CODING OF ANIMATED 3-D WIREFRAME MODELS FOR INTERNET STREAMING APPLICATIONS

Socrates Varakliotis[1], Jörn Ostermann[2], Vicky Hardman[1]

[1]Computer Science Dept.
University College London
Gower Street, London WC1E 6BT, UK
{S.Varakliotis, V.Hardman}@cs.ucl.ac.uk

[2]AT&T Labs - Research
200 Laurel Ave South
Middletown, NJ 07748, USA
osterman@research.att.com

ABSTRACT

In this paper we present a coding technique for 3-D animated wireframe models, suitable for Internet streaming. First we present the MPEG-4 like coding scheme with simple RTP packetisation, and provide details of the achieved compression efficiency. Then, we describe a distortion metric for such a signal and we conduct experiments to study the effect of packet loss, following a bursty packet loss model that approximates a simulated IP network. Our experiments examine the performance of the coding scheme for a simple streaming scenario with different sequence configurations. The results show that short-term and short average length burst losses of up to 30% have a logarithmic effect on the decrease of the animation smoothness in the case of simple differential coding. This logarithmic decrease is corrected to linear by inserting I-frames to the sequence at the expense of reduced compression. The result is smoother animation.

1. INTRODUCTION

Image, video, audio and computer graphics represent a major source of multimedia in our time. Increasing demand of 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, international standardisation effort took 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 [1] in the context of MPEG-4.

MPEG-4 attempts to provide the 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 Scenes (BIFS) [3]. BIFS is the compressed binary format in which 3-D scenes are defined, modified (BIFS-Command) and animated (BIFS-Anim) [2]. It is derived from VRML that it extends by preserving backwards compatibility. In addition to BIFS, the MPEG-4 SNHC tools yield reasonably high compression for still textures, face and body animation, and 3-D mesh coding. In MPEG-4, the two main animation tools, BIFS-Anim and 'faceanim' are based on a DPCM system and arithmetic encoding that allows for low delay coding. The coding scheme we are describing in detail in the next section follows a similar block structure.

Designing a 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 [6] and vic [7] use RTP [5] with adaptive playout buffering algorithms to cope with variable delay. Loss is not possible to predict, 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. Best common practice guidelines for writers of RTP payload formats are provided in [9] and influence the design of our coding scheme throughout.

The rest of this paper is organised as follows. In section 2 we present our coding method, the bitstream packetisation scheme and discuss the achieved compression efficiency. In section 3 we describe our simulation model, the metric we used for the experiments and we analyse our results. Section 4 concludes this paper with reference to future work.

2. WIREFRAME CODING FOR THE INTERNET

The coding process assumes as a reference the initial wireframe model that is already present or downloaded at the receiver by other means. An *I-frame*, I_k, describes changes from the reference model I_0 to the model at the current time instant t_k. A *P-frame*, P_k, describes the changes of a model from the previous time instant t_{k-1} to the current time instant t_k. In the rest of the text we are using the following terms: A *vertex* is a point in 3-D space defined by a 3-tuple (its x, y, z coordinates). A *node* is the fundamental building block of a VRML world. Nodes may contain fields, which hold the data for the node. An *IndexedFaceSet* is the field that specifies which vertices to connect, and in what order, to draw the surfaces of a model.

Figure 1 shows a block diagram of our coding scheme. Diagram A depicts a DPCM coder that takes advantage of the temporal correlation of the displacement of each vertex along every axis in the 3-D space. Our signal is the set of non-zero displacements of all vertices and all nodes ($s_i\,[n, v]$) at time t_k. The decoded set (animation frame) of the previous instance is used as the predicted value ($\hat{s}_{i-1}[n, v]$). Then, the prediction error $e_i[n, v]$, i.e. the difference between the current displacement set and the predicted one, is computed and quantised ($\hat{e}_i[n, v]$). Finally, the quantised samples are entropy coded ($c_i[n, v]$) using an adaptive arithmetic coding algorithm to handle the unknown data statistics [10]. The predictive scheme described above prevents quantisation error accumulation. A DPCM decoder first decodes arithmetically the received samples ($c'_i[n, v]$) and computes the decoded samples ($\hat{s}'_i[n, v]$). Similar coding schemes have been used in MPEG-4 for the compression of facial animation parameters [4] and for BIFS-Anim [2].

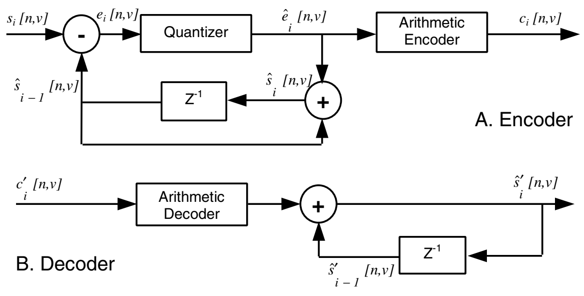


Figure 1: The block diagram for the encoder and decoder.

It is noted that our method encodes those vertices with with non-zero displacements in at least one of the three dimensions. At a frame level, we consider only those nodes with at least one vertex changing between frames. This process may generate sequences where not all nodes appear from frame F_k to frame $F_{k+1.}$ It may even be the case that no nodes change between consecutive frames, thus generating 'empty' frames. This property resembles the silence period inherent in speech audio streams and can be exploited in the application layer using RTP to build delay adaptive receivers that absorb network jitter. Furthermore, interstream synchronisation can be achieved, which is paramount for many applications (e.g. lip synchronisation of a virtual salesman with packet speech).

In order to describe which nodes of the model are to be animated we define the animation masks, NodeMask and VertexMasks, similar to BIFS-Anim [2]. The NodeMask is essentially a bitmask where each bit, if set, denotes that the corresponding IndexedFaceSet node in the Node Table will be animated. The Node Table (a list of all IndexedFaceSet nodes in the scene) is known a priori at the receiver since the wireframe model exists already, or is downloaded through other means. Similar to the NodeMask we have defined the VertexMasks, one per axis, for the vertices to be animated.

In its simplest form one RTP packet would contain one P-frame, which represents one Application Data Unit (ADU). In this sense, the codec's output bitstream is 'naturally packetisable' according to the Application Level Framing (ALF) principle [8]. We considered a packet format as the one shown in Figure 2. In principle, we start with the NodeMask and VertexMasks, followed by the encoded samples along each axis. The M bit in the RTP header must be set for the first of a series of 'empty' frames, which can be grouped together. For the degree of animation present in our sample sequence this simple format suffices. Other sequences with the same order of animation and scene complexity would be covered without any particular grouping and fragmentation rules or special consideration of the path MTU limits. A more sophisticated RTP payload format for both I- and P-frames and packet loss robustness is an issue of further design.

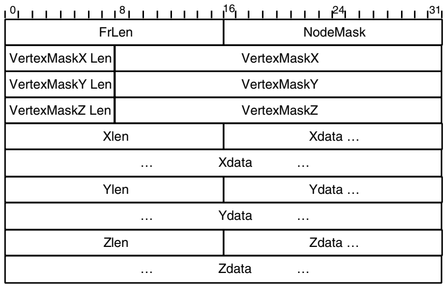


Figure 2: Packet format.

With the coding scheme described in this section we achieved the average compression efficiency per vertex component x, y, z shown in Table 1, assuming the same quantisation step size for all nodes. We determined the quantisation range of each node for the x-axis as the difference between the maximum and minimum displacement of the node along this axis. The same was done along axes y and z.

Table 1: Compression efficiency per component x, y, z for various quantiser step sizes and I-frame position.

	Component Compression Efficiency (bits/sample)								
I-frame:	Only frame 1			Every 20 frames			Every 8 frames		
Quantiser levels	x	y	z	x	y	z	x	y	z
256	4.80	5.67	5.85	4.84	5.79	5.89	4.91	5.95	5.97
128	4.37	5.21	5.27	4.41	5.32	5.31	4.47	5.45	5.39
64	3.84	4.75	4.70	3.88	4.83	4.74	3.94	4.95	4.81
32	3.39	4.28	4.23	3.41	4.33	4.25	3.46	4.42	4.29

3. EXPERIMENT SETUP AND RESULTS

For our experiments we have used the talking head model *Telly* shown in Figure 3. The wireframe is constructed in VRML 2.0 with the scene hierarchy shown in the same figure. The sequence *Telly* shows a talking head speaking a sentence that lasts for 190 frames at 30 Hz (≈6sec). The model smiles towards the end of the sentence. The mouth movements and facial expressions are achieved by shifting vertices of the IndexedFaceSets. With this technique we animated the nodes Skin, UpperLip, LoweLip, EyeCorner, EyeLash, EyeBrow and Nostril, appearing in italics in the scene graph of Figure 3. We used the model in its neutral position as the reference model. To extend the sequence, we concatenated the symmetric (mirrored) of the original sequence 3 times, totalling to 764 frames (≈25sec).

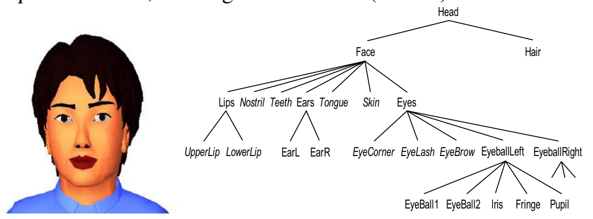
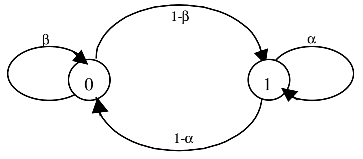


Figure 3: The wireframe model Telly with its scene graph.

2001 IEEE International Conference on Multimedia and Expo
ISBN 0-7695-1198-8/01 $17.00 © 2001 IEEE

0-7695-1198-8/01/$10.00 (C) 2001 IEEE

To simulate the packet loss process in the Internet we rely on the 2-state Gilbert model of Figure 4. States 0 and 1 represent no loss and loss state respectively. The probabilities of being in each of the states are expressed by *a* and *ß*, as shown in the diagram. P_{ul} denotes the probability of the model being in the loss state. We say that a burst loss has occurred when the model remains in state 1 for multiple consecutive packets. The average length of a burst is expressed by *b*. Due to the limited space of this publication the reader is referred to our past work in [11] for the analysis of this loss model. Suffice here to say that the equations in Figure 4 for $b \in [2..4]$ capture well the short-term bursty packet loss behaviour inherent in the Internet [12]. Medium and long-term outages are better described by higher-level models. A more detailed study of modelling the packet loss process in the Internet for streaming applications is presented in [13]. We used a 64-bit system timer to seed the random number generator. We repetitively ran the model simulation to generate loss patterns with loss rates $0 = L_k = 30$ (in percentage), for $b \in [2...4]$.



$$\alpha = 1 - \frac{1}{b}$$

$$\beta = 1 - p_{ul}\frac{1-\alpha}{1-p_{ul}}$$

Figure 4: A Gilbert model for packet loss simulation with its parameters.

To evaluate performance aspects of our coding scheme and the effects of packet loss on the resulting stream we consider the peak Signal-To-Noise Ratio (PSNR), as an objective distortion metric. For this particular type of signal coding, the PSNR does not relate to the quality of the animated picture as in natural video. It rather provides a quantitative expression of animation smoothness in time.

The frame PSNR is calculated based on the peak Mean Square Error (PMSE) per vertex and per node. Note how this formula captures the animation samples present in the P-frames by considering only those vertices and nodes that have changed since the previous frame.

$$PMSE = \frac{\sum_{n=1}^{N}\frac{1}{N_n}\sum_{v=1}^{P}\frac{1}{P_v}\sum_{i=x,y,z}\frac{\left[s_i(n,v) - \tilde{s}_i'(n,v)\right]^2}{3}}{R^2}$$

$$PSNR = -10 \cdot \log_{10}\left(PMSE\right)$$

$s_i(n,v)$ = original sample of vertex v in node n on axis i
$\hat{s}_i'(n,v)$ = decoded sample of $s_i(n,v)$
R = maximum displacement of $s_i(n,v)$
N_n = number of changed nodes in a P - frame
P_v = number of changed vertices in a node of a P - frame

In the following subsections we describe two experiments we conducted in a simulated IP network with packet loss for two different sequence configurations.

3.1 *Experiment 1, sequence: I, P, P, P, …*

In the first experiment we measured the average PSNR for each node in the 3-D wireframe model for different packet loss rates up to 30%. We first examined a sequence made of P-frames only. We assume here that the 3-D reference model is already available at the decoder. This is equivalent to an I-frame preceding the sequence of P-frames. The results for two of the nodes (Skin and Teeth) are shown in the lower part (2 plots) of Figure 5.

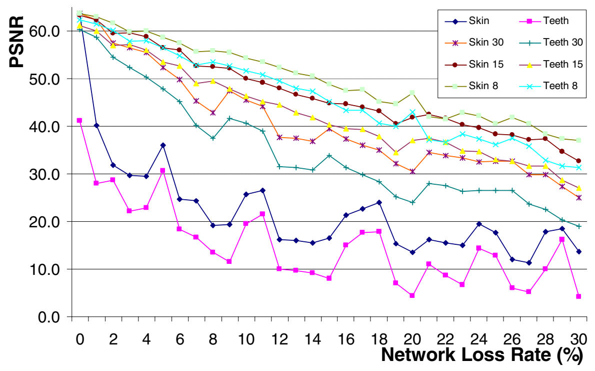


Figure 5: The average node PSNR against network loss rate (nodes Skin & Teeth) for various I-frame repetitions (none, every 30, 15, or 8 frames).

The graphs show a logarithmic decreasing trend in the PSNR for the two nodes, with a steep fall in the loss range 0-8% and a more linear decrease for loss greater than 9%. Node Skin in the initial model has an IndexedFaceSet of 471 vertices with 170 of them moving on average (with motion on all axes x, y, z) between two consecutive frames, whereas node Teeth has a total of 128 vertices with average activity of 75.

It is also worth noticing the graph at loss rates 5, then at 10 and 11, at 16 to 18, and at 24 and 29%. In these cases the average PSNR increases from the previous experiment iteration with lower loss rate. This does not imply a false result. In fact, after examining the simulated loss pattern we noticed more frequent and longer loss bursts taking place in the specified loss rates where this seeming anomaly occurs. Recall that the animated sequence is a head-and-shoulders talking avatar with many repetitive movements around the mouth area as a result of it talking. In the case of a lost frame we apply a frame replenishment algorithm that repeats the last successfully received frame (or packet, since we fit one frame in every packet), thus linearly extrapolating the motion of the previous frame. In our coding scheme this frame replenishment technique results in the model preserving its 'motion vectors' and amplitude of animation samples, in an attempt to minimise the perceptual distortion of the model's animation. We are currently investigating further the effects of error resiliency in 3-D animation streams through different replenishment algorithms.

During the course of this experiment we noticed that the anomaly in the average PSNR described above may, or may not, appear with various non-cumulative[1] loss patterns. Had the loss pattern been cumulative between successive loops of the experiment we would have expected a smoother plot. To better investigate the effect of the loss pattern on the average node PSNR we would need much longer sequences and possibly higher-level animation. We iterated this experiment several times and we acquired similar graphs, with PSNR anomalies as discussed above. Each time they are positioned at different loss rates. This is a result of the non-deterministic loss patterns generated in every step of our simulation.

3.2 Experiment 2, sequence: I, P, P, …, I, P, P, …

For the second experiment, we repeated the same process as in experiment 1 after inserting I-frames at regular intervals in the *Telly* sequence.

Table 2 shows the average output bitrate at the encoder for 8-bit quantization and full frame rate (30 Hz) for various *Telly* sequence configurations (I-frames every 30, 15, 8 frames or none). This corresponds to a high-quality stream for the sequence *Telly* achieved with our encoding, which shows very good and smooth quality animation. Lower quantization step sizes and frame rates produce proportionally lower bitrates.

Table 2: Average output bitrate for various I-frame positions.

I-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

We also plotted the modified average PSNR for 2 nodes, Skin and Teeth, first with an I-frame every 30 frames and, then, one every 15 and every 8 frames. For comparison, we used the same loss pattern as in the previous experiment. The PSNR results can be seen in the upper part (6 plots) of Figure 5.

The graph in Figure 5 shows that the average PSNR per node increases as the I-frame frequency in the sequence increases. It is clear that the PSNR increase is noticeable at loss rates above 3%, however, this is an average figure. It is also noticeable that the logarithmic trend of PSNR decrease in the animation stream without I-frames (lower 2 plots), tends to become linear in the case of I-frames (upper 6 plots). The actual effect of this on the sequence is smoother animation.

A closer look on Figure 5 also points out that for some loss rates (i.e. 21-22, 25-26) at high I-frame rates (15 and 8 Hz) the average PSNR does not show considerable increase in either case of one I-frame every 15 or 8 frames. The justification to this effect comes by examining again the loss pattern in the previous loss rates of 20% and 24%. In each of these cases we noticed that packet loss has affected I-frames, resulting in a lower actual I-frame frequency in the sequence than the nominal (15 or 8 Hz).

[1] Cumulative here means to keep the same loss pattern P_k in the experiment between two successive loops k (with loss rate R_k) and k+1, as if it was a deterministic process, and just drop some extra packets on P_k, until we achieve the requested loss rate R_{k+1} for loop k+1, where $R_k < R_{k+1}$.

4. CONCLUSIONS

In this paper, we have presented a coding scheme for 3-D wire-frame models suitable for Internet streaming applications. The compression scheme gives efficient coding of IndexedFaceSet nodes and generates animation bitmasks similar to those in BIFS-Anim. The resulting bitstream has a flexible format for direct RTP packetisation to use in IP streaming applications. Combined with powerful endpoint stations our compression and animation method is suitable for real-time applications such as videoconferencing and e-commerce avatars.

Further considerations on the compression technique include the addition of error resilience in a content-based fashion that respects the ALF principle. The enhanced bitstream (and resulting packet stream) would then cope in a robust manner with delay and jitter in an IP environment and would be better suited for higher-level animations. Finally, synchronisation of such a stream with other high-quality streams (i.e. high-quality audio) will enable rich multimedia services.

5. REFERENCES

[1] E. S. Jang. “3-D Animation Coding: its History and Framework”. In *International Conference on Multimedia and Expo 2000 (ICME2000)*. New York, Aug. 2000.

[2] J. Signès. “BIFS: Combining MPEG-4 media to build rich multimedia services”. *Signal Processing: Image Communication*, Vol. 15, No. 4-5, Jan. 2000.

[3] J. Signès, Y. Fischer, A. Eleftheriadis. “MPEG-4’s Binary Format for Scene Description”. *Signal Processing: Image Communication*, Vol. 15, No. 4-5, pp. 321-345. Jan. 2000.

[4] A. M. Tekalp, J. Ostermann. “Face and 2D Mesh Animation in MPEG-4”. *Signal Processing: Image Communication*, Vol. 15, No. 4-5, pp. 387-421. Jan. 2000.

[5] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. “RTP: A Transport Protocol for Real Time Applications”. RFC 1889, Internet Engineering Task Force. Jan. 1996.

[6] V. Hardman, M. A. Sasse, I. Kouvelas. “Successful Multiparty Audio Communication over the Internet”. *Communications of the ACM*, Vol. 41, No. 5, 1998. http://www-mice.cs.ucl.ac.uk/multimedia/software/rat

[7] S. McCanne, V. Jacobson. “vic: A Flexible Framework for Packet Video”. In *ACM Multimedia ’95*, San Francisco, CA. November 1998. http://www-nrg.ee.lbl.gov/vic

[8] D. Clark, D. Tennenhouse. “Architectural Considerations for a New Generation of Network Protocols”. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 200-208. Sept. 1990.

[9] M. Handley, C. Perkins. “Guidelines for Writers of RTP Payload Format Specifications”. RFC 2736, Internet Engineering Task Force. Dec. 1999.

[10] A. Moffat, R. Neal, I. Witten. “Arithmetic Coding Revisited”. *ACM Trans. on Information Systems*, Vol. 16, No. 3, pp. 256-294. July 1998.

[11] A. Basso, S. Varakliotis, R. Castagno, F. Lohan. “Transport of MPEG-4 over IP/RTP”. In *European Trans. on Telecommunications: special issue on Packet Video 2000 Workshop*. To appear.

[12] J-C.Bolot. “End-to-End Packet Delay and Loss Behavior in the Internet”. In *SIGCOMM ’93*, pp. 289-298. Sept. 1993.

[13] H. Sanneck. “Packet Loss Recovery and Control for Voice Transmission over the Internet”. Ph.D. thesis, Technical Univ. of Berlin. Oct. 2000.