either associate QoS with a generic congestion control (CC) management service

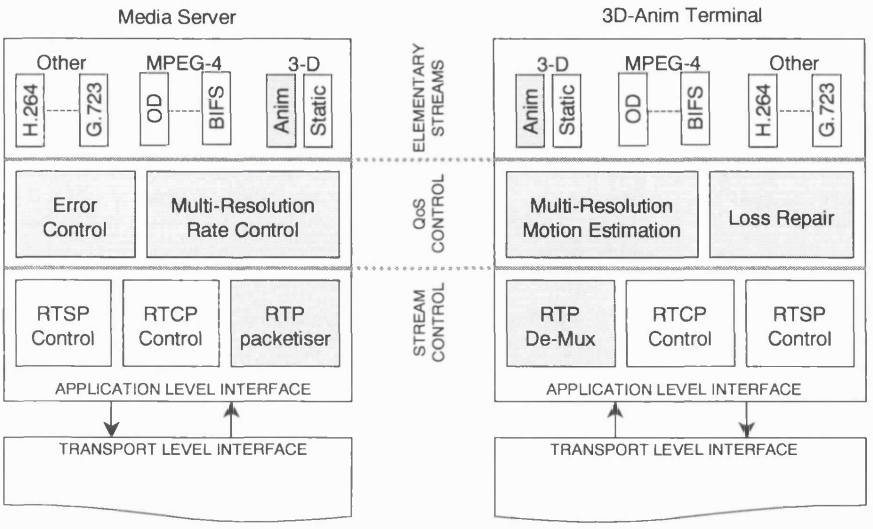


Figure 3.13: Block diagram of the Internet streaming architecture.

[Rej03, BRS99], or express QoS through a number of application-level provisions tailored to a specific medium, such as static 3-D graphics [RLS+02, Mar02]. Architectures of the latter category, however, do not integrate any existing CC mechanism of the former category properly, if at all, and rather provide their QoS mechanisms on an ad-hoc basis.

The 3-D streaming architecture described in this section enables QoS control in the application level and shows how QoS mechanisms can be integrated within a wider CC module. The block diagram is depicted in figure 3.13. The design is based on a generic streaming architecture for MPEG-4 [BVCL01, VBH00], which has been enhanced with the functionality of the greyed blocks, i.e., the 3D-Anim family Elementary Stream (ES) codecs and the QoS control module.

For adaptive 3D-Anim streams in particular, QoS-enabled streaming refers to the functionalities of multi-resolution rate control at the server side and multi-resolution motion estimation at the client terminal. The functionalities of these two sub-modules are those described in paragraphs 3.4.4.2 and 3.4.4.4. The combination of end-to-end multi-resolution adaptivity and error control is in accordance with the minimal QoS requirements of a 3-D streaming application, as defined in subsection 2.4.2.

Issues associated with 3D-Anim packet loss repair at the receiver are further analysed in chapter 4. The error control sub-module of the server side is the subject of chapter 5.

3.5.1 Application-level protocol considerations — RTP

The Real-time Transport Protocol (RTP) has been adopted for the carriage of 3D-Anim codec family data as it is developing as the common denominator in multimedia transport. RTP pro-

vides end-to-end network transport functions suitable for conventional multimedia applications transmitting real-time data (i.e., audio or natural video), over unicast or multicast network services. Transport of 3D-Anim is currently restricted to unicast although multicast transmission is not prohibited, but may raise media synchronisation, and service admission issues.

RTP represents a new style of protocol design philosophy shaped around the principles of Application Level Framing (ALF) and Integrated Layer Processing (ILP) proposed by Clark and Tennenhouse [CT90]. That is, *"RTP is intended to be malleable to provide the information required by a particular application and will often be integrated into the application processing rather than being implemented as a separate layer"* [SCFJ03]. This design philosophy, which does not address resource reservation and does not guarantee any level of quality-of-service, maps well to the design and provisions of the 3-D streaming architecture described earlier, where 3-D QoS is integrated into graphics applications.

3-D data transmission benefits from RTP since the protocol allows synchronisation with other media types for which a large base of payload format descriptions already exist. Furthermore, 3D-Anim RTP payload principles may influence the design of other 3-D streams payloads. For example, there is lack of transport and fragmentation rules for BIFS-Anim if carried over RTP as an independent stream, or for the now developing Lightweight Graphics and AFX elementary streams of MPEG-4.

3.5.1.1 Payload design

In its simplest form, one RTP packet would contain one D-frame or P-frame, which represent one Application Data Unit (ADU). This is in accordance with the ALF principle [CT90], which defines an ADU as the smallest unit of payload data that can be processed out of order with respect to other ADUs. In this sense, the codec's output bitstream is directly packetisable with RTP, whereby the boundaries of one ADU take the place of the RTP payload boundaries for purposes of end-to-end data manipulation.

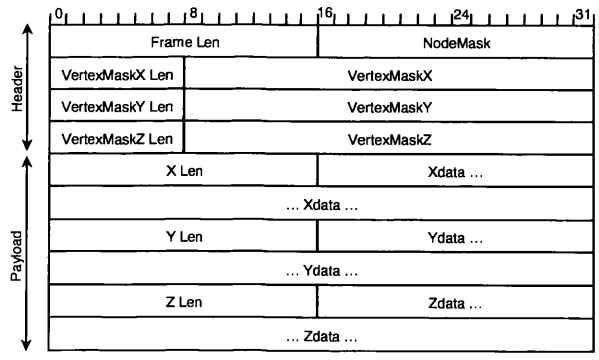


Figure 3.14: 3D-Anim stream generic packet format.

The 3D-Anim packet format design as an encapsulated RTP payload is shown in figure 3.14. The packet is split into *header* and *payload* parts.

The header part starts with the Node Mask and is followed by one or more Vertex Masks. Animation masks are kept together and placed in the header to facilitate control operations that may take place in the Application Layer. For example, the motion compensation feature of P3D-Anim and the associated rate adaptation algorithm rely on the animation masks to operate. Furthermore, the packet format specifies that uncompressed masks should be sent to allow for fast control and compatibility with MPEG-4's BIFS-Anim.

The header is followed by the 3D-Anim payload part, which contains encoded samples for x, y, and z axes in this order. The geometry components are separately packed to allow for the same payload format to be easily used by both 2-D and 3-D animated models. For example, 2D-Anim's bitstream can be accommodated simply by eliminating the z-axis component masks (from the header) and data (from the payload). The vertex mask and geometry data *length* fields serve as pointers for fast access to mask and geometry component data.

The above design has been kept simple and linear also to facilitate further extensions for attribute data (colours and normals) that may follow geometry data. The M bit in the RTP header must be set for the first of a series of empty RTP payload frames, which can also be grouped together. For wireframe models with up to a few hundreds of vertices, or detailed models with more than a few thousands of vertices but sparse animation, the resulting ADU size is short enough to fit in the path MTU of a wired Ethernet network.

3.5.1.2 Fragmentation and reassembly

Model sequences with higher density of animation would not be covered by the previous design without *grouping* and *fragmentation* rules. Special consideration of the path MTU limits are also required when the network is not wired end-to-end.

In such cases, the previous ADU definition is relaxed to accommodate the maximum size logical data units within an P-frame or a D-frame. The fragmentation rules are listed below.

1. In 3D-Anim, logical data units for individual processing are mapped to the node boundaries including the corresponding Vertex Masks.

2. The complete Node Mask, as defined in paragraph 3.2.2.5, must be encapsulated in the payload of the first RTP packet of the higher level ADU (P or D frame) along with the first unit of the fragmented ADU (first node of the Node Table).

3. For each subsequent fragmented ADU, a modified Node Mask is prefixed, in which there is only one bit set. This bit corresponds to the node being carried in the payload. Such an

arrangement allows each node to be processed as soon as it arrives at its destination and prevents the receiver from discarding a whole high-level ADU (frame) in case of a node loss.

4. Lost ADUs are considered stationary in the 3D-Anim sense, and therefore, reassembling the fractional ADUs before processing is not a requirement of the architecture.

The payload design described in this subsection allows for low delay streaming of 3D-Anim data and is suited to those applications with real-time needs. If higher delay is tolerable and there exist requirements for end-to-end redundant error control, a different payload format will be shown in chapter 5.

3.5.1.3 Rich media synchronisation

It was mentioned earlier in subsection 3.2.1 that D_t's dimension is reduced to $p \leq n$ compared to M_t, since it does not contain vertices for which the displacement on all axes is zero. This property is an advantage against MPEG-4's BIFS-Anim [Sig00], which does not allow for reduced animation frames. For sparse D_t matrices it may also be the case that a whole node is not animated thus allowing great animation flexibility and generating a scalable bitstream. Furthermore, in the case where $F_{i,t} = \mathbf{0}$, for all $i \in [1..l]$, the displacement matrix D_t is zero, leading to empty payload frames.

This property resembles the silence period inherent in speech audio streams and can be exploited in the application layer of RTP-based receivers to absorb network jitter. Inter-stream synchronisation can also be achieved, which is paramount for many applications (e.g., lip synchronisation of a 3-D animated virtual salesman with packet speech).

3.6 Conclusion

This foundation chapter, proposed and evaluated a coding scheme for 3-D wireframe models suitable for Internet streaming applications. The proposed compression scheme, 3D-Anim, leads to efficient coding of sequences of `IndexedFaceSet` structures, which hold geometric information of dynamic meshes, and generates single-resolution compressed animation data along with their corresponding bitmasks. 3D-Anim and its variant, E3D-Anim, exploit temporal coherence to achieve better compression than MPEG-4's 3DMC, which can only compress 3-D model sequences statically, on a per-frame basis. Compressed animation data are streamed to 3D-Anim-capable clients with a QoS-enabled 3-D model streaming architecture. The degree of QoS offered by the architecture at the Application Layer is that of adaptation to the available network bandwidth, through 3D-Anim's proposed multi-resolution variant (P3D-Anim).

Chapter 4

Repair Options

To date, almost all work within the field of 3-D animation has focussed on raising perceptual appeal to an adequate level. However, it is a non-trivial task to match the requirements of real-time streams representing tri-dimensional data to the reality of the Internet. One of the key problems that must be addressed is that of how to best conceal errors in the event of packet loss. This chapter proposes, models, and evaluates the effect of different possible concealment schemes under different conditions of loss. It concludes that frame insertion methods show good performance at low loss rates and loss patterns with isolated packet losses, but that frame interpolation methods exhibit better performance at both low and higher burst packet loss rates, though at the cost of added delay and complexity.

4.1 Introduction

In MPEG-4, the two early animation tools, BIFS-Anim and 'face-anim' were based on an adaptive arithmetic encoder that output a series of differential changes to scene components. This allowed for low delay coding. The remainder of this chapter will refer to 3D-Anim, as was described in detail in chapter 3; a zero-order prediction-based coder that arithmetically encodes vertex displacements of animated 3-D meshes that do not suffer topological changes through time. However, there is considerably more to realising a codec suitable for use over the Internet than this.

In order to design a codec suitable for widespread Internet use, it is just as important to take into account the likely channel characteristics as to consider what happens in the signal processing domain. In fact, it is most unwise to design the latter without having the former in mind. The best-effort Internet is a somewhat hostile environment; one must cope with packet loss, packet reordering and duplication, delay, delay variation (jitter) and even fragmentation. Consequently, network-awareness is very desirable. To take a simple example, packet loss on the fixed network is most often associated with congestion at routers and is a relatively common

occurrence in the wide area. Unless coding schemes are robust and unless the transmission rate can be adapted to take account of the congestion, perceptually significant problems will arise.

The network problems that arise from the best-effort nature of the Internet have been investigated and addressed over some period of time within the now established audio/video tools, such as vic and rat [uu00]. These tools use RTP [SCFJ03] with adaptive playout buffering algorithms to cope with variable delay. However, for the reasons outlined above, congestive loss is a reality, ranging from individual packet loss through burst loss to network outage. As a consequence, online loss parameter estimation methods (notably Bernoulli and 2-state Markov chain models) have been studied with the aim of enabling adaptive applications to monitor the network congestion levels and adapt encoder transmission rates. Given this possibility to estimate loss and monitor congestion, loss concealment or redundancy techniques have been devised [PHH98], along with suitable RTP payload formats, and network-friendly streaming protocols.

In the remainder of this chapter, we bring to bear the depth of experience in streaming media repair techniques for the Internet [PHH98] on the problems faced by novel media streams such as 3-D animation data in the same environment. We also study their performance for emerging applications and standards, such as MPEG-4. The suggested repair schemes may operate in tandem with network-friendly rate control mechanisms, such as the one described in paragraph 3.4.4.4.

4.2 Options for Repair of Animation Streams

Even given the obvious similarity in presentation, it is not obvious that perceptual effect of loss in a stream containing animation of 3-D models should be anything like the effect of loss in a stream containing natural video. This derives from three factors:

- video represents a 2-D projection of a 3-D scene, whereas the third dimension remains within animation data
- there is a fundamentally different underlying data representation, so a lost packet has a fundamentally different visual effect in each case
- synthetic animation data does not necessarily follow a naturalistic model in terms of chrominance versus luminance. The HVS, however, is tuned to the naturalistic model.

It is probable, therefore, for these reasons that the error sensitivity of animation data is significantly greater than that for video. For obvious reasons, much effort has been expended in developing realistic model rendering, colouring, texturing and animation techniques within the animation community. Likewise, recent 3-D animation coding attempts

[Len99, GSK02, AKKH01, AKKH02, YKL01, YKL02, AM00] focus on compression. In all, there is a small body of literature that addresses the problem of error concealment for animation data in the presence of packet loss. Based on the above remarks and past research experience with traditional multimedia coding, some of the possible options for robust animation coding are explored below.

4.2.1 Sender-based repair

A number of audio repair techniques require the participation of the sender of an audio stream to achieve recovery from packet loss. There are two major classes of sender-based techniques: active retransmission and passive channel coding. Channel coding, in turn, is further subdivided into interleaving and forward error correction (FEC). FEC data may be either *media-independent*, typically based upon parity operations and Reed-Solomon codes, or *media-specific*, based on the properties of a signal. Detailed information on this taxonomy specifically for audio streams is provided in [PHH98].

Clearly, the above taxonomy could equally well be applied to 3-D animation data, as the left half of table 4.1 shows. The sender side taxonomy is discussed below.

Retransmissions of large animation frames often imply retransmitting a plurality of packets which are likely to suffer variable delays in the sender-receiver path. Even in the case of small frames comprised of just a couple or few packets, delay variability (jitter) can be harmful for interactivity. Thus retransmission-based methods are likely to be too costly for animated data in the same way that they are for traditional multimedia streams. This is particularly true for 3-D animations where possible application areas include those with strong real time requirements, such as multi-user virtual worlds and distributed games, where excessive delays can not be tolerated.

Channel coding methods indirectly repair errors by allowing redundant information to be transmitted through the data channel, which the receiver can use appropriately to reconstruct data (packets or frames) that have been lost (or corrupted) during transmission. The following list of options is typical of channel coding methods for conventional multimedia streams that can also be applied to animation:

1. *Media-independent* FEC schemes do not depend on the contents of the packet and consequently can only attempt to ensure that the repair is an exact replacement for a lost packet. Such techniques are easy to implement and cheap to execute but, because they cannot take advantage of media-specific knowledge, they may be relatively expensive in terms of bandwidth consumed. This is because, for a constant overall stream rate,

Table 4.1: Summary of repair options for 3-D animation sequences.

SENDER-BASED			RECEIVER-BASED			
	Channel coding		Insertion		Interpolation	
Retransmission	Media-independent	Media-specific	Repetition	Prediction	Tracking curve	Parameter based

any media-independent FEC scheme requires corresponding reduction in the source rate thereby introducing source coding error to the stream. However, channel optimisation methods can be effective, especially for FEC schemes provided unequally to the packets (or frames) of the source signal, according to the perceptual significance of the packets in the animation stream. Media-independent FEC under an optimisation framework is discussed in chapter 5.

2. *Media-specific* FEC is an attractive proposition for 3-D animation streams, but suffers two drawbacks relative to media-independent FEC: it requires increased computational effort, which increases latency; and it requires the modification of animation codecs, which may be hard for existing 3-D animation codecs that rely on lower level technologies and specialised hardware. In its simplest form, however, media-specific FEC can use the same engine to produce both the primary and secondary encodings, at high and low quality respectively.

Finally, another repair method that could make part of sender-based channel-coding taxonomy is packet *interleaving*. Its potential benefits on generic animation though, as in the case of media-specific FEC, are unclear and will undoubtedly require extensive subjective evaluation tests before useful conclusions may be drawn. Consequently, media-specific methods and interleaving will not be investigated in this thesis.

4.2.2 Receiver-initiated concealment techniques

Receiver-initiated error concealment methods are desirable when either the sender of a stream is unable to participate in the recovery, or when sender-based recovery schemes fail to correct all loss. Receiver-based methods ameliorate the effects of lost data by attempting to produce something that is perceptually as close to the original as possible, often by taking advantage of the short-term self similarity of the signal.

For audio signals, these methods perform well for low to average loss rates ($< 15\%$) and for small packets ($4-40$ ms), since these figures represent audio signals shorter than the typical duration of a phoneme ($5 - 100$ ms). As the loss length approaches the length of a phoneme,

these techniques are believed to break down, since the listener may miss whole phonemes. In particular, loss of a phoneme positioned at the start of a word may result in significant perceptual loss.

The lack of a direct equivalent to phonemes in visual animation signals means that the error bounds for audio are unlikely to apply in the same way in this case. Thus the true perceptual effect of errors and its relationship to stream content, packet sizes and loss lengths for any given concealment technique can only be determined by experimentation. The first step in this is a re-examination of the generic techniques that are available for receiver-initiated error concealment.

In the case of 3-D animation with free face deformations, which pertains to the family of 3D-Anim codecs of chapter 3, the following categories for receiver-based concealment are identified, as shown in the right half taxonomy of table 4.1. In the description of the receiver-based concealment methods in this thesis it is assumed that one animation frame corresponds to one RTP packet in respect of the Application Level Framing principle. This is particularly true for animations of models with small number of vertices, or for sparse animations on larger models, or on layered animation data (described in chapter 5) where each layer's frames tend to be short. All previously described cases suggest that it is meaningful to evaluate the concealment methods at the frame level, where the assumption of one frame corresponding to one packet represents a rather worst case packetisation scenario. In general, any frame fragmentation approach could potentially be combined with other inter-frame concealment techniques (e.g., spatial interpolation), which would only improve the performance of receiver-based concealment methods. Section 4.4 will provide further reasoning on this.

Insertion based schemes repair losses by inserting a fill-in packet. Usually, the fill-in packet can be very simple, such as a repetition of the last successfully received packet. In zero-order prediction-based codecs, similar to the 3D-Anim developed in subsection 3.2.2, the decoded frame usually contains vertex differences between the model's current position and its previous position in 3-D space. The decoder, then, has several options:

1. To *repeat* a previously decoded frame 'as is' (Frame Repetition), in order to avoid erroneous motion on the model's faces. Good candidate frames are those from the short motion history. This is easy to implement, but requires additional memory at the receiver, especially in the case of loss bursts, where concealment could be based on repetition of multiple past frames to reach the animation sequence frame rate.

2. To *predict* the missing frame from the previously received one, thus *extrapolating* the motion of the missing frame (Motion Vectors). Prediction can be linear, in the case of a

single frame concealment, or non-linear, for consecutive frame drops. With this scheme the model preserves its motion vectors in an attempt to minimise the perceptual distortion of the animation. This technique is also relatively easy to implement, but on average it requires more buffering and has greater complexity than scheme 1.

Interpolation based schemes use the packets surrounding the loss to produce a replacement for a missing sequence of packets. These techniques account for the changing characteristics of the animation signal and, therefore, are well suited for longer loss bursts when the animation pattern is mainly directional instead of alternating. The options for repair at the receiver, using interpolation methods, can be categorised as following:

1. The receiver constantly updates a *tracking curve* of the model's animation path. In the event of loss, this curve is used to interpolate the position and predict the velocity of the animated vertices for missing frames, approximating a smoother animation path. For better results, a receiver could trade delay for quality, in order to collect a reasonable set of sampling points along the animation path, and employ non-linear interpolation and optimisation techniques. In its proposal for distributed animation in multi-user worlds, the SNHC group of MPEG-4 advocate a similar method in [ISO02] in conjunction with a dead reckoning principle algorithm. Although the proposed animation streams in [ISO02] are not frame-based, the ideas can be applied to error repair. Such techniques require increased processing power at the receiver and imply playout delays. Thus, they can be well applied to near real-time or stored animation avatars.

2. The receiver can apply more specific *parameter-based* recovery techniques. In special cases, where the animation exhibits some form of regularity or short-term stationary behaviour in the samples of consecutive frames, it is possible to describe the animation path piecewise using regressive analysis methods on the last received set of samples. A synthesis step during concealment, based on an estimate of motion parameters (e.g., velocity), will generate an approximation of the missing frames. These schemes may be hard to implement for animation and there is hardly any experience yet with regards to their performance.

Table 4.1 summarises the repair options on 3-D mesh sequences. The remainder of this chapter proposes and evaluates the performance of receiver-based concealment methods that follow the taxonomy of table 4.1.

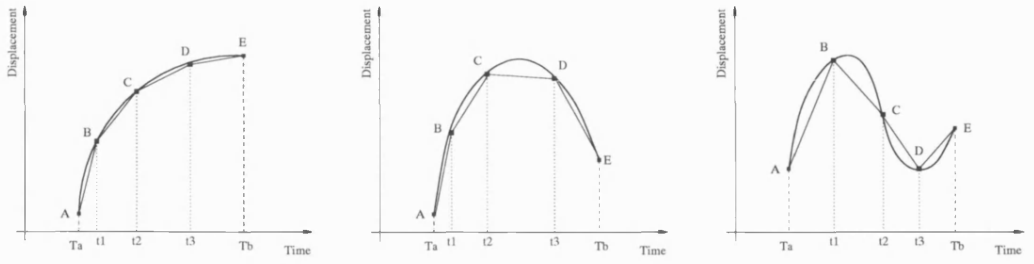


Figure 4.1: Three scenarios using linear interpolation in a 3-D animation sequence: (a) monotonic, (b) asymptotic, (c) free-form trajectory.

4.3 Receiver Concealment of Vertex-based Animation

The trajectory of animated vertices can take several forms depending on the content and the intensity of the animation. Figure 4.1 shows three possible scenarios of the trajectory, where it may (a) remain monotonic, (b) present a single peak, or (c) alternate, exhibiting local minima/maxima. When applying an error concealment method to time-dependent meshes the aim is to achieve a good approximation of the vertex animation trajectory for relatively short packet loss bursts (that are common in the Internet).

For a concealment scheme to be successful, it is assumed that the principle of *locality of vertex motion* holds true. According to this principle, the motion trajectory is unlikely to change much for a single component (x, y, or z) within a short time interval. In formal notation, the trajectory functions F_x, F_y, F_z remain monotonic within time interval $[T_a..T_b]$. The visual result of assuming monotonicity in an animation trajectory that is non-monotonicity would be seen as artifacts on the mesh surface. Depending on the error concealment scheme in effect and the packetisation scheme used these artifacts can take the form of:

(i) *blips* on the rendered surface, if vertices of one sub-mesh are split between different packets, or

(ii) irregularly *enlarged*, or *shortened* sub-meshes, if all vertices of one sub-mesh are contained in the same lost packet(s).

Such artifacts, however, are almost invisible in high frame-rate animation streams.

In the following subsections, three receiver-based error concealment schemes covering the taxonomy of table 4.1 are proposed and their suitability for different loss patterns and scenarios is analysed. The performance of the schemes under variable loss conditions are further evaluated in section 4.4.

4.3.1 Frame repetition (FR)

A simple form of frame concealment, following the notation introduced in 3.2.1, is to repeat the vertex positions of the last received frame in the current missing frame, as soon as it is detected missing. Formally, if the j-th vertex of frame k is lost, its position $v'_{j,k}$ can be estimated as:

$$v'_{j,k} = v_{j,k-1} \ . \tag{4.1}$$

where $v_{j,k-1}$ is the position of the corresponding properly received vertex v_j in the previous time instant $k-1$.

The advantage of this simple concealment method is that it only depends on the previously decoded frame, which in all cases could be maintained in the receiver buffer and be reused before it is discarded, and does not require any additional calculations at the receiver.

As described earlier in sections 3.2 and 3.3, the 3D-Anim codec can tag those vertices with insignificant displacement over the previous time instant, without needing to repeatedly encode all the details. The simple FR scheme will perform well if such *stationary* vertices dominate the model when loss occurs. This situation is likely to arise in sparse animations suffering isolated losses on D-frames where the majority of a frame's vertices are either not animated or where they only exhibit very small displacements from the previous frame. Then, in the event of a single D-frame loss, FR can perform accurate concealment of the missing frame at very low computational cost. Section 4.4 evaluates and further discusses the performance of the FR method in comparison to the other proposed methods following this subsection.

4.3.2 Motion vector repetition (MV)

In the previous subsection, equation (4.1) implicitly assumes the majority of vertices to be stationary. Intuitively, this is not the optimal choice if non-stationary vertices form the majority of a time-dependent mesh, so that most or all of $v_{j,k}$ vertices in frame k are animated. In fact, by definition of the proposed 3D-Anim scheme in subsection 3.2.2, stationary vertices are excluded from coding and the corresponding Vertex Mask bits in the RTP payload described in subsection 3.5.1 are reset. This is not the case in dense, or complete, animations.

An improved alternative in such cases is to assign motion to each vertex of frame k equal to the motion in the previous frame $k-1$, expressible as:

$$v'_{j,k} - v_{j,k-1} = v_{j,k-1} - v_{j,k-2} \ . \tag{4.2}$$

The right part in the above equation represents motion of vertex v_j from time $k-2$ to time $k-1$. Let $\vec{\mathbf{V}}_{j,k-1}$ denote this motion vector in the 3-D space, and equivalently let $\vec{\mathbf{V}}'_{j,k}$ represent the

estimated motion vector of the left part of equation (4.2), where $v'_{j,k}$ is the estimated position of vertex v_j at frame k. Then, equation (4.2) can be re-written in a simpler form as:

$$\vec{\mathbf{V}}'_{j,k} = \vec{\mathbf{V}}_{j,k-1} \ . \tag{4.3}$$

The case previously described implies the assumption that vertices are moving with *constant velocity*. (Note how in the special case of zero velocity, $\vec{\mathbf{V}}_{j,k-1} = \mathbf{0}$, equation (4.3) captures the FR method). Equation (4.3) can be augmented to account for the average *linear* estimation of $\vec{\mathbf{V}}_{j,k}$ from K past motion vectors:

$$\vec{\mathbf{V}}'_{j,k} = \frac{1}{K} \sum_{i=1}^{K} \vec{\mathbf{V}}_{j,k-i} \ . \tag{4.4}$$

The concealment scheme described by equation (4.4) linearly estimates a missing vertex position $v'_{j,k}$ assuming constant velocity for the motion vectors. Non-linear approaches lead to vertex geometry estimation after loss assuming *constant acceleration* of the motion vectors, expressible as:

$$\vec{\mathbf{V}}'_{j,k} = \gamma_{j,k-1} \cdot \vec{\mathbf{V}}_{j,k-1} \ , \tag{4.5}$$

where $\gamma_{j,k-1}$ is the acceleration of vertex v_j at frame $k-1$, calculated by the ratio of the norms $|\cdot|$ of motion vectors $\vec{\mathbf{V}}_{j,k-1}$ and $\vec{\mathbf{V}}_{j,k-2}$ at frames $k-1$ and $k-2$ respectively:

$$\gamma_{j,k-1} = \frac{|\vec{\mathbf{V}}_{j,k-1}|}{|\vec{\mathbf{V}}_{j,k-2}|} \ . \tag{4.6}$$

The acceleration-based concealment may provide good estimates for $v'_{j,k}$ in those particular situations where the vertex motion is not linear. However, the value of this concealment method depends on the temporal characteristics (content) of the animation. For example, considering the case where motion is low for vertex v_j in frame $k-2$ and which sharply accelerates thereafter, the acceleration ratio $\gamma_{j,k-1}$ may tend to infinity, thus yielding poor estimates for $v'_{j,k}$. This situation is not unlikely to occur at the end of a GOV sequence of animation frames, where the DPCM coding method may exhibit higher average distortion (error vectors) during a coding transition from a D-frame to an P-frame. Although this transition in frame coding aims at refreshing the error vectors that will now be coded relative to the reference model, this error vector correction is expected to result in unstable γ's. The symmetric case at the beginning of the GOV (with P to D frame coding transition) may have similar effects. Thus the performance of MV repetition in such cases is unpredictable and can only be evaluated heuristically at the expense of additional calculations. In most cases, the constant velocity approach is computationally simple yet with sufficient performance.

From the above analysis, it now becomes clearer that a frame concealment method based on motion vector repetition is suitable for small burst losses on D-frames where there is no transition from a P to a D frame, and possibly on sparse individual losses on P-frames, and thus provide an improvement over simple FR methods. This is expected to be more apparent in cases where the P frame refresh rate is low and when considering dense or complete animations. The relative performance of MV concealment verifying the above is shown in figure 4.5 and is discussed in the following section 4.4.

4.3.3 Linear interpolation (IN)

Both previous methods attempted to estimate the position of a vertex $v'_{j,k}$ at a frame k by using only information from the near past $(k-i)$ of the previous i frames. It can be intuitively inferred that neither FR or MV require additional latency in order to be effective; a single frame loss can be directly detected and concealed using the previously received frame.

It has also been argued that, assuming frame $k-i$ has been received without errors, small and sparse bursts (i=2 at low loss rates) can be adequately concealed with low latency using MV repetition (or FR in certain situations) from frame $k-i$ to $k-i+1$, and subsequently the predicted vertex positions $v'_{j,k-i+1}$ in frame $k-i+1$ can be reused to estimate motion towards frame $k-i+2$ (or k, if i=2). However, this consecutive use of MV repetition may lead to the undesirable situation of overestimating motion, in particular for sparse sequences, where mostly stationary vertices may be overly displaced.

To overcome these situations and yet provide robustness against burst losses, a concealment scheme can trade latency for increased geometry precision of predicted vertices. This can be achieved with a bidirectional temporal interpolation scheme between two frames which are spaced T frames apart, and it can be described with equation (4.7) and gradation factor τ as:

$$\vec{\mathbf{V}}'_{j,k+t} = (1-\tau)\cdot\vec{\mathbf{V}}_{j,k} + \tau\cdot\vec{\mathbf{V}}_{j,k+T}, \quad \text{with} \quad \tau = \frac{t}{T}, \quad 0 < t < T\ . \tag{4.7}$$

With the above equation the geometry of vertices $v'_{j,k+t}$ belonging to intermediate frames $k+t$ can be estimated, which may have been partially or completely lost due to a large loss burst spanning frames k+1 to k+T−1 (see example in figure 4.2). The gradation factor τ weights the impact that the vertex geometry of the endpoint frames has on estimated frames (k, ..., k+T), in such a way that for interpolated vertices $v'_{j,k+i}$ closer to frame k the geometry of $v_{j,k}$ dominates over $v_{j,k+T}$ and vice versa. It is assumed that all vertices belonging to the endpoint frames have been previously received without loss. It is also assumed that the animation system can tolerate latency of up to T frames. Figure 4.2 depicts the case of linear interpolation on the wireframe CHICKEN after a loss burst of 4 frames.

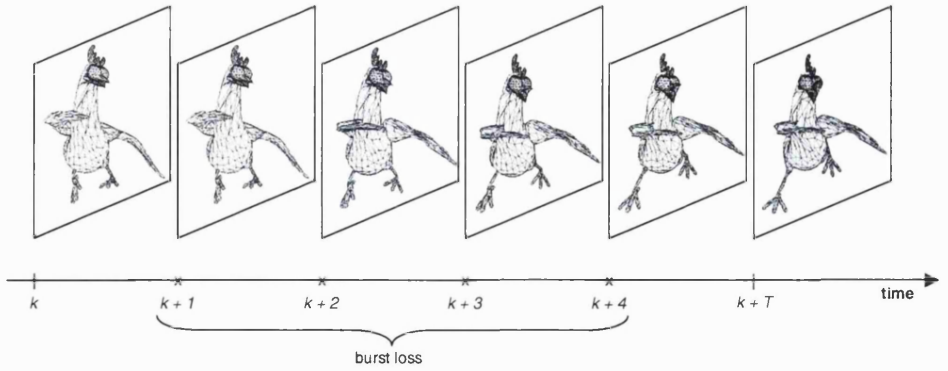


Figure 4.2: Linear interpolation applied to a hypothetical loss burst (wireframe CHICKEN with k=295 and T=5). Frames 295 and 300 correspond to the original uncompressed wireframes of CHICKEN, whereas the intermediate frames 296 – 299 have been interpolated from the above endpoints.

The interpolation scheme described above provides good estimates of the lost frames geometry even in the case of sparse animations, where some vertices are animated while others remain stationary. Since there is no computationally cheap way of predicting the performance of FR or MV methods in sparse animations, the proposed IN scheme can be used in all situations where frame concealment is required, either due to isolated or long burst losses. It is worth noting that even in the case of isolated loss, the IN method incurs one frame delay, which is not the case for FR or MV methods.

The above formalisation of equation (4.7) for vertex-wise linear interpolation in 3-D meshes resembles to a large extent the idea of geomorphs formalised by Hoppe [Hop96]. The motivation in Hoppe's expression of geomorphs is to achieve smooth transitions between a mesh representation M^i and its forward or backward representations M^{i+1} and M^{i-1} produced with a vertex split or edge collapse operation respectively. Geomorphing ensures that no visible 'snapping' occurs between such transitions, as figure 4.3 shows.

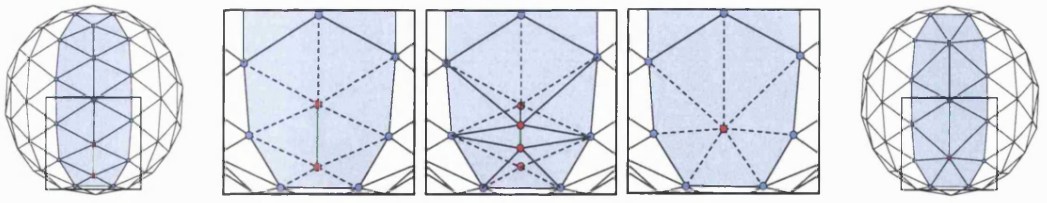


Figure 4.3: Snapping avoidance by *geomorphing* on wireframe BOUNCEBALL. The leftmost snapshot corresponds to the original mesh with 320 faces (M^{320}) at frame 24 of the wireframe sequence. The rightmost snapshot corresponds to the backwards representation M^{300} of the same frame, after decimating M^{320} with 10 edge collapses. The three intermediate frames show the geomorphing process applied to the marked rectangular area of interest.

In a similar way, the proposed interpolation concealment scheme tries to avoid such visual effects of snapping when a sequence of animation frames suffers burst loss. Consecutive missing frames can be reconstructed by 'smoothing' the distance between the last received frame at time $k+T$ and the previously received frame at time k, with τ being mapped to the *blend parameter* of geomorphing. Such techniques are typical of frame-rate up-sampling in natural video.

From the previous reasons of smoothing, and due to the locality of reference principle discussed earlier, it can be intuitively inferred that interpolation-based methods exhibit very good repair capabilities for burst frame errors. Even if the average burst loss length $\bar{b}$ is large enough ($\bar{b} \approx 4$), the proposed interpolation-based concealment achieves good visual transitions between the endpoint frames, not only masking potential snapping effects, but also achieving good estimations of the intermediate vertex positions. In particular, where the vertices maintain a relatively constant velocity over small frame spans and regardless of their direction the average objective error to the original mesh position remains small. This performance, as captured by the 3-D PSNR metric, is discussed in the following section 4.4, highlighting the effectiveness of this method over FR or MV for recovering burst frame losses.

The increased performance comes at the expense of some computational effort and increased latency. Typical values of $\bar{b}$, however, in the wide area of the wired Internet are in the order of two packets ($\bar{b} = 2$), thus not requiring extensive interpolations. However, if the sender-receiver path involves a wireless link, loss of a single lower-layer data segment may result in a larger application-level loss burst where interpolation-based concealment can be applied.

4.4 Experimental Evaluation

In this section, experimental evidence on the relative performance of the concealment methods proposed in the previous paragraphs is presented. The experiments covered here focus on receiver-initiated techniques described in the previous section 4.3. Sender-based mechanisms are further discussed and evaluated in chapter 5.

4.4.1 Packet loss model

As reported by Bolot [Bol93], the short-term bursty loss process of the Internet is quite complex, but it can be closely approximated by a 2-state Markov model (known as the Gilbert model). Sanneck [San00] not only confirmed this claim, but further studied the packet loss process in the wired Internet for streaming applications and suggested that medium and long-term outages can be also described by higher-level models.

The model derived by both these citations is depicted in figure 4.4. The two states are state **G** (good), where packets are timely and correctly received, and **B** (bad), where packets are either lost or delayed to the point that they that can be considered lost. The model captures a burst loss when it remains in the bad state, **B**, for multiple consecutive packets. The average burst length is denoted by $\overline{b}$. The conditional state transition probabilities p_{GB} and p_{BG} can fully describe the model but, since they are not sufficiently intuitive, the model can be better expressed using the unconditional average loss probability P_B (or probability of being in state **B** on average, $Prob(\mathbf{B})$), and $\overline{b}$, as:

$$P_B = Prob(\mathbf{B}) = \frac{p_{GB}}{p_{GB} + p_{BG}} , \tag{4.8}$$

$$\overline{b} = \frac{1}{p_{BG}} . \tag{4.9}$$

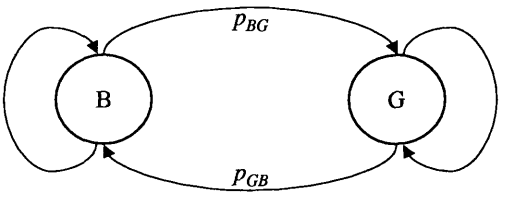


Figure 4.4: Gilbert model for packet loss simulation.

Equations (4.8)-(4.9) with $\overline{b} \in [2..4]$ tend to capture well the short-term burst loss patterns of the Internet. Where experiments with packet loss are described in this thesis, the model was repetitively run in order to generate loss patterns with average loss rates, P_B, up to a high percentage. The loss patterns can be generated in either *cumulative* or *non-cumulative* way. Cumulative means to keep the same loss pattern P_k between two successive loops k and $k+1$ (with respective loss rates R_k and R_{k+1}), as a deterministic process, and drop additional packets from P_k, until the requested loss rate R_{k+1} is achieved for loop $k+1$, where $R_k < R_{k+1}$. Experiment details and parameters like these are described individually, where necessary.

4.4.2 Evaluation and results

For the evaluation, a number of animated wireframe models are used: TELLY, FRED, ROBOT, BOUNCEBALL, and CHICKEN. Results are discussed here for the first three wireframes. A description of these models' characteristics (such as number of vertices, nodes, frame frequency, number of frames, and other features) is given in appendix A. For the following experiments, all sequences have been encoded either with P-frames at various regular positions, namely at multiples of 8, 15, or 30 frames (denoted in the graphs by P/8, P/15, P/30 respectively), or without P-frames apart from the first frame in the sequence (denoted in the graphs by P/0). Sequence

TELLY contains *sparse* animation data in accordance to the definition of equation (3.8) in section 3.3, as opposed to all other sequences' *dense* animation. In all sequences, the number of quantisation levels was maintained constant at 11 bits. This is to guarantee uniform effect of quantisation across all sequences when measuring distortion at the receiver, so that the effect of lossy source coding does not mask the distortion effects introduced by packet and frame loss, which are the focus of the experiments in this section.

It is assumed that one encoded animation frame fits into one RTP packet, following the design of subsection 3.5.1, with respect to the path MTU. (The value of the Internet path MTU is roughly 1500 bytes.) In fact, this is true for sequences TELLY, BOUNCEBALL and FRED. CHICKEN and ROBOT are larger models for which frame fragmentation is required. However, the assumption that no frame fragmentation takes place reflects the worst case scenario where whole animation frames get lost. For small animated models or sparse animations on larger models (such as localised effects and surface deformations), having one frame being contained in a single packet is a very realistic case. Equally realistic is the case where the frame may consist of a subset of nodes only. In addition, the animation system described in section 3.5 caters for *layered* streaming of time-dependent wireframe meshes (details of which can be found in a later chapter, section 5.3). In short, wireframe layering is achieved with node segmentation, which leads to the design choice of one node (or node segment) per packet. All previously mentioned cases suggest that it is meaningful to evaluate the concealment methods at the frame level. The results discussed in this section refer to frames that contain a single node. In general, any frame fragmentation approach could potentially be combined with other inter-frame concealment techniques (e.g., spatial interpolation), which would only improve the performance results presented in this section.

Burst packet losses were simulated with the 2-state Markov model described in subsection 4.4.1. Using this model, various non-cumulative loss patterns were generated and applied to the sequences within the packet loss percentage range [0..30%]. The graphs presented in figure 4.5 were generated by averaging experimental results over 60 loss patterns for FRED, and 40 patterns for TELLY and ROBOT. This was done to eliminate any bias in the results occurring from the loss pattern. For example, as will be mentioned below, some loss patterns may hit P-frames more than other patterns, thus slightly increasing the measured distortion. In the remaining text describing the experiment, distortion is measured as 3-D PSNR, which is calculated as the average Euclidean Distance between the original and coded sequences. This was defined in [VOH01] and discussed in paragraph 2.5.3.5. Note that in these experiments a 3-D PSNR value is calculated per node using equation (2.16) with $N=1$.

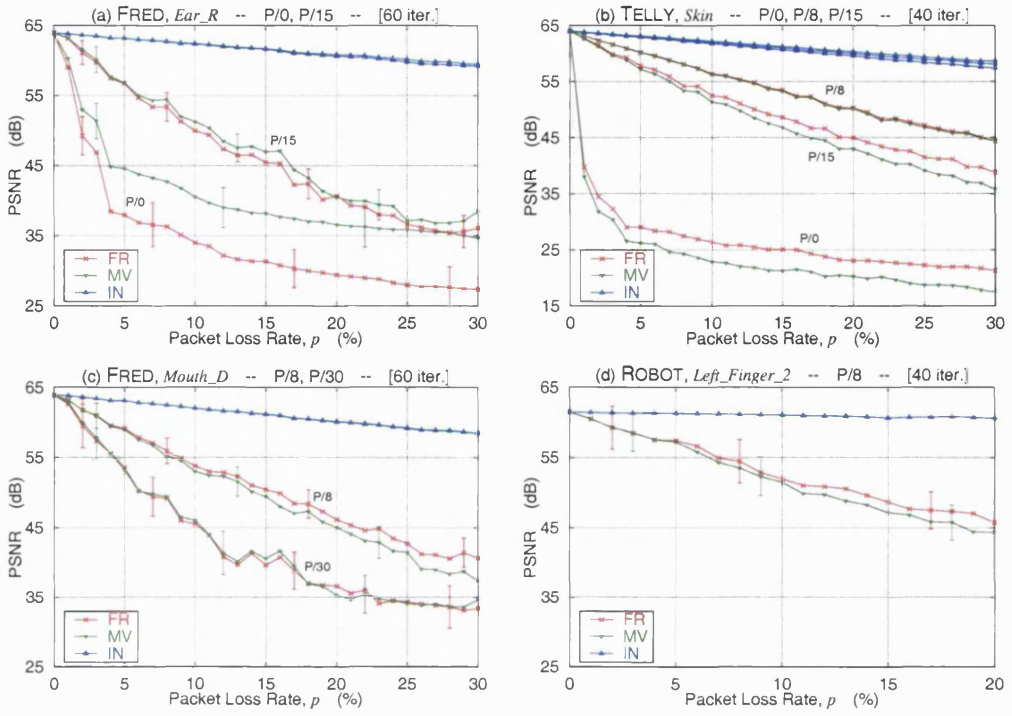


Figure 4.5: 3-D PSNR versus Loss Rate for different sequences, nodes, and P-frame combinations.

Figure 4.5 shows the PSNR achieved after repairing with either of the three proposed methods, namely Frame Repetition, Motion Vector repetition, and Interpolation, referenced as FR, MV, and IN, respectively. As the encoding has been iterated over a large number of loss patterns, the plots show the average performance with min-max error bars. At first glance, the graphs verify the expected decreasing trend of PSNR as loss rate increases, highlighting the relative performances of each repair method. Certain sudden PSNR drops or peaks are visible mainly expressing the effect of loss on P-frames; these drops are in the order of 1-2 dB for sequences with P-frames, or less in sequences without P-frames (denoted by P/0), as can be observed in sequence FRED with P-frames inserted every 8, 15 and 30 frames in graphs (a) and (c). The anomaly is due to the non-cumulative loss patterns, as has been explained in subsection 4.4.1, which result in random P-frame losses.

From plots (a)-(c) it is evident that MV performs better than FR in FRED for P-frames inserted up to every 15 frames. When P-frames are inserted every 8 frames, the relative performance of MV and FR is already reversed for node EarRight (a), suggesting that the frequency of P-frames is important to the relative performance of the repair techniques. This behaviour is observed in all nodes of sequence FRED with a swap-in performance at P/10 for the majority of nodes.

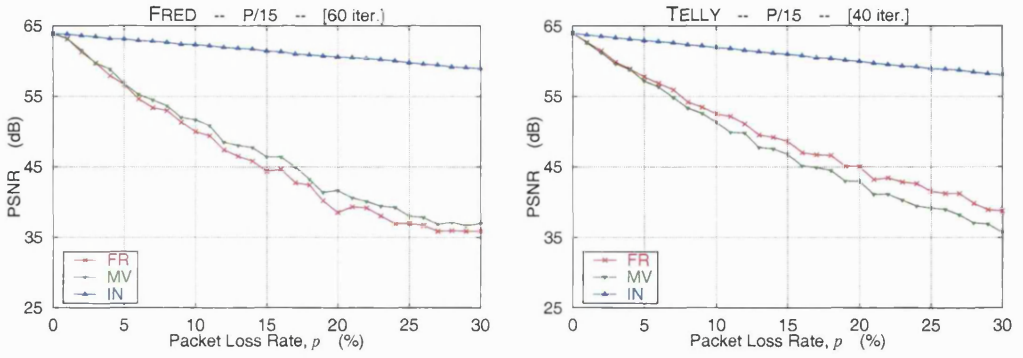


Figure 4.6: Mesh-wide 3-D PSNR versus Loss Rate for sequences FRED and TELLY.

Observation of TELLY data in plot (b) shows inverted results, with performance swap threshold of P/5. In TELLY, sparse vertex animation takes place, and different vertices are animated from one frame to the next. Repeating the previous frame means animating the vertices present in the previous frame only, in which case FR performs better than MV even in the absence of P-frames. This evidence suggests that FR tends to be more suitable for sparse animations where vertices tend to be stationary on average.

Knees are also observed early at around 4% loss in the PSNR curves without P-frames (P/0) in plots (a) and (b). This observation validates the intuitive result that lack of P-frames in the encoded sequences sharply deteriorates their animation quality in the event of frame losses.

Plot (d) shows the same experiment performed on ROBOT for a shorter loss rate range, where the loss patterns have been edited (here called *biased* loss) in order to minimise the effects of loss on P-frames. In the range shown, the relative performance of FR and MV is as in TELLY, but without observing any P-frame frequency threshold. This sequence also validates the effect of loss on P-frames, as was observed earlier in sequence FRED by sharp drops or peaks on PSNR. In ROBOT, where the loss pattern is biased towards not dropping P-frames these effects are hardly, if at all, noticeable.

It was mentioned earlier in this section that results obtained as 3-D PSNR in the graphs of figure 4.5 have been shown per node. Mesh-wide plots would have not made easily apparent the effects of loss on P-frames. Furthermore, the swap-in performance of MV and FR could have been slightly masked by the averaging of node PSNR in equation (2.16). Figure 4.6 shows mesh-wide PSNR measurements for models FRED and TELLY.

All plots show that IN outperforms FR and MV concealment methods. In all graphs, the IN plots lie very close together and increasing the P-frame frequency in the animation sequence contributes little PSNR gain. Interpolation is, therefore, the preferred frame conceal-

ment method irrespective of the loss pattern or P-frame frequency, but it achieves this performance at the expense of additional computational load and increased latency. As can also be seen in all plots, IN hardly exhibits any PSNR anomalies (sudden drops) pointed out earlier for FR and MV. IN will fail altogether for loss bursts that span a longer packet duration than the maximum frame latency allowed at the receiver, but such bursts are unlikely to occur unless they are the consequence of a disconnection, about which one can do nothing.

With this knowledge of the performance of the receiver-based concealment methods over different loss patterns and P-frame placements, a wireframe animation streaming system may be tuned to cope effectively with loss. As these methods are completely receiver-based, and the receiver is capable of monitoring network conditions and the animation content, a method that is expected to perform better under the experienced conditions can be chosen for repair in cases of sustained packet loss.

4.5 Conclusion

This chapter focussed on receiver-based error concealment options for 3-D animations streams, by studying the effects of packet loss on 3D-Anim streams. It developed three concealment schemes, evaluated their performance under conditions of burst packet loss, and inferred network loss behaviour scenarios where each method is more appropriate. The chapter concludes that frame insertion methods show good performance at low loss rates and loss patterns with isolated packet losses, but that frame interpolation methods exhibit better performance at both low and higher burst packet loss rates, though at the cost of added delay and complexity.

Chapter 5

Optimised Error Resilient Streaming of 3-D Wireframe Animations

This chapter addresses the problem of how to achieve optimal resilience of dynamic 3-D models to channel errors in terms of the perceptual effect at the receiver by providing data redundancy at the sender. The perceptual effects of packet loss on 3-D wireframe meshes have been poorly addressed in the context of animation to date and much of the work that there has been in this field has relied on objective measures such as PSNR in lieu of those that take subjective effects into account.

In order to achieve a degree of error resilience for animated 3-D wireframes, concepts from the related field of video streaming are brought together. In particular, this chapter introduces options for organising the animation stream into a number of layers. The independent application of Reed-Solomon (RS) forward error correction (FEC) codes to each layer is then proposed. FEC codes are applied *unequally* to each layer, in such a way as to maintain the same overall animation bitrate whilst minimising the perceptual effects of error, as measured by the visual smoothness metric presented in section 5.5.

5.1 Introduction

To date, the most prevalent representations for 3-D static models are polygonal or triangle meshes. These representations allow the approximation of models of arbitrary shape and topology within some desired precision or quality. Efficient algorithms and data structures exist to generate, store, modify, compress and transmit such static meshes, and state-of-the-art algorithms for encoding were presented in section 2.2. Enhanced, non-static, 3-D media stream types that introduce the time dimension, such as those generated by the encoding methods of section 2.3, would require scalable solutions to survive with respect to the network's limited resources (bandwidth) and characteristics (channel errors).

The previous chapter introduced the classification of receiver-based and sender-initiated methods for error resilience, and proposed and evaluated receiver-based schemes. The classification was summarised in table 4.1. This chapter focuses on sender-initiated techniques for error resilient streaming of time-dependent 3-D meshes. The proposed techniques in this chapter are optimised with respect to network bandwidth and consider the bursty loss nature of the channel.

In subsection 4.2.1, the classification of sender-based redundancy distinguished between media-independent and media-specific error resilience. According to the discussion in that subsection, redundancy methods that are called media-independent do not depend on the content of the packets in order to provide redundant packet-replacement information. The redundancy methods proposed in this chapter are generally considered to follow the media-independent trend. However, the proposed redundancy methods also assume that the animation data have been organised into packets in a layered fashion, as elaborated in section 5.3. Some of the proposed layering methods rely on ranking or ordering of the vertex position data according to a distortion quantity, which, in the cases considered by this chapter, is the smoothness of the mesh surface. In this sense, by ordering the vertex data (content of the packet) one could arguably consider any redundancy method applied to such layers as being media-specific. Despite the validity of this argument, the error resilience process does apply redundancy to the packets of each layer independently; thus, this chapter maintains the distinction between media-independent and media-specific resilience and adheres to the former paradigm.

The problem of 3-D wireframe animation streaming addressed in this chapter can be stated as follows. Assume:

(i) a time-dependent 3-D mesh has been scalably compressed as a sequence of wireframe animation frames,

(ii) the available transmission rate R is known (or determined with respect to the corresponding TCP-friendly rate),

(iii) the channel error characteristics are known, and

(iv) a fraction R_C of the available transmission rate ($R_C < R$) can be reserved for channel coding.

In a media-independent approach, the optimal number of bits is initially sought that can be allocated *unequally* to each level of importance (layer) in the animation scene so that the perceived quality of the time-dependent mesh at the receiver is maximised.

Section 5.2 explores the motivation for researching sender-based methods of error resilience for 3-D animation streams. Section 5.3 discusses layering options for animated wireframe meshes. Section 5.4 describes the media-independent error correction codes used for providing redundancy to the appropriately packetised output bitstream of the 3D-Anim codec described in section 3.2. The channel error model is also analysed in relation to the redundancy codes and the proposed unequal error protection method is presented in a formal expression that leads to solutions yielding optimal smoothness. Analysis and discussion of experimental evidence is given in section 5.7.

5.2 Background and Motivation

The following subsections serve as background to FEC-based error protection schemes for layered streaming media. A review of this background outlines the motivation for the design of error protection solutions tailored to animated 3-D wireframes.

5.2.1 Layering streaming media

From the point of view of the network, organising streamed data into multiple layers (in the general sense) is a mechanism that has been used primarily by natural video transmission systems to address two distinct problems:

(i) Client bandwidth *heterogeneity* and the related problem of network congestion and,

(ii) Channel *unreliability* due to packet loss.

With the goal of achieving adaptivity to client bandwidth heterogeneity and the network congestion level, several approaches in the literature have exploited the properties of cumulative layered coding of a continuous source (mainly video) efficiently to transmit to clients at different bitrates. To achieve efficiency and scalability, the paradigm of layered source data coding is assisted by multicast distribution to a potentially large set of heterogeneous receivers. Most notably, in Receiver-driven Layered Multicast (RLM) by McCanne [McC96], low-bandwidth clients may choose to receive only a base layer by joining a single multicast group, while high-bandwidth clients would receive both a base layer and a multitude of enhancement layers at will by continuously monitoring the reception quality and network conditions and accordingly joining or leaving multicast groups that receive enhancement layers.

However, quality issues have been raised (Rejaie, [Rej99]) for layer-switching streaming video such as RLM and the overheads associated to cumulatively layered encodings have been highlighted by Kim and Ammar [KA01] and de Cuetos et al. [dCSR01] among others. The

latter two provide simulated and experimental evidence of the overheads, and advocate the replication of streams, or 'multiple stream versions', instead of conventional cumulative layer encodings, for multicast and unicast distribution respectively of stored video.

To achieve robustness to packet loss, the approach of encoding a continuous media source with multiple descriptions (MD) has been taken. Proposed solutions in the literature rely on the content distribution network's (CDN) architecture or are applied end-to-end. In the first paradigm, Apostolopoulos et al. [AWTW02] advocate the use of path diversity with MD coding for streaming video on-demand over CDNs. This is similar to combining parallel downloads of different video descriptions from multiple server nodes. In an end-to-end context, Lee et al. [LPK+00] calculate optimisations to the MD coder descriptions using feedback from an AIMD-like congestion control protocol.

In a novel approach to layering, Chou et al. addressed the joint problem of heterogeneous receivers and channel unreliability for multicast video with the hybrid Layered Multiple Description Coding (LMDC) [CWP03] which, if supported by multiple diverse distribution trees, also provides a good solution to peer-to-peer streaming [CPW03]. With LMDC they transmit base layer descriptions to low bandwidth clients while both base and enhancement layer descriptions get transmitted to higher bandwidth clients.

5.2.2 Channel reliability through forward error correction codes

Graceful degradation of streaming multimedia over packet erasure channels has often been the topic of research on various types of media. Mohr's work on image transmission [MRL00] advocates the application of unequal amounts of FEC to progressively compressed images, with the FEC codes optimised according to the importance of various parts of the image stream. Coded images, however, are represented by a rather non-time-lined bitstream, unlike other continuous media types that are either inherently time-lined, such as audio, or temporally encoded such as natural video. For the latter media types, the idea of graceful degradation at higher packet loss rates has been expressed in general through two types of algorithms:

(i) Those algorithms that optimise the amount of FEC redundancy codes applied unequally to layers of compressed digital video (Stuhlmüller et al. [SFLG00]) or,

(ii) FEC codes provided selectively to those frames of an audio codec that disturb speech intelligibility if lost and are 'marked' with higher importance (Sanneck et al. [SLW00]).

The above approaches fall respectively into the areas of media-independent and media-specific error resilience schemes of the classification in table 4.1 and the former are the subject of

this chapter. Rejaie provides a systematic and more complete classification of error resiliency schemes that can operate in best-effort networks [Rej99].

The simplicity and popularity of FEC schemes in multimedia streaming has led the research community and the industry to attempt their standardisation. The AVT group of IETF recommends the use of FEC in two threads for the protection of multimedia streams against packet loss. In one such thread it is recommended that parity FEC be applied generically, albeit unequally, to selected fields of RTP packet headers and to the payload. This effort, called Unequal Level Protection (ULP) [LLV+03], is media-independent and stems from the observation that almost all media formats have their frame header information at the beginning of a data packet, which is therefore the most vital part of the packet. The second thread recommends the Unequal Erasure Protection scheme (UXP) [LWPW03], which applies interleaving and redundancy information obtained through Reed-Solomon operations to the media payload of progressively encoded multimedia streams.

The above distinction between ULP and UXP exposes the fundamental requirements difference between the applications to which these schemes are tailored. The interleaving and block structure redundancy of UXP incurs delays at both encoding and decoding sides even when loss does not occur, whereas, in ULP, decoding delays are only introduced at the receiver in case of packet loss. UXP offers, however, the advantage of changing the packet size independent of media payload type and error code used. It is also important to reemphasise here that although both ULP and UXP are considered media-independent schemes, the importance of each part or 'layer' in UXP is rather dependent on the stream content. Reflecting the different importance, redundancy is then applied unequally but independently to each part. This distinction is maintained in this chapter where a method adhering to the media-independent UXP paradigm is proposed for animated 3-D wireframes without interleaving the transmitted packets.

This chapter focuses on the use of FEC codes and optimisations for use with layered 3-D animation streams. Optimised error-resilient approaches for the transmission of statically encoded models have been proposed by Al-Regib [RA02, RA01] in related literature. However, layering particularly tailored to animated 3-D wireframe meshes has not been proposed in the literature. Application of FEC codes to static models, as proposed by Al-Regib, also bears the limitation that the decoding process would stop in case of high losses, if the provided FEC-based redundancy is not adequate to cover severe loss conditions.

MPEG-4 attempts to elaborate on the area of 3-D animation within its Animation Effects Framework (AFX) [ISO02, Jan00] in version 5 of the standard, which recently reached draft standard status. However, the MPEG-4 approach does not provide a scalable error resilience method regarding animation streams at the Elementary Streams Layer, that accounts for the channel bandwidth or channel error characteristics. It rather leaves any transport-level loss protection at the Systems Layer.

Given these facts, transmission of animation based on each frame being individually encoded with a static codec, such as those proposed by Al-Regib [RARM02], MPEG-4's 3DMC, or other methods described in section 2.2, will not have the same visual result as a temporally coded sequence where the sender-optimised redundancy is complemented by receiver-based concealment. To further improve the perceptual result, the proposed FEC code optimisations of this chapter for 3-D animation are designed around a perceptual measure of distortion capturing model surface smoothness, as opposed to raw distance metrics used in the literature of static mesh coding.

The following section proposes options for the 'layering' of animation data in the wider sense, motivated by similar approaches for other streaming media described earlier, in a step towards formulating an optimisation problem to sender-based redundancy.

5.3 Options for 3-D Wireframe Layering

In 3-D animated wireframe models, options for organising the encoded vertex position data into layers can be defined from different standpoints. Layering options from three different standpoints are proposed below.

5.3.1 Grouping and partitioning

From the point of view of the *3-D scene composition* two possible layering options are identified: node grouping and node partitioning.

1. **Node grouping** leads to a layered organisation of the vertex data where the nodes of the animated mesh are allocated to different sets according to some criterion and each such set represents a streaming layer. This scheme is the commonest method for layering a 3-D mesh having a plurality of nodes. The number of nodes to be grouped in the same layer is a design choice, and dictates the output bitrate of the layer. The criteria for node grouping could be any of the error metrics or other measurable quantities on the mesh, like those used by mesh simplification methods (global ranking or cost value of candidate topological operations). Such criteria have been discussed as error metrics in

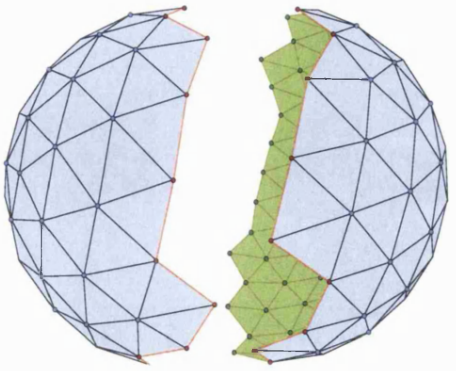


Figure 5.1: Node grouping and partitioning methods for animation layering. The original model BOUNCEBALL has only one node consisting of 320 triangular faces, and has been split into two smaller nodes (partitioning). A *symmetric* partitioning is shown here where each sub-node has 180 faces each. Vertices and edges on the split boundary have been duplicated.

subsection 2.5.3 and in the mesh decimation literature of paragraph 2.2.2.3.

2. **Node partitioning** is suitable for meshes with only a single or few nodes but a large number of vertices per node. For such meshes, node partitioning is required according to which a node's vertices are allocated to two or more sets, where each set or sub-node represents a streaming layer. Node partitioning would restructure the 3-D mesh's vertices into a new mesh with more nodes than were originally present. This process does not affect connectivity, or the overall rendered model.

When layering is done with node grouping, it is good practice to consider grouping together nodes which are spatially close or adjacent in the mesh. This should be done in order to minimise the chance of cracks on the surface of the model, as described in paragraph 2.2.1.2. When layering is achieved by partitioning a node into sub-nodes, partitioning should not be arbitrary, but should reflect the natural objects these new nodes represent in the 3-D scene and their corresponding motion. If partitioning is not possible in the above sense, one could partition the mesh or the node into arbitrary sized sub-meshes (sub-nodes) and subsequently allocate these sub-meshes into layers. Figure 5.1 shows model BOUNCEBALL layered according to the above scheme of node partitioning into sub-nodes. In general, mechanisms for efficient mesh partitioning may require complex pre-processing steps or connectivity changes, but these mechanisms are beyond the scope of this thesis.

By assuming static connectivity, the 3D-Anim codec described in chapter 3 can afford layered encoding of vertex position data by node partitioning or node grouping.

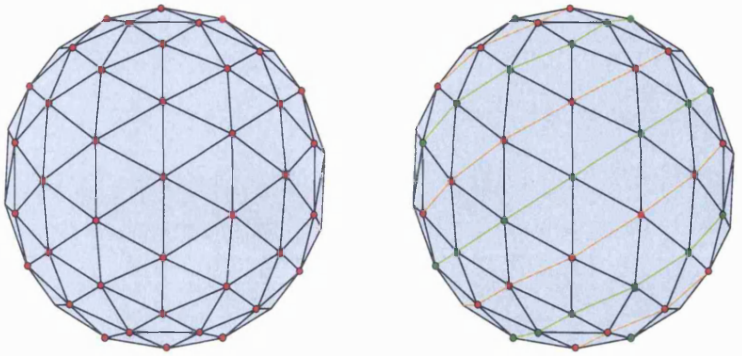


Figure 5.2: Organising animation data into independent layers: The red vertices of the original wireframe BOUNCEBALL in the left image have been organised into two independent layers. The first layer is made up of the green vertices shown in the right image, whereas the red vertices form the second layer. Connectivity information needs to be updated to reflect the layering configuration.

5.3.2 Cumulative and independent layering

A second distinction in layering schemes can be made from the standpoint of the *source encoder*, between independent layers and layers with cumulative effect on mesh detail.

1. **Independent layer** schemes suggest that animation data (signal samples, such as vertex positions in the case of 3D-Anim encoder) in each layer is not related to animation data of any other layer belonging to the same stream. Each independent layer can feed the animation samples it contains to the decoder, and these samples can be rendered immediately without requiring the samples of a 'lower' layer to have been decoded first. Independent layering of sequence BOUNCEBALL is shown in figure 5.2.

2. **Cumulative layer** schemes, on the other hand, suggest that for the decoding and rendering of animation samples contained in layer L_i, animation data of the coarser quality layer L_{i-1} have been decoded first. Otherwise layer L_i data are unused. It is assumed that the signal has been encoded in such a way so that layer L_i animation adds 'detail' to, or refines, the coarser layer L_{i-1}.

To organise the animation data into independent layers seems to be different from what is considered common layering practice in natural 2-D video coding, where layers containing encoded video samples are rather cumulative in nature, and a 'higher' layer is usually added to the quality of a coarser quality layer.

Independent layering may also serve as an elegant mechanism for mapping scalable bitstreams of 3-D meshes encoded hierarchically, or progressively, to layers for animation streaming. Suppose, for example, that a static mesh decimation scheme results in: (i) a batch of

base layer vertices forming a coarse representation of a 3-D object, and (ii) other batches of refinement layer vertices (not included in the base layer) that add detail to the model. Had the mesh been animated, these resolution-based batches can be mapped one-to-one to streaming layers, where each streaming layer contains animation data for vertices only present in the corresponding resolution batch. This setting, formed by independent layers, emulates the effect of cumulative layering and bears a strong resemblance to multiple description (MD) video coding.

5.3.3 Fixed and adaptive layering

Seeing the layering mechanism from the standpoint of the *end-user* experiencing the 3-D stream, a further distinction can be made between fixed and adaptive layering schemes.

1. **Fixed layering** assumes that the group of nodes (or a sub-node) allocated to a certain layer are static and this allocation does not change over time.

2. **Adaptive layering** mechanisms permit the group of nodes that form a layer to change over time, or follow a node partitioning scheme that can dynamically change during a layer's transmission.

An example of static layer allocation is the *distortion-aware layering scheme* where layering is performed in a way such that the average distortion of a layer reflects its importance in the animation sequence. To achieve this, the distortion metric (such as the visual smoothness, VS, defined in equation (5.5)) is computed independently for every node in a mesh and the nodes are ranked according to their average distortion in the sequence. The node, or group of nodes, with the highest average distortion forms the first and most important layer visually, L_0. This is the layer that a streaming system may want to make more resilient to packet loss than other layers. Subsequent importance layers $L_1, ..., L_M$ are created by subsequent nodes, or group of nodes, in the distortion ranking order. The experiments with sender-based optimised redundancy using the 3D-Anim system and Visual Smoothness as distortion metric described in section 5.7 follow a fixed layering scheme.

Adaptive layering schemes allow dynamic changes to their vertex contents and are expected to be less usual. An example of adaptive layering is the *visibility-based layering* resulting from user interactivity in the streamed 3-D scene. According to this scheme, transmitted layers contain only those vertices that are expected to be visible at the receiving terminal. As the user interacts with the streamed object scene, different parts of these objects are visible at any time instant, and only the nodes forming these visible parts are grouped or partitioned to form layers. This mechanism relies on regular and timely camera position updates being communicated back to the server.

Table 5.1: Summary of layering approaches and options for 3-D animated wireframe sequences.

3-D SCENE		SOURCE ENCODER		END-USER	
Node Grouping	Node Partitioning	Independent Layers	Cumulative Layers	Fixed Layering	Adaptive Layering

The three approaches to layering 3-D animated wireframes and their corresponding options are summarised in table 5.1. It has to be noted that layering options proposed in this section are not restricted by the topological constructs forming the signal. The discussion above was made assuming the topological construct in hand is vertices, but equivalent layering schemes based on faces or edges could be shaped accordingly. It is also worth commenting the fact that layering options from each of the standpoints described previously can be combined together to form layering schemes suitable for a wide range of applications. In the experiments described in section 5.7, fixed independent layers based both on node grouping (for sequence TELLY) and node partitioning (for BOUNCEBALL) are used.

The following section formalises the RS erasure code optimisation problem, describes the channel model, and a block structure that is suitable for the design of an efficient packetisation scheme for each layer. A closed form relation of the layer design parameters is also derived.

5.4 Error Resilient 3-D Wireframe Streaming

5.4.1 Channel model and error correction codes

The idea of Forward Error Correction (FEC) is to transmit additional redundant packets which can be used at the receiver to reconstruct lost packets. In the proposed FEC scheme Reed-Solomon (RS) codes are used across packets. RS codes are the only non-trivial maximum distance separable codes known, hence they are suitable for protection against packet losses over bursty loss channels. An RS (n, k) code of length n and dimension k is defined over the Galois Field $GF(2^q)$ and encodes k q-bit information symbols into a codeword of n such symbols, i.e., $n \leq 2^q - 1$. A sender with k information packets will generate $n - k$ redundancy packets. The resulting n packets are stacked in a Block-Of-Packet (BOP) structure described in more detail in subsection 5.4.2 and shown in figure 5.3. The BOP packet structure has been previously used in the literature for error resilient video and static mesh coding [HSLG99, SFLG00, RA02]. To maintain constant total channel data rate, the source rate is reduced by the fraction k/n, called the code rate, resulting in an initially reduced animation quality. A receiver

can begin decoding as soon as it receives any k correct symbols, or packets of a BOP.

In reality, the underlying bursty loss process of the Internet is quite complex, but it can be closely approximated by a 2-state Markov model, as described in subsection 4.4.1 and depicted in figure 4.4. The two states are state **G** (good), where packets are timely and correctly received, and **B** (bad), where packets are either lost or delayed to the point that they that can be considered lost. The state transition probabilities p_{GB} and p_{BG} fully describe the model but, since they are not sufficiently intuitive, the model can be better expressed using the average loss probability P_B (or probability of being in state **B** on average, $Prob(\mathbf{B})$), and the average burst length $\bar{b}$, as:

$$\begin{aligned} P_B = Prob(\mathbf{B}) &= \frac{p_{GB}}{p_{GB} + p_{BG}} , \\ \bar{b} &= \frac{1}{p_{BG}} . \end{aligned} \tag{5.1}$$

For the selection of the appropriate RS code one needs to know the probability that a BOP cannot be reconstructed by the erasure decoder as a function of the channel and the RS code parameters. For an RS (n, k) code, this is the probability that more than $n - k$ packets are lost within a BOP, and it is called the *block error rate*, P_{BER}. Let $P(m, n)$ be the probability of m lost packets within a block of n packets, also called the *block error density function*. Then, it can be calculated:

$$P_{BER} = \sum_{m=n-k+1}^{n} P(m, n) . \tag{5.2}$$

The average loss probability P_B and the average loss burst $\bar{b}$ corresponding to the 2-state Markov model described above, relate to block error density function $P(m, n)$. The exact nature of their relationship has been extensively studied and derived in the literature. Here, the derivation for bit-error channels is adapted to a packet loss channel [Ell65].

Such a model is determined by the distribution of error-free intervals (gaps). If there occurs an event of gap length ν such that $\nu-1$ packets are received between two lost packets, then the gap density function $g(\nu)$ gives the probability of a gap length ν, i.e., $g(\nu) = Prob(0^{\nu-1}1|1)$, where 0 denotes a received, and 1 a lost packet. The gap distribution function $G(\nu)$ gives the probability of a gap length greater than or equal to $\nu - 1$, i.e., $G(\nu) = Prob(0^{\nu-1}|1)$. In state **B** of the model, all packets are lost, while in state **G** all packets are received, yielding:

$$g(\nu) = \begin{cases} 1 - p_{BG}, & \nu = 1, \\ p_{BG}(1 - p_{GB})^{\nu-2} p_{GB}, & \nu > 1. \end{cases}$$

$$G(\nu) = \begin{cases} 1, & \nu = 1, \\ p_{BG}(1 - p_{GB})^{\nu-2}, & \nu > 1. \end{cases}$$

Let $R(m,n)$ be the probability of $m-1$ packet losses within the next $n-1$ packets following a lost packet. This probability can be calculated from the recurrence:

$$R(m,n) = \begin{cases} G(n), & m = 1, \\ \sum_{\nu=1}^{n-m+1} g(\nu) R(m-1, n-\nu), & 2 \leq m \leq n. \end{cases}$$

Then, the block error density function $P(m,n)$ or probability of m lost packets within a block of n packets is given by:

$$P(m,n) = \begin{cases} 1 - \sum_{\nu=1}^{n} P(m,\nu), & m = 0, \\ \sum_{\nu=1}^{n-m+1} P_B G(\nu) R(m, n-\nu+1), & 1 \leq m \leq n, \end{cases}$$

where P_B is the average error probability.

From equation (5.2) it can be seen that $P(m,n)$ determines the performance of the FEC scheme, and $P(m,n)$ can be expressed as a function of P_B and $\bar{b}$ using equations (5.1). Later, in section 5.4, this expression of $P(m,n)$ is used in a RS (m,n) FEC scheme for optimised source/channel rate allocation that minimises the visual distortion.

5.4.2 Bitstream format and packetisation

The single-layer output bitstream of the 3D-Anim codec without error resilience at the sender has been appropriately packetised for streaming with RTP as the application-level transport protocol. This process was described in detail in subsection 3.5.1 and its main features are summarised below:

- In order to describe which nodes of the model are animated, two animation masks, Node Mask and Vertex Mask, were defined. A bit set in the Node Mask denotes that a corresponding node in the Node Table will be animated. A Vertex Mask per axis has been defined in a similar way for animated vertices.

- The M bit in the RTP header must be set for the first of a series of 'empty' frames, which — if they exist — can be grouped together.

- One Application Data Unit (ADU) is contained in one RTP packet. For sparse animations one ADU usually corresponds to an animation frame.

- The RTP packet payload format of subsection 3.5.1 starts with the Node Mask and Vertex Mask, followed by the encoded samples along each axis.

In this sense, the 3D-Anim codec's output bitstream is 'naturally packetisable' according to the Application Level Framing (ALF) principle [CT90].

The simple payload format summarised above suffices for light animations with a modest number of vertices. However, sequences with high scene complexity or high resolution meshes, may generate a large amount of coded data after compression, resulting in frames which potentially exceed the path MTU. In such cases, raw packetisation in a single layer would require the definition of fragmentation rules for the RTP payload, which may not always be straightforward in the ALF sense. Furthermore, frames directly packetised in RTP as described above generate a variable bitrate stream due to their varying lengths.

For sender-based redundancy schemes, like the one proposed in this chapter, a more efficient packetisation scheme is sought that satisfies the requirements set out above: (a) to accommodate layered bitstreams, (b) to produce a constant bitrate stream, and (c) to be able to multiplex redundant information with source data. There is related effort in the literature in designing efficient block data structures conforming to these requirements. Redmill and Kingsbury [RK96] have proposed the error-resilient entropy code (EREC) to multiplex the variable-length blocks of data produced by standard compression algorithms into an error-resilient structure. Their algorithm works by reorganising the variable-length blocks and allocating them into slots such that each block starts at a known position within the code (the beginning of a slot). This means that the decoder is automatically synchronised at the start of each block at the expense of some source bitrate that defines the boundary of the slot. Albanese and Luby, in Priority Encoded Transmission (PET) [AL96], suggest that the various parts of a compressed source signal (*original message*) be ordered according to their importance in the decoded signal (*decoded message*). They then propose a generic error protection scheme according to the priority ordering of the data parts. In PET, the priorities of each part are assigned by the user. Although PET does not directly propose a block data structure, to a large extent it implies inherent similarity to EREC for the partitioning and prioritisation of the data.

The BOP data structure introduced earlier in this chapter satisfies the three requirements above for efficient packetisation of multiplexed source and redundant data. Yet it maintains simplicity compared to EREC as it does not waste additional source bits to achieve synchronisation. Furthermore, the method of providing unequal redundancy to different parts of the encoded message (3-D animation data in this case) does not rely on the user arbitrarily assigning priorities to the various parts of the message as done in the PET scheme, but rather explicitly extracts such priorities from the visual importance of each part (based on the calculation of the visual smoothness metric). The ordered 3-D animation data can then be naturally allocated to

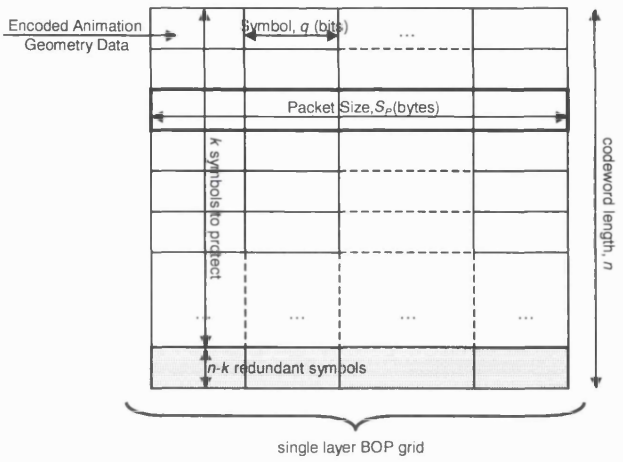


Figure 5.3: The Block-Of-Packets (BOP) grid structure.

layers for streaming. Finally, the BOP structure naturally respects the ALF principle at the packet level and makes a suitable choice for efficient packetisation of redundant 3D-Anim data.

The BOP structure has been previously used in error resilient video or static mesh coding [HSLG99, SFLG00, RA02], and it is adapted here for use with 3-D animation data. Encoded frames of a single layer are placed sequentially in line order of an n-line by S_P-column grid structure and then RS codes are generated vertically across the grid. For data frames protected by an RS (n, k) erasure code, error resilience information is appended so that the length of the grid is n for k frames of source data, as shown on figure 5.3. This method is most appropriate for packet networks with burst packet errors, and can be fully described by the sequence frame rate f, the packet size S_P, the data frame rate in a BOP f_{BOP}, and the RS code (n, k).

Intuitively, for a BOP frequency of f_{BOP} (BOPs/sec), with S_P byte long packets, with frame rate f, the total source and channel bitrate R is given by:

$$R = R_C + R_S = \frac{n \cdot f \cdot S_P}{f_{BOP}} , \tag{5.3}$$

where R_S and R_C are the source and channel bitrates respectively.

This equation serves as a guide to the design of efficient packetisation schemes by appropriately balancing the parameters f_{BOP}, n and S_P. It also encompasses the trade-off between delay and resilience. For a layered bitstream, the design of one BOP structure per layer is needed. By varying the parameters in equation (5.3), different RS code rates can be allocated to each layer, thus providing an unequal level of error protection to each layer. The way these parameters are adjusted in practice for the application of 3-D animation streaming, considering a measure of visual error, is explained in sections 5.6 and 5.7.

5.5 Visual Distortion Metric

Typical automatic distortion metrics reviewed in the literature (subsection 2.5.3), such as the SNR and 3-D PSNR defined in [GSK02] and [VOH01] respectively, are derived by equivalence to natural video coding but they are tailored to the statistical properties of the specific signal they encode, failing to give a uniform measure of user perceived distortion across a number of signals and encoding methods over different media. Moreover, especially for 3-D meshes, all these metrics give only objective indications of geometric closeness, or signal-to-noise ratios, and they fail to capture the more subtle visual properties the human eye appreciates, such as surface smoothness.

One attempt that was made in this direction was reported by Karni and Gotsman in [KG00] as being undertaken whilst evaluating their spectral compression algorithm for 3-D mesh geometries. In this, the suggested 3-D mesh distortion metric normalises the objective error computed as the Euclidean distance between two vertices, by each vertex's distance to its adjacent vertices. This type of error metric captures the surface smoothness of the 3-D mesh. This may be achieved by a *Laplacian operator*, which takes into account both topology and geometry. The value of this geometric Laplacian at vertex v_i is:

$$GL(v_i) = v_i - \frac{\sum_{j \in n(i)} l_{ij}^{-1} v_j}{\sum_{j \in n(i)} l_{ij}^{-1}} , \tag{5.4}$$

where $n(i)$ is the set of indices of the neighbours of vertex i, and l_{ij} is the geometric distance between vertices i and j. Figure 5.4 provides intuition on the geometric Laplacian quantity GL. The surface on the left of the image represents a hypothetical fragment of the animated mesh at time t, consisting of vertices denoted by heavy dots and triangular connectivity denoted by straight and dotted lines. The right hand side mesh represents the corresponding decoded fragment. Assume a scenario where the decoded coordinates of vertices B, C, D, E introduce no coding error, whereas vertex A's y-coordinate contains coding error. Obviously, the decoded surface suffers a bump on vertex A. This distortion is what the Laplacian GL captures. A complete numerical example of the VS metric is given in appendix C.

The new metric is defined as the weighted sum of the norm of the geometric distance between meshes and the norm of the Laplacian difference ($m_t, \hat{m}_t$ are the vertex sets of meshes $M_t, \hat{M}_t$ respectively, and n the set size of $M_t, \hat{M}_t$):

$$VS_{(t)} = \frac{1}{2n} \sum_{r=1}^{n} (w_G \|m_{rt} - \hat{m}_{rt}\| + w_L \|GL(m_{rt}) - GL(\hat{m}_{rt})\|) . \tag{5.5}$$

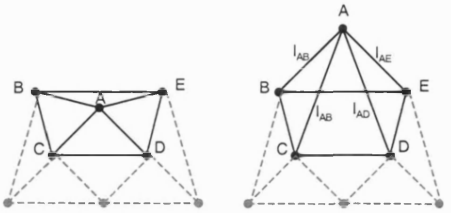


Figure 5.4: Example of a Laplacian operator on a hypothetical surface. Left: original surface. Right: reconstructed surface with distortion on vertex A.

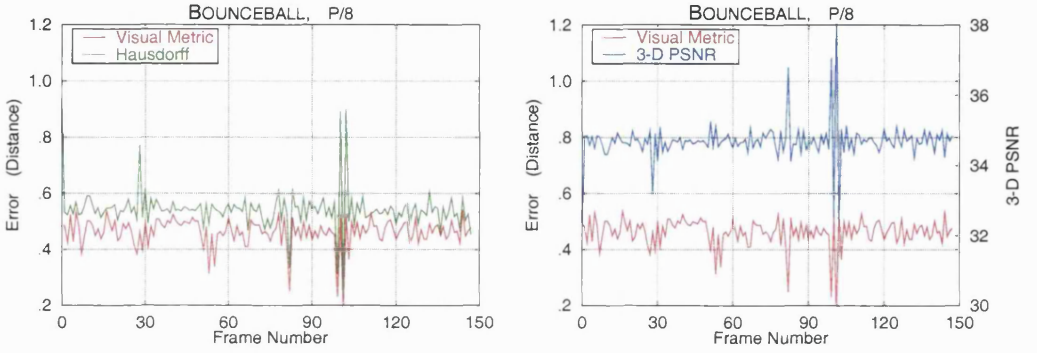


Figure 5.5: Comparative plot of distortion metrics: 3-D PSNR, Hausdorff Distance, and Visual Smoothness for 150 frames of the animated sequence BOUNCEBALL and P-frames inserted every 8 frames.

Depending on the value of the weights w_G, w_L one can put more emphasis on either the geometric or the visual term of the equation according to the animation content.

This metric in equation (5.5) is used in the evaluation of the proposed redundancy optimisation scheme, and it will be referred to hereafter as the *Visual Smoothness* metric (VS). For the experiments of this chapter the values of $w_G = w_L = 1$ have been used, leading to a balanced VS metric between geometric and visual distortion. A comparative plot providing intuition on the notion of the VS metric against 3-D PSNR and the Hausdorff Distance is given in figure 5.5 for the 3-D sequence BOUNCEBALL. The sequence has been encoded and decoded with 3D-Anim and the plots show the Geometric Distance (as either Hausdorff or 3-D PSNR) against the Visual Distortion. Both graphs indicate that the Visual Distortion might be low in case where the Geometric Distance is high and vice-versa. This is most evident around frames 16 and 114.

Obviously, the VS metric requires connectivity information: the adjacent vertices of every vertex m_t. For the case of the 3D-Anim codec, where it was assumed no connectivity changes occur during the animation, the vertex adjacencies can be precomputed.

5.6 Problem Formulation

The expected distortion of the animation at the receiver at time t is the sum of the product quantities $P_{jt} \cdot VS_{jt}$, where j is the layer index, VS_{jt} is the visual distortion incurred by missing information in layer j at time t, and P_{jt} is the probability of having an irrecoverable packet loss in layer j. By the way the layers were constructed, the probabilities P_{jt} are independent, and a burst packet loss in a layer contributes to its own visual distortion VS_{jt} in the decoded sequence. Formally, the expected visual smoothness $VS_{(t)}$ of an animation at the decoder at time t can be expressed as:

$$VS_{(t)} = \sum_{j=0}^{L-1} P_{jt} \cdot VS_{jt} \ , \tag{5.6}$$

where L is the number of layers. In the equation above, P_{jt} is the block error rate P_{BER} as given by equation (5.2), or the probability of losing more than $n - k_j$ packets in layer j. Using the block error density function $P(m, n)$, we can write:

$$P_{jt} = \sum_{m=n-k_{jt}+1}^{n} P(m, n) \ . \tag{5.7}$$

From equation (5.6) and (5.7) the quantity $VS_{(t)}$ can now be expressed as:

$$VS_{(t)} = \sum_{j=0}^{L-1} \sum_{m=n-k_{jt}+1}^{n} P(m, n) \cdot VS_{jt} \ . \tag{5.8}$$

Equation (5.8) estimates in a statistical sense the expected visual smoothness experienced per frame at the decoder. The objective is to minimise this distortion with respect to the values of k_{jt}'s in equation (5.8). With distortion-aware layering it is expected that the optimisation process allocates more redundancy to the layer that exhibits the greatest visual distortion (coarse layer), and gradually reduces the redundancy rate on layers with finest contribution to the overall smoothness. There are L values of k_{jt} that need to be calculated at every time t, that follow the conditions $0 \leq k_{jt} \leq n$ and $\sum_{j=0}^{L-1} (n - k_{jt}) = \frac{R_C}{q}$, where R_C the redundancy bits, and q is the symbol size. The above problem formulation yields a non-linear constrained optimisation problem that can be solved numerically[1]. .

The anticipated behaviour of the model for $P_B = 0$ is to produce equal values for k_{jt}'s, whereas at high P_B, k_{jt}'s would vary unequally. Note that for the calculation of smoothness distortions in equation (5.8), possible concealment at the receiver was not taken into account.

In chapter 4, it was shown that techniques based on vertex linear temporal interpolation are an efficient method of error concealment for 3D-Anim frames. According to this technique, a burst of missing frames can be concealed by interpolating the missing vertex values from

[1] A solution is provided here using the non-linear optimisation tools of *Mathematica*®

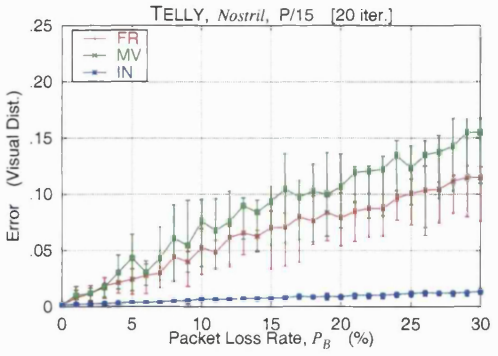


Figure 5.6: Performance of three error concealment methods for the sparse node *Nostril* of sequence TELLY, with $P_B = [0..30]$ and $\bar{b} = 4$ using the Visual Smoothness distortion metric. (FR = Frame repetition, MV = Motion vector repetition, IN = Interpolation). The plot shows the average performance over 20 iterations with different loss patterns.

two sets of vertices: the ones in the last received frame before the burst occurred and those in the currently received frame. This relies on the 'locality of reference principle', according to which high-frame rate animations are unlikely to exhibit vertex trajectories other than linear or piece-wise linear. If higher complexity can be accommodated, higher order interpolation can be employed by using information from the neighbouring frames. Interpolation and other concealment methods discussed in section 4.3 are generic in that they can be used by any other decoder.

Figure 5.6 shows the relative performances of three error concealment methods adapted to the experimental parameters of this work, namely $P_B = [0..30\%]$ and $\bar{b} = 4$. It is evident that linear interpolation outperforms frame repetition or motion vector repetition methods under all experimental conditions. (Detailed descriptions of all error concealment methods at the receiver were given in subsections 4.3.1 to 4.3.3 and detailed evaluation of their performances with 3-D PSNR metric was presented in 4.4.) The plot shows average values for 20 iterations with different loss patterns. It is clear on the plot (as seen by the error bars) that the interpolation concealment method exhibits very low variance, verifying the locality of reference principle. In high frame-rate sequences (30 Hz in the case of TELLY) the animation trajectory of each vertex remains monotonic for frame lengths higher than the average loss burst length $\bar{b} = 4$. This leads to very good concealment performances of interpolation-based methods.

The remainder of this chapter will assume the use of interpolation-based error concealment at the receiver in the case where the channel decoder receives less than $n - k_{jt}$ BOP packets. In fact, the k_{jt}'s that provide a solution to the optimisation problem, will also give minimum distortion if combined with concealment techniques. The experienced distortion in such cases

will be lower than the distortion without error concealment.

5.7 Experiments and Results

5.7.1 Experiment details

In the following experiments, the efficiency of the proposed Unequal Error Protection (UEP) scheme combined with Error Concealment (EC) for streaming 3-D wireframe animations is demonstrated through simulation. In particular, UEP and EC are compared to simple UEP, to Equal Error Protection (EEP) and to No Protection (NP). The comparison is based on the Visual Smoothness metric, which is known to yield a distortion measure that captures the surface smoothness of the time-dependent mesh during the animation. For the calculation of the parameters k_{jt} the constrained minimisation problem of equation (5.8) was numerically solved, given the channel rate R_C. Furthermore, n was calculated from equation (5.3) such that the rate characteristics of the original source signal are met for a design of a BOP particular to this experiment. The other parameters used in equation (5.3) are given below for the two sequences in the experiments, and are also summarised in table 5.2. For the EEP case, a constant k is considered that can be derived directly from the selection of the channel rate, which was set to 15%. For the NP case, all available channel rate was allocated to the source. Finally, an EC scheme based on interpolation was used for the case of UEP with residual losses. In all experiments, $\bar{b} = 4$ was used.

Sequences TELLY and BOUNCEBALL with density factors of $df_{\text{TELLY}} = 0.75$ and $df_{\text{BOUNCEBALL}} = 1.0$ given by equation (3.8) were used in the experiment. TELLY consists of 9 nodes (out of which 3 are relatively sparse, and the remaining 6 are complete) and totals 780 frames at 30 Hz as shown in table 5.2. Its average source bitrate is $R_{S,\text{TELLY}} = 220$ kbps. BOUNCEBALL only has one complete node and 528 frames at 24 Hz, forming one layer of source rate $R_{S,\text{BOUNCEBALL}} = 61$ kbps average. Both sequences have been coded with P-frames at every 15 frames. Roughly 15% of channel coding redundancy was allowed, resulting in total source and channel rates of $R_{\text{TELLY}} = 253$ kbps and $R_{\text{BOUNCEBALL}} = 70.15$ kbps. Choosing $n = 32$ the parameters calculated from equation (5.3) for each layer's packetisation are tabulated in table 5.2. The value of n is chosen as a compromise between latency and efficiency, since higher n makes the RS codes more resilient, by sacrificing delay and buffer space.

Sequence TELLY was split into three fixed layers according to the *distortion aware* layering method presented in section 5.3. Each layer consisted of the nodes shown in table 5.2. Each layer's fraction of the total number of animated vertices in the 3-D mesh is $(L_0, L_1, L_2) = (0.48, 0.42, 0.10)$ on average. This splitting is expected to reflect the source bitrates of each

Table 5.2: Animation sequence parameters used in the redundancy experiments: TELLY & BOUNCEBALL.

Sequence	TELLY
df_{TELLY}	0.75
Nodes	9
Frame Rate, f	30 Hz
Source Rate, R_S	220 kbps
Channel Rate, R_C	33 kbps
Frames	780
Layer 0	UpperLip LowerLip Tongue
Layer 1	Skin Teeth
Layer 2	EyeLash EyeBrow EyeCorner Nostril

Sequence	BOUNCEBALL
$df_{\text{BOUNCEBALL}}$	1.0
Nodes	1
Frame Rate, f	24 Hz
Source Rate, R_S	61 kbps
Channel Rate, R_C	9.15 kbps
Frames	528
Layer 0	Bball 1^{st} half
Layer 1	Bball 2^{nd} half

		TELLY	BBALL
L_0	S_P	264	200
	f_{BOP}	16	35
L_1	S_P	264	200
	f_{BOP}	19	35
L_2	S_P	150	N/A
	f_{BOP}	50	

layer proportionally. As can be seen, the suggested layering scheme allocated 2 out of 3 sparse nodes to the same layer, L_1. The total number of vertices of these two sparse nodes represents 65% of the vertices in the reference mesh. The third sparse node, Nostril, was allocated to layer L_2, but its individual motion relates to a very small fraction of the model's total number of vertices ($\approx$ 1.3%). This fact may bear some significance if one wishes to relate the node-to-layer allocation (using the VS metric) to the density factor df_L, calculated per layer[2] (by equation (3.8)), and to the output bitrates. However, in order to derive any useful such relation a large bank of animated 3-D model data is required. If such a relation exists, an adaptive layering scheme may be developed for applications with such needs.

Sequence BOUNCEBALL initially contains only one node. The sequence represents a soft ball with inherent symmetry around a centre point as its shape implies. The ball also deforms slightly as it bounces. Given the shape symmetry, layers were designed according to the *node partitioning* method. The BOUNCEBALL mesh was partitioned into two nodes of equal number of vertices without respect to the VS metric for each node. The logic behind this partitioning is to attempt to verify the effect the VS metric has on the proposed UEP resilience scheme. All other source coding parameters are constant between the two layers, most importantly the quantisation step size. It is anticipated that both layers will receive roughly equal average

[2]This calculation is possible if k in equation (3.8) refers only to the nodes allocated to the particular layer.

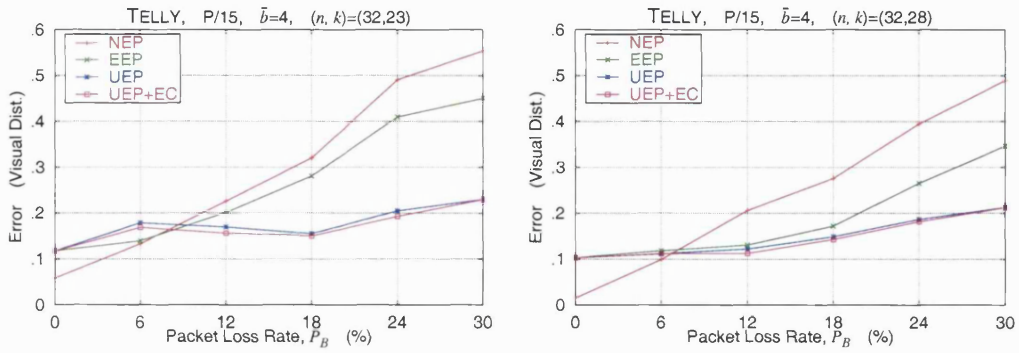


Figure 5.7: Comparison of Visual Smoothness (VS) between transmitted and decoded frames of 3 layers of the wireframe animation TELLY. The equivalent EEP RS codes are respectively: left $(n, k) = (32, 23)$, right $(n, k) = (32, 28)$. Average burst length: $\bar{b} = 4$.

protection bits, so that UEP performance will approach that of EEP.

5.7.2 Experimental results

Figure 5.7 (left) depicts visual smoothness, VS, as a function of the average packet loss rate, P_B, for TELLY. The four curves on the plot represent each suggested resilience method, for the code $(32, 23)$. The average calculated codes for the UEP are as follows (rounded to nearest integer): $(n, \bar{k_0})=(32, 20)$, $(n, \bar{k_1})=(32, 24)$, $(n, \bar{k_2})=(32, 29)$. It is clear that UEP, and UEP+EC outperform NP and EEP for medium to high loss rates of $P_B > 9\%$. Recall that layering here was performed in such a way that the lowest layer exhibited high average visual distortion. Since the UEP method allocates higher codes to the lower layer (L_0), better resilience is expected for L_0 at high loss rates. This factor dominates in the average distortion, resulting in better performance. At low loss rates it is noticeable that EEP and UEP behave in approximately the same way, as the RS codes are more than sufficient to recover all or most errors. It can be also noted that the NP method under conditions of no loss is much better than any other. This is an intuitive result, since source information takes all available channel rate, thus better encoding the signal. It is also worth noticing the effect of EC: the distortion of the UEP+EC scheme is slightly improved over the simple UEP case. This is also expected.

The results for the $(32, 28)$ RS code on sequence TELLY, shown in figure 5.7 (right), are similar. Here, the threshold where the UEP methods (with or without EC) take over EEP or NP is around $P_B = 7\%$. Note how the initial NP performance (low P_B's) is steep compared to the $(32, 23)$, highlighting again the fact that channel coding bits are actually 'wasted' since they do not contribute much resilience in this low loss region, at the expense of source rate. The corresponding average codes per layer are: $(n, \bar{k_0})=(32, 27)$, $(n, \bar{k_1})=(32, 29)$, $(n, \bar{k_2})=$

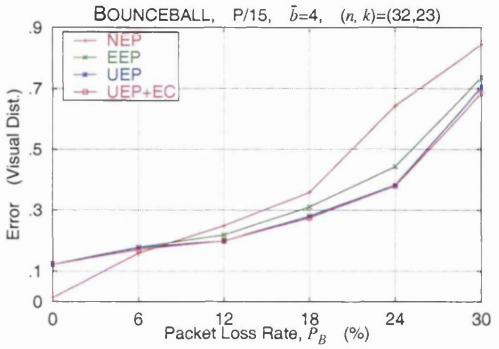


Figure 5.8: Comparison of Visual Smoothness (VS) between transmitted and decoded frames of 2 layers of the wireframe animation BOUNCEBALL. The equivalent EEP RS code is: $(n, k) = (32, 23)$. Average burst length: $\bar{b} = 4$.

$(32, 31)$. There is a slight improvement again in the UEP method's performance resulting from the error concealment's interpolation algorithm. As this quantity has not been accounted for in the optimisation problem, it is expected to contribute a small reduction to the visual error.

Figure 5.8 shows the results achieved for the same experiment repeated over the sequence BOUNCEBALL, which was 'symmetrically' layered as described in the previous subsection. The same $(32, 23)$ EEP code was used again as before for comparison. The graph shows the same trends and relative performances as in TELLY, with UEP+EC being the one giving the best overall performance. It can be noted, however, that the distance of the UEP curves from the EEP ones decreased considerably compared to the TELLY sequence at high P_B's. The average integer calculated RS codes for the UEP case are: $(n, \bar{k_0})=(32, 23)$, $(n, \bar{k_1})=(32, 23)$, i.e., equivalent to the EEP case. This may be a surprising result at the first glance, but careful reasoning suggests that equally balanced layers in terms of the amount of the animation they contain (same number of vertices, nodes, very similar motion in the scene, and same encoding parameters) correspond to visually balanced distortions. This was exactly the anticipated result when layering was discussed for the BOUNCEBALL sequence in the previous subsection. In fact, the real values of k_{0t}, k_{1t} computed as the solution to the optimisation problem, vary around the average integer value of 23. Furthermore, recall that the original symmetric BOUNCEBALL mesh was partitioned into two arbitrary nodes without consideration of their individual visual distortions, which were assumed to be similar. In fact, the soft ball's deformation at the bouncing points reduces the symmetry of the original shape. These facts reasonably explain why there is not an accurate fit to the UEP and EEP curves at higher P_B's, as one would normally expect. Finally, it can be noted that the UEP+EC method provides a slight, but hardly noticeable, improvement to the visual distortion as in the previous experiment.

5.8 Conclusion

This chapter addressed the fundamental problem of how best to utilise the available channel capacity for streaming 3-D wireframe animation in such a way as to achieve optimal subjective resilience to error. In summary, channel coding, packetisation, and layering have been linked with a subjective measure that measures visual smoothness in the reconstructed image (animation frame). The proposed method optimises the distribution of the bit budget allocation reserved for channel coding amongst different layers, using a metric that reflects the human eye's visual property of detecting surface smoothness on time-dependent meshes. Using this metric it has been shown how the encoded bitstream can initially be partitioned into layers of visual importance, and that unequal error protection provided at the sender combined with receiver-based error concealment yields good resilience against burst packet errors occurring on the Internet.

Chapter 6

Conclusions

This thesis addressed the problem of streaming 3-D wireframe animations over best-effort networks and proposed an end-to-end solution that spans source compression and error and rate control. The main strength of the proposed 3D-Anim family of source compression schemes lies in the flexibility that can be achieved with the rates of the outgoing bitstream, which can be adapted to the available network bandwidth either by reducing the geometric accuracy of the model, or by dynamically simplifying it.

However, to realise the full potential of such a scheme in the real application environment of the unreliable Internet, issues related to the robustness of the animation stream to packet loss need to be resolved. Among these issues, the fields of error control and error concealment have been identified as of immediate interest. In the former field, the thesis advocates and proposes a solution based on redundant animation data being transmitted in-band with the basic animation data in an optimised way that caters for the visual smoothness of the wireframe model's surface. In the latter field, performance of low-delay error concealment schemes suitable for animation stream repair have been studied and proposed.

The major contributions of this thesis are outlined below.

6.1 Summary of Contributions

Flexible source coding: A generic predictive compression scheme, 3D-Anim, has been proposed for time-dependent 3-D wireframe meshes as a baseline source coder. It was shown that with minimum processing effort over the geometric signal, temporal compression of wireframe sequences performs better than intra- only coding of 3-D models. The flexibility is such that the generic encoder can also produce layers of animated 3-D data, or can be used for two-dimensional wireframe geometry signals, and can be easily integrated with existing platforms and standards such as MPEG-4. To improve on the generic coding method, an enhanced mode called E3D-Anim significantly optimises the output bitrate by selectively thinning out geomet-

ric constructs that have the least effect on the quality of the mesh.

Progressive coding for 3-D stream quality adaptation: Another main contribution in source coding is the progress made on formally expressing dynamic multi-resolution meshes. The formalisation led to the design of P3D-Anim, an extension of the generic animation codec able to compress 3-D meshes while they are successively decimated or refined in time. This type of simultaneous mesh simplification and geometry compression establishes a significant new direction to 3-D mesh signal processing. This development enables dynamic graphics to adapt their quality to the available resources of the network or the receiving terminal, in direct analogy to quality adaptation of streamed natural video, and may well breed animated 3-D meshes as a mainstream medium.

Rate-controlled end-to-end transmission: In order to achieve adaptation of dynamic 3-D meshes, the progressive source codec P3D-Anim operates in tandem with a rate-control method. The method relies on a motion estimation scheme at the receiver that, for a given reference coded vertex with a 1-ring neighbourhood, is able to predict motion of the neighbouring vertices which have not been coded in the bitstream. The motion estimation method has been investigated with different wireframe sequences and shown to operate well within limits.

Sender-based redundancy for optimal visual smoothness: An important theme in this thesis is the notion of fidelity of decompressed animated models, which can be optimally considered when such models are streamed over the best-effort Internet suffering burst packet loss. Optimality was achieved using an expression of the reconstructed model's surface smoothness as quality criterion. This is a significant departure from mainstream work on error-resilient transmission of static 3-D models, which so far has only focused on objective (or automatic) metrics to quantify distortion. Using visual smoothness to drive transmission of dynamic 3-D models, it was shown that a layered 3-D graphics source supported by forward error control unequally distributed among layers, but with constrained available channel bandwidth, can improve upon transmissions that are unaware of the channel error characteristics. With the visual smoothness metric, it is possible to exploit the sensitivity of human vision in perceiving smooth dynamic surfaces by allowing more redundancy to be provided to layers that are perceptually more salient.

Receiver-based concealment: Investigation of the effects of packet loss on 3-D animation reveals that, even after loss, there is significant signal information at the receivers to help conceal missing animation frames. 3D-Anim and its derivatives structure their frames around motion vectors, which the proposed animation frame concealment solutions utilise either through prediction, repetition, or interpolation, to achieve good substitutes for the lost vectors. This is

the first attempt at the intersection of the fields of networking and computer graphics to tackle the problem and establish solutions for repair of streamed dynamic graphics. Receiver-based model concealment also complements the joint source-channel optimised animation coding described previously.

6.2 Future Directions

There are several potential avenues for further research relating to streaming 3-D models. Despite the popularity of 3-D models in many fields of computer graphics and the intense research carried out in the area of static 3-D mesh compression and transmission in the last decade, there has been little research in coding, and even less in QoS-enabled transmission of animated 3-D models. As this is the first major work addressing these problems, many paths remain as yet unexplored.

Below, a critical view is taken on assumptions and solutions described in the thesis, which re-state the limitations of solutions provided by, or associated with 3D-Anim, and which lead to ideas for future research. These ideas involve either natural extensions to the solutions presented, whereas others are related, yet they are distinctly different approaches to the problems addressed in this thesis.

Coding of animated geometry with attributes and topology. The prototype 3D-Anim assumes that an isomorphic reference model already exists at the receiver, which can be streamed or downloaded beforehand with other means. This assumption implies that a reference model resides fully at the receiver, including its connectivity and attribute information, which remain unaltered during animation. Streaming, however, animated geometry data with 'constant' attributes may not always be what the application requires. Animated models may form part of an interactive virtual environment which is appropriately illuminated, shaded, or textured, to offer increased realism to the user. Ideally, these additional attributes should be compressed along with geometry data. Furthermore, compressed attributes need to be streamed with geometry in a synchronised manner.

The problem of compressing static 3-D models with their attributes has been addressed before, but the simultaneous, or synchronous, compression and transmission of *time-dependent 3-D meshes with attributes and dynamic connectivity* remains an open problem. The models used in this thesis have been carefully selected as appropriate representatives of those models potentially used in typical 3-D applications, e.g., news or advertising clips (TELLY), cartoons and recreational animation (FRED), movies with visual effects (CHICKEN), games (ROBOT, BOUNCEBALL). Naturally, a broader range of models is needed for experimentation with time-

dependent attributes. Furthermore, if the new models are not isomorphic, this violates the basic assumption behind the basic 3D-Anim, and a compression scheme that jointly exploits spatial with temporal redundancy is required.

Improved error metrics. The proposed solutions for concealing dynamic 3-D meshes suffering geometry loss at the receiver have been evaluated mainly with automatic objective metrics, such as the 3-D PSNR. Although the proposed method already represents a step forward to streaming 3-D animation over best effort networks, further investigation of the proposed solutions is desirable with improved distortion metrics that also correlate to dynamic attributes, or to human vision perception of rendered models. In fact, as has also been established through natural video coding, the human eye is more sensitive to colour changes. Therefore, a colour artifact on an animated surface not captured by a geometry-only concealment scheme, even if its geometry has been perfectly concealed, may still result in higher visual distractions than geometric cracks.

In this context, extending the visual smoothness metric to capturing *attribute smoothness* has significant potential. Furthermore, it is worth investigating the use of a 3-D-to-2-D projection of a model's geometry in an improved metric as it may reveal significant correlation to the way human vision perceives a dynamic 3-D mesh. Subjective quality assessment of the resulting decoded models with attributes will be beneficial to determine the degree to which an extended visual metric correlates to the actual quality of a dynamic model.

Spatial prediction aided by interleaving. In receiver-based concealment for loss-suffering 3-D animation streams, the MV repetition method can benefit from a vertex *interleaving* scheme. A vertex interleaver can spread out the effect of loss on individual vertices, thus enabling better spatial prediction of a motion vector from its correctly received neighbouring motion vectors. Assuming frame fragmentation takes place, a vertex interleaver would spread neighbouring vertices into different packets. Loss of a single packet, or a short burst of packets, would affect selected vertices distributed over the whole mesh surface, rather than a local segment, had no interleaving taken place (as is currently the case with 3D-Anim).

LoD-based and RoI-aware animation compression. The proposed layering mechanisms can be revisited and extended to follow level-of-detail-based (LoD-based) compression of 3-D animation. As discussed in chapter 3, the proposed multi-resolution coding of animated 3-D models relied on the idea of simultaneous decimation and temporal compression of a 3-D mesh sequence. Following the inverse paradigm, that of simultaneous *refinement and encoding of dynamic meshes*, a dynamic 3-D object examined during a remote walkthrough may be better compressed by a source coding mechanism that is able to efficiently express the

successive LoD refinement levels.

In a parallel extension, calculating face normals is now a cheap operation in most modern CPUs. Normal vectors can be used to determine the visible part of a mesh's surface, and optimally allocate more bits to encode this part. In accordance with the adaptive layer allocation, a *dynamic visibility-based layering scheme* can be built, able to compute the user's region of interest (RoI) in real-time, and encode and transmit this region to a potential receiver. Such a solution may result in significant rate savings over 'static' transmissions.

Better primitives for hardware-assisted temporal compression. A greater challenge is to find a small set of compression/decompression primitives that are well-matched to current, and next generation, graphics hardware, but general enough to capture the wide variety of animated geometry. Matching the existing triangle strips and fans may not lead to better compression ratios. Not matching existing primitives may yield to inefficient use of the rendering engine. This seeming trade-off could be addressed by experimenting with new rendering pipeline technologies, that consider temporal coherence of animations.

Appendix A

3-D Animation Sequences

A.1 Sequence Details

This appendix summarises the animation sequence details.

For each sequence, the table on the left column lists the frame rate, number of frames in the sequence, and number of nodes in the 3-D mesh. The lower part of the table lists the number of faces (rectangles or triangles) and vertices for each node and the total number of faces and vertices in the 3-D model.

On the right column a node hierarchy map and a screenshot of each rendered model are provided. If a model was not originally rendered, only its wireframe is given.

A.1.1 TELLY

This sequence is by courtesy of Jörn Ostermann (AT&T Research).

Table A.1: Node details of wireframe TELLY.

Frame Rate	30 Hz
Frames	780
Nodes	9

	Rectangles	*Vertices*
SKIN	552	471
UPPERLIP	24	29
LOWERLIP	30	34
TEETH	32	128
TONGUE	170	192
EYELASH	20	22
EYEBROW	12	24
EYECORNER	2	6
NOSTRIL	6	12
Total	848	918

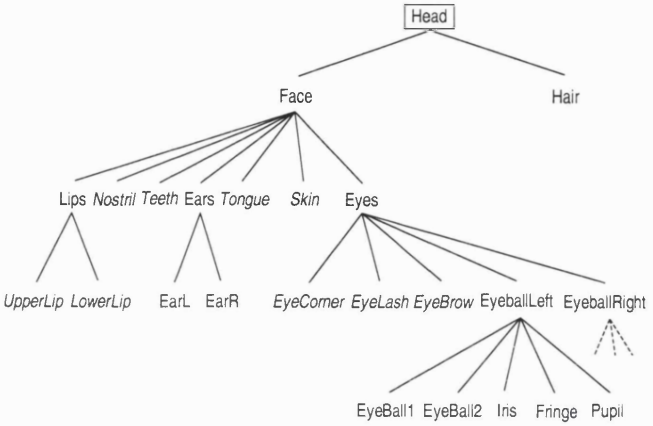


Figure A.1: Node hierarchy of wireframe TELLY.

Figure A.2: Wireframe model TELLY after rendering.

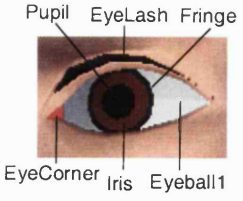


Figure A.3: Eye detail of wireframe TELLY.

A.1.2 BOUNCEBALL

This sequence is by courtesy of Jeong-Hwan Ahn (Seoul National University).

Table A.2: Node details of wireframe BOUNCEBALL.

Frame Rate	24 Hz
Frames	528
Nodes	1

	Triangles	*Vertices*
BOUNCEBALL	320	162
Total	320	162

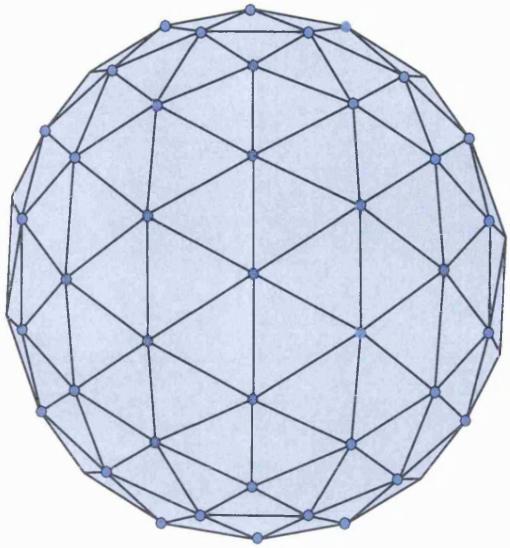


Figure A.4: Wireframe model BOUNCEBALL after rendering.

A.1.3 CHICKEN

This sequence is by courtesy of Jerome Lengyel (Microsoft Research).

Table A.3: Node details of wireframe CHICKEN.

Frame Rate	15 Hz
Frames	400
Nodes	1

	Triangles	*Vertices*
CHICKEN	5664	3030
Total	5664	3030

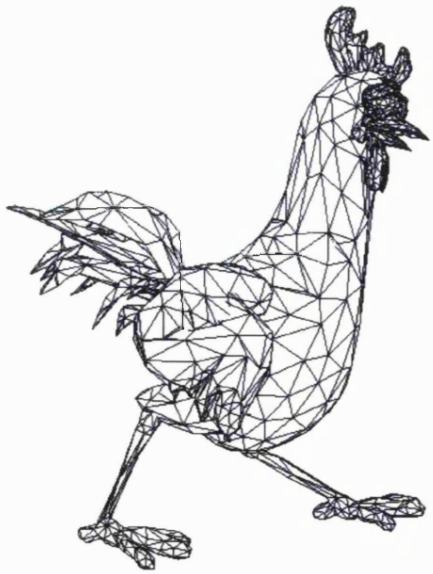


Figure A.5: Wireframe model CHICKEN without rendering.

A.1.4 FRED

This sequence is by courtesy of MPEG-4.

Table A.4: Node details of wireframe FRED.

Frame Rate	15 Hz
Frames	100
Nodes	24

	Triangles	Vertices
HEAD	226	121
EAR_L	82	43
EAR_R	82	43
MOUTH_BO	6	8
LIPS_U	16	12
NOSTRILS	10	12
MOUTH_U	78	48
MOUTH_BI	6	8
LIPS_D	16	12
MOUTH_D	32	22
EYEBROW_L	22	13
EYEBROW_R	22	13
EYELID_LU	40	25
EYELID_LD	40	25
EYEBALL_L	40	25
IRIS_L	8	9
PUPIL_L	8	9
EYELID_RU	40	25
EYELID_RD	40	25
EYEBALL_R	40	25
IRIS_R	8	9
PUPIL_R	8	9
HAND_L	106	55
HAND_R	106	55
Total	1082	651

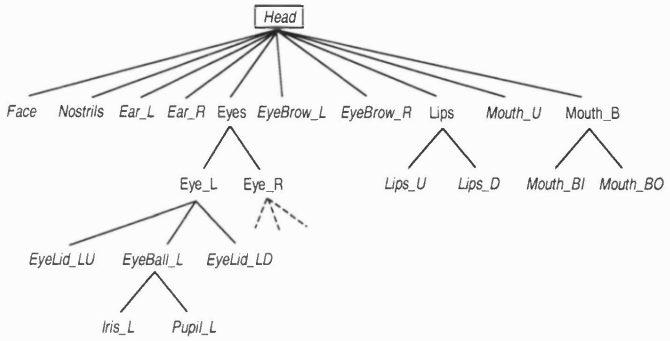


Figure A.6: Node hierarchy of wireframe FRED.

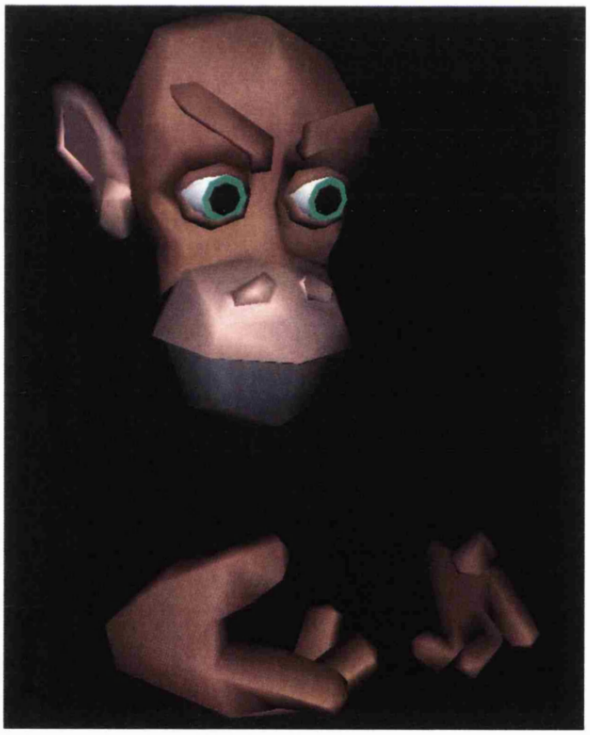


Figure A.7: Wireframe model FRED after rendering.

A.1.5 ROBOT

This sequence is by courtesy of MPEG-4.

Table A.5: Node details of wireframe ROBOT.

Frame Rate	15 Hz
Frames	100
Nodes	32

	Triangles	*Vertices*
CIRCLE05	694	355
CIRCLE10	694	355
LEFT_ARM	812	438
LEFT_FINGER_1	228	116
LEFT_FINGER_2	228	116
LEFT_FINGER_3	228	116
LEFT_PALM	224	114
MOLUSK_BODY	46	92
MOLUSK_BODY-1	438	311
MOLUSK_BODY-2	138	184
MOLUSK_BODY-3	4262	2259
OBJECT02	224	118
OBJECT03	224	118
RECTANGLE08	30	20
RECTANGLE09	30	20
RECTANGLE10	30	20
RECTANGLE11	30	20
RECTANGLE12	30	20
RECTANGLE13	30	20
RECTANGLE14	30	20
RECTANGLE15	30	20
RECTANGLE16	30	20
RECTANGLE17	30	20
RECTANGLE18	30	20
RECTANGLE19	30	20
RIGHT_ARM	812	438
RIGHT_FINGER_1	228	116
RIGHT_FINGER_2	228	116
RIGHT_FINGER_3	228	116
RIGHT_PALM	224	114
THIGH_L	1008	524
THIGH_R	1008	524
Total	12536	6880

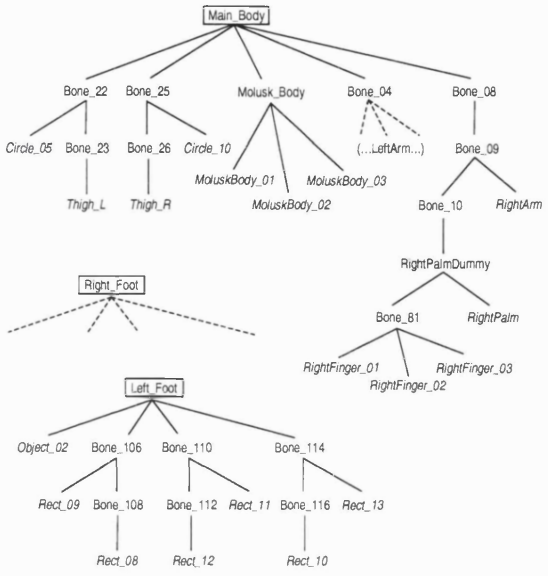


Figure A.8: Node hierarchy of wireframe ROBOT.

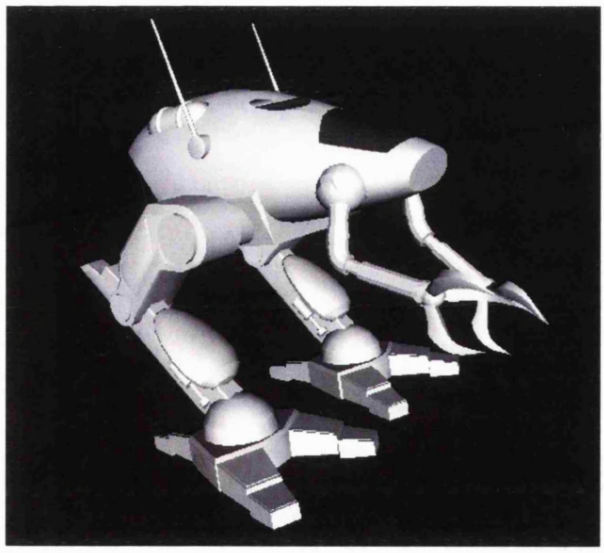


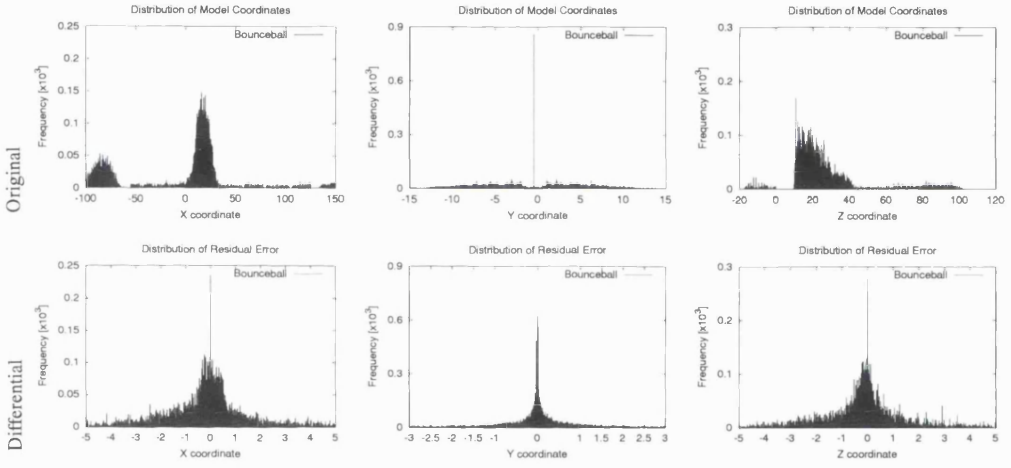
Figure A.9: Wireframe model ROBOT after rendering.

A.2 Histograms

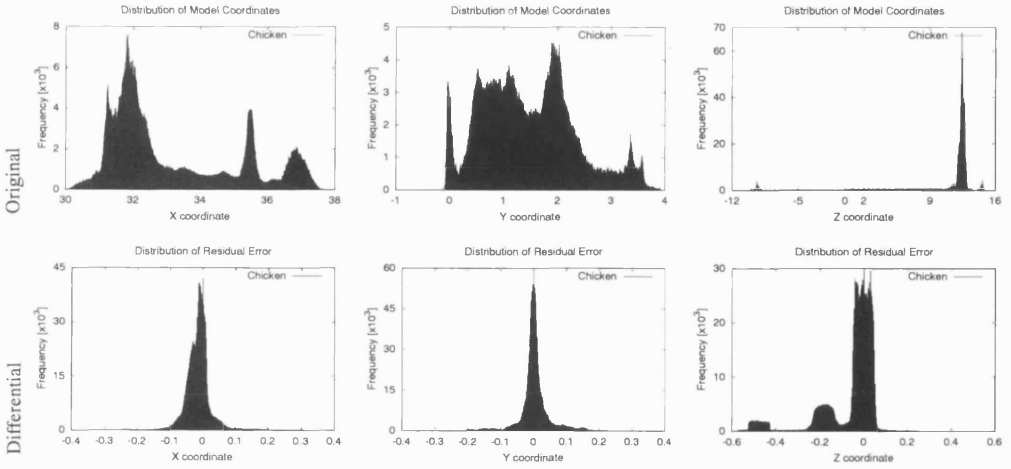
This section shows how the dynamic range of the differential signal is significantly reduced in comparison to the original absolute vertex coordinates. Each vertex coordinate component X, Y, Z, of the animated wireframes is shown. The table below summarises entropy estimates and actual rates.

Sequence	Rate (bits/vertex component)	Entropy (per component)
BOUNCEBALL	5.0	4.149
CHICKEN	3.9	2.741
FRED	5.5	4.305
ROBOT	4.9	3.221
TELLY	6.2	4.877

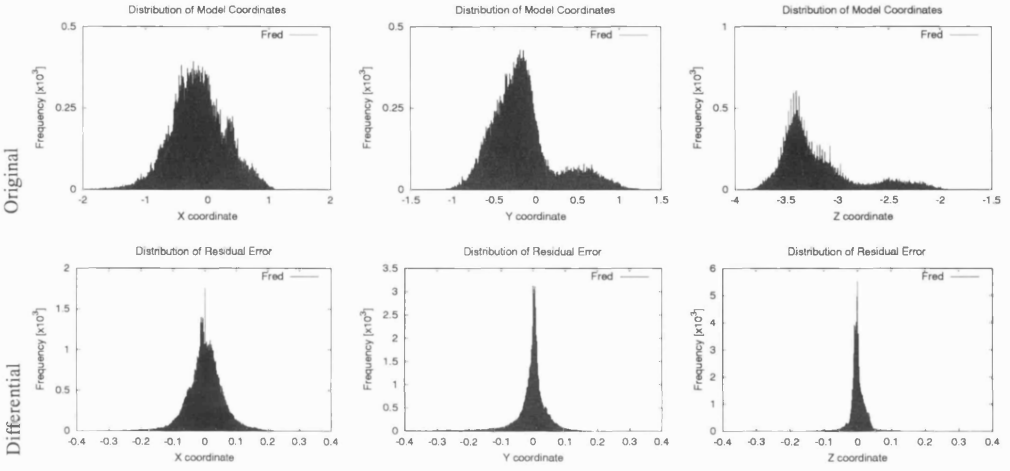
BOUNCEBALL



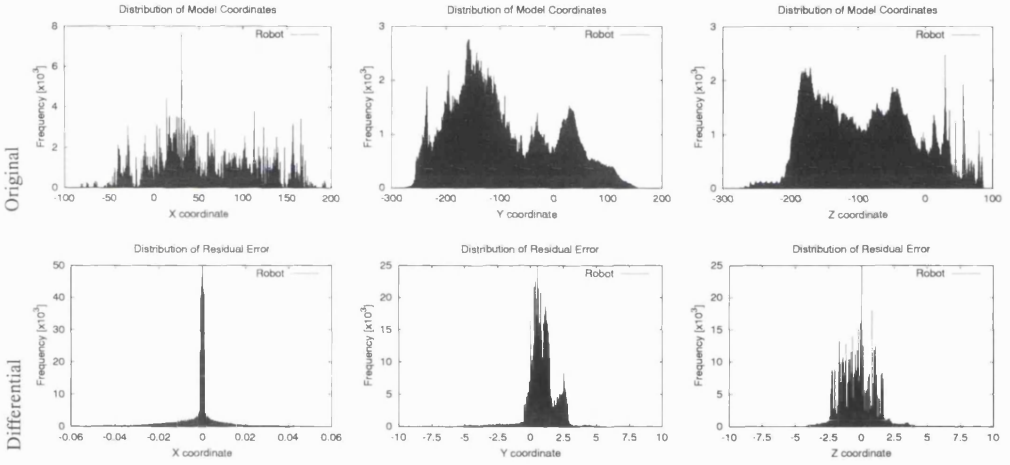
CHICKEN



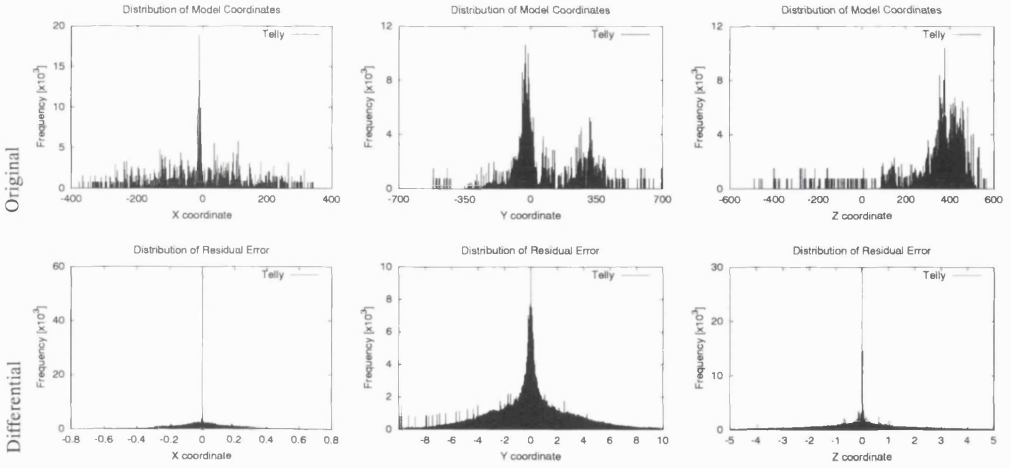
FRED



ROBOT



TELLY



Appendix B

Bitstream Syntax of 3D-Anim

This appendix provides the bit stream syntax of the 3-D animation coding scheme. The convention followed throughout the syntax is as follows. The second column of the table specifies the number of bits the entity in the first column occupies in the bit stream. The third column specifies the data type of the entity in the first column.

The syntax is quite straightforward. First, the explanation of constants occurring in the beginning of the bitstream are given as shown in the packet payload format in figure 3.14. These are followed by the syntax table as described in the previous paragraph.

- **FrameLen** - Length of frame in bytes.
- **NodeMask** - Bit mask that specifies which nodes in the model are animated.
- **VertexMaskXLen** - Length in bytes of the Vertex Mask (see below) for the X coordinate.
- **VertexMaskYLen** - Length in bytes of the Vertex Mask (see below) for the Y coordinate.
- **VertexMaskZLen** - Length in bytes of the Vertex Mask (see below) for the Z coordinate.
- **VertexMaskX** - Bit mask that specifies which vertices are animated in the current node along the X axis.
- **VertexMaskY** - Bit mask that specifies which vertices are animated in the current node along the Y axis.
- **VertexMaskZ** - Bit mask that specifies which vertices are animated in the current node along the Z axis.
- **XdataLen** - Length in bytes of the animation data for the X coordinate.
- **YdataLen** - Length in bytes of the animation data for the Y coordinate.
- **ZdataLen** - Length in bytes of the animation data for the Z coordinate.
- **Xdata** - Vertex displacements along the X axis.
- **Ydata** - Vertex displacements along the Y axis.
- **Zdata** - Vertex displacements along the Z axis.

decode_frame() {		
num_nodes = *number of nodes counted in NodeMask*	8	byte
for (k=1; k ≤ num_nodes; k++)		
{		
decode_node();		
}		
}		
decode_node() {		
num_vertices_X = *number of vertices counted in VertexMaskX*	16	ushort
num_vertices_Y = *number of vertices counted in VertexMaskY*	16	ushort
num_vertices_Z = *number of vertices counted in VertexMaskZ*	16	ushort
for (k=1; k ≤ num_vertices_X; k++)		
{		
decode_vertices(X);		
}		
for (k=1; k ≤ num_vertices_Y; k++)		
{		
decode_vertices(Y);		
}		
for (k=1; k ≤ num_vertices_Z; k++)		
{		
decode_vertices(Z);		
}		
}		
decode_vertices(axis) {		
vtxbits = *number of bits next vertex is encoded at*		
while (!vtx_bitstream_end)		
{		
vtxbits = ArithDecode(nextVertex);	vtxbits	bits
}		
}		

Appendix C

The Visual Smoothness Metric: An Example

This appendix provides intuition on the Visual Smoothness metric (VS), by use of a practical example on a hypothetical surface. The example aims at highlighting the importance of the VS metric in 3-D wireframe animation measurements.

Assume the original surface shown on the left of figure C.1 with 5 vertices $\{A, B, C, D, E\}$ and the corresponding coordinates shown in table C.1 for 2 examples. Assume also that the decoding process introduces coding error only for the y-coordinate of vertex A, whereas the remaining 4 vertices $\{B, C, D, E\}$ are perfectly decoded, without coding errors. The decoded coordinates at the receiver, denoted by $\{\hat{A}, \hat{B}, \hat{C}, \hat{D}, \hat{E}\}$, are displayed in table C.1 for two numerical examples.

Table C.1: Two numerical examples of vertex coordinates for the surface in figure C.1.

Example 1		Example 2	
A(0,1,0)	$\hat{A}$(0,5,0)	$A(0,\frac{1}{2},0)$	$\hat{A}$(0,2,0)
B(-1,0,0)	$\hat{B}$(-1,0,0)	$B(-\frac{3}{2},0,-1)$	$\hat{B}(-\frac{3}{2},0,-1)$
C(0,0,1)	$\hat{C}$(0,0,1)	C(-1,0,1)	$\hat{C}$(-1,0,1)
D(1,0,0)	$\hat{D}$(1,0,0)	D(1,0,1)	$\hat{D}$(1,0,1)
E(0,0,-1)	$\hat{E}$(0,0,-1)	$E(\frac{3}{2},0,-1)$	$\hat{E}(\frac{3}{2},0,-1)$

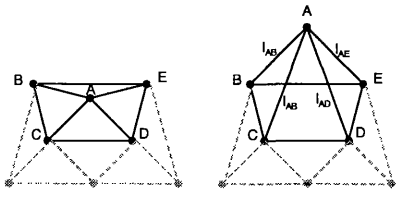


Figure C.1: Example of a Laplacian operator on a hypothetical surface. Left: original surface. Right: reconstructed surface with distortion on vertex A.

Then, the calculation of the quantities GL(m_A) and GL($\hat{m}_A$) for the original and decoded meshes in equation (5.5) for the first example (left of table C.1) is as follows:

$$\begin{aligned} GL(m_A) &= m_A - \frac{\frac{m_B}{l_{AB}} + \frac{m_C}{l_{AC}} + \frac{m_D}{l_{AD}} + \frac{m_E}{l_{AE}}}{\frac{1}{l_{AB}} + \frac{1}{l_{AC}} + \frac{1}{l_{AD}} + \frac{1}{l_{AE}}} \\ &= m_A - \frac{\frac{m_B}{\sqrt{2}} + \frac{m_C}{\sqrt{2}} + \frac{m_D}{\sqrt{2}} + \frac{m_E}{\sqrt{2}}}{\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}} \\ &= m_A - \frac{1}{4}(m_B + m_C + m_D + m_E) \ , \end{aligned}$$

and:

$$GL(\hat{m}_A) = \hat{m}_A - \frac{1}{4}(\hat{m}_B + \hat{m}_C + \hat{m}_D + \hat{m}_E) \ .$$

Similarly, the quantities GL for the other vertices can be derived:

$$\begin{aligned} GL(m_B) &= m_B - \frac{1}{3}(m_A + m_C + m_E) \ , \\ GL(\hat{m}_B) &= \hat{m}_B - \frac{1}{2 + \frac{1}{\sqrt{13}}}(\frac{1}{\sqrt{13}}\hat{m}_A + \hat{m}_C + \hat{m}_E) \ , \\ \\ GL(m_C) &= m_C - \frac{1}{3}(m_A + m_B + m_D) \ , \\ GL(\hat{m}_C) &= \hat{m}_C - \frac{1}{2 + \frac{1}{\sqrt{13}}}(\frac{1}{\sqrt{13}}\hat{m}_A + \hat{m}_B + \hat{m}_D) \ , \\ \\ GL(m_D) &= m_D - \frac{1}{3}(m_A + m_C + m_E) \ , \\ GL(\hat{m}_D) &= \hat{m}_D - \frac{1}{2 + \frac{1}{\sqrt{13}}}(\frac{1}{\sqrt{13}}\hat{m}_A + \hat{m}_C + \hat{m}_E) \ , \\ \\ GL(m_E) &= m_E - \frac{1}{3}(m_A + m_B + m_D) \ , \\ GL(\hat{m}_E) &= \hat{m}_E - \frac{1}{2 + \frac{1}{\sqrt{13}}}(\frac{1}{\sqrt{13}}\hat{m}_A + \hat{m}_B + \hat{m}_D) \ . \end{aligned} \tag{C.1}$$

Note, also, that for the above example it holds $m_A \neq \hat{m}_A$, but $m_B = \hat{m}_B$, $m_C = \hat{m}_C$, $m_D = \hat{m}_D$, $m_E = \hat{m}_E$.

Let $a = -\frac{1}{3}$ and $b = \frac{1}{2\sqrt{13}+1}$. Substituting the GL derivations in equation (C.1) above to equation (5.5), and $n = 5$, we can rewrite equation (5.5), as:

$$\begin{aligned} VS = \frac{1}{2 \cdot 5}(&2 \cdot \|m_A - \hat{m}_A\| + \\ &+ 2 \cdot \|a \cdot m_A + b \cdot \hat{m}_A + (a+b) \cdot m_C + (a+b) \cdot m_E\| + \\ &+ 2 \cdot \|a \cdot m_A + b \cdot \hat{m}_A + (a+b) \cdot m_B + (a+b) \cdot m_D\|) \ . \end{aligned}$$

We can now easily calculate the overall VS distortion for the surface of figure C.1:

$$VS = 0.910 \ .$$

Equivalently, the VS distortion for the second example on table C.1 is calculated as:

$$VS = 0.463 \ .$$

Bibliography

[AD01a] Pierre Alliez and Mathieu Desbrun. Progressive compression for lossless transmission of triangle meshes. In *SIGGRAPH 2001 Conference Proceedings*, pages 198–205, 2001.

[AD01b] Pierre Alliez and Mathieu Desbrun. Valence-driven connectivity encoding of 3-D meshes. In A. Chalmers and T.-M. Rhyne, editors, *Eurographics Conference Proceedings, (volume 20)*, pages 480–489. Blackwell Publishers, September 2001.

[AKKH01] Jeong-Hwan Ahn, Chang-Su Kim, C.-C. Jay Kuo, and Yo-Sung Ho. Motion-compensated compression of 3-D animation models. *IEE Electronics Letters*, 37(24):1445–1446, November 2001.

[AKKH02] Jeong-Hwan Ahn, Chang-Su Kim, C.-C. Jay Kuo, and Yo-Sung Ho. Motion compensated coding of 3-D animation models. In *Visual Communications and Image Processing*, San José, California, January 2002. SPIE.

[AL96] Andres Albanese and Michael Luby. PET – Priority Encoding Transmission. *High-Speed Networking for Multimedia Applications – Kluwer Academic Publishers*, March 1996.

[ALSS99] Pierre Alliez, Nathalie Laurent, Henri Sanson, and Francis Schmitt. Mesh approximation using a volume-based metric. In *Pacific Graphics '99 Conference Proceedings*, pages 292–301, October 1999.

[AM00] Marc Alexa and Wolfgang Müller. Representing animations by principal components. *Computer Graphics Forum*, 19(3):411–418, August 2000. ISSN 1067–7055.

[ASCE02] Nicolas Aspert, Diego Santa-Cruz, and Touradj Ebrahimi. MESH: Measuring errors between surfaces using the Hausdorff distance. *IEEE International Conference on Multimedia and Expo*, August 2002.

[AWTW02] John Apostolopoulos, Tina Wong, Wai-Tian Tan, and Susie Wee. On Multiple Description Streaming with Content Delivery Networks. In *IEEE INFOCOM*, 2002.

[BCPZ98] Chandrajit Bajaj, Steven Cutchin, Valerio Pascucci, and Guozhong Zhuang. Error resilient streaming of compressed VRML. Technical report, TICAM, The University of Texas at Austin, 1998.

[BK02a] Stephan Bischoff and Leif Kobbelt. Streaming 3-D geometry data over lossy communication channels. *IEEE International Conference on Multimedia and Expo*, August 2002.

[BK02b] Stephan Bischoff and Leif Kobbelt. Towards robust broadcasting of geometry data. In *Computers & Graphics*, 2002.

[BM98] Mark R. Bolin and Gary W. Meyer. A perceptually based adaptive sampling algorithm. In *Proceedings of the 25^{th} annual conference on computer graphics and interactive techniques*, 1998.

[BM03] Ioana M. Boier-Martin. Adaptive graphics. *IEEE Computer Graphics & Applications*, 23(1):6–10, January 2003.

[Bol93] Jean-Chrysostome Bolot. End-to-end packet delay and loss behavior in the Internet. *ACM SIGCOMM*, pages 289–298, September 1993.

[Bos99] Frank Bossen. *On The Art Of Compressing Three-Dimensional Polygonal Meshes And Their Associated Properties.* PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), June 1999.

[BPZ98] Chandrajit Bajaj, Valerio Pascucci, and Guozhong Zhuang. Compression and coding of large CAD models. Technical report, TICAM, The University of Texas at Austin, 1998.

[BRS99] Hari Balakrishnan, Hariharan Rahul, and Srinivasan Seshan. An integrated congestion management architecture for Internet hosts. In *ACM SIGCOMM*, pages 175–187, Cambridge, MA, September 1999.

[BS02] Mikaël Bourges-Sévenier. An introduction to MPEG-4 Animation Framework eXtension (AFX). *IEEE International Conference on Image Processing ICIP 2002*, 3:1–4, September 2002.

[BTP97] Petrus J.L. van Beek, Murat Tekalp, and Atul Puri. 2-D mesh geometry and motion compression for efficient object-based video representation. In *Intl. Conf. on Image Processing (ICIP '97)*, volume 3, pages 440–443, October 1997.

[BVCL01] Andrea Basso, Socrates Varakliotis, Roberto Castagno, and Florin Lohan. Transport of MPEG-4 over IP/RTP. *European Trans. on Telecommunications: special issue on Packet Video 2000 Workshop*, 12(3), May–June 2001.

[CCMS97] Andrea Ciampalini, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. Multi-resolution decimation based on global error. *The Visual Computer*, 13(5):228–246, 1997.

[CDM03] Cyril Concolato, Jean-Claude Dufourd, and Jean-Claude Moissinac. Encoding of Cartoons Using MPEG-4 BIFS: Methods and Results. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(11):1129–1135, November 2003.

[Cho97] Mike M. Chow. Optimized geometry compression for real-time rendering. In *Proceedings of the conference on Visualization '97*, 1997.

[CKL+00] Jin-Soo Choi, Yong-Han Kim, Ho-Jang Lee, In-Sung Park, Myoung-Ho Lee, and Chieteuk Ahn. Geometry compression of 3-D mesh models using predictive two-stage quantization. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(2):312–322, March 2000.

[CMS98] Paolo Cignoni, Claudio Montani, and Roberto Scopigno. A comparison of mesh simplification algorithms. *Computers and Graphics*, 22(1):37–54, 1998.

[CN02] Bing-Yu Chen and Tomoyuki Nishita. Multiresolution streaming mesh with shape preserving and QoS-like controlling. In *Proceeding of the Seventh International Conference on 3D Web Technology*, 2002.

[COI02] Daniel Cohen-Or and Revital Irony. Multi-way geometry coding. Technical report, The School of Computer Science, Tel-Aviv University, 2002.

[COLR99] Daniel Cohen-Or, David Levin, and Offir Remez. Progressive compression of arbitrary triangular meshes. In David Ebert, Markus Gross, and Bernd Hamann, editors, *IEEE Visualization '99*, pages 67–72, San Francisco, 1999.

[COMF99] Daniel Cohen-Or, Yair Mann, and Shachar Fleishman. Deep compression for streaming texture intensive animations. In Alyn Rockwood, editor, *SIGGRAPH 1999, Computer Graphics Proceedings*, pages 261–268, Los Angeles, 1999. Addison Wesley Longman.

[CPW03] Philip Chou, Venkat Padmanabhan, and Helen Wang. Resilient peer-to-peer streaming. Technical Report MSR-TR-2003-11, Microsoft Research, Redmond, WA, 2003.

[CRS98] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. METRO: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, June 1998.

[Cru03] Diego Santa Cruz. *Compression of parametric 3-D models with NURBS*. PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), May 2003.

[CT90] David D. Clark and David L. Tennenhouse. Architectural considerations for a new generation of network protocols. *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 200–208, September 1990.

[CVM+96] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick Brooks, and William Wright. Simplification envelopes. *Computer Graphics*, 30:119–128, 1996.

[CWP03] Philip Chou, Helen Wang, and Venkat Padmanabhan. Layered multiple description coding. *13^{th} International Packet Video Workshop*, April 2003.

[Dal93] Scott Daly. Digital images and human vision. In A.B. Watson, editor, *The visible differences predictor: an algorithm for the assessment of image fidelity, (chapter 14)*, pages 179–206. The MIT Press, 1993.

[dCSR01] Philippe de Cuetos, Despina Saparilla, and Keith Ross. Adaptive streaming of stored video in a TCP-friendly context : Multiple versions or multiple layers. *11^{th} International Packet Video Workshop*, April 2001.

[Dee95] Michael Deering. Geometry Compression. In *22^{nd} ACM Annual Conference on Computer Graphics and Interactive Techniques*, pages 13–20, September 1995.

[Ell65] E.O. Elliott. A model of the switched telephone network for data communications. *Bell Systems Technical Journal*, 44(1):89–109, January 1965.

[FCOIZ01] Efi Fogel, Daniel Cohen-Or, Revital Ironi, and Tali Zvi. A web architecture for progressive delivery of 3D content. In *Proceedings of the sixth international conference on 3D Web technology*, pages 35–41, Paderbon, Germany, 2001. ACM Press.

[FHPW00] Sally Floyd, Mark Handley, Jitendra Padhye, and Jörg Widmer. Equation-based congestion control for unicast applications. In *ACM SIGCOMM*, pages 43–56, Stockholm, Sweden, August 2000.

[FvdZT+00] James Folwer, John van der Zwaag, Shivaraj Tenginakai, Raghu Machiraju, and Robert Moorhead. Decoding of large terrains using a hardware rendering pipeline. Technical Report MSSU-COE-ERC-00-13, Mississippi State University, Mississippi, MI, 2000.

[Gar99a] Michael Garland. Multiresolution modeling: Survey & future opportunities. *EUROGRAPHICS – State of the Art Reports*, pages 111–131, May 1999.

[Gar99b] Michael Garland. *Quadric-Based Polygonal Surface Simplification.* PhD thesis, Carnegie Mellon University, School of Computer Science, May 1999.

[GB99] Enrico Gobbetti and Eric Bouvier. Time-critical multiresolution scene rendering. In David Ebert, Markus Gross, and Bernd Hamann, editors, *IEEE Visualization '99*, pages 123–130, San Francisco, 1999. IEEE Computer Society Press.

[GBTS99] André P. Guéziec, Frank Bossen, Gabriel Taubin, and Cláudio T. Silva. Efficient compression of non-manifold polygonal meshes. In David Ebert, Markus Gross, and Bernd Hamann, editors, *IEEE Visualization '99*, pages 73–80, San Francisco, 1999.

[GGK02] Craig Gotsman, Stefan Gumhold, and Leif Kobbelt. Simplification and compression of 3-D meshes. *Tutorial on Multiresolution in Geometric Modelling*, 2002.

[GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. *Computer Graphics Annual Conference Series*, 31:209–216, 1997.

[GH98] Michael Garland and Paul S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In David Ebert, Hans Hagen, and Holly Rushmeier, editors, *IEEE Visualization '98*, pages 263–270, 1998.

[GS98] Stefan Gumhold and Wolfgang Straßer. Real time compression of triangle mesh connectivity. *Computer Graphics*, 32:133–140, 1998.

[GSK02] Sumit Gupta, Kuntal Sengupta, and Ashraf A. Kassim. Compression of 3D Dynamic Geometry Data using Iterative Closest Point Algorithm. *IEEE Computer Vision and Image Understanding*, 87(3):116–130, July 2002.

[GSK03] Sumit Gupta, Kuntal Sengupta, and Ashraf A. Kassim. Registration and partitioning-based compression of 3-D dynamic data. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(11):1144–1155, November 2003.

[GTHL99] André Guéziec, Gabriel Taubin, Bill Horn, and Francis Lazarus. A framework for streaming geometry in VRML. *IEEE Computer Graphics & Applications*, 19(2):68–78, March/April 1999.

[Gué95] André P. Guéziec. Surface simplification inside a tolerance volume. In *Second Annual International Symposium on Medical Robotics and Computer Assisted Surgery*, pages 132–139, Baltimore, MD, 1995.

[HDD+93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *ACM SIGGRAPH'93*, pages 19–26, 1993.

[HM00] Hugues Hoppe and Steve Marschner. Efficient minimization of new quadric metric for simplifying meshes with appearance attributes. Technical report, Microsoft Research, MSR-TR-2000-64, June 2000.

[Hop96] Hugues Hoppe. Progressive Meshes. *ACM SIGGRAPH*, 30:99–108, 1996.

[Hop97] Hugues Hoppe. View-dependent refinement of progressive meshes. In *ACM SIGGRAPH'97*, pages 189–198, 1997.

[Hop99] Hugues Hoppe. New quadric metric for simplifying meshes with appearance attributes. In David Ebert, Markus Gross, and Bernd Hamann, editors, *IEEE Visualization '99*, pages 59–66, San Francisco, 1999.

[HP03] Mark Handley and Colin Perkins. *Guidelines for Writers of RTP Payload Format Specifications*. RFC 2736, IETF, December 2003.

[HSLG99] Uwe Horn, Klaus Stuhlmüller, Michael Link, and Bernd Girod. Robust Internet Video Transmission Based on Scalable Coding and Unequal Error Protection. *Signal Processing: Image Communication, Special Issue on Real-time Video over the Internet*, 15(1–2):77–94, September 1999.

[IA02] Martin Isenburg and Pierre Alliez. Compressing polygon mesh geometry with parallelogram prediction. *Visualisation 2002*, pages 141–146, October 2002.

[ISO98] ISO/IEC JTC1/SC29/WG11. W2066, Description of Core Experiments on 3-D model coding, February 1998.

[ISO99] ISO/IEC JTC1/SC29/WG11. 14496/Amd1 extensions (MPEG-4 v.2), December 1999.

[ISO01a] ISO/IEC JTC1/SC29/WG11. 14496-1: Information technology – Coding of audio-visual objects – Part 1, Systems (2^{nd} edition), 2001.

[ISO01b] ISO/IEC JTC1/SC29/WG11. 14496-2: Information technology – Coding of audio-visual objects - Part 2, Visual (2^{nd} edition), 2001.

[ISO01c] ISO/IEC JTC1/SC29/WG11. N7199, challenges of mapping cartoons to BIFS, July 2001.

[ISO02] ISO/IEC JTC1/SC29/WG11. N4852, study of ISO/IEC 14496-1:2001/PDAM4: Animation Framework eXtension and Multi User Worlds (MPEG-4 v.5), May 2002.

[Jan00] Euee S. Jang. 3-D Animation Coding: its History and Framework. *IEEE International Conference on Multimedia and Expo*, August 2000.

[JMT02] Chris Joslin and Nadia Magnenat-Thalmann. MPEG-4 animation clustering for networked virtual environments. *IEEE International Conference on Multimedia and Expo*, August 2002.

[KA01] Taehyun Kim and Mostafa H. Ammar. A comparison of layering and stream replication video multicast schemes. In *NOSSDAV'01*, June 2001.

[KADS02] Andrei Khodakovsky, Pierre Alliez, Mathieu Desbrun, and Peter Schröder. Near-optimal connectivity encoding of 2-manifold polygon meshes. In *The Journal of Graphical Models (special issue)*, 2002.

[KBB+00] Leif Kobbelt, Stephan Bischoff, Mario Botsch, Kolja Kähler, Christian Rössl, Robert Schneider, and Jens Vorsatz. Geometric modeling based on polygonal meshes. *Tutorials of the European Association for Computer Graphics 21st Annual Conference (Eurographics-00)*, pages 1–47, August 2000.

[KG00] Zachi Karni and Craig Gotsman. Spectral compression of mesh geometry. In Kurt Akeley, editor, *SIGGRAPH 2000, Computer Graphics Proceedings*, pages 279–286. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[KH03] Sung-Yeol Kim and Yo-Sung Ho. View-dependent transmission of 3-D mesh models using hierarchical partitioning. In *Visual Communications and Image Processing*, Lugano, Switzerland, July 2003. SPIE.

[KJH+02] James D.K. Kim, Seok Yoon Jung, Mahnjin Han, Euee S. Jang, Sang Oak Woo, Shin Jun Lee, and Gyeong Ja Jang. Animation data compression in MPEG-4: Interpolators. *IEEE International Conference on Image Processing ICIP 2002*, 3:33–36, September 2002.

[KLS96] Reinhard Klein, Gunther Liebich, and Wolfgang Straßer. Mesh reduction with error control. In Roni Yagel and Gregory M. Nielson, editors, *IEEE Visualization '96*, pages 311–318, 1996.

[Kob97] Leif Kobbelt. Discrete fairing. In *Seventh IMA Conference on the Mathematics of Surfaces*, pages 101–131, 1997.

[Kri03] Jan Krikke. Samurai Romanesque, J2ME, and the Battle for Mobile Cyberspace. *IEEE Computer Graphics & Applications*, 23(1):16–23, January 2003.

[KSS00] Andrei Khodakovsky, Peter Schröder, and Wim Sweldens. Progressive geometry compression. In Kurt Akeley, editor, *SIGGRAPH 2000, Computer Graphics Proceedings*, pages 271–278. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[LDW97] Michael Lounsbery, Tony DeRose, and Joe Warren. Multi-resolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 1(16):34–73, January 1997.

[Len99] Jerome Edward Lengyel. Compression of time-dependent geometry. In *ACM Symposium on Interactive 3-D graphics*, pages 89–95, Atlanta, Georgia, 26–29 April 1999.

[LGTW98] Rynson W.H. Lau, Mark Green, Danny S.P. To, and Janis Wong. Real-time continuous multi-resolution method for models of arbitrary topology. *Presence: Teleoperators and Virtual Environments*, 7(1):22–35, 1998.

[LKSS00] Ulf Labsik, Leif Kobbelt, Robert Schneider, and Hans-Peter Seidel. Progressive transmission of subdivision surfaces. *Computational Geometry: Theory and Applications*, 15:25–39, 2000.

[LL98] Yong-Gu Lee and Kunwoo Lee. Geometric detail supression by the Fourier Transform. *Computer-Aided Design*, 30(9):677–693, 1998.

[LLK97] Jiankun Li, Jin Li, and C.-C. Jay Kuo. Progressive compression of 3D graphic models. In *International Conference on Multimedia Computing and Systems*, pages 135–142, 1997.

[LLV+03] Adam Li, Fang Liu, John Villasenor, Jeong-Hoon Park, Dong-Seek Park, Yung-Lyul Lee, Jonathan Rosenberg, and Henning Schulzrinne. An RTP payload format for generic FEC with ULP. Internet draft, Internet Engineering Task Force, 2003. Work in progress.

[LNDB00] Gauthier Lafruit, Lode Nachtergaele, Kristof Denolf, and Jan Bormans. 3-D computational graceful degradation. *IEEE ISCAS, Workshop and Exhibition on MPEG-4*, pages 547–550, May 2000.

[LPK+00] Kang-Won Lee, Rohit Puri, Tae-Eun Kim, Kannan Ramchandran, and Vaduvur Bharghavan. An Integrated Source Coding and Congestion Control Framework for Video Streaming in the Internet. In *IEEE INFOCOM*, pages 747–756, 2000.

[LT98] Peter Lindstrom and Greg Turk. Fast and memory efficient polygonal simplification. In *IEEE Visualization*, pages 279–286, 1998.

[LT00] Peter Lindstrom and Greg Turk. Image-driven simplification. *ACM Transactions on Graphics (TOG)*, 19(3), 2000.

[Lub95] Jefrey Lubin. A visual discrimination model for imaging system design and evaluation. In Eli Peli, editor, *Vision Models for Target Detection and Recognition, (chapter 10)*, pages 245–283. World Scientific Publishing, 1995.

[LWPW03] Guenther Liebl, Marcel Wagner, Juergen Pandel, and Wenrong Weng. An RTP payload format for erasure-resilient transmission of progressive multimedia streams. Internet draft, Internet Engineering Task Force, 2003. Work in progress.

[Mar02] Ioana M. Martin. Hybrid transcoding for adaptive transmission of 3-D content. *IEEE International Conference on Multimedia and Expo*, August 2002.

[McC96] Steven McCanne. *Scalable Compression and Transmission of Internet Multicast Video*. PhD thesis, Computer Science Division, University of California, Berkeley, December 1996.

[MGS+02] Francisco Morán, Patrick Gioia, Michael Steliaros, Mikaël Bourges-Sévenier, and Narciso García. Subdivision surfaces in MPEG-4. *IEEE International Conference on Image Processing ICIP 2002*, 3:5–8, September 2002.

[MNW98] Alistair Moffat, Radford Neal, and Ian H. Witten. Arithmetic Coding Revisited. *ACM Transactions on Information Systems*, 16(3):256–294, July 1998.

[MRL00] Alexander E. Mohr, Eve A. Riskin, and Richard E. Ladner. Unequal loss protection: Graceful degradation of image quality over packet erasure channels through forward error correction. *IEEE Journal on Selected Areas in Communication*, 18(6):819–829, June 2000.

[NCO03] Yuval Noimark and Daniel Cohen-Or. Streaming scenes to MPEG-4 video-enabled devices. *IEEE Computer Graphics & Applications*, 23(1):58–64, January 2003.

[NRL+02] Nam-Pham Ngoc, Wolfgang Van Raemdonck, Gauthier Lafruit, Geert Deconinck, and Rudy Lauwereins. A QoS framework for interactive 3-D applications. *The International Conference in Central Europe on Computer Graphics and Visualization (WSCG)*, pages 317–325, February 2002.

[OP99] Stephan Olbrich and Helmut Pralle. Virtual reality movies – real-time streaming of 3-D objects. *Computer Networks – The Challenge of Gigabit Networking*, 31(21):2215–2225, November 1999.

[PH97] Jovan Popović and Hugues Hoppe. Progressive simplicial complexes. In *ACM SIGGRAPH'97*, pages 217–224, 1997.

[PHH98] Colin Perkins, Orion Hodson, and Vicky Hardman. A survey of packet loss recovery techniques for streaming audio. *IEEE Network*, 12(5):40–48, September 1998.

[PR00] Renato Pajarola and Jarek Rossignac. Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):79–93, March 2000.

[RA01] Ghassan Al Regib and Yucel Altunbasak. A system level framework for streaming 3-D meshes over packet networks. In *International Conference on Networking ICN2001 (LNCS 2094)*, pages 745–753. Springer-Verlag, 2001.

[RA02] Ghassan Al Regib and Yucel Altunbasak. An unequal error protection method for packet loss resilient 3-D mesh transmission. In *IEEE INFOCOM*, 2002.

[RA03] Ghassan Al Regib and Yucel Altunbasak. 3TP: An application-layer protocol for streaming 3-D graphics. *IEEE International Conference on Multimedia and Expo*, July 2003.

[RARM02] Ghassan Al Regib, Yucel Altunbasak, Jarek Rossignac, and Russell Mersereau. Protocol for streaming compressed 3-D animations over lossy channels. *IEEE International Conference on Multimedia and Expo*, August 2002.

[RB93] Jarek Rossignac and Paul Borrel. Multi-resolution 3-D approximation for rendering complex scenes. *2^{nd} Conference on Geometric Modelling in Computer Graphics*, pages 453–465, 1993.

[Red96] Martin Reddy. SCROOGE: Perceptually-driven polygon reduction. *Computer Graphics Forum*, 15(4):191–203, 1996.

[Rej99] Reza Rejaie. *An end-to-end architecture for quality adaptive streaming applications in the Internet.* PhD thesis, Faculty of the Graduate School, University of Southern California, September 1999.

[Rej03] Reza Rejaie. On integration of congestion control with internet streaming applications. *13^{th} International Packet Video Workshop (Poster Session)*, April 2003.

[RFT02] Michaël Roy, Sebti Foufou, and Frédéric Truchetet. Generic attribute deviation metric for assessing mesh simplification algorithm quality. In *Proceedings of IEEE International Conference on Image Processing*, pages 817–820, 2002.

[RHE99] Reza Rejaie, Mark Handley, and Deborah Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet. In *IEEE INFOCOM*, pages 1337–1345, 1999.

[RK96] David W. Redmill and Nick G. Kingsbury. The EREC: An error-resilient technique for coding variable-length blocks of data. *IEEE Transactions on Image Processing*, 5(4):565–574, April 1996.

[RLS+02] Wolfgang Van Raemdonck, Gauthier Lafruit, Liesbeth E.F.M. Steffens, Clara C.M. Otero Pérez, and Reinder J. Bril. Scalable 3-D graphics processing in consumer terminals. *IEEE International Conference on Multimedia and Expo*, August 2002.

[Ros97] Jarek Rossignac. Geometric simplification and compression, multiresolution surface modeling, course notes #25. In *ACM SIGGRAPH'97*, pages 279–286, 1997.

[Ros99] Jarek Rossignac. Edgebreaker : Connectivity Compression for Triangle Meshes. *IEEE Transactions on Visualization and Computer Graphics*, pages 47–61, 1999.

[RR96] Rémi Ronfard and Jarek Rossignac. Full-range approximation of triangulated polyhedra. *Eurographics, Computer Graphics Forum*, 15(3):67–76, 1996.

[San00] Henning Sanneck. *Packet loss recovery and control for voice transmission over the Internet.* PhD thesis, Technical University of Berlin, October 2000.

[SBM01] Dinesh Shikhare, Sushil Bhakar, and Sudhir P. Mudur. Compression of Large 3D Engineering Models using Automatic Discovery of Repeating Geometric Features. In 6^{th} *International Fall Workshop on Vision, Modeling and Visualization (VMV2001)*, November 2001.

[SBM02] Dinesh Shikhare, S. Venkata Babji, and Sudhir P. Mudur. Compression Techniques for Distributed Use of 3D Data: An Emerging Media Type on the Internet. In *ICCC-2002*, Mumbai, India, 12–14 August 2002.

[SCFJ03] Henning Schulzrinne, Steve Casner, Ron Frederick, and Van Jacobson. *RTP: A Transport Protocol for Real Time Applications.* RFC 3550, IETF, July 2003.

[SCOT03] Olga Sorkine, Daniel Cohen-Or, and Sivan Toledo. High-pass quantization for mesh encoding. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry Processing*, pages 42–51. Eurographics Association, 2003.

[SFE00] Julien Signès, Yuval Fischer, and Alexandros Eleftheriadis. MPEG-4s Binary Format for Scene Description. *Signal Processing: Image Communication*, 15(4–5):321–345, January 2000.

[SFLG00] Klaus Stuhlmüller, Niko Färber, Michael Link, and Bernd Girod. Analysis of video transmission over lossy channels. *IEEE Journal on Selected Areas in Communications, Special Issue on Error-Resilient Image and Video Transmission*, 18(6):1012–1032, June 2000.

[Sha93] Jerome M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, December 1993.

[Shi00a] Dinesh Shikhare. Signal Processing over Triangle Meshes. Technical report, Graphics & CAD division, National Center for Software Technology, December 2000.

[Shi00b] Dinesh Shikhare. State of the Art in Geometry Compression. Technical report, Graphics & CAD division, National Center for Software Technology, December 2000.

[Sig00] Julien Signès. BIFS: Combining MPEG-4 media to build rich multi-media services. *Signal Processing: Image Communication*, 15(4–5), January 2000.

[SKL02] Jae-Young Sim, Chang-Su Kim, and Sang-Uk Lee. 3-D mesh compression using triangle fan structure. *ISCAS – 2002*, May 2002.

[SKL03] Jae-Young Sim, Chang-Su Kim, and Sang-Uk Lee. An efficient 3-D mesh compression technique based on triangle fan structure. *Signal Processing: Image Communication*, 18(1):17–32, January 2003.

[SLW00] Henning Sanneck, Nguyen-Tuong Le, and Adam Wolisz. Efficient QoS Support for Voice-over-IP Applications Using Selective Packet Marking. *First International Workshop on Intelligent Multimedia Computing and Networking (IMMCN 2000)*, 1:553–556, February 2000.

[SMG+02] Alexandru Salomie, Adrian Munteanu, Augustin Gavrilescu, Gauthier Lafruit, Peter Schelkens, Rudi Deklerck, and Jan Cornelis. MeshGrid - A compact, Multi-Scalable and Animation-Friendly surface representation. *IEEE International Conference on Image Processing ICIP 2002*, 3:13–16, September 2002.

[SP01] Ariel Shamir and Valerio Pascucci. Temporal and spatial level of details for dynamic meshes. In *ACM VRST'01*, pages 77–84, Alberta, Canada, November 2001.

[SPB00] Ariel Shamir, Valerio Pascucci, and Chandrajit Bajaj. Multiresolution dynamic meshes with arbitraty deformations. Technical report, TICAM, The University of Texas at Austin, 2000.

[SZL92] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *Computer Graphics (ACM SIGGRAPH'92)*, 26(2):65–70, 1992.

[Tau99] Gabriel Taubin. 3-D geometry compression and progressive transmission. *EUROGRAPHICS – State of the Art Report*, 1999.

[TG98] Costa Touma and Craig Gotsman. Triangle Mesh Compression. *Graphics Interface '98*, pages 26–34, 1998.

[TGHL98] Gabriel Taubin, André Guéziec, William Horn, and Francis Lazarus. Progressive forest split compression. *Computer Graphics Annual Conference Series*, 32:123–132, 1998.

[TLG99] Danny S.P. To, Rynson W.H. Lau, and Mark Green. A method for progressive and selective transmission of multi-resolution models. In *ACM VRST'99*, pages 88–95, London, November 1999.

[TO00] Murat Tekalp and Jörn Ostermann. Face and 2D mesh animation in MPEG-4. *Signal Processing: Image Communication*, 15(4–5):387–421, January 2000.

[TR98] Gabriel Taubin and Jarek Rossignac. Geometric compression through topological surgery. *ACM Transactions on Graphics*, 17(2):84–115, 1998.

[uu00] `http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/` `http://www-nrg.ee.lbl.gov/vic/`.
Robust Audio Tool (RAT) and Video Conferencing Tool (VIC), 1995–2000.

[VBH00] Socrates Varakliotis, Andrea Basso, and Vicky Hardman. Delivering MPEG-4 content over IP: An architecture and issues towards standardisation. *IEEE International Conference on Telecommunications*, May 2000.

[VHO02] Socrates Varakliotis, Stephen Hailes, and Jörn Ostermann. Repair options for 3-D wireframe model animation sequences. *IEEE International Conference on Multimedia and Expo*, August 2002.

[VHO03] Socrates Varakliotis, Stephen Hailes, and Jörn Ostermann. Optimally smooth error resilient streaming of 3-D wireframe animations. *SPIE Visual Communications & Image Processing (VCIP)*, July 2003.

[VHO04] Socrates Varakliotis, Stephen Hailes, and Jörn Ostermann. Progressive coding for QoS-enabled streaming of dynamic 3-D meshes. *IEEE International Conference on Communications*, June 2004.

[VOH01] Socrates Varakliotis, Jörn Ostermann, and Vicky Hardman. Coding of animated 3-D wireframe models for Internet streaming applications. *IEEE International Conference on Multimedia and Expo*, August 2001.

[W3C03] W3C. Scalable Vector Graphics (SVG) 1.1 specification, World Wide Web Consortium recommendation, January 2003.

[WFM01] Benjamin Watson, Alinda Friedman, and Aaron McGaffey. Measuring and predicting visual fidelity. In *Proceedings of the 28^{th} annual conference on computer graphics and interactive techniques*, pages 213–220. ACM Press, 2001.

[WL00] Hongwu Wang and Jin Li. OctMesh – Interactive mesh browsing over the Internet. In *IEEE International Conference on Information Technology: Coding and Computing (ITCC'00)*, Las Vegas, Nevada, March 2000.

[XV96] Julie C. Xia and Amitabh Varshney. Dynamic view-dependent simplification for polygonal models. In Roni Yagel and Gregory M. Nielson, editors, *IEEE Visualization '96*, pages 335–344, New York, 1996.

[YKK01] Sheng Yang, Chang-Su Kim, and C.-C. Jay Kuo. View-dependent progressive mesh coding for graphic streaming. In *SPIE International Symposium on Convergence of IT and Communications*, Denver, August 2001.

[YKL01] Jeong-Hyu Yang, Chang-Su Kim, and Sang-Uk Lee. Compression of 3D triangle mesh sequences. In *Proc. IEEE Workshop on Multimedia Signal Processing (MMSP-01)*, pages 181–186, Cannes, France, IEEE 2001.

[YKL02] Jeong-Hyu Yang, Chang-Su Kim, and Sang-Uk Lee. Compression of 3-D triangle mesh sequences based on vertex-wise motion vector prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(12):1178–1184, December 2002.