

The Analysis of MPEG-4 Core Profile and its system design

Author: Liang Cheng(lcheng@uci.edu), Magda El Zarki(*IEEE member*)
Department of Information and Computer Science, University of California

Abstract--- The recently proposed MPEG-4 standard is aimed at supporting content-based coding of audio, text, image, synthetic/natural video data, multiplexing of the coded data, as well as the composition and representation of audio-visual scenes. However, it is unnecessary and nearly impossible to implement all these functions in one application. This paper will be discussing the major coding tools of a scalable core profile MPEG-4 visual codec and will outline the design of a practical but reliable MPEG-4 client-server mechanism with the goal of reducing the unnecessary complexity as much as possible.

1. Introduction

These days, there is a tremendous interest in providing more advanced real-time multimedia services. However, the huge amount of data involved with video streaming hinders the wide acceptability and adoption of multimedia applications over the Internet. Traditionally, pure data compression is a good solution for reducing bit rates. However, in the case of video, we found that we encountered a new dilemma: the compression efficiency is close to the maximum limit defined by Shannon's information theory. So there is a need to find some other kind of solution to the problem. It is well known that the human visual system filters and processes information in many varied ways. For example, humans are much more sensitive to the low-frequency signals than the high-frequency signals, this feature has been used by the MPEG-1/2 and H.263 video standards. Humans also pay more attention to foreground objects than they do to background scenes. It is therefore natural to conclude that encoding video based on its content, usually referred to as "content-based" coding or "object-based" coding, is a good direction to take in attempting to reduce the bit rate whilst maintaining visual quality.

MPEG-4 claims to be an object-based encoding system. However, it is also well known for its complexity in implementation. This paper will discuss the encoding tools of a scalable core profile MPEG-4 visual codec by comparing it with the simple profile version that is based on H.263, and then outline the design of an MPEG-4 compliant server-client model based on an IP network. Our work can be applied to a lot of IP based broadcast applications [3], such as long-distance learning; Internet TV broadcast system, etc.

2. MPEG-4 Core profile Codec Engine

MPEG-4's claim for being "object-based" is embodied mainly in its visual part. Although MPEG-4 provides several visual object types, the scalable core profile fits our design requirements most, because it has obvious advantages over the simple profile as it supports arbitrary-shaped coding, temporal/spatial scalability, etc. At the same time, a scalable core profile encoder has an acceptable design complexity as it excludes the features for DTV broadcasting and computer graphics, such as "interlace," "2D mesh," etc. However, there remain several unanswered questions: for situations for which shape information cannot be easily obtained, is MPEG-4 still more desirable than H.263? Will the shape information, that needs to be included, lower the coding efficiency of MPEG-4 thereby wiping out the gains in compression that can be achieved?

2.1 Comparison of frame-based MPEG-4 visual codec and H.263

The first MPEG-4 natural video texture encoders were based on the H.263 baseline encoder without any supplemental new features. As a result, people now define a "short Header" option in MPEG-4 to allow it to be backward compatible with the H.263 baseline encoder. Because of this history, our performance comparisons will be made against an H.263 encoder.

Over the past few years, more and more video coding algorithms have been adopted by MPEG-4, such as intra DC/AC prediction, MPEG quantization table, nonlinear DC quantization and more advanced error resilience, etc. Even without the content-based features, MPEG-4 is supposed to be more efficient than the H.263 baseline encoder. Although H.263+ defines some new features to make H.263 more efficient, it still cannot compete with MPEG-4 which incorporates such features as quantization method 1 and the MPEG intra/inter quantization weight matrix, etc. However, a performance study that we conducted, showed that rectangular-shaped MPEG-4 visual coding demonstrates its efficiency mostly in some very specific situations.

Figure 1 and figure 2 show the results of both the rectangular-shaped MPEG-4 encoder and the H.263 encoder for different values of the quantization parameter Q . For the convenience of comparison, we turned off the rate-control option of the encoders. Figure 1 shows that the MPEG-4 encoder has a higher compression rate under the same quantization parameter for the typical "Head and Shoulder" scene, "Akiyo". However, MPEG-4 does not always display higher compression efficiency as shown in Figure 2. Here we used a different video sequence, "Stefan". A close analysis revealed that it is mainly because of the intra AC/DC prediction, which is used in MPEG-4 but not in H.263. AC/DC prediction in the intra texture coding is very efficient and contributes the most to the improved results in figure 1, though this feature does make the video more error-prone (see [4]). In addition, MPEG-4 uses a non-linear DC quantization, which enhances its coding efficiency while maintaining the subjective quality as close to that of H.263 as possible.

Therefore, the efficiency of rectangular-shaped MPEG-4 coding is dependent on the content of the video. For more complex video sequences with intensive motion and camera panning, such as found in "Stefan", MPEG-4 failed to improve the compression efficiency. This is primarily due to the fact that inter coding plays more an important role than intra coding in such a sequence, the AC/DC prediction is unable to decrease the bit rate as much as it did in "Akiyo". Therefore, the compression rate of "Stefan" will be close to that of H.263 as demonstrated in Figure 2.

2.2 Analysis of arbitrary-shaped coding

When comes to "content-based", the most significant feature of MPEG-4 is that it allows the separate encoding of foreground figures and background scene. According to [2], the scalable core profile uses the "binary shape" tool, which uses a constant non-zero value to represent the front figure while padding all of the remaining pixels with "0".

Compared with rectangular-shaped MPEG-4 visual coding, arbitrary-shaped coding is supposed to keep the quality of what concerns people the most while cutting the bit budget considerably. However, we notice that arbitrary-shaped coding will include shape information in the compressed stream, which is additional overhead not found in the rectangular-shaped codec. Figure 3 shows the comparison, of the average cost in terms of bits, for different components in the rectangular-based VOP (video object plane) and arbitrary-shaped VOP. Obviously, in "Akiyo", the shape information overhead did not impact the coding efficiency considerably. Figure 5 shows the subjective quality of the results. We can observe that, although arbitrary-shaped encoding uses fewer bits (Figure 3), it achieves similar perceptual result as rectangular-based encoding.

For this experiment, we obtained the mask of Akiyo by using the semi-automatic segmentation tool [5] (see Figure 4). Unfortunately, the segmentation result is not guaranteed to be unique, and some bad segmentation results can tradeoff the efficiency obtained when using arbitrary-shaped coding. What is more, extracting the shape information of a long sequence is a nontrivial task and the whole arbitrary-shaped coding process is hindered by this preprocessing drawback. For this purpose, we are evaluating the method that is commonly used in the virtual reality broadcasting studio, "Chroma Key/Blue Screen"[8]. Using this technique, real-time arbitrary coding can be optimized.

2.3 Scalability

In order to give the user more flexibility when choosing content and to adapt to different client side environments and link bandwidths, the temporal and spatial scalability are included in our system. The base layer can be decoded independently, while the enhancement layer can only be decoded together with base layer. The spatial scalability will allow people to add one or more enhancement VOL (video object layers) to the base VOL to achieve different video scenes. For example, when someone is watching a tennis match, which constitutes the base layer of a scene, some other person may want to add an enhancement layer, demonstrated here by the inclusion of a swimming fish [as shown in 6]. In addition, one can use temporal scalability to achieve higher frame rates, which results in smoother playback.

3: System Design

The system part of MPEG-4 allows people to create or watch a multimedia sequence with hybrid elementary streams, which can be encoded and decoded with the best suitable codec for each stream [1]. However, to manipulate those streams synchronously and compose them onto a screen in real time [10] is very complicated. In this paper, we will assume that all of the media streams have been encoded and are stored on a server (see Figure 7). All of the ES (elementary streams) consist of either a multiplexed (using the MPEG4 defined FLEXMUX) stream or a single stream, but all of them have been packetized by the MPEG-4 SL (synchronization layer).

3.1 Message passing and Interactivity between server and client

First, the client will use an HTTP-based CGI submission to ask for a desired media presentation (see Figure 8). The submission will only contain the index of the preformatted BIFS for the case of a pre-created and stored presentation or a text based description of the user's authored presentation. The server receives the request, converts it to the BIFS format, stores it and sends to the client the BIFS stream together with the object descriptor in the form of an *initial object descriptor* stream to initialize the client's decoder. In addition, the initial object descriptor should also contain the evaluation result of the decoding complexity and buffer capacities, etc, because the multistream decoding/composition is usually very computation intensive.

If the client side can satisfy the decoding requirements, it will send the server a confirmation to start the presentation; otherwise, the client will send a negative response and the server will repeatedly down grade to a lower-profile until it meets the decoding capabilities of the client side. After this exchange of messages, the server will initiate the establishment of the necessary sessions for the SD (scene description) stream (BIFS format) and the OD (object description) stream referenced by the `ctl_ES_IDs` obtained from the initial object description stream. The client can now retrieve the compressed media stream by using the `media_ES_ID` or URL contained in the ES descriptor stream in real time. The BIFS is used to lay out the media elementary stream in the presentation, because it provides the spatial and temporal relationship of those objects by referencing their `ES_IDs`. When client users want to modify the received scene, such as adding an enhancement layer to the current

scene, they can send a BIFS update command to the server and obtain a reference to the new media elementary stream. Thus, interactivity can be achieved in our IP-based MPEG-4 system.

3.2 Decoding, Composition and Synchronization

At the client site, the decoding process may be much more complex than H.263, MPEG1/2, etc, because MPEG-4 needs to handle multiple streams. The need for synchronization is beyond the traditional “lip-sync” of audio and video. MPEG-4 needs to sync between different objects and between the different elementary streams of a single object (e.g., base layer and enhancement layer).

The Synchronization Layer is responsible for synchronizing the Elementary Streams. This layer receives the data in the form of SL-packets that are encapsulated by lower layer packets such as IP packet. Each SL-packet consists of an AU (access unit) or a fragment of an AU. AU needs time stamps for synchronization and it constitutes the data unit that is actually consumed by the decoder. In our system, an AU will be consisting of a VOP. Before an AU enters its decoder, it has to be placed into a decoding buffer, whose size is specified by the sending terminal and conveyed within object descriptors during the setup stage of the MPEG-4 session. After that, each AU comes to the decoder at the time instance specified by DTS (Decoding Time Stamp.) The decoder then transforms the uncompressed data into composition units and puts them into composition memory. The composition units include the CTS (Composition Time Stamp), which specifies the time instance for composition units to enter the compositor. The compositor is actually responsible for that deciding on either to consume or skip the composition units by using the scene description information.

To relate the elementary streams to media objects within a scene, our system uses an object descriptor framework, which allows identifying, describing and properly associating elementary streams to each other. Each object descriptor is a collection of descriptors that describe one object, which is usually associated with more than one elementary stream to allow scalable content representation or multiple alternative streams that convey the same content, such as multiple scalability styles. The object descriptors are obtained from their specified OD decoder. After content composition, we write our real-time presentation directly to the video buffer memory under the support of Microsoft’s DirectX.

Conclusion:

We have presented a practical MPEG-4 compliant system. The video coding engine gives the end user multiple choices to tradeoff complexity and compression rate, such as the choice between rectangular-shape and arbitrary-shape, the choice among different scalability modes, etc. In addition, our system supports interactivity and realtime scene update over an IP network.

Reference:

1. ISO/IEC 14496-1: 1999(E), Information Technology Coding of audio-visual objects-Part 1: Systems
2. ISO/IEC 14496-2: 1999(E), Information Technology Coding of audio-visual objects-Part 2: Visual
3. ISO/IEC JTC1/SC29/WG11 MPEG 99/N2724 March 1999/Seoul *MPEG-4 Applications*
4. Ximin Zhang, Anthony Vetro, H. Sun, "Robust Decoding for reduced error propagation of DC/AC prediction errors", *Proceedings of workshop and exhibition on MPEG-4*, June 18th, 2001
5. Daras Petros, R. Theodoros, K. Ioannis, Prof. Strintzis Michael G, *MPEG-4 ToolBox 2.0*
6. Simon A. J. Winder, ISO/IEC 14496 (MPEG4) Video Reference Software,
7. H.263 encoder software (TMN8) <http://www.ee.ubc.ca/image>
8. <http://www.seanet.com/Users/bradford/bluscrn.html>
9. Alexandros Eleftheriadis, MPEG4 Systems Systems, *Proceedings of SPIE VI. 4209(2001)*
10. Yong He, Ishfaq ahmad, Ming L. Liou, “Real-Time Interactive MPEG-4 System Encoder Using a Cluster of Workstations”, *IEEE Transactions on Multimedia VOL. 1, No. 2, June 1999*

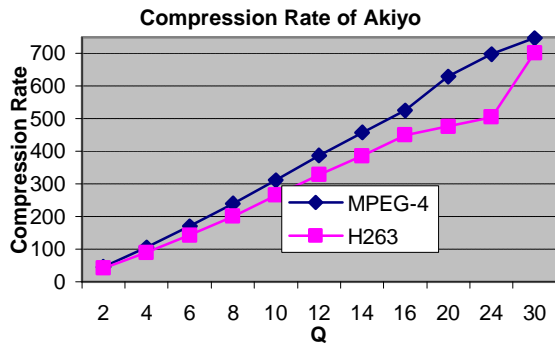


Figure 1 Akiyo, MPEG-4 vs. H263

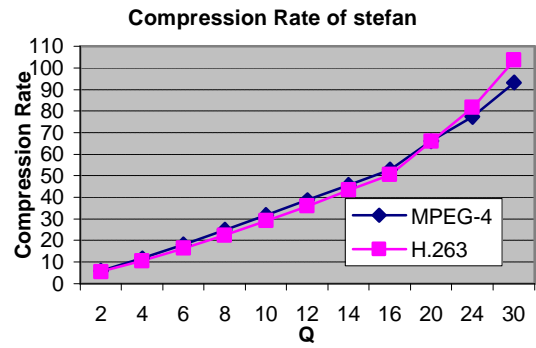


Figure 2 "Stefan", MPEG-4 vs H.263

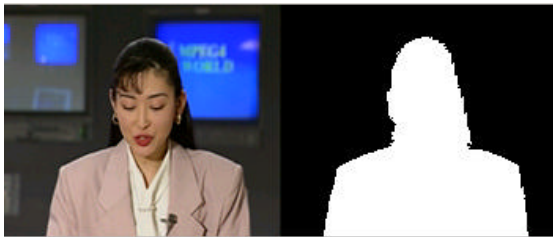


Figure 4 Original Akiyo and its mask (Frame #40 Format: qcif)

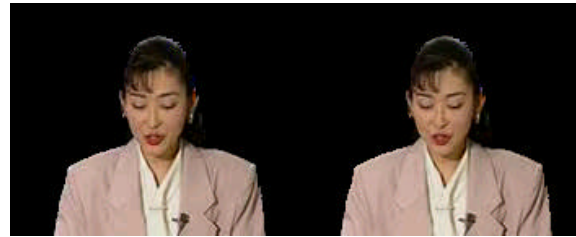


Figure 5 Arbitrary-shape coding (Left) vs. Rectangular-based coding (Right) (Q = 6). The bit budget is shown in Figure 3.

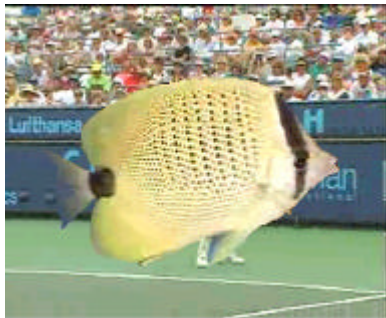


Figure 6 Frame #10 VOP of new combined scene (base + enhancement)

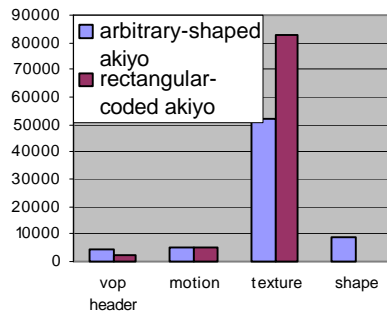


Figure 3 Comparison of Arbitrary-shape coding and Rectangular-based coding (Q = 6). Total no. of frames: 40, format: cif

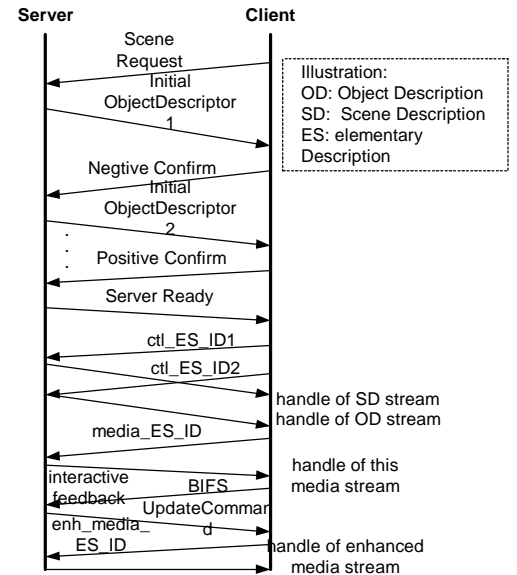


Figure 8 Message passing of server and client

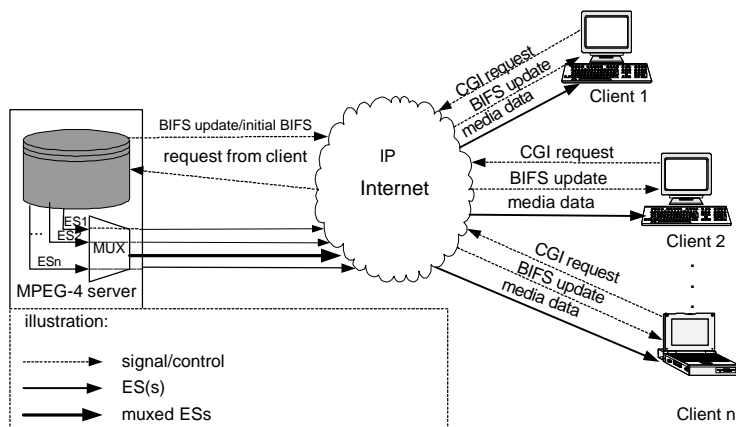


Figure 7 MPEG-4 system design