Controlling and Efficient Coding of MPEG-4 Compliant Avatars

Tolga K. Capin, Daniel Thalmann

Computer Graphics Laboratory (LIG)
Swiss Federal Institute of Technology (EPFL)
1015 Lausanne, Switzerland
{capin, thalmann}@lig.di.epfl.ch

Abstract

The MPEG-4 includes support not only for natural video and audio, but also for synthetic graphics and sounds. MPEG-4 Version 2 includes the representation of human bodies in addition to faces. In the MPEG-4 Version 2 Committee Draft, Body Animation Parameters (BAPs) and Body Definition Parameters (BDPs) allow virtual bodies to be streamed in very low bit rates and provide deformation of body surface based on skeleton posture. In this paper, we overview body coding using MPEG-4, discuss MPEG-4 compliant avatar control, and we present our experimental results.

Keywords: *MPEG-4, SNHC, Body Animation*

1. Introduction

ISO/IEC JTC1/SC29/WG11 (Moving Pictures Expert Group - MPEG) is currently working on the Version 2 of the MPEG-4 standard scheduled to become International Standard in December 1999. In a world where audio-visual data is increasingly stored, transferred and manipulated digitally, MPEG-4 sets its objectives beyond 'plain' compression. Instead of regarding video as a sequence of frames with fixed shape and size and with attached audio information, the video scene is regarded as a set of dynamic objects. Thus the background of the scene might be one object, a moving car another, the sound of the engine the third etc. The objects are spatially and temporally independent and therefore can be stored, transferred and manipulated independently. The composition of the final scene is done at the decoder, potentially allowing great manipulation freedom to the consumer of the data.

Video and audio acquired by recording from the real world is called natural. In addition to the natural objects, synthetic, computer generated graphics and sounds are being produced and used in ever increasing quantities. MPEG-4 aims to enable integration of synthetic objects within the scene. It will provide support for 3D Graphics, synthetic sound, Text to Speech, as well as synthetic faces and bodies. In this paper we concentrate on the representation of bodies in MPEG-4, and in particular the efficient coding of body animation.

The following section provides the introduction to the representation of bodies in MPEG-4. We explain how Body Animation Parameters and Body Definition Parameters are used to define the shape and animation of bodies. Then we present our algorithm for efficient encoding of Body Animation Parameters and Body Definition Parameters. In the final sections we present the results and conclusions, as well as the ideas for future work.

2. Body Animation in MPEG-4

Conceptually the FBA object consists of a collection of nodes in a scene graph which are animated by the FBA object bitstream. The shape, texture and expressions of the face are generally controlled by the bitstream containing instances of Facial Definition Parameter (FDP) sets and/or Facial Animation Parameter (FAP) sets. Upon construction, the FBA object contains a generic face with a neutral expression and a generic body with a default posture. This model can already be rendered. It is also immediately capable of receiving the FAPs and BAPs from the bitstream, which will produce animation of the face and body. If FDPs and BDPs are received, they are used to transform the generic model into a particular model determined by its shape and (optionally) texture.

Upon construction, the Body object contains a generic virtual human or human-like body with the default posture. This body can already be rendered. It is also immediately capable of receiving the BAPs from the bitstream, which will produce animation of the body. If BDPs are received, they are used to transform the decoder's generic body into a particular body determined by the parameter contents. Any component can be null. A null component is replaced by the corresponding default component when the body is rendered. Similar to the face, the BAPs can be transmitted also without first downloading BDPs, in which case the decoder animates its local model.

No assumption is made and no limitation is imposed on the range of defined mobilities for humanoid animation. In other words the human body model should be capable of supporting various applications, from realistic simulation of human motions to network games using simple human-like models.

2.1 Structure of the FBA bitstream

A face and body object is formed by a temporal sequence of face and body object planes. An FBA object represents a node in an ISO/IEC 14496 scene graph. An ISO/IEC 14496 scene is understood as a composition of Audio-Visual objects according to some spatial and temporal relationships. The scene graph is the hierarchical representation of the ISO/IEC 14496 scene structure (see ISO/IEC 14496-1).

Alternatively, an FBA object can be formed by a temporal sequence of FBA object plane groups (called segments for simplicity), where each FBA object plane group itself is composed of a temporal sequence of 16 FBA object planes

2.2 Body Animation Parameters

BAP parameters comprise joint angles connecting different body parts. These include: toe, ankle, knee, hip, spine (C1-C7, T1-T12, L1-L5), shoulder, clavicle, elbow, wrist, and the hand fingers. The detailed joint list, with the rotation normals, are given in the following section.

Note that the normals of rotation move with the body, and they are fixed with respect to the parent body part. That is to say, the axes of rotation are not aligned with the body or world coordinate system, but move with the body parts.

The hands are capable of performing complicated motions and are included in the body hierarchy.

The unit of rotations is defined as 10^{-5} radians. The unit of translation BAPs (BAPs tr_vertical, tr_lateral, tr_frontal) is defined in millimeters.

2.3 BAP Grouping

In order to further decrease the bandwidth requirements and facilitate communication, the joints comprising the body can be partitioned into a finite set of groups with respect to their interrelationships and importance. For example, joints related to the spine can be grouped. In this way, if the motion affects only one part of the body, only the joints of that part of the body which change in the frame are coded and sent through the bitstream to the server, and then other clients. For example, if the virtual human is waving with their right arm, only joints involved in moving the right arm are sent through the network.

We divide the body degrees of freedom into the groups shown in Table 1. Complete degrees of freedoms are given in MPEG-4 Version 2 (PDAM1) specification. The groups can be sent separately by introducing a mask for each group, and inserting this mask in the beginning of the message. The mask has the following format:

<13-bit mask><4-bit mask><dofs for each group in mask>...

For example, to send only arm joints, the message has the following format:

<0000000011000><0000><5floats><7floats>

This decreases the size of the message from 408 bytes to 60 bytes. Thus, with an additional overhead of 3 bytes, we can decrease the message size significantly.

The detailed list of BAP groups and the associated BAPs are given in MPEG-4 PDAM1.

Pelvis:
sacroiliac_tilt, sacroiliac_torsion, sacroiliac_roll
Left leg1:
l_hip_flexion,l_hip_abduct,l_knee_flexion,l_ankle_flex
Right leg1:
r_hip_flexion,r_hip_abduct,r_knee_flexion,r_ankle_flex
Left leg2:
l_hip_twisting, l_knee_twisting, l_ankle_twisting,
l_subtalar_flexion,l_midtarsal_flexion,l_metatarsal_flex
Right leg2:
r_hip_twisting, r_knee_twisting, r_ankle_twisting,
r_subtalar_flexion,r_midtarsal_flexion,r_metatarsal_flex
Left arm1:
l_shoulder_flexion,l_shoulder_abduct,l_shoulder_twist
l_elbow_flexion, l_wrist_flexion
Right arm1:
r_shoulder_flexion,r_shoulder_abduct,r_shoulder_twist
r_elbow_flexion, r_wrist_flexion
Left arm2:
l_sternoclavicular_abduct, l_sternoclavicular_rotate,
l_acromioclavicular_abduct, l_acromioclavicular_rot,
l_elbow_twisting, l_wrist_pivot, l_wrist_twisting
Right arm2:
r_sternoclavicular_abduct, r_sternoclavicular_rotate,
r_acromioclavicular_abduct,r_acromioclavicular_rot,
r_elbow_twisting, r_wrist_pivot, r_wrist_twisting
Spine1:
skullbase_roll, skullbase_torsion, skullbase_tilt,
vc4roll, vc4torsion, vc4tilt,vt6roll, vt6torsion, vt6tilt,
vl3roll, vl3torsion, vl3tilt,
Spine2:
vc2roll, vc2torsion, vc2tilt,vt1roll, vt1torsion, vt1tilt,
vt10roll,vt10torsion,vt10tilt, vl1roll, vl1torsion, vl1tilt,
vl5roll, vl5torsion, vl5tilt
Spine3:
vc3roll, vc3torsion, vc3tilt,vc6roll, vc6torsion, vc6tilt,
vt4roll, vt4torsion, vt4tilt,vt8roll, vt8torsion, vt8tilt,
vt12roll, vt12torsion, vt12tilt,vl4roll,vl4torsion, vl4tilt,
Spine4:
vc5roll, vc5torsion, vc5tilt, vc7roll, vc7torsion, vc7tilt
vt2roll, vt2torsion, vt2tilt, vt7roll, vt7torsion, vt7tilt,
vt11roll, vt11torsion, vt11tilt,vl2roll,vl2torsion, vl2tilt,
Spine5:

vc1roll, vc1torsion, vc1tilt, vt3roll, vt3torsion, vt3tilt, vt5roll, vt5torsion, vt5tilt, vt9roll, vt9torsion, vt9tilt,
Left hand1:
l_pinky1_flexion,l_pinky2_flexion,l_pinky3_flexion, l_ring1_flexion, l_ring2_flexion, l_ring3_flexion, l_middle1_flexion,l_middle2_flexion,l_middle3_flex l_index1_flexion, l_index2_flexion,l_index3_flexion, l_thumb1_flexion, l_thumb1_pivot, l_thumb2_flexion, l_thumb3_flexion
Right hand1:
r_pinky1_flexion, r_pinky2_flexion,r_pinky3_flexion, r_ring1_flexion, r_ring2_flexion, r_ring3_flexion, r_middle1_flexion,r_middle2_flexion,r_middle3_flex r_index1_flexion, r_index2_flexion, r_index3_flexion, r_thumb1_flexion, r_thumb1_pivot, r_thumb2_flexion,r_thumb3_flexion
Left hand2:
l_pinky0_flexion,l_pinky1_pivot,l_pinky1_twisting, l_ring0_flexion,l_ring1_ pivot,l_ring1_twisting, l_middle0_flexion,l_middle1_pivot,l_middle1_twist l_index0_flexion,l_index1_ pivot, l_index1_twisting,l_thumb1_twisting
Right hand2:
r_pinky0_flexion, r_pinky1_pivot,r_pinky1_twisting, r_ring0_flexion,r_ring1_ pivot, r_ring1_twisting, r_middle0_flexion,r_middle1_pivot,r_middle1_twist r_index0_flexion, r_index1_pivot, r_index1_twisting, r_thumb1_twisting
Global positioning:
HumanoidRoot_tr_vertical,HumanoidRoot_tr_lateral, HumanoidRoot_tr_frontal,HumanoidRoot_rt_turn, HumanoidRoot_rt_roll,HumanoidRoot_rt_tilt

2.4 BAP Coding

For each joint in the state vector, the quantization module stores a quantum value. The quantum value indicates what the step size is going to be for that joint angle in the compressed representation of the joint angle parameter. Thus, a quantum value of 1 indicates the angle will be encoded with the most precision, and 255 indicates a lower precision. Note that each degree of freedom has a different precision requirement. Therefore different quantization step sizes are applied to each degree of freedom. The base quantization step sizes for each joint angle are presented in the next chapter.

The actual formula to obtain quantized state vector *S'* from *S* is

Quantized Value (*i*) =
StateVector(*i*)/(Quantum(*i*)*Global_Quantization_Value)
→Rounded to the nearest integer
(for each joint angle *i*)

During decoding, the dequantization formula works in reverse:

StateVector'(*i*) = QuantizedValue(*i*) * Quantum (*i*)
(for each joint angle *i*)

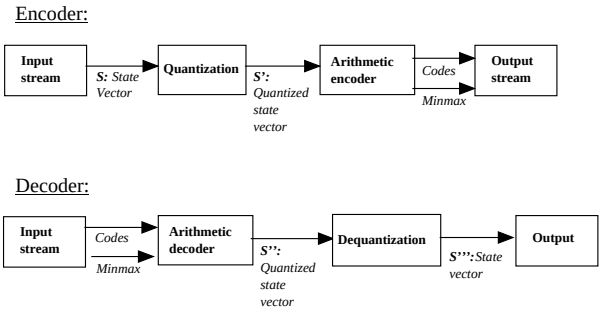

Figure 1: Dataflow of scalable compression

The bit rate is controlled by adjusting the quantization step via the use of a quantization scaling factor called *Global_Quantization_Value.* This value is applied uniformly to all DOFs. The magnitude of the quantization parameter ranges from 1 to 31. By modifying this value, we can control the bit rate requirements. For example, a global quantization value of 1 requires higher bit rates, changing it to 31 gives less accurate quantized values, letting the next step, arithmetic coding, to compress for lower bit rates. We measure the precision requirement for *Quantum*(*i*)

3. Controlling of MPEG-4 Avatars

The participant should animate their virtual human representation in real time; however, the human control is not straightforward: the complexity of virtual human representation needs a large number of degrees of freedom to be tracked. In addition, interaction with the environment increases this difficulty even more. Therefore, the human control should use higher-level mechanisms to be able to animate the representation with maximal facility and minimal input. We divide the virtual human control methods into three classes (Thalmann et al. 1995):

- *Directly controlled virtual humans*: the state vector of the virtual human is modified directly (e.g. using sensors attached to the body) by providing the new DOF values directly (e.g. by sensors attached to the body).
- *User-guided virtual humans*: the external driver *guides* the virtual human by defining tasks to perform, and the virtual human uses its motor skills to perform this action by coordinated joint movements (e.g. walk, sit).
- *Autonomous virtual humans*: the virtual human is assumed to have an internal state which is built by its goals and sensor information from the environment, and the participant modifies this state by defining high-level motivations, and state changes (e.g. turning on vision behaviour).

The control methods are not independent, higher-level controls require lower-level capabilities. Autonomous behaviour assumes motor skills to accomplish control, and motor skills modify individual DOFs.

We have developed the following motion control techniques for MPEG-4 avatars:

- *Directly control of avatar* using Flock of Birds magnetic trackers from Ascension Technologies.
- *User-guided control of avatar* for walking, picking, breathing, bending.
- *Autonomous virtual human* example: chatting with Eliza-based autonomous actor.

4. Encoding of MPEG-4 Avatars

We are developing an efficient method for encoding BAPs that optimize the bitrate vs. resulting quality, letting the user choose the required bandwidth. The required bandwidth should be used together with the current animation to find the right grouping and quantization values. We will present the detailed algorithm in the final version.

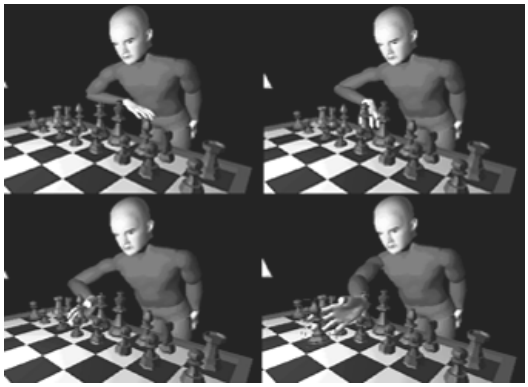

Figure 2: User-guided avatar sequence

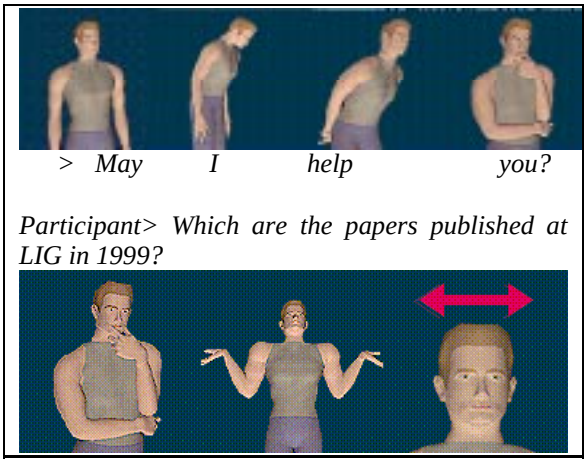

Figure 3: Eliza-based helper application

5. Experimental Results

We present the results below. In the final version of the paper we will present more detailed results and a videotape that shows the resulting animations.

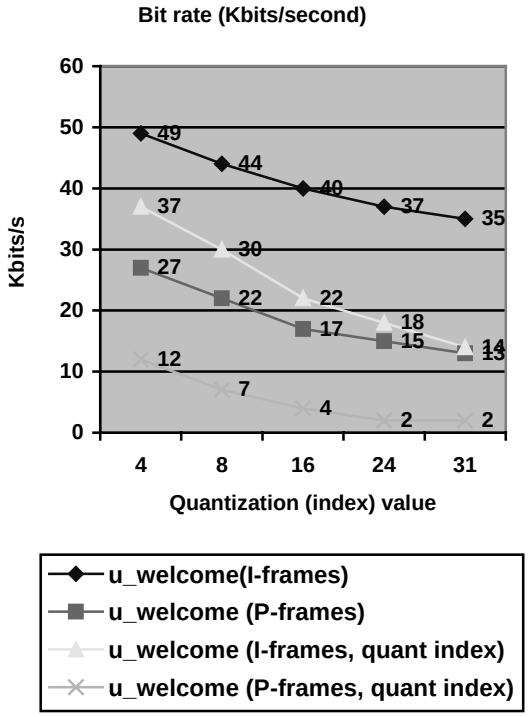

Figure 4: Comparison of bit rate requirements of the proposed quant index technique compared to linear quantization values. (Note that these bit rates contain all the BAPs, No BAP masking was used for these figures. Using BAP masking decreases bitrates further).

6. Conclusion

In this paper, we present the body animation coding in MPEG-4, and MPEG-4 compliant avatar control. We believe that research on efficient encoding algorithms of face and body models is only starting, and we present our current approach.

References:

Capin T.K., Pandzic I.S., Thalmann D., Magnenat Thalmann N. *Avatars in Networked Virtual Environments*, John Wiley, June 1999.

ISO/IEC 14496-1: MPEG-4 PDAM1, available on MPEG official web site: www.cselt.stet.it/mpeg

Thalmann D., Çapin T.K., Magnenat Thalmann N., Pandzic I.S. (1995) Participant, User-Guided and Autonomous Actors in the Virtual Life Network VLNET, *Proc. ICAT/VRST '95*, Chiba, Japan, pp. 3–11.