MPEG-4 Very Low Bit-rate Video Compression by Adaptively Utilizing Sprite to Short Sequences

Kumi Jinzenji*, Shigeki Okada*, Naoki Kobayashi*,and Hiroshi Watanabe+
*NTT Cyber Space Laboratories, NTT Corporation
+Global Institute of Telecommunication and Information, Waseda University

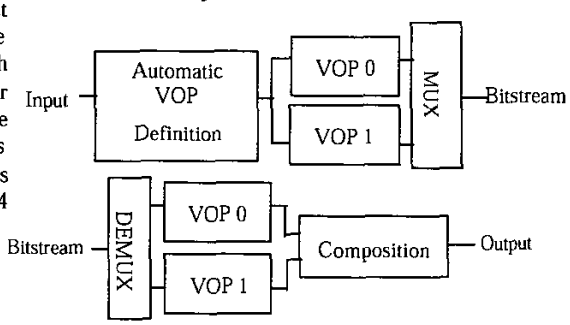
Abstract

In MPEG-4, a video sequence can be divided into foreground object and background objects that are independently encoded. Using sprites can dramatically compress the overall bit rate but not all video sequences can be so encoded. This paper introduces MPEG-4 multi-mode coding; it offers automatic coding mode decision, video object generation and high compression efficiency. The source video sequence is segmented and each segment is automatically categorized as either "normal", which is encoded using MPEG-4 simple profile, or as "sprite". Coding experiments shows that if the bit rate is low, multi-mode coding offers more high coding efficiency than regular MPEG-4 in terms of frame rate and image quality.

1. Introduction

We focused on sprite coding in MPEG-4 [1] and proposed the two-layer video object model and "Sprite Mode"[2]. Figure 1 shows our concept of sprite mode. In sprite mode, the video data is automatically divided into two object types: foreground and background. The foreground object contains all moving regions without camera motion. Each background object is effectively a still image, a sprite, so coding efficiency can be dramatically improved. The foreground object is compressed using MPEG-4 object coding, while background object is compressed using sprite coding. For some video sequences, sprite coding is not suitable. Examples include images with no camera motion or large foreground objects. To enhance overall coding efficiency and lower manual operation costs, we introduce a new MPEG-4 encoding algorithm that offers automatic sprite generation and application. We call it multi-mode coding. First, the source video sequence is split into short segments. Multi-mode coding examines each segment and processes it using the conventional coding scheme, "normal mode", or using sprite coding, "sprite mode". A simulation shows the effectiveness of multi-mode coding in low bit-rate compression from the viewpoint of image quality and frame rate.

Section 2 introduces the algorithm of multi-mode coding. Sections 3 and 4 explain details of the video processing and coding routines, respectively. Some coding experiments are discussed in Section 5, and our study is then concluded.



VOP1: Foreground moving object
VOP0: Background object

Fig.1 Concept of "sprite mode" coding [2]

2. The algorithm of "multi-mode" coding

2.1 Automatic coding mode decision and video data processing

Figure 2 shows the concept of multi-mode coding. Figure 3 indicates the multi-mode coding flow chart. The source video data is first cut into short segments of duration T seconds. If the segment contains a cut point, it is automatically tagged as "N" (indicating that normal mode coding is to be applied) and no further assessment is performed on this segment.

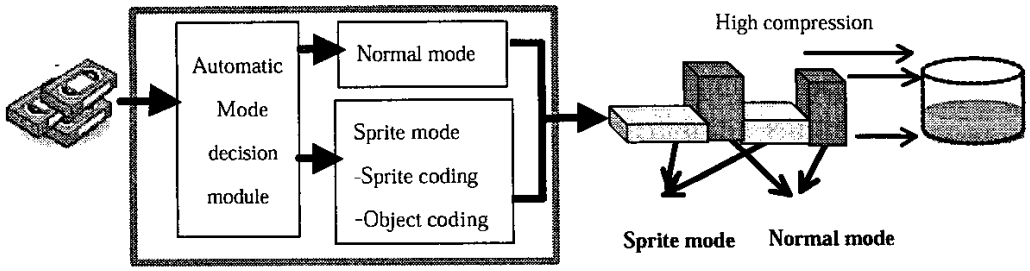


Fig.2 The concept of multi-mode coding

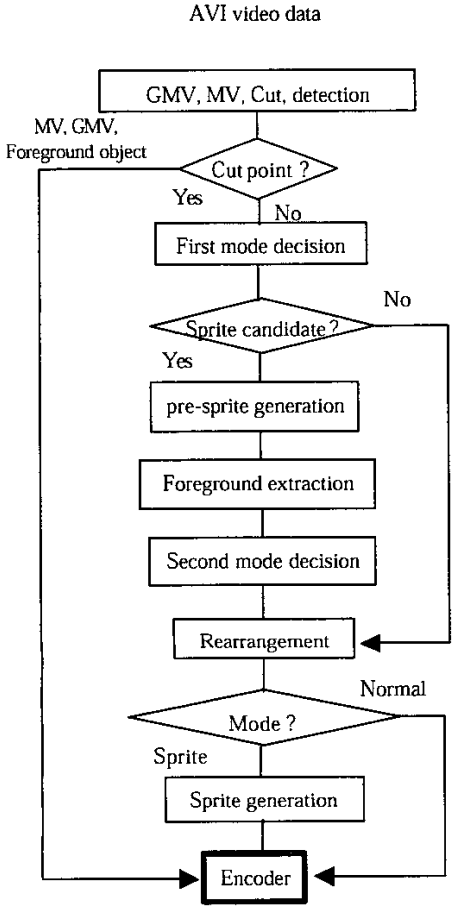


Fig. 3 The algorithm of -coding

2.2 The first coding mode decision by GMV

Global motion vectors (GMV) are estimated for each of the segments remaining after cut point detection using the modified method described in [2]. If the size of GMV, particularly panning and tilting parameter, of a segment falls under a threshold, the camera motion in the stack is estimated as "still", and the segment is tagged N. otherwise, it is tagged a candidate of "S" (indicating that sprite mode coding is to be applied).

2.3 The second coding mode decision by foreground ratio

A provisional sprite is generated for each S segment using the method of [2]. Briefly, the provisional sprite is generated by calculating the median of the pixel intensity at the same position, when all frames are aligned in XYT space. Macro-block based foreground objects are then extracted frame by frame. Reference [2] notes that the sprite mode is effective only if the foreground ratio is relatively small, so if the segment has an excessive foreground ratio, it retagged as N.

2.4 The final mode decision

Since isolated S segments are relatively ineffective, all S segments that are bounded by N segments are retagged as N. Finally, continuous runs of S and N segments are combined and coded appropriately.

3. The details of video processing

We have improved some of the video processing tools necessary for coding mode decision and video object generation. We introduce here improved algorithms for global motion estimation (GME) and sprite generation. The basic algorithms are shown in reference [2].

3.1 Improved GME algorithm

The improved GME algorithm offers higher quality sprite generation and foreground object extraction. Generally speaking, in the bottom-up approach, GME accuracy depends on the accuracy of determining local motion vectors. A video sequence with large smooth areas produce a lot of inaccurate local motion vectors (LMV), so the resulting GMV does not always represent real camera motion. This is a critical problem for generating sprites. On the other hand, the top-down approach provides generally very much computational cost.

Accordingly, we combine the bottom-up and top-down approaches as detailed below.

(1) Provisional GMV is calculated using [2].
(2) The difference between LMV' and LMV in each macro-block position is determined. LMV' is the local motion value of GMV in the macro-block position.
(3) If he sum of the difference is more than a certain threshold, GMV is recalculated using the top-down approach. Otherwise, GMV of the next frame is calculated in the bottom-up approach.

3.2 Improved sprite generation algorithm

The background objects, what are left after extracting the foreground objects, are processed to generate a high quality sprite [2]. The sprite image is generally corrupted by the residuals of foreground objects created by imperfect extraction. Therefore, after aligning all frames of the segment in XYT space, a provisional sprite (median sprite) is generated. At each macro-block position, the macro-block MB(x,y,t) in XYT space that has the smallest difference from the provisional sprite is selected as part of the final sprite SP(x,y).

4. Details of video coding

To improve sprite mode quality, we propose two methods: multi video object rate control using sprite and sprite transformation.

4.1 Multi video object rate control for sprite

We use the rate control algorithm of VM17 [4] for normal mode. It may appear that the multi video object rate control algorithm proposed by Vetro et al. [3] could not be used for sprite coding. Video objects are assumed to be "motion pictures", so still images such as sprites are not suitable as coding subjects. Moreover, we emphasize the importance of lowering the latency of sprites and so developed a new multi video object rate control method.

(1) First, foreground and background objects are encoded, using an initial constant quantization parameter (QP). The coding distribution ratio is settled by the code bit ratio.
(2) The optimal constant QP to achieve the total number of bits assigned to the sprite is found.
(3) The remaining bits are assigned to the foreground object.
(4) The foreground object is encoded and the frame rate is determined.
(5) The sprite is encoded by in a low-latency manner at the same frame rate as the foreground.

4.2 Efficient coding using sprite transformation

Sprites can be transformed because they are still images. After deformation, global motion is recalculated to regenerate the frame from the transformed sprite. It is known that scaled-down still images sometimes provide better quality at low bit-rates and this is also true for sprites. Moreover, MPEG-4 has AC/DC prediction for Intra MB coding that effectively compresses intra-frame images; the same is true for sprites. Low-latency sprites are not always encoded from the left to the right; such sprites should be rotated.

Table 1 Example of mode decision.

Start frame	End frame	First decision	Second decision	FG ratio	Final decision	Final shot
1	30	N	N		N	N
31	60	N	N		N	
61	90	N	N		N	
91	120	S	N	0.214	N	
121	150	S	S	0.092	S	S
151	180	S	S	0.101	S	
181	210	S	S	0.120	S	
211	240	S	S	0.063	S	S
241	270	S	S	0.097	S	
271	300	S	S	0.106	S	S
301	330	S	S	0.093	S	
331	360	N	N		N	N
361	390	N	N		N	

5. Coding experiment

We conducted coding experiments using the standard video sequence "stefan" (150frames, SIF) and "skateboard" (390frames, SIF, a skateboarder's follow-shot with right-to-left panning camera motion) is used. Two coding conditions were used: 15fps at 128kbps and 30fps at 384kbps. We compared multi-mode coding to single mode coding in which the video sequences were encoded by VM17.0[4].

Table 1 shows an example of the coding mode decisions made for "skateboard". This example does not mention isolated "S" explaned section2.4. The "stefan" sequence contained only S segments. Table 2 shows the coding results. Figure 4 plots assigned QP. Multi-mode coding yielded lower QPs than single mode coding. Most of the QPs for "stefan" (128kbps, 15fps) are around 31, while multi-mode coding provide much lower QP constantly. Table 2 shows multi-mode coding achieved around 15fps in both sequences at 128kbps, but the average QPof single mode coding is around 9. Figure 5 shows a part of the coding images for "stefan". From the viewpoint of image quality, multi mode coding, in this case sprite mode, provides better subjective visual quality than single mode coding.

Table 2 Coding result

Video Sequence	Mode	Object type	128kbps 15fps			384kbps,30fps		
			Ave. QP	Ave. SNR[dB]	frame rate [fps]	Ave. QP	Ave. SNR[dB]	frame rate [fps]
skateboard	Multi	FG	18.97	24.29	15.00	9.36	30.07	27.57
		BG	18.71	20.29	15.00	9.62	20.25	27.57
		Norm.*	24.07	14.46	14.83	10.95	26.83	30.00
		Total	---	23.16	14.92	---	24.46	28.69
	Single	---	26.40	24.46	12.85	18.15	26.38	30.00
Stefan	Multi	FG	17.07	26.44	15.00	8.34	31.48	30.00
		BG	17.00	18.19	15.00	8.30	18.23	30.00
		Norm.	---	---	---	---	---	---
		Total	---	19.38	15.00	---	19.52	30.00
	Single	---	30.84	24.15	8.90	23.67	25.37	30.00

*Norm. : normal (rectangle) object. This type of object is normally encoded by MPEG-4 Simple Profile.

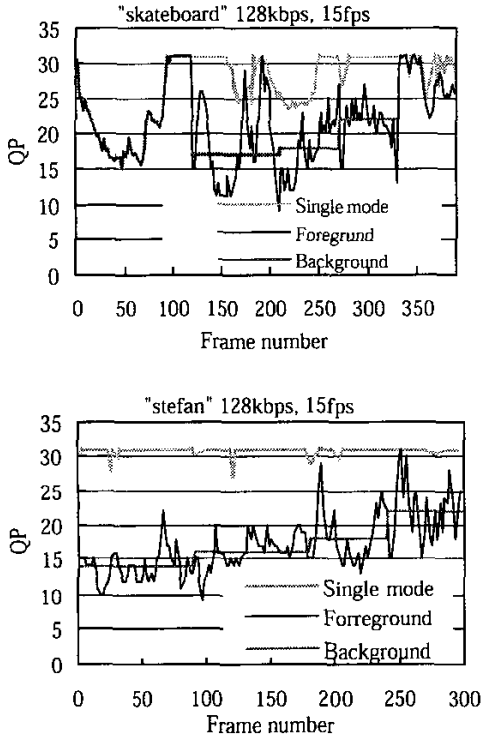


Fig. 4 Assigned QPs .

6. Conclusion

In this paper, we proposed the multi-mode coding system that fully utilizes the advantages of sprite coding. A two-stage coding mode decision (sprite or normal) algorithm was introduced. Modified algorithms were introduced for global motion estimation and sprite generation. A simple method of multi video object rate control using sprites was proposed. The multi-mode coding system achieves higher frame rates and higher quality image than single mode coding for the same bit rate.

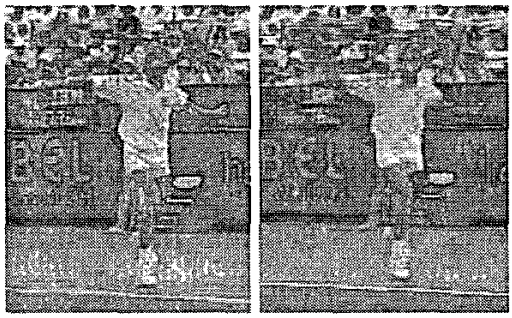


Fig. 5 Final image ("stefan"). Left image produced by multi-mode coding, right one produced by conventional method. (128kbps)

References

[1] "Information technology- Coding of audio-visual objects-Patrt2:Visual Amendment 1:Visual extensions ISO/IEC 14496-2

[2] Kumi Jinzenji, Hiroshi Watanabe, Shigeki Okada, Naoki Kobayashi, "MPEG-4 Very Low Bit-rate Video Compression Using Sprite Coding," IEEE ICME'01, TA0.02.

[3] Anthony Vetro, Huifang Sun, Yao Wang, "MPEG-4 Rate Control for Multiple Video Objects," IEEE Trans. on Circuits And Systems for Video Technology, Vol.9, No.1, pp.186-199, February 1999.

[4] "MPEG-4 Video Verification Model version 17.0," ISO/IEC JTC1/SC29/WG11/N3515.