MODEL–BASED MPEG COMPRESSION OF SYNTHETIC VIDEO SEQUENCES

Davide Quaglia, Angelo Gattuso

Dipartimento di Automatica e Informatica
Politecnico di Torino
Corso Duca degli Abruzzi, 24 — I-10129 Torino, Italy
E-mail: davide.quaglia@polito.it

ABSTRACT

The paper addresses the problem of improving the MPEG compression of synthetic video sequences by exploiting the knowledge about the original 3D model. Two techniques are proposed for the specific case of a virtual walkthrough in which the point of view is the unique moving object in the scene. Technique 1 consists of using only P–frames when position and direction of the point of view do not change since, in this case, each frame is equal to the previous one; P–frames can be simply repeated without any encoding effort thus reducing the computational complexity. Technique 2 consists of increasing the quantization parameter when the direction of the point of view is changing, since the resulting increase of distortion is not perceived clearly for fast–moving objects because of the temporal masking effect. Experimental results compared with model–unaware encoding shows that Technique 1 reduces the bitstream size by about 9% without any appreciable decrease of perceptual quality while CPU encoding time is reduced by about 18%. The combination of both techniques reduces the bitrate by about 13% with a slight increase of the quantization noise which is partially hidden by the temporal masking effect. Video samples are available at
http://media.polito.it/mpeg3d/.

1. INTRODUCTION

Multimedia distributed applications are going to play a key role for both entertainment and training, pushed by the growing synergy between digital video, computer graphics, and networking technologies. In this context, synthetic video sequences are used in animation movies, video games and virtual reality applications (e.g., immersive collaborative environments and scientific visualization tools). In such applications, video sequences have to be stored or transmitted and, therefore, compression should be applied to them. In general, even for natural sequences, better compression results can be obtained if the underlying model is extracted and coded; MPEG–4 [1] object–based video coding standard is an example of this effort. In case of synthetic sequences, the highest compression efficiency can be achieved by coding the original model using a standard format like the Virtual Reality Modeling Language (VRML) [2], the Extensible 3D (X3D) [3], and the MPEG–4 Binary Format for Scenes (BIFS) [4]. The distribution of the original model to end–users may not be advisable for copyright reasons and for the need of a rendering application in each client. An alternative approach consists of distributing a compressed version [5, 6, 7, 8]. Even if specific compression techniques for synthetic sequences have been proposed [9, 6, 7], the use of a traditional frame–based video coding standard like MPEG–1 or MPEG–2 can be an appealing approach since MPEG codecs are often embedded in many clients (e.g., DVD players) and no additional software would be required. Coding techniques in MPEG are mainly designed for natural video, not graphics; however, the knowledge of the synthetic model can contribute to reduce bitrate and computational complexity and to enhance quality. In particular, computational complexity reduction is a desirable factor in a distributed environment where encoding is performed on the server for many clients [8].

In this work the motion information provided by the 3D animation engine is exploited to 1) quickly estimate interframe correlation, and 2) establish when fast–moving objects are present in the scene. Both techniques are applied to the coding of the *virtual walkthrough* as in [8]. The knowledge of the movements of the point of view in the 3D environment allows a computational efficient identification of the intervals in which scene content does not change and the intervals of high motion; in the first case, predictive coding without motion compensation can be exploited to lower bitrate and computational complexity without loss of quality; in the second case, coarse quantization can be applied since eyes cannot appreciate details of fast–moving objects.

The paper is organized as follows. Section 2 briefly introduces the system from the point of view of the animation and coding environment. Section 3 describes two model–

This work was supported in part by MIUR, Project FIRB-PRIMO, http://primo.ismb.it.

0-7803-8554-3/04/$20.00 ©2004 IEEE.

based techniques for the coding of the virtual walkthrough. Experimental results are reported in Section 4. Finally, conclusions are drawn in Section 5.

2. SYSTEM OVERVIEW

Figure 1 shows the layout of a client–server system to which the proposed techniques can be applied. The 3D model is stored on the server. The animation engine is a computer graphics application which applies a set of geometrical transformations to the model obtaining an animation; the geometrical transformations can be driven by the remote user (e.g., he can move the point of view of the scene through client's keyboard). The animation engine also creates a bi–dimensional view of the scene transforming the animation in a sequence of frames (e.g., arrays of luminance and chrominance samples). Frames are fed into a frame–based video encoder belonging to the widespread MPEG or H.26X families. A subset of the motion information held by the animation engine is also transferred to the video encoder to improve compression; they consist of the translation and rotation parameters of the point of view of the scene. The client receives the compressed stream, decodes it, and displays frames. Many clients such as mobile devices and set–top boxes have a standard MPEG-2 decoder (e.g., DVD players).

In our work the frame–based video encoder is a standard MPEG encoder modified to manage motion information given by the animation engine. MPEG exploits the general similarity of adjacent frames by using motion vectors for each 16×16 pixel macroblock to point to another 16×16 block in a previous or future frame. The MPEG coder records the motion vector itself and the differences between previous and current pixels. Since many of the pixels will be largely the same, the difference will contain many zeros or small terms, requiring fewer bits to be encoded.

MPEG provides three frame coding types: *intra–coded* I–frames, *predicted* P–frames, and *bi–directional* B–frames. Quality and coding efficiency of each of them are related to the encoding method and to the video content. Intra–coded frames are compressed without references to other frames; as a result they consume more bits, but have less error and are useful when the frame being coded is quite different from its neighbors. Predicted frames contain motion vectors which reference the previous I– or P–frame, along with the difference between blocks; because the differences are usually small, they compress well. Bi–directional frames contain motion vectors which reference the nearest I– or P–frames in their past and future. They increase the coding efficiency with respect to P–frames when the content of the scene changes and new objects cannot be predicted from the past but only from future frames. Even if many MPEG sequences follow a repeating pattern of these frame types such as III, IPPP, or IBBPBB, this feature is not constrained by the MPEG standard.

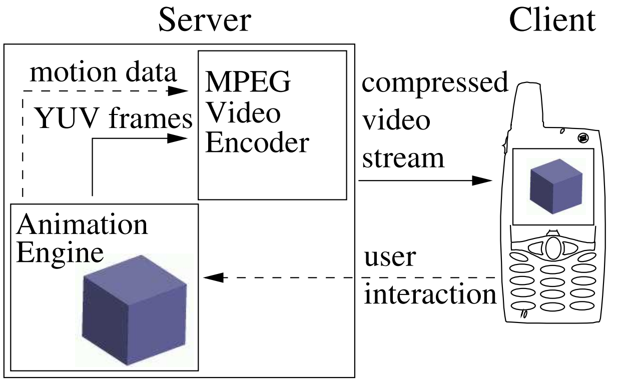

Fig. 1. Overview of the system.

3. VIRTUAL WALKTHROUGH

The virtual walkthrough is a particular computer graphics scenario in which the user holds the point of view in the 3D scene and can walk and look at objects. This model is used in virtual museum explorations and many well–known video–games. Since moving objects introduce changes in the frame content, information about their trajectory can be exploited to quickly determine inter–frame correlation. In this work we assume that the subject is the only moving object in the scene as in many virtual scene explorations. According to this assumption, when the subject does not change its position, the animation engine generates a number of identical frames. As described above, different frame encoding types are provided by MPEG. We propose to adapt the choice of the encoding mode to the correlation between the current frame and its neighbors which is estimated from the motion information provided by the animation application.

In Technique 1, we propose to use only P–frames when the point of view of the subject does not change since, in this case, each frame is equal to the previous one. Since frame content does not change, motion vectors and pixel differences are zero and the resulting P–frames are very small. In this way, many bits required by I–frames are saved and the resulting bitrate is more constant than in case of regular IPB pattern. P–frames are encoded with the same quality of I–frames and, therefore, encoding distortion does not increase. Moreover, compressed frames can be simply repeated without any encoding effort thus reducing the computational complexity. Figure 2 shows the bitrate as a function of frame number for a synthetic test sequence representing a virtual walkthrough; from frame 142 to 190 position and direction of the point of view do not change.

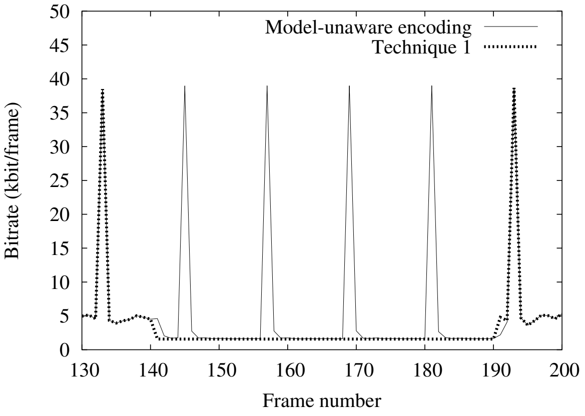

Fig. 2. Bitrate of the compressed video as a function of frame number; from frame 142 to 190 the position of the point of view does not change.

Model–unaware encoding is performed using a regular pattern in which an I–frame is followed by eleven P–frames. The regular frame pattern produces a variable bitrate whose peaks correspond to I–frames. In fact, the size of I–frames is independent of inter–frame correlation. Therefore, the knowledge of the movements of the subject allows to optimize the choice of the frame encoding mode in order to minimize bitrate and computational complexity without degrading video quality. Moreover, bitstreams with a more regular bitrate can be easier transmitted over traditional networks.

The second technique exploits the knowledge of the direction of the subject's gaze. When the subject turns its head to the left or to the right, new elements enter or leave the scene. The resulting frames may require many more bits to be encoded since temporal prediction fails. On the other hand, due to the temporal masking effect [10], there is a small latency period during which the new appeared objects are not perceived clearly and, therefore, they can be encoded with coarse quantization. The wider is the rotation of the head, the more significant is the temporal masking effect. In our proposed technique, the quantization parameter (QP) is chosen for each frame according to (1), where $|\alpha_i|$ is the absolute value of the angular variation of gaze direction between the i–th frame and the previous one; since time interval between frames is constant, α_i is proportional to the angular speed. QP_{base} is the default value of the quantization parameter (e.g., obtained with a rate control algorithm); K is a normalization constant.

$$QP_i = K|\alpha_i| + QP_{base} \tag{1}$$

It is worth noting that while the first proposed technique is virtually lossless since only the frame coding mode is changed, this technique is lossy since more quantization noise is introduced during gaze rotations.

Table 1. Performance of the first and the second technique with respect to model–unaware MPEG encoding.

	Avg. QP (1–31)	Size (kbyte)	PSNR (dB)	t_{CPU} (s)
Model–unaware	13.0	1257	41.9	477.9
Technique 1	13.0	1148	41.7	390.0
Technique 1+2	13.9	1093	41.5	390.0

4. EXPERIMENTAL RESULTS

The performance of the proposed techniques is tested with a 3D scene created and animated by Pov–Ray v. 3.5. The 3D scene consists of a number of boxes of different sizes and colors laying on the floor. To simplify the rendering process no texture is applied to the objects. The horizon is at infinite distance from the viewer. The light source is placed on the point of view and the illumination model is radial. The subject moves horizontally among boxes, stops and rotates. Rotations and translations are independent and are combined in different ways.

The generated sequence consists of 1000 frames; the resolution is 512×384 pixels and the chrominance format is 4:2:0. The 25% of the sequence corresponds to intervals in which the subject does not change its position and gaze direction; changes in gaze direction are distributed along the whole sequence.

Frames are encoded with the TM5 MPEG Encoder [11] which has been modified to support the proposed techniques. The default GOP structure consists of an I–frame followed by eleven P–frames; with reference to (1) we set QP_{base}=13 and K=2 for the whole sequence.

Bitstreams encoded with the proposed techniques are compared with a model–unaware bitstream encoded using the default parameters. Table 1 compares the performance of the three approaches. Technique 1 reduces the bitstream size by about 9% without any appreciable decrease of perceptual quality; CPU encoding time is also reduced by about 18%. The combination of both techniques reduces bitrate by about 13% with respect to model–unaware encoding. The mean value of QP is higher with Technique 2 since QP is increased when gaze direction changes; this leads to a slight increase of the distortion which should not be perceived because of the temporal masking effect.

Figure 2 shows the bitrate as a function of frame number for a fragment of the compressed stream; from frame 142 to 190 position and direction of the point of view do not change. Technique 1 is compared with model–unaware encoding with regular I–P pattern; bits required by I–frames are saved and the resulting bitrate is more constant.

Figure 3 compares the effect on bitrate of Technique 2

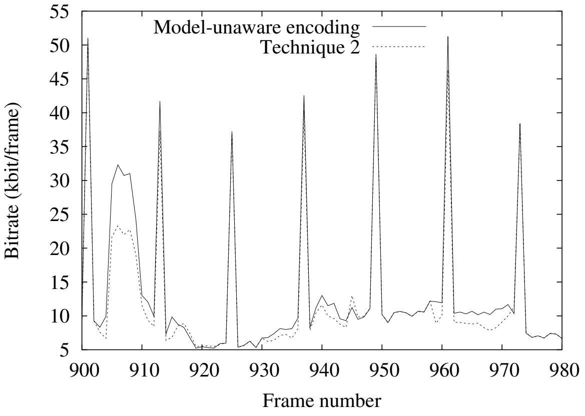

Fig. 3. Bitrate of the compressed video as a function of frame number when the gaze direction changes.

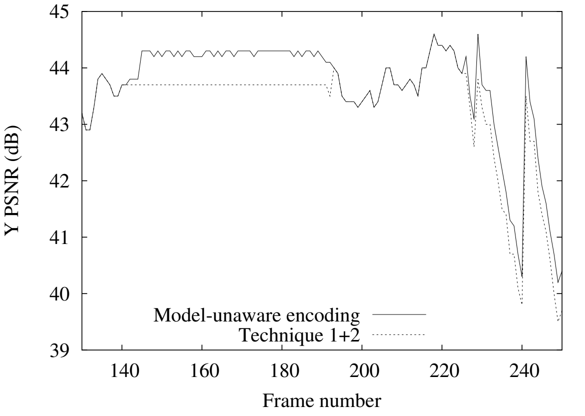

Fig. 4. PSNR of the compressed video as a function of frame number; from frame 142 to 190 the position of the point of view does not change.

with respect to model–unaware encoding for a fragment of the compressed stream; Technique 2 increases the quantization parameter when gaze direction changes thus reducing the bitrate except for frame 945 in which the efficiency of prediction from the previous coarse–quantized frame is reduced. This issue shall be addressed in future works.

Figure 4 shows the PSNR as a function of frame number for a fragment of the compressed stream. The proposed techniques is compared with model–unaware encoding. When the position of the point of view does not change (i.e., from frame 142 to 190) the PSNR is lower but more constant and, indeed, the decoded video does not exhibit the flickering effect present in the model–unaware bitstream. From frame 225 to 250 the PSNR decrease is mainly located where the temporal masking effect is expected to be more significant.

5. CONCLUSIONS

We have shown that the compression of synthetic animations with traditional frame–based video encoding standards can be improved by exploiting the knowledge of the 3D model. We presented two techniques for the specific scenario of the virtual walkthrough encoded with MPEG–2. The first technique adapts the frame encoding type to the inter–frame correlation which is estimated from the motion information provided by the animation software; in particular, P–frames are simply repeated when the position and the direction of the point of view do not change. The second technique increases the quantization parameter when objects move quickly because of the rotation of subject's gaze; in this case the effect of temporal masking compensates the increase of the quantization noise. Experimental results compared with model–unaware encoding shows that Technique 1 reduces the bitstream size by about 9% without any appreciable decrease of perceptual quality while CPU encoding time is reduced by about 18%. The combination of both techniques reduces bitrate by about 13%.

6. REFERENCES

[1] ISO/IEC, "MPEG-4 - Information technology - Coding of audio-visual objects," *ISO/IEC 14496*, 2000.

[2] ISO/IEC, "Information technology – Computer graphics and image processing – The Virtual Reality Modeling Language (VRML) – Part 1: Functional specification and UTF-8 encoding," *ISO/IEC 14772-1*, 1997.

[3] ISO/IEC, "Information technology – Computer graphics and image processing – Extensible 3D (X3D)," *ISO/IEC FDIS 19775*, 2003.

[4] J. Signes, Y. Fisher, and A. Eleftheriadis, "MPEG-4's binary format for scene description," *Signal Processing: Image Communication*, vol. 15, no. 4–5, pp. 321–345, 2000.

[5] M. Levoy, "Polygon-assisted jpeg and mpeg compression of synthetic images," in *Proc. ACM SIGGRAPH*, , 1995, pp. 21–28.

[6] D. Cohen-Or, "Model-based view-extrapolation for interactive vr web-systems," in *Proc. IEEE Int. Conf. Computer Graphics*, , June 1997, pp. 104–112.

[7] I. Yoon and U. Neumann, "Compression of computer graphics images with image-based rendering," in *Proc. of SPIE Multimedia Computing and Networking*, , 1999, pp. 66–75.

[8] Y. Noimark and D. Cohen-Or, "Streaming scenes to MPEG-4 video-enabled devices," *IEEE Computer Graphics and Applications*, vol. 23, no. 1, pp. 58–64, Jan–Feb 2003.

[9] B. Guenter, H. Yun, and R. Mersereau, "Motion compensated compression of computer animation frames," in *Proc. ACM SIGGRAPH*, , 1993, pp. 297–304.

[10] N. S. Jayant, J. Johnson, and R. Safranek, "Signal compression based on models of human perception," *Proc. IEEE*, vol. 81, no. 10, pp. 1385–1422, Oct. 1993.

[11] S. Eckart and C. Fogg, "ISO/IEC MPEG-2 software video codec," *Proc. SPIE*, vol. 2419, pp. 100–118, Apr. 1995.