A JAVA-BASED MPEG-4 LIKE VIDEO CODEC

Dung-Yung Liu, Chien-Wu Tsai and Ja-Ling Wu
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

Abtract—**The MPEG-4 specification was developed in response to the growing need for a coding method that can facilitate dynamic access to audiovisual objects for various applications such as digital storage media, internet, various forms of wired or wireless communication etc.. Although the specification is still in the working draft stage, the concepts and technologies of the specification are very encouraging and will inspire a lot of creative works in the multimedia applications. This paper is devoted to the rsoftware implementation of the video part of MPEG-4. We develop a prototype of MPEG4-like video encoder and a prototype of Java-based video decoder, which can be executed on Internet, in the client-server model by using Java Applet. The prototype can show us the strong functionality of MPEG-4 that includes user interactivity and object scalability, etc.**

I. INTRODUCTION

The goal of MPEG-4[4][6] Video is to provide standardized kernel technologies allowing efficient storage, transmission and manipulation of video data in multimedia environments. The MPEG-4 Video Verification Model[5] describes a fully defined core video coding algorithm and platform including encoder, decoder as well as bitstream syntax and semantics for the development of the standard. The MPEG-4 Video coding algorithm will eventually support all functionalities already provided by MPEG-1[1] and MPEG-2[2]. The functionalities common to cluster of applications considered by MPEG-4 will include efficient compression, object scalability, spatial and temporal scalability, and error resilience[5][8].

To this end, the main idea of MPEG-4 in video compression relies on content-based visual data representation. The concept of the so-called content-based visual data representation supports the separation of encoding and decoding for physical objects in a scene. And the ability to identify and selectively decode and reconstruct Video Objects of interest is referred to as content-based scalability or object scalability in MPEG-4. These MPEG-4 features provide the most elementary mechanism for interactivity and content manipulation of Video Objects at the receiver.

To support the content based interactivity, the MPEG-4 Video Verification Model introduces the concept of Video Object Planes (VOP's). It is assumed that each frame of an input video sequence is segmented into a number of arbitrarily shaped image regions that describe physical objects within scenes the so called Video Object planes. Compared to the video source format used for MPEG-1 and MPEG-2 standards, the video input for MPEG-4 Verification Model can be arbitrary shape, rectangle block convention is just a special case of it.

Due to the rapid advance of the CPU technology, the computing power of personal computer is faster than ever. Nowadays many computation-intensive algorithms used in video coding (e.g. MPEG-1, MPEG-2, H.261, and H.263[3]) can be implemented by software only[10][11]. It follows that hardware devices used in applications of video coding will be replaced with pure software programs in the near future. That is the reason why we try to implement this MPEG-4 like software codec. Furthermore, MPEG-4 also provides the functionality for downloading the tools required in the decoding stage from the remote site. To reach the goal, the Java-based software video decoder is presented. The capability of machine independence, user interaction, decoding downloadability, object scalability of the MPEG-4 coding standard can be demonstrated by this decoder.

This paper is organized as follows. Sections II and III present the details and the experimental results of encoder and decoder, respectively. And Section IV concludes this work.

II. MPEG-4 LIKE VIDEO ENCODER

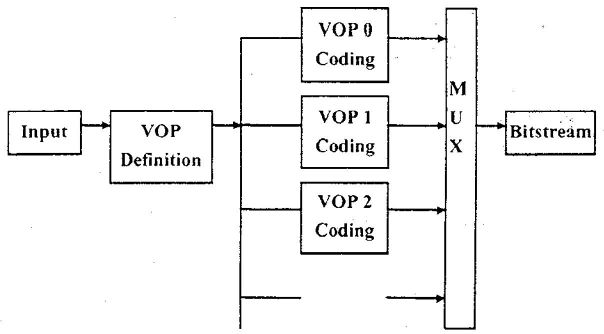


Figure 1: VM encoder structure

There are two main components in the MPEG-4 video encoder: the VOP encoder and the multiplexer as shown in Fig.1. The multiplexer is to combine the output bitstreams

Contributed Paper
Manuscript received January 12, 1998

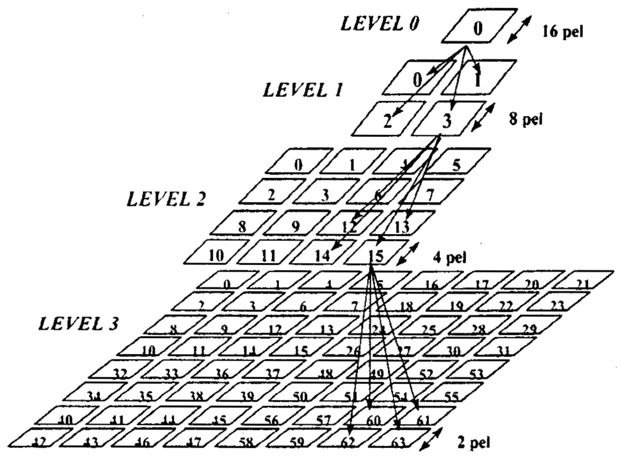


Figure 2: The quadtree structure

from VOP encoders into a single bitstream; therefore, the kernel is on the VOP encoder.

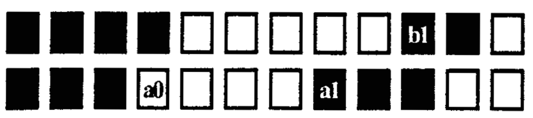
The input to be coded can be a VOP image region of arbitrary shape and the shape and location of the region can vary from frame to frame. The Video Object (VO) correspond to successive VOP's belonging to the same physical object in a scene. The shape, motion and texture information of the VOP's belonging to the same VO is encoded and transmitted or coded into a separate VOL (Video Object Layer). In addition, the composition information is also included in the bitstream. This allows the separate decoding of each VOP and the required flexibility in manipulation of the video sequence. In MPEG-4 video test sequence, the video source input assumed for the VOL structure is either known by construction of the sequences (i.e. is generated by blue screen technology) or is generated by means of semi-automatic segmentation.

Notice that, if the original input image sequences are not decomposed into several VOLs of arbitrary shape, the coding structure simply degenerates into a single layer representation which supports conventional image sequences of rectangular shape.

In our implementation, the input for VOP encoder is the bitstream consisting of shape, position, and texture information of Video Object. Moreover, the VOP encoder will encode the Video Object into one base layer and three enhancement layers to support scalability. The VOP encoder is mainly composed of two parts: the shape coding and the motion and texture coding. We will describe some feasible methods for each coding in the following sections.

A. Shape Coding

In MPEG-4, a grey scale alpha plane represents the shape and transparency of a Video Object. For simplifying our codec, we just take the binary alpha plane as the shape information without transparency information. We realize two methods to code the binary shape. One is Quadtree, and the other is the MMREAD (Modified Modified



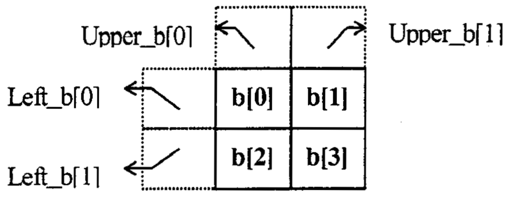
where: a0 is the current point in the coding line to be processed.
a1 is the first point that makes a change from a0.
b1 is the first point that makes a change from next position above a0 in the reference line.

Figure 3: The relationships between the reference points

Relative Element Address Detection)[14].

A1. Quadtree Binary Shape Coding

The method adopted is based on binary alpha blocks (16x16 samples) in the binary alpha plane. We can build a quadtree like Fig.2. And we calculate all the index values in level 3 according to (1), then do swapping by the rules R1. The same processes are repeated for all levels up to level 0. Finally, from level 0, we assign each value a variable length code.



where b[i]=2 if sample value=255
b[i]=0 if sample value=0

$$\text{Index}=27*b[0]+9*b[1]+3*b[2]+b[3] \qquad (1)$$

Rule R1:

(1) If upper_b[0]<upper_b[1], then swapping b[0] and b[1], and b[2] and b[3].
(2) If left_b[0]<left_b[1], then swapping b[0] and b[1], and b[2] and b[3].
(3) If upper_b[0]+upper_b[1]<left_b[0] + left_b[1], then swapping b[1] and b[2].

Since this method do not consider the correlation between adjacent lines in space and consecutive frames in time, the compression ratio is low.

A2. MMREAD Binary Shape Coding

MMREAD is derived from the G3 2-D modes of fax standard. We simplify the definition of the standard to move up the coding efficiency. Fig.3 shows the relationships between the reference points used in the coding. The three major reference points are shown in Fig.3. And the modified algorithm is as follow.

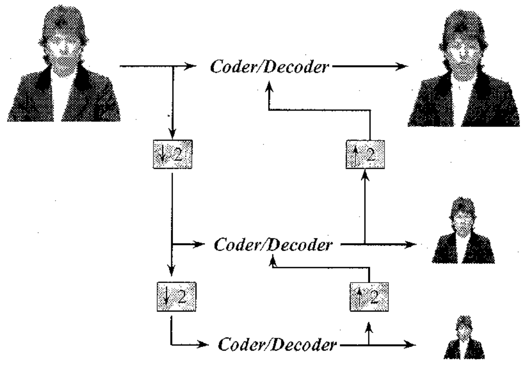


Figure 4: Spatial Scalability

```
Algorithm MMREAD:
1. Detect a1
2. Detect b1
3. if(b1 detected)
        check distance between a1 and b1 below
        if(distance <= 5)
            vertical_mode
        else
            check a1 and a0
            if(distance < WIDTH)
                horiznotal_mode(ULB=a1-a0)
            else
                vertical_pass_mode(b1 detected)
4. else b1 not detected
    check a1 and a0
    if(distance < WIDTH)    horiznotal_mode(RLB=a1-a0)
    else
        vertical_pass_mode(b1 no detected)
5. a0 = a1 , goto 1 until a1 not detected
```

This algorithm has three coding modes: vertical mode, horizontal mode, and vertical pass mode.

(1) Vertical mode:
It is the best situation, and only the distance between a0 and b1 is coded.

(2) Horizontal mode:
First it sends the makeup code of the horizontal mode, then codes the ULB (unchanged length) if b1 is detected, or the RLB (residual length) if b1 is not detected.

(3) Vertical pass mode:
It is the worst situation. First, it sends the code according to the horizontal mode. It depends on if b1 is detected or not. If b1 is detected, then ULB is equal to Width, the width of scanline, and RLB is equal to a1-a0-Width. If b1 is not detected, then RLB is equal to a1-a0. Then, the codes of vertical mode will be send next. The number of the codes for vertical mode is dependent on how many blocks are covered by the run with length RLB. Finally, a code for horizontal mode with the remaining residual length is sent.

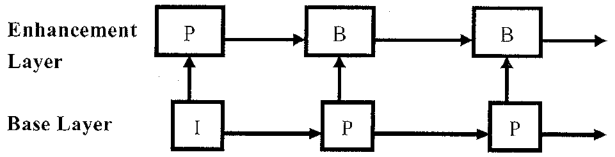


Figure 5: Enhancement structure with B-VOP's for spatial scalability

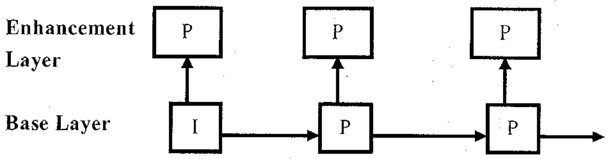


Figure 6: Enhancement structure with P-VOP's for spatial scalability

Since this method do consider the correlation in space and time, the compression ratio is increased. But the performance is descended due to this algorithm accompanying many redundant scans.

B. Motion And Texture Encoding

The coding methods used in this part are almost adopted from H.263 directly. But in order to process the arbitrary shaped objects, some methods need to be changed or modified[3][9][10]. The traditional DCT can be replaced by padding DCT or shape adaptive DCT which are proposed by MPEG-4 VM. Since the shape adaptive DCT is in testing stage and not defined completely, we choose the padding DCT for our implementation. Also, the block matching algorithm is replaced by the polygon matching algorithm. All the details can be obtained from the MPEG-4 VM document, and it will not be mentioned here.

C. Scalability Encoding

Scalability of Video Objects means the ability to achieve video of more than one resolution and/or quality simultaneously. There are two main types of scalability: the spatial scalability, and the temporal scalability. In the following sections, we will describe how to support these scalabilities.

C1. Spatial scalability

The spatial scalability means that the resolution of a Video Object can be changed by the interaction of users. In Fig.4, we show the processing diagram. The base layer has the lowest resolution, and the enhancement layers having higher resolution are the layers above the base layer. According to this way of division, the bitstream of VOP's can be arranged as in Fig.5 and Fig.6.

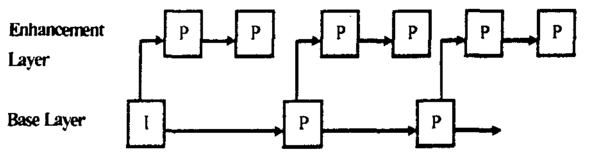


Figure 7: Enhancement structure with P-VOP's for temporal scalability

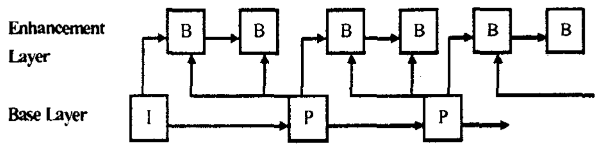


Figure 8: Enhancement structure with B-VOP's for temporal scalability

We choose the second one to place P-VOP in the enhancement layer in our codec. Since it is easy and convenient for the decoder and encoder to manipulate them, and the decoder can decode only the enhancement layers of interest, instead of all the enhancement layers, to obtain the demanded quality. The drawback of this selection is that the compression ratio is worse than the first one.

C2. Temporal scalability

The temporal scalability means that the frame rate of a Video Object can be negotiated with user's request. There are two types of enhancement structures as shown in Fig.7 and Fig.8. We decide to place only one P-VOP in the enhancement layer for the same reason mentioned in the spatial scalability.

D. Performance Evaluation

We will show the compression ratios and the frame rates of the encoder by adopting the two methods, i.e. Quadtree and MMREAD, used in binary shape coding. Because we do not have any test sequences used in MPEG-4 VM, we prepare it by ourselves. Just for demonstration purpose, we provide only two Video Objects. One is a background Video Object and the other one is a foreground Video Object. The binary alpha planes of these two Video Obects are obtained by manually pointing out, frame by frame. The background is the CLARIE sequence with 113 frames, and each frame size is 128x128. The foreground is the GARDEN sequence with 113 frames, and each frame size is 176x144. And the testing is executed on Pentium-200 MMX machine.

Fig.9 shows the result obtained by encoding the first 27 frames of the testing Video Objects. We can see that the MMREAD shape coding algorithm has compression ratio about two times of the Quadtree algorithm. Table 1 shows the performance of encoder by adopting the two methods respectively.

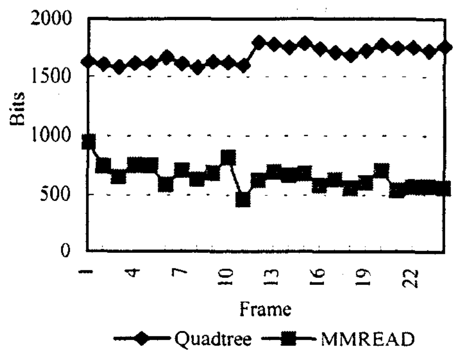


Figure 9: The comparison of the compression ration obtained by Quadtree and MMREAD

	Quadtree	MMREAD
Bitstream Size(bytes)	598,489	316,297
Frame rate	3.0	6.6

Table 1: The performance of encoder by using Quadtree and MMREAD respectively.

	2 VO(1 Layer)	2 VO(4 Layer)
Bitstream size (bytes)	321,296	800,270
User interactivity	Change position and size of Video Objects.	Change position, size, and spatial and temporal scalability of Video Objects.
Frame rate	6.6	2.4
Decoding Frame rate	0.41	0.4

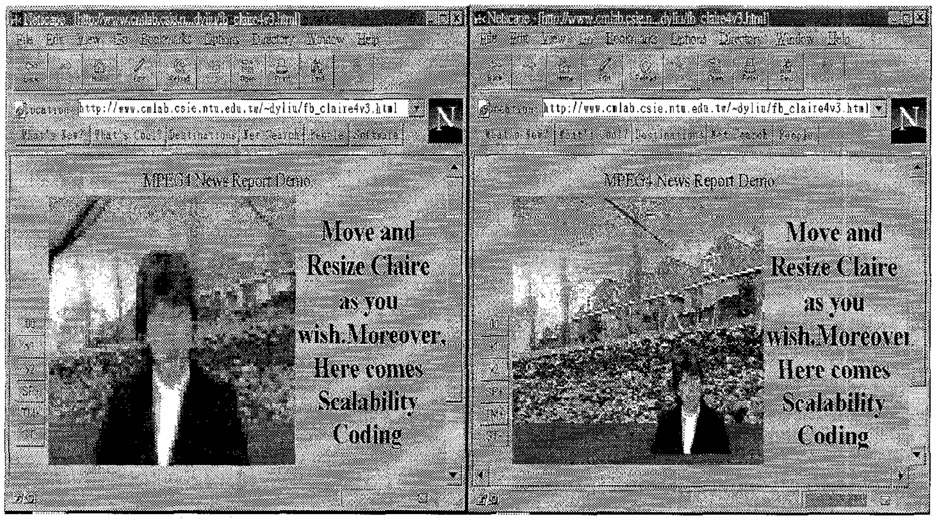
Table 2: The performance variation of encoder when including the enhancement layers.

In order to see the influence of enhancement capability in performance of encoder, we obtain the result in Table 2 by considering Video Objects with different number of Video Object Layers.

III. MPEG-4 LIKE VIDEO DECODER

A. The Structure Of Video Decoder

Since MPEG-4 video coding is object-based coding, a scene is composed of Video Objects. The structure of MPEG-4 video decoder can be divided into three components including demultiplexing part, Video Object decoder, and compositor as shown in Fig.10. At first, we



http://www.cmlab.csie.ntu.edu.tw/~dyliu/fb_claire4v3.html

MPEG4 News Report Demo

Move and Resize Claire as you wish.Moreover, Here comes Scalability Coding

(a) (b)

The buttons at the left side are for controlling the size and scalability of video objects.
Where 00: restore the original object, X1 reset the original size of object,
X2: double the size of object, SP+: spatial enhancement
TM+: temporal enhancement, ST-: SP or TM degradation.

Figure 10: The execution snapshot of Java applet decoder in the Netscape. (a) Only the base layers of the background and foreground objects are decoded. (b) After enhancing the background object, and enhancing and resizing the foreground object.

use a demultiplexer to distribute each Video Object to its corresponding decoder, then decoding the object. And finally, the compositor combines the decoded objects into a scene and displays the scene. That is why we decide to adopt multithreading in the decoder. The details of each part will be explained in the following subsections.

B. The Demultiplexer

The demultiplexer sends separately the bitstreams of Video Objects to the individual decoder for decoding. So it is the responsibility of demultiplexer for generating the new Video Object decoder processes and extracting the bitstreams of Video Objects from the original single input bitstream. Using this method, each Video Object can be manipulated separately which simplifies the complexity of the decoder.

C. The Video Object Plane Decoder

For the scalability decoding, the demanded resolution and frame rate depends on the interaction of user. By default, the VOP decoder decodes only the bistream belonging to the base layer, and skips the enhancement layer. When the user requesting higher resolution and/or higher frame rate, then the VOP decoder will further decode the enhancement layer. In this way, both the decoding efficiency and user's interactivity of the VOP decoder can be taken into account.

D. The Compositor

The outputs of the decoders are the reconstructed VOP's that are passed to the compositor. The compositor then put these reconstructed VOP's in a scene according to their locations and orders. Since each VOP decoder is responsible for a Video Object, the degree of decoding process may be different for all VOP decoders and this will result in the asynchronous problem. To solve this problem, in our implementation, the compositor will stop the VOP decoders who have already done the job to continually decode new Video Objects until all other decoders have done their jobs.

Also, the compositor can receive the control messages from user. The control messages are about the location, the size, and scalability of Video Objects. The compositor will send the control message to the corresponding decoder to reflect the demand of user's interaction.

E. Java Applet Decoder

We adopt Java as our implementing language for its abilities to cross the boundaries of platforms and to download decoding tools from the remote sites. You can use any Java-supported WWW browser to execute this decoder and see the apperance shown in Fig.10.

In Table 3, we demonstrate the decoding efficiency of this Java decoder. The testing Video Objects include one background, one foreground. The background is the CLARIE sequence with frame size 128x128, and the foreground is the GARDEN sequence with frame size

Enhancements	Pentium-200 Frames/sec	Sun Ultra Sparc Frames/sec
No	6.6	7.2
Background	3.6	4.2
Foreground	3.0	3.0
Background & Foreground	2.4	2.4

Table 3: The performance of Java applet decoder

176x144. The scalabilities used in the testing consist of temporal and spatial enhancements.

IV. CONCLUSION

The goal of MPEG-4 is to provide an standard for the applications of multimedia, e.g. VRML, MIDI, and the synthesis of natural images and computer graphics[13]. The applications are not limited to some specific areas, but all the multimedia applications can apply it[7]. We can foresee that it will be a very important standard in the future.

In this paper, we practically develop a prototype of MPEG4-like video encoder and a prototype of Java-based video decoder, which can be executed on the Internet. The prototype demonstrates the feasibility of MPEG-4 and some functionalities that have been strongly recommended in MPEG-4.

REFERENCES

[1] ISO/IEC JTC1 CD 11172, "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/s," December 1991

[2] ISO/IEC Draft International Standard 13818, "Information Technology-Generic Coding of Moving Pictures and Associated Audio," March 1994

[3] Draft ITU-T Recommendation H.263, "Video CODEC for low bitrate communication," October 1995

[4] ISO/IEC JTC1/SC29/WG11, "MPEG-4 Video Verification Model Version 2.1," May 1996

[5] ISO/IEC JTC1/SC29/WG11, "MPEG-4 Video Verification Model Version 7.0," April 1997

[6] ISO/IEC JTC1/SC29/WG11, "Working Draft3.0 of ISO/IEC 14496-2," April 1997

[7] Leonardo Chiariglione, "MPEG and Multimedia Communications," IEEE Trans. Circuits and System for Video Tech. Vol. 7, No.1,February 1997

[8] Thomas Sikora, "The MPEG-4 Video Standard Verification Model," IEEE Trans. Circuits and System for Video Tech. Vol. 7, No.1,February 1997

[9] Junavit Chalidabhongse and C.C. Jay Kuo, "Fast Motion Vector Estimation Using Multiresolution-Spatio-Temporal Correlations," IEEE Trans. Circuits and System for Video Tech.,April 1996

[10] Den-Yuan Hsiau, "The Realtime Implementaion of H.261 Video CODEC," Master Thesis, National Taiwan University, June 1996

[11] 翁詠泉, "Implementaion of Software MPEG-2 Video CODEC and Its Related Research," Master Thesis, National Taiwan University, June 1996

[12] Fenando Pereira, Rob Koenen, "Very low bit-rate audio-visual applications," Signal Processing: Image Communication, September 1994

[13] Caspar Horne (editor), "SNHC Verification Model 3.0," JTC1/SC29/WG11 N1545, February 1997

[14] International Telegraph and Telephone Consulative Committee. Facsimile coding schemes and coding control fuctions for Group 4 facsimile apparatus, Recommendation T.6. ISO, 1984