Optimally Smooth Error Resilient Streaming of 3-D Wireframe Animations

Socrates Varakliotis[a] Stephen Hailes[a] Jörn Ostermann[b]

[a]Computer Science Dept., University College London, UK
[b]AT&T Labs - Research, USA

ABSTRACT

Much research has been undertaken in the area of streaming video across computer networks in general and the Internet in particular, but relatively little has been undertaken in the field of streaming 3-D wireframe animation. Despite superficial similarities, both being visual media, the two are significantly different. Different data passes across the network, so loss affects signal reconstruction differently. Regrettably, 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.

In this paper, we bring together concepts from a number of fields to address the problem of how to achieve optimal resilience to errors in terms of the perceptual effect at the receiver. To achieve this, we partition the animation stream into a number of layers and apply Reed-Solomon (RS) forward error correction (FEC) codes to each layer independently and in such a way as to maintain the same overall bitrate whilst minimizing the perceptual effects of error, as measured by a distortion metric derived from related work in the area of static 3-D mesh compression. Experimental results show the efficacy of our proposed scheme under varying network bandwidth and loss conditions for different layer partitionings. The results indicate that with the proposed Unequal Error Protection (UEP) combined with Error Concealment (EC) and efficient packetization scheme, we can achieve graceful degradation of streamed animations at higher packet loss rates than other approaches that do not cater for the visual importance of the layers and use only objective layering metrics. Our experiments also demonstrate how to tune the packetization parameters in order to achieve efficient layerings with respect to the subjective metric of surface smoothness.

Keywords: 3-D animation compression, 3-D mesh streaming, 3-D mesh error metrics, mesh QoS optimization, error resilience, forward error correction, unequal loss protection, MPEG-4.

1. INTRODUCTION

The Internet has evolved rapidly during the past few years from a low-bandwidth, text-only collaboration medium, to a richer, interactive, real-time, audio-visual virtual world. It involves many users, environments and applications, where 3-D animations constitute a driving force. Animated 3-D models enable intuitive and realistic interaction with displayed objects and allow for effects that cannot be achieved with conventional audio-visual animations. Consequently, the current challenge is to integrate animated 3-D geometry as a new data stream in the existing evolving infrastructure of the Internet, in a way that both enhances the existing networked environment and respects its limited resources. Although static 3-D mesh geometry compression has been actively researched in the past decade, very little research has been conducted in compressing dynamic 3-D geometry, a logical extension of static 3-D meshes to the temporal domain.

To date, the most prevalent representations for 3-D static models are polygonal or triangle meshes. These representations allow to approximate models of arbitrary shape and topology within some desired precision or quality. Efficient algorithms and data structures exist to generate, modify, compress, transmit and store

Further author information: (Send correspondence to S.V.)
S.V./S.H.: {S.Varakliotis, S.Hailes}@cs.ucl.ac.uk, Computer Science Dept., University College London, WC1E 6BT, UK. Tel.: +44 20 7679.3679, Fax: +44 20 7387.1397
J.O.: osterman@research.att.com, 200 Laurel Ave South, Middletown, NJ-07748, USA. Tel.: +1 732 420.9116, Fax: +1 732 368.9475

such static meshes. Future, non-static, stream types that introduce the time dimension, would require scalable solutions to survive with respect to the network's limited resources (bandwidth) and characteristics (channel errors).

In this paper we concentrate on source and channel coding techniques for error resilient time-dependent 3-D mesh streaming over the Internet, that respects network bandwidth and considers the bursty loss nature of the channel.

The problem of 3-D wireframe animation streaming we address can be stated as follows: *Assume (i) a time-dependent 3-D mesh has been scalably compressed in 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 C of the available transmission rate ($C < R$) can be reserved for channel coding. Then, what is the optimal number of bits to be allocated to each level of importance (layer) in the animation scene that maximizes the perceived quality of the time-dependent mesh at the receiver?*

Section 2 presents an overview of literature related to the fields involved in this work. Section 3 describes the 3-D wireframe animation codec and its bitstream content, along with the FEC codes used. The visual distortion metric is detailed in section 4. The proposed UEP method and receiver-based concealment is discussed in Section 5. In Section 6 an analysis of the experimental results is given. Finally, significant findings and future work are presented in the concluding Section (7).

2. RELATED LITERATURE

The field of static 3-D mesh compression has been an active topic for research in the past decade. Efficient algorithms have been developed that compress a 3-D model's connectivity to very high ratios and decompress it in real time. Consequently, all recent efforts in 3-D mesh streaming only focus on the efficient transmission of a 3-D model's layers that correspond to additive Levels of Detail (LOD). These LODs aim at refining the base layer of a mesh that usually represents a coarse mesh approximation. Usually these streaming efforts accompany some mesh simplification algorithm[1,2] or progressive mesh compression techniques[3,4] to reduce the downloading time from a server. The most recent approach in the area of static mesh transmission, considers scalability issues with regards to the channel bandwidth and the channel error characteristics.[5,6] None of the above works, though, looks at the problem of robustness for time-evolving 3-D meshes.

One way to compress the time-dependent geometries would be to compress its individual frames using those static schemes mentioned above. However, a more reasonable approach would be to exploit the temporal coherence inherent in such signals. To this end, only a very limited number of research efforts that tackle the problem of 3-D animation compression exist. Lengyel's[7] pioneering work is based on an algorithm that calculates and transmits motion parameters by coding residuals using spatial prediction and a greedy clustering technique. This landmark work laid the foundation for a couple more approaches later. Alexa[8] et al. introduces principal component analysis and proposes the computation and off-line transmission of a set of basic shapes. Yang[9] et al. in approach the problem using vertex-wise motion vector prediction. More recently Sengupta[10] et al. propose a compression scheme based on image registration techniques and partition the 3-D mesh using an iterative algorithm. This work also caters for connectivity changes in the time-dependent mesh. None of the above literature addresses streaming issues and QoS in particular.

Our work, according to Wu's[11] classification, addresses the problem of application-level QoS control in 3-D wireframe animation streaming, a specialized area that has rarely formed the subject of research before. In particular, we provide a 3-D wireframe animation compression and streaming solution, optimized with source/channel coding FEC. Where the optimization procedure fails due to heavy channel losses, we propose the use of interpolation-based 3-D wireframe error concealment as set out in our previous work.[12]

Related source/channel optimization work can be found in the literature for a limited spectrum of media, namely speech audio and natural video. Some landmark works are listed below.

Podolsky[13] et al. systematically study the issue of source/channel coding for audio signals and provide extensive analytical simulation results to evaluate the 'scalability' of signal processing-based FEC (SFEC) for packet audio i.e., the ability for a coding algorithm to improve aggregate performance when used by all sources

in the network, and find that optimal signal quality is achieved when sources react to network congestion not by blindly adding FEC, but rather by adding FEC in a controlled fashion that simultaneously constrains the source-coding rate.

Stuhlmüller[14] et al. present detailed analysis of a video transmission over lossy channels. They discuss in detail a complete video transmission system including rate-distortion performance of the video encoder, modelling of a 2-state Markov burst-loss channel, forward error correction using RS erasure codes, interleaving and the effect of error concealment and inter-frame error propagation at the decoder. Their analysis objective is to optimize the video transmission system based on objective PSNR quality metrics and constant channel to source coding ratios for RS codes per source coding layer. Obviously, despite the exhibited accuracy of their analytical model, optimization of an objective metric does not necessarily reflect on improved subjective assessments.

Zhang[15] et al. propose a network-adaptive congestion control scheme for scalable video streaming with Unequal Loss Protection (ULP). In this scheme, end-to-end rate-distortion optimized source and channel codes are allocated on the fly to best utilize the estimated available network bandwidth. The authors employ an alternative to the BOP[14] packetization scheme, known as EREC,[16] which allows the formation of constant length packets out of variable length blocks of video data. Given such a packetization scheme, the optimal source to channel coding ration is sought that minimizes end-to-end distortion, again in terms of PSNR.

MPEG-4 attempts to elaborate on the area of 3-D animation within its Animation Effects Framework[17,18] (AFX) in version 5 of the standard, which has 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.

3. BACKGROUND

This work builds upon our recently proposed *3D-Anim*[19] animation coder previously presented. The following section provides an overview of the 3D-Anim codec and introduces the related notation. An overview of the error correcting Reed-Solomon (RS) codes follows together with derivation of the channel model, as well as the design of Unequal Error Protection (UEP) packetization for the encoded bitstream.

3.1. 3-D Wireframe Animation Coding

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

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

We partition the indexed set of vertices into intuitively natural partitions, called *nodes**. The position matrix corresponding to the i^{th} node is denoted by $N_{i,t}$. Note that, without loss of generality, the vertex matrix can now be expressed as:

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

for k such nodes. For notational convenience, we use $N_{i,t}$ $(i = 1, 2, ..., k)$ both for representing a matrix and to refer to the i^{th} node as well. The objective of the 3D-Anim compression algorithm is to compress the sequence of matrices M_t that form the synthetic animation, for transmission over a communications channel. Obviously, for free-form animations of a 3-D mesh the coordinates of the mesh may exhibit high variance, which makes the M_t matrices unsuitable for compression. Hence, we define our signal as the set of *non-zero displacements* of all vertices in all nodes at time t:

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

*The term is chosen to highlight the correspondence of such nodes with the nodes as defined in VRML.

Following our notation, we can express the above with a displacement matrix, D_t, as:

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

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

or equivalently, using node matrices:

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

where $F_{i,t}$ the displacement matrix of node i, for l such nodes ($i = 1, 2, ..., l$). Note that D_t's dimension is reduced to $p \leq n$ compared to M_t, since D_t does not contain vertices for which the displacement on all axes is zero. Note, too, that $l \leq k$ holds, in the event that no vertices in a node get displaced ($F_{i,t} = 0$). In 3D-Anim terminology, we call *sparse* animations those sequences with $p < n$ and $l < k$. It is evident that depending on the parameters p and l, the encoder can generate a layered bitstream (by adjusting parameter l), where every layer L can be scalable (by adjusting parameter p). We quantify the sparsity (or density) of the animation by the *density factor*, defined as:

$$df = \frac{1}{N}\frac{1}{k}\sum_{f=1}^{F}\sum_{j=1}^{l}\frac{p_{jf}}{n_{jf}}, \text{ with } l \leq k \text{ and } p \leq n \tag{2}$$

in the range [0..1], where N is the number of animation frames and k the number of nodes in the reference model. For $p \to n$ and $l \to k$ we have $df \to 1$, therefore a *dense* animation.

The scheme described and summarized in equation (1) above, is suited to a DPCM coder, as detailed below. 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 one recently proposed by Al-Regib[5] and colleagues, along with error protection if we assume the transmission is done over the same lossy channel as the time-dependent mesh. Our method can accommodate and interoperate with such static mesh transmissions and will not be discussed further here.

In the 3D-Anim codec's context, an *I-frame* describes changes from the reference model M_0 to the model at the current time instant t. A *P-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 I and P frames are denoted respectively by M_t^I, M_t^P, D_t^I, D_t^P.

Figure 1 shows a block diagram of our coding scheme. The top Figure 1 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 P-frame, the decoded set (animation frame or displacement matrix) of the previous instance is used as the predicted value, $\hat{M}_{t-1}^P$. (Equivalently, for encoding an I-frame the predicted matrix is $\hat{M}_{t-1}^I$, where $\hat{M}_{t-1}^I = 0$ 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 quantized ($\hat{E}_t$). Finally, the quantized samples are entropy coded (C_t) using an adaptive arithmetic coding algorithm[20] to handle the unknown data statistics. The predictive scheme described prevents quantization error accumulation.

A DPCM decoder (bottom of Figure 1) first decodes arithmetically the received samples (C'_t) and computes the decoded samples ($\hat{D}'_t$). The quantization step size can be assumed to be the same for all nodes, or can vary in order to shape the encoded bitstream rate†. Similar coding schemes have been used in MPEG-4 for the compression of facial animation parameters[21] and for BIFS-Anim.[22]

We mentioned before 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,[22] which does not allow for reduced animation frames. For sparse D_t matrices it may also be the case

†Allowing different quantization step sizes for different nodes may result in artifacts, such as mesh cracks, especially in the boundaries between nodes.

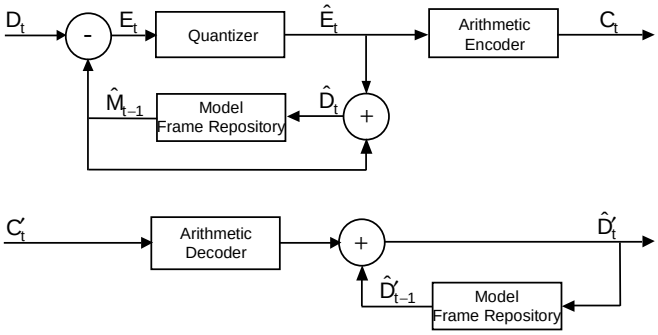

Figure 1. Block diagram of the 3D-Anim codec. Top: Encoder, Bottom: Decoder.

that a whole node is not animated thus allowing great animation flexibility and generating a scalable bitstream. Furthermore, in the case where $F_{i,t} = 0, \forall i \in [1..l]$, the displacement matrix D_t is zero, leading to an 'empty' frame. 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 synchronization can also be achieved, which is paramount for many applications (e.g. lip synchronization of a 3-D animated virtual salesman with packet speech).

3.2. 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 our FEC scheme we use Reed-Solomon (RS) codes 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 needs to store copies of k information packets in order to calculate $n - k$ redundancy packets. The resulting n packets are stacked in a block of packet (BOP) structure, commonly used in the literature[5, 14] and described later in Figure 2. 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. 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, we express the model using the average loss probability P_B, and the average burst length L_B, as:

$$P_B = Pr(B) = \frac{p_{GB}}{p_{GB} + p_{BG}} \tag{3}$$

$$L_B = 1/p_{BG} \tag{4}$$

For the selection of the RS code parameters we need 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, we can calculate:

$$P_{BER} = \sum_{m=n-k+1}^{n} P(m, n) \tag{5}$$

The average loss probability P_B and the average loss burst L_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 we adapt the derivation for bit-error channels[23] to a packet loss channel.

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. $g(\nu) = Pr(0^{\nu-1}1|1)$, 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 $\nu-1$, i.e. $G(\nu) = Pr(0^{\nu-1}|1)$. In state B of our 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 Eq. (5) we see that $P(m,n)$ determines the performance of the FEC scheme, and can be expressed as a function of P_B, L_B using Eq. (3) and (4). In a later section we will see how we can use this expression of $P(m,n)$ in a RS (m,n) FEC scheme for optimized source/channel rate allocation that minimizes the visual distortion.

3.3. Bitstream Format and Packetization

The output bitstream of the 3D-Anim codec needs to be appropriately packetized for streaming with an application-level transport protocol, e.g. RTP. This process for a single layer bitstream has been previously described in detail[19] and its main features are summarized below:

- In order to describe which nodes of the model are to be animated we define the animation masks, NodeMask and VertexMasks, in a similar way to BIFS-Anim.[22] The NodeMask is essentially a bit-mask where each bit, if set, denotes that the corresponding node in the Node Table will be animated. The Node Table (an ordered list of all nodes in the scene) is either known a priori at the receiver since the reference wireframe model exists there already, or is downloaded by other means. In a similar way, we have defined the VertexMasks, one per axis, for the vertices to be animated.

- In its simplest form, one frame (which represents one Application Data Unit (ADU)), is contained in one RTP packet. In this sense, the 3D-Anim codec's output bitstream is 'naturally packetizable' according to the Application Level Framing[24] (ALF) principle. We considered an RTP packet payload format starting with the NodeMask and VertexMasks, followed by the encoded samples along each axis.

This simple format 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 packetization in a single layer would require the definition of fragmentation rules for the RTP payload, which may not always be straightforward

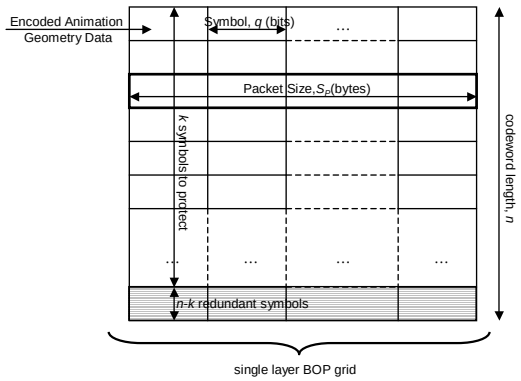

Figure 2. The Block-Of-Packets (BOP) grid structure.

in the ALF sense. Furthermore, frames directly packetized in RTP as described above generate a variable bitrate stream due to their varying lengths.

We seek a more efficient packetization scheme that satisfies the requirements set out above: (a) to accommodate layered bitstreams, and (b) to produce a constant bitrate stream. We can achieve this efficiency by appropriately adapting the block structure presented by Horn[25] et al. known as Block-Of-Packets (BOP). In this method, 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 we append error resilience information so that the length of the grid is n for k frames of source data, as shown on Figure 2. This method is most appropriate for packet networks with burst packet errors, and can be fully described by the sequence frame rate FR, the packet size S_P, the data frame rate in a BOP F_{BOP}, and the RS code (n, k).

Intuitively, for a BOP consisting of F_{BOP} data frames, with S_P bytes long packets, at FR frame rate, the total source and channel bitrate R is given by:

$$R = \frac{n \cdot FR \cdot S_P}{F_{BOP}} \tag{6}$$

This equation serves as a guide to the design of efficient packetization schemes by appropriately balancing the parameters F_{BOP}, n and S_P. It also encompasses the trade-off between delay and resilience. Obviously, for a layered bitstream we need to design one BOP structure per layer. By varying the parameters in Eq. (6) we can allocate different RS code rates to each layer, thus providing 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 Section 5.

4. VISUAL DISTORTION METRIC

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 difference between the original mesh M_t at time t and its decoded equivalent $\hat{M}_t$. The simplest measure is the RMS geometric distance between corresponding vertices. Alternatively, the Hausdorff Distance has been commonly used in some literature[5, 26] as an error metric. The Hausdorff distance, is defined in our case as the maximum minimum distance between the vertices of two sets, M_t and $\hat{M}_t$, in such a way that every point M_t lies within the distance $H(M_t, \hat{M}_t)$ of every point in $\hat{M}_t$ and vice versa. This can be expressed as:

$$H(M_t, \hat{M}_t) = max(h(M_t, \hat{M}_t), h(\hat{M}_t, M_t)) \tag{7}$$

where $h(M_t, \hat{M}_t) = \max_{m_t \in M_t} \min_{\hat{m}_t \in \hat{M}_t} \|m_t - \hat{m}_t\|$, and $\|\cdot\|$ is the Euclidean distance between the two vertices, m_t and $\hat{m}_t$.[27] Many other distortion metrics for 3-D data can be derived by equivalence to natural video coding in 2-D,

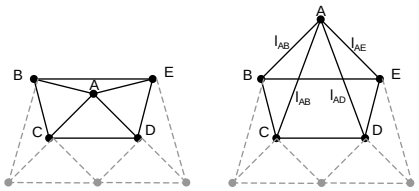

Figure 3. Example of a Laplacian operator on a hypothetical surface. Left: original surface. Right: reconstructed surface with distortion on vertex A.

such as SNR[10] and PSNR,[19] but these 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[28] as being undertaken whilst evaluating their spectral compression algorithm for 3-D mesh geometries. In this, the suggested 3-D mesh distortion metric normalizes 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:

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

where $n(i)$ is the set of indices of the neighbors of vertex i, and l_{ij} is the geometric distance between vertices i and j. Figure 3 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.

Hence, the new metric is defined as the average 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 $M_t, \hat{M}_t$):

$$VS_{(t)} = f(M_t, \hat{M}_t) = \frac{1}{2n} \sum_{r=1}^{n} (\|m_{rt} - \hat{m}_{rt}\| + \|GL(m_{rt}) - GL(\hat{m}_{rt})\|) \tag{8}$$

This metric in Eq. (8) is used in our proposed scheme, and we will be referring to it hereafter as the *Visual Smoothness* metric (VS).

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

5. ERROR RESILIENT 3-D WIREFRAME STREAMING

In Section 3.3 we described the BOP structure, which is suitable for the design of an efficient packetization scheme that employs redundancy information based on RS erasure codes. We also expressed in closed form the relation of its design parameters F_{BOP}, n and S_P in Eq. (6). This equation, though, does not reflect any information about layering. In this section, the layering design approach is given first, followed by the proposed error resilient method for 3-D wireframe streaming.

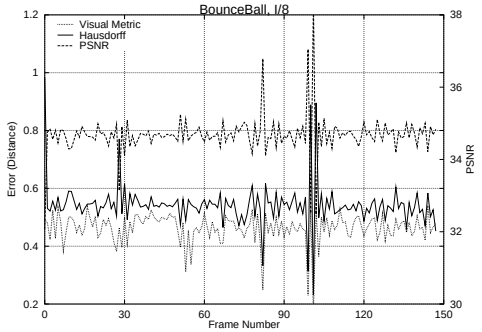

Figure 4. Comparative plot of distortion metrics: PSNR, Hausdorff Distance, and Visual Smoothness for 150 frames of the animated sequence `BOUNCEBALL` with I-frame freq. at 8Hz. The two upper plots (PSNR-Hausdorff) show the expected correlation between the corresponding metrics of Geometric Distance and Hausdorff Distance (eq. 7) they represent. The two lower plots, indicate that the Visual Distortion (eq. 8) might be low in case where the Geometric Distance is high and vice-versa.

The layering is performed in a way that the average VS value of each layer reflects its importance in the animation sequence. To achieve this, we need to compute the VS from Eq. (8) for every node in the mesh independently and order the nodes according to their average VS in the sequence. A node, or group of nodes, with the highest average VS forms the first and most important layer visually, L_0. This is the layer that we want to make more resilient to packet errors than other layers. Subsequent importance layers $L_1, ..., L_M$ are created by correspondingly subsequent nodes, or group of nodes, in the VS order.

If a 3-D mesh has more nodes than the desirable number of layers, then the number of nodes to be grouped in the same layer is a design choice, and dictates the output bitrate of the layer. For meshes with only a few nodes but a large number of vertices per node, node partitioning might be desirable. The partitioning would restructure the 3-D mesh's vertices into a new mesh with more nodes than were originally present. This process will not affect connectivity, or the overall rendered model. Following the discussion on visual importance in this paper, one realizes that mesh partitioning into nodes, if it is possible, should not be arbitrary, but should rather reflect the natural objects these new nodes will represent in the 3-D scene and their corresponding motion. If partitioning is not possible in the above sense, one could partition the mesh into arbitrary sized sub-meshes (nodes) that are allocated to the same layer. Mesh partitioning may require complex pre-processing steps and goes beyond the scope of this paper. Recall, however, that the 3D-Anim codec assumes static connectivity.

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 we constructed the layers, 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:

$$VS_{(t)} = \sum_{j=0}^{L-1} P_{jt} \cdot VS_{jt} \tag{9}$$

where L is the number of layers. In the equation above, P_{jt} is the block error rate P_{BER} as given by Eq. (5), 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:

$$P_{jt} = \sum_{m=n-k_{jt}+1}^{n} P(m, n) \tag{10}$$

From Eqs. (9) and (10) we can express $VS_{(t)}$ as:

$$VS_{(t)} = \sum_{j=0}^{L-1} \sum_{m=n-k_{jt}+1}^{n} P(m, n) \cdot VS_{jt} \tag{11}$$

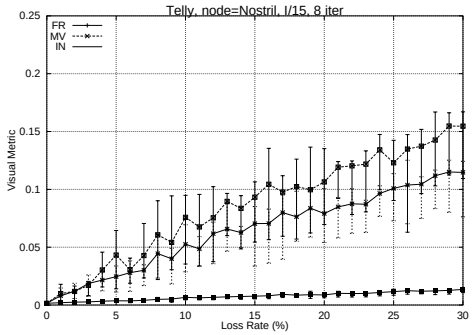

Figure 5. Performance of three error concealment methods for sequence TELLY, for $P_B = [0..30]$ and $L_B = 4$. FR=Frame Repetition, MV=Motion Vectors, IN=Interpolation. The plot shows the average performance over 8 iterations with different loss patterns.

Equation 11 estimates in a statistical sense the expected visual smoothness experienced per frame at the decoder. Our objective is to minimize this distortion with respect to the values of k_{jt}'s in Eq. (11). From the way we split the bitstream into layers we would expect the optimization process to allocate more redundancy to the layer that exhibits the greatest visual distortion (coarse layer), and gradually reduce 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 $\sum_{j=0}^{L-1}(n - k_{jt}) = R_C/q$, where R_C the redundancy bits, and q is the symbol size. The above problem formulation yields a non-linear constraint optimization problem that can be solved numerically.

The anticipated behavior of the model for $P_B = 0$ is to produce equal values for k_{jt}'s, whereas at high P_B we would get unequally varying k_{jt}'s. Note that for the calculation of smoothness distortions in Eq. (11) we assumed that no error concealment takes place at the receiver.

In our recent work,[12] we have shown that techniques based on vertex linear temporal interpolation are a sufficient and 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 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 neighboring frames. Interpolation and other concealment methods discussed elsewhere[12] are generic in that they can be used by any other decoder.

Figure 5 shows the relative performances of three error concealment methods adapted to the experimental parameters of this work, namely $P_B = [0..30]$ and $L_B = 4$. It is evident that linear interpolation outperforms Frame Repetition or Motion Vector-based methods[‡]. The plot shows average values for 8 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 (the average loss burst length $L_B = 4$ is much lower than the sequence frame rate of 30 Hz.) We, therefore, propose 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 optimization problem, will also give minimum distortion if combined with concealment techniques. The expected distortion in such cases will be lower than the distortion without error concealment.

In the following section we explain the experimental procedure and we tune the values and the optimization process for a real-world case of 3-D wireframe animation, along with discussion of our findings.

6. EXPERIMENTS AND RESULTS

In the following experiments, we are demonstrating through simulation the efficiency of the proposed Unequal Error Protection (UEP) scheme combined with Error Concealment (EC) for streaming 3-D wireframe animations.

[‡]Detailed descriptions of the other error concealment methods at the receiver are given in our past related work.[12]

In particular, we are comparing UEP and EC 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} we numerically solved the constrained minimization problem of Eq. (11), given the channel rate R_C. Furthermore, we calculated n from Eq. (6) such that we meet the rate characteristics of the original source signal for our particular design of a BOP. The other parameters we used in Eq. (6) are given below for the two sequences in the experiments, and are also summarized in Table 1. For the EEP case we consider a constant k that can be derived directly from the selection of the channel rate, which we set to 15%. For the NP case we allocate all available channel rate to the source. Finally, we used an EC scheme based on interpolation for the case of UEP with residual losses. In all experiments we use $L_B = 4$.

We utilized the sequences `TELLY` and `BOUNCEBALL` with density factors of $df_{\texttt{TELLY}} = 0.75$ and $df_{\texttt{BBALL}} = 1.0$ given by Eq. (2). `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 1. Its average source bitrate is $R_{S,\texttt{TELLY}} = 220$ Kbps. `BOUNCEBALL` only has 1 complete node and 528 frames at 24 Hz, forming 1 layer of source rate $R_{S,\texttt{BBALL}} = 61$ Kbps average. Both sequences have been coded with I-frames at every 15 frames. We allow roughly 15% of channel coding redundancy, resulting in total source and channel rates of $R_{\texttt{TELLY}} = 253$ Kbps and $R_{\texttt{BBALL}} = 70.15$ Kbps. Choosing $n = 32$ the parameters we calculated from Eq. (6) for each layer's packetization are tabulated in Table 1. 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 3 layers according to the suggested layering method presented in Section 5, each consisting of the nodes shown in Table 1. 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 layer proportionally. We noticed that 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 we wish to relate the node-to-layer allocation (using the VS metric) to the density factor df_L, calculated per layer[§] (Eq. 2), and to the output bitrates. We do not have enough model data at the moment to derive any useful relation. If such relation exists, a dynamic 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 center point as its shape implies. The ball also deforms slightly as it bounces. Given the shape symmetry, we decided to partition the mesh into 2 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 quantization step size. It is anticipated that both layers will receive roughly equal average protection bits, so that UEP performance will approach that of EEP.

Figure 6 (top) 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 $(31, 22)$. The average calculated codes for the UEP are as follows (rounded to nearest integer): $(n, \bar{k_0}) = (31, 19), (n, \bar{k_1}) = (31, 23), (n, \bar{k_2}) = (31, 28)$. It is clear that UEP, and UEP+EC outperform NP and EEP for medium to high loss rates of $P_B > 9\%$. Recall that we performed the layering 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 we notice that EEP and UEP behave in approximately the same way, as the RS codes are more than sufficient to recover all or most errors. We also note 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.

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

Table 1. Animation sequence parameters used in the redundancy experiments: TELLY & BOUNCEBALL.

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

Sequence	BOUNCEBALL
df_{BBALL}	1.0
Nodes	1
Frame Rate	24 Hz
Source Rate	61 Kbps
Channel Rate	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_{BOF}	16	35
L_1	S_P	264	200
	F_{BOF}	19	35
L_2	S_P	150	N/A
	F_{BOF}	50	

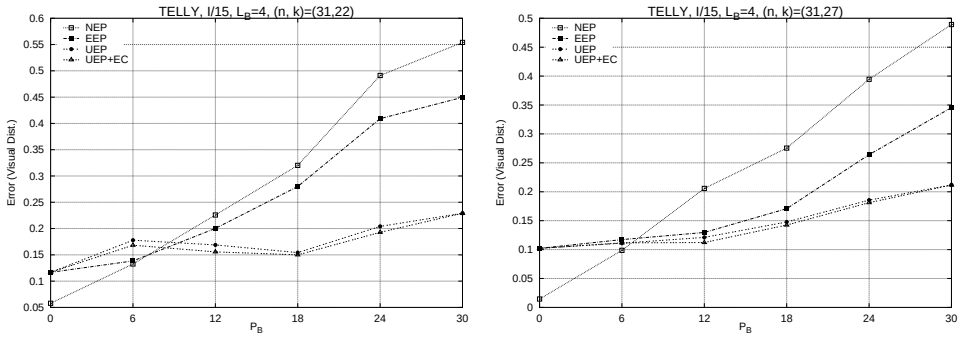

Figure 6. Comparison of Visual Smoothness (VS) between transmitted and decoded frames of 3 layers of the wireframe animation TELLY. The equivalent EEP RS codes are respectively: top $(n, k) = (31, 22)$, bottom $(n, k) = (31, 27)$. Average burst length: $L_B = 4$

The results for the (31,27) RS code on sequence TELLY, shown in Figure 6 (bottom), 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 $(31, 22)$, 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: $(n, \bar{k_0}) = (31, 26), (n, \bar{k_1}) = (31, 28), (n, \bar{k_2}) = (31, 30)$. 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 optimization problem it is expected to contribute a small reduction to the visual error.

Figure 7 shows the results achieved for the same experiment repeated over the BOUNCEBALL sequence, which was 'symmetrically' layered as described earlier in this section. We used the same $(31, 22)$ EEP code 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. We note, 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: $(n, \bar{k_0}) = (31, 22), (n, \bar{k_1}) = (31, 22)$, 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 animation they contain (same number of vertices, nodes, very similar motion in the scene, and same encoding parameters) correspond to visually balanced distortions. This is exactly the result we were anticipating when we discussed layering for the BOUNCEBALL sequence earlier in this section. In fact, the real values of k_{0t}, k_{1t} computed as the solution to the optimization problem, vary around the average integer value of 22. Furthermore, recall that we partitioned the original symmetric BOUNCEBALL mesh into two arbitrary nodes

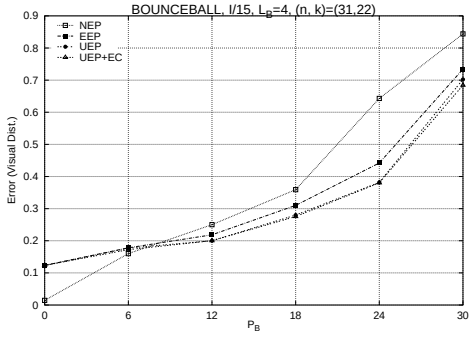

Figure 7. Comparison of Visual Smoothness (VS) between transmitted and decoded frames of 2 layers of the wireframe animation `BOUNCEBALL`. The equivalent EEP RS code is: $(n, k) = (31, 22)$. Average burst length: $L_B = 4$

without consideration to their individual visual distortions, which we 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 the UEP and EEP curves are not accurately fit at higher P_B's as one would normally expect. Finally, we note that the UEP+EC method provides a slight, but hardly noticeable, improvement to the visual distortion as in the previous experiment.

7. CONCLUSION

In this paper we have addressed the fundamental problem of how best to utilize the available channel capacity for streaming 3-D wireframe animation in such a way as to achieve optimal subjective resilience to error. In short, we have linked channel coding, packetization, and layering with a subjective measure that measures visual smoothness in the reconstructed image. On this basis, we believe that our result may help open the way for 3-D animation to become a serious networked media type. Our methods attempt to optimize 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 we initially partition the encoded bitstream into layers of visual importance, and show with experimental results that UEP combined with EC yields good protection against burst packet errors occurring on the Internet.

ACKNOWLEDGMENTS

The authors wish to thank Adam Greenhalgh for technical support on the experiments and Athanasios Zacharopoulos on mathematical modelling.

REFERENCES

1. G. Taubin and J. Rossignac, "Geometric compression through topological surgery," *ACM Transactions on Graphics* **17**(2), pp. 84–115, 1998.
2. C. Touma and C. Gotsman, "Triangle Mesh Compression," *Graphics Interface '98* , pp. 26–34, 1998.
3. H. Hoppe, "Progressive Meshes," *Computer Graphics* **30**, pp. 99–108, 1996.
4. R. Pajarola and J. Rossignac, "Compressed progressive meshes," *IEEE Transactions on Visualization and Computer Graphics* **6**, pp. 79–93, March 2000.
5. G. Al-Regib and Y. Altunbasak, "An unequal error protection method for packet loss resilient 3-D mesh transmission," *IEEE INFOCOM* , 2002.
6. G. Al-Regib and Y. Altunbasak, "A system level framework for streaming 3-D meshes over packet networks," in *International Conference on Networking ICN2001, LNCS 2094*, P.Lorenz, ed., pp. 745–753, Springer-Verlag, 2001.
7. J. Lengyel, "Compression of time-dependent geometry," in *ACM Symposium on Interactive 3-D graphics*, (Atlanta), August 1999.

8. M. Alexa and W. Müller, "Representing animations by principal components," *EUROGRAPHICS* **19**(3), pp. 411–418, 2000.
9. J.-H. Yang, C.-S. Kim, and S.-U. Lee, "Compression of 3D triangle mesh sequences," in *Proc. IEEE Workshop on Multimedia Signal Processing (MMSP-01)*, pp. 181–186, (Cannes, France), IEEE 2001.
10. S. Gupta, K. Sengupta, and A. A. Kassim, "Compression of 3D Dynamic Geometry Data using Iterative Closest Point Algorithm," *IEEE Computer Vision and Image Understanding* **87**, September 2002.
11. D. Wu, Y. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming video over the Internet: Approaches and directions," *IEEE Transactions on Circuits and Systems for Video Technology* **11**, pp. 1–20, Feb. 2001.
12. S. Varakliotis, S. Hailes, and J. Ostermann, "Repair options for 3-D wireframe model animation sequences," *IEEE International Conference on Multimedia and Expo* , August 2002.
13. M. Podolsky, C. Romer, and S. McCanne, "Simulation of FEC-based error control for packet audio on the internet," in *INFOCOM (2)*, pp. 505–515, 1998.
14. K. Stuhlmüller, N. Färber, M. Link, and B. 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**, pp. 1012–1032, June 2000.
15. Q. Zhang, G. Wang, W. Zhu, and Y.-Q. Zhang, "Robust scalable video streaming over Internet with network-adaptive congestion control and unequal loss protection," 11^{th} *Packet Video Workshop* , April 2001.
16. D. W. Redmill and N. G. Kingsbury, "The EREC: An error-resilient technique for coding variable-length blocks of data," *IEEE Transactions on Image Processing* **5**, pp. 565–574, April 1996.
17. ISO/IEC JTC1/SC29/WG11, "N4415: Systems AMD 4 PDAM (AFX and Multi User Worlds)," Feb. 2002.
18. E. S. Jang, "3-D Animation Coding: its History and Framework," *IEEE International Conference on Multimedia and Expo* , August 2000.
19. S. Varakliotis, J. Ostermann, and V. Hardman, "Coding of animated 3-D wireframe models for Internet streaming applications," *IEEE International Conference on Multimedia and Expo* , August 2001.
20. A. Moffat, R. Neal, and I. H. Witten, "Arithmetic Coding Revisited," *ACM Transactions on Information Systems* **16**, pp. 256–294, July 1998.
21. M. Tekalp and J. Ostermann, "Face and 2D mesh animation in MPEG-4," *Signal Processing: Image Communication* **15**, pp. 387–421, January 2000.
22. J. Signès, "BIFS: Combining MPEG-4 media to build rich multi-media services," *Signal Processing: Image Communication* **15**, January 2000.
23. E. O. Elliott, "A model of the switched telephone network for data communications," *Bell Systems Technical Journal* **44**, pp. 89–109, January 1965.
24. D. Clark and D. Tennenhouse, "Architectural considerations for a new generation of network protocols," *SIGCOMM Symposium on Communications Architectures and Protocols* , pp. 200–208, September 1990.
25. U. Horn, K. Stuhlmüller, M. Link, and B. 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**, pp. 77–94, September 1999.
26. N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "Mesh: Measuring errors between surfaces using the hausdorff distance," *IEEE International Conference on Multimedia and Expo* , August 2002.
27. J. Choi, Y. Kim, H.-J. Lee, I.-S. Park, M. Lee, and C. Ahn, "Geometry compression of 3-D mesh models using predictive two-stage quantization," *IEEE Transactions on Circuits and Systems for Video Technology* **10**, pp. 312–322, March 2000.
28. Z. Karni and C. Gotsman, "Spectral compression of mesh geometry," in *Siggraph 2000, Computer Graphics Proceedings*, K. Akeley, ed., pp. 279–286, ACM Press / ACM SIGGRAPH, 2000.