SYNCHRONOUS AND ASYNCHRONOUS MULTIPLE OBJECT RATE CONTROL FOR MPEG-4 VIDEO CODING

Yu Sun, Ishfaq Ahmad, Jiancong Luo, and Xiaohui Wei

Department of Computer Science and Engineering,
University of Texas at Arlington, Arlington, Texas 76019, USA

ABSTRACT

Video scenes containing multiple objects can potentially achieve higher degree of compression and better visual quality with individual coding for each object. Not always are video objects synchronous, implying each object may have a separate temporal resolution. This paper proposes a rate control algorithm for multiple video object encoding. Using a novel bit allocation strategy, the algorithm achieves accurate target bit rate, provides good visual quality, and decreases buffer overflow/underflow. Experimental results for both synchronous and asynchronous multiple video object encoding demonstrate that, when compared with the existing rate control scheme recommended by the MPEG-4 standard, the proposed algorithm provide better temporal-spatial tradeoff with more accurate rate regulation.

1. INTRODUCTION

In object-based videos, a video object (VO) in a scene may be individually coded and may correspond to an elementary bitstream that can be individually accessed, manipulated and transmitted, while the information regarding the inter-object relationship is sent in a separate stream [1]. The exploitation of the specific characteristics of each object can improve coding efficiency, as well as provide additional flexibility and functions [2]. Rate control (RC) is crucial to provide an optimum peak signal-to-noise ratio (PSNR) within the budget of available transmission bit rate. A few researchers have studied the RC problem for multiple video objects (MVOs) with different temporal resolutions. To our knowledge, with the exception of the work by Nunes and Pereira who presented a scene level RC algorithm for MVOs encoded at different VOP rates [3], the other algorithms [4-7] assume that VOs are synchronous, meaning that all VOs are coded with the same video object plane (VOP) rate. However, this assumption does not always hold because several VOs in a scene may have different temporal resolutions. and thus are asynchronous. Proper asynchronous RC can provide significant savings in bits in object-based videos.

In addition, current MEPG-4 RC schemes adopt a similar bit allocation approach: allocates target bits to a scene for each encoding time, and then distributes the allocated bits among several VOs in a scene. This approach works well for synchronous RC, but it is not very effective when used in asynchronous case for MVOs. The reason is that for each encoding time instant, the number of objects in a scene keeps varying along the coding time, and the VOP types of MVOs may also be different in one coding time instant.

This paper proposes a bit estimation and allocation algorithm for asynchronous multiple object RC, while treating the synchronous RC as a special case. The proposed algorithm, named "SAS" (Synchronous and Asynchronous), considers each VO as a *relatively independent* VO, like a single object encoding case. It divides objects into different "object streams", such that each VO is an independent stream along the coding time. The advantage of this approach is that the asynchronous RC problem is decomposed into multiple sub-problems, with each sub-problem regarded as a single-object variable bitrate control problem. Another advantage is that at the top level, SAS dynamically distributes the bit budget among multiple object streams at each encoding time by jointly adjusting visual qualities of MVOs. At the same time, at the lower level the algorithm can exploit efficient individual single-object or frame-level RC schemes to solve each stream's RC and simplify the traditional bit allocation method.

The rest of this paper is organized as follows: Section 2 presents the details of the SAS algorithm. Section 3 includes the experimental results demonstrating the performance of the SAS. Section 4 concludes the paper with final observations.

2. THE SAS ALGORITHM

Figure 1 presents the basic idea of MVO encoding with different VOP rates. Here, the Foreground Object$_1$ with fast motion has a higher VOP rate, forming an "*object stream$_1$*" along time t_1, t_2, ..., t_n. The Background Object$_2$ with slow motion has a lower rate and forms its own stream. The number of VOPs to be encoded at each encoding time is different, e.g, there are 2 VOPs need to be encoded at time t_1 and one at time t_2. The following sub sections describe the principles and foundations of the algorithm.

2.1. Initial Target Bit Estimation

Bit Ratio Computation for Object Streams: SAS calculates the average number of bits actually used to encode per VOP_i during the previous coding time period before the current time t:

$$\overline{A}_{i,t} = (\sum_k A_{i,k}) / n_i \qquad (k = t-1, \ldots, t-n_i-1),$$

here, $A_{i,k}$ represents the actual bits used to code VOP_i at time k, $t-1$ is the first previous encoding time for VOP_i before t, $t-n_i-1$ is the n_i^{th} previous encoding time, n_i represents the number of VOP_i in a given time period, it equals to the encoding VOP_i rate in our experiments.

Considering different objects have different encoding VOP rates, SAS computes the previous bit ratio $L_{i,t}$ for VOP_i at time t:

$$L_{i,t} = (VR_i \cdot \overline{A}_{i,t}) \Big/ \sum_{j=1}^{M_t} (VR_j \cdot \overline{A}_{j,t}) \quad ,$$

where VR_i means the encoding VOP_i rate. M_t is the number of objects presenting at time t.

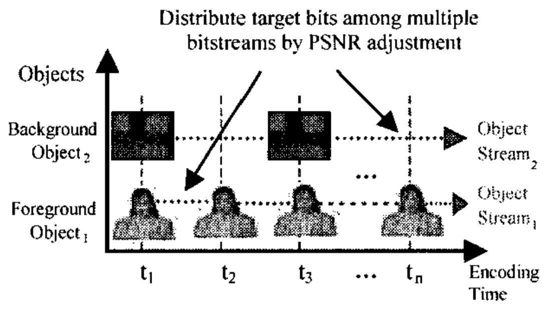


Figure 1: Asynchronous rate control for multiple VOs.

Initial Target Bit Estimation: If VO_i appears at the current time t, then according to its VOP type and $L_{i,t}$, its initial target number of bits is set to a weighted average bitcount:

$$\overline{T}_{i,t} = (\beta_{i,t} \Big/ \sum_{l=0}^{D_i} \beta_{i,l} \cdot N_{i,l}) \cdot L_{i,t} \cdot R_{r,t} \quad ,$$

where D_i is the number of VOP types for VO_i, $N_{i,l}$ is the remaining number of VOP_i with type l, $R_{r,t}$ is the remaining number of bits at time t, $\beta_{i,l}$ is the weight of VOP_i with type l, $\beta_{i,t}$ is the weight corresponding to the current VOP type, l = 0, 1, 2 indicate I-, P- and B-VOP, respectively.

Weight Adjustment among MVOs: To avoid large perceptual quality differences among MVOs, SAS allows adjusting the weight for each object. The larger the weight for VO_i, the more target bits should be allocated to it. Initially, the weight for each object is set to 1.0. The average PSNR at time $t-1$ is:

$$\overline{Q}_{t-1} = \sum_{j=1}^{M_t} (V_{j,t-1} \cdot Q_{j,t-1}) \Big/ \sum_{j=1}^{M_t} V_{j,t-1} \quad ,$$

where, $Q_{j,t-1}$ is the coding quality of VOP_j at time $t-1$, $V_{j,t-1}$ is the number of non-transparent MacroBlocks (MBs) in VOP_j at time $t-1$, it indicates the size of VOP_j. The weight for VOP_i at time t can be adjusted by:

$$W_{i,t} = W_{i,t-1} \times (\overline{Q}_{t-1} / Q_{i,t-1})^{\Gamma} \quad ,$$

where Γ is set to 2 in the experiments. This means if $Q_{i,t-1}$ is lower than $\overline{Q}_{t-1}$, $W_{i,t}$ is increased and VOP_i would get more bits during target bit allocation, thus obtain a higher PSNR; otherwise, $W_{i,t}$ is decreased a little and gets lower PSNR. Then the normalized weight for VOP_i is calculated by:

$$W'_{i,t} = W_{i,t} / \sum_{j=1}^{M_t} W_{j,t} \quad .$$

Object-Level Coding Complexity Analysis: It is important to analysis each object's coding complexity before allocate bits to the object. We adopt our coding complexity measure ([7]) as:

$$C_{i,t} = \sum_{m=1}^{V_{i,t}} \sqrt[k]{\sum_{j=1}^{n_m} (P_{m,j} - \overline{P}_m)^2 / n_m} \qquad (1)$$

where $C_{i,t}$ is the coding complexity of VOP_i at time t, $P_{m,j}$ is the luminance value of the pixel j in the m^{th} MB (MB_m) of a motion-compensated residual VOP_i, $\overline{P_m}$ is the arithmetic average pixel value of MB_m. n_m is the number of non-transparent pixels in MB_m. k equal to 4 in our experiments. $C_{i,t}$ naturally combines the object size ($V_{i,t}$) and average variance of each MB in a VOP, and thus, can reflect the instantaneous characteristics of this VOP [7]. The normalized coding complexity of VOP_i at time t can be obtained by:

$$C'_{i,t} = W'_{i,t} \cdot C_{i,t}$$

Target Bits Adjustment: Considering VOP_i's coding complexity and its average complexity $\overline{C}_{i,t}$ of previous n_i time instants for VOP_i before time t, its target bits budget is then estimated by:

$$T_{i,t} = (C'_{i,t} / \overline{C}_{i,t}) \cdot \overline{T}_{i,t} \qquad (2)$$

The number of target bits is estimated only for P- and B-VOP. We do not estimate target bits for I-VOP, QP of I-VOP can be obtained using our method in [7]. Equation (2) is determined for the following reasons: If $C'_{i,t}$ is higher than $\overline{C}_{i,t}$, more bits should be allocated to VOP_i than $\overline{T}_{i,t}$, and vice versa.

2.2. Buffer Control Strategy

To get more accurate target bit estimation, the initial bit target is further adjusted based on the buffer fullness. Here, we adopt our Proportional-Integral-Differential (PID) buffer control technique [7]. SAS adjusts the total target bits T_t (the sum of each VOP's target bits $T_{i,t}$) allocated to time t. Before adjusting T_t, we should calculate the target bit ratio $g_{i,t}$ for VOP_i, obtained by comparing $T_{i,t}$ with T_t. The PID buffer adjusting factor is computed as:

$$PID_t = K_p \cdot E_t + K_i \cdot \int E_t \cdot dt + K_d \cdot dE_t / dt$$
$$\text{with} \quad E_t = (B_s/2 - B_{f,t}) / (B_s/2) \quad ,$$

where B_s is the buffer size, E_t is the relative error between the target buffer fullness ($B_s/2$) and the current buffer fullness $B_{f,t}$, K_p, K_i and K_d are the Proportional, Integral and Differential control parameters, respectively, and are set empirically to 1.0, 0.05 and 0.9 respectively in the experiments. Then the total target bits T_t can be further adjusted by:

$$T_t = T_t \times (1 + PID_t) \, .$$

The final target bits $T_{i,t}$ for VOP_i after buffer adjustment can be obtained by $g_{i,t}$ times T_t.

2.3. QP Calculation, Encoding, and Post-Encoding

Once the number of target bits for VOP_i is obtained, QP for texture encoding is computed based on the R-D model in VM8 [1], [4-5]. Then, the encoder encodes VOP_i. After encoding, the encoder updates the R-D model for VO_i based on the encoding results [4]. $\beta_{i,l}$ is also updated by: the average number of bits used in coding previous n_I_i I-VOPs or n_B_i B-VOP divided by the average number of bits used in coding previous n_P_i P-VOPs. Considering the tradeoff between keeping the algorithm stability and rapidly reflecting the influence of VO_i's variations, we choose the window size (n_I_i+ n_P_i+ n_B_i) to the number of

VOPs for VO_i in one second in the experiments. The number of bits to be output from the buffer after encoding VOP_i can be computed by:

$$Bpp_{i,t} = (\beta_{i,t} \cdot L_{i,t} \Big/ \sum_{l=0}^{D_i} \beta_{i,l} \cdot N_{i,l}) \cdot R_{r,t} \quad .$$

Then, the virtual buffer fullness can be modified by:

$$B_{f,t} = B_{f,t} + \sum_{i=1}^{M_t} (A_{i,t} - Bpp_{i,t}) \quad .$$

To effectively avoid buffer overflow, the encoder checks the current buffer fullness before encoding next VOPs: If the buffer occupancy exceeds 80% of the buffer size, the encoder skips next VOPs [1]. When VOP skipping happens, the buffer fullness is updated by:

```
While   $((B_{f,t} + \sum_{i=1}^{M_t} (A_{i,t} - Bpp_{i,t})) \geq 80\% \cdot B_s)$
{   // Skip encoding VOPs at the next coding time $t+1$;
    For ($i = 1; i <= M_{t+1}; i++$)
    {       Deciding the VOP type $l$ of the skipped $VOP_{i,l}$;
            $N_{i,l}$--; // Decrease the remaining number of $VOP_{i,l}$;
            Re-calculate $Bpp_{i,t+1}$;
            $B_{f,t+1} = B_{f,t} - Bpp_{i,t+1}$;        }         }
```

3. EXPERIMENTAL RESULTS

Our experiments are conducted for synchronous and asynchronous MVOs. The results are compared with those achieved using the VM8 RC algorithm suggested by the MPEG-4 standard [1] when possible. The initial values of $\beta_{i,0}$ for I-VOP_i, $\beta_{i,1}$ for P-VOP_i and $\beta_{i,2}$ for B-VOP_i are 3.0, 1.0 and 0.5, respectively. $\beta_{i,1}$ is fixed to 1.0, $\beta_{i,0}$ and $\beta_{i,2}$ are dynamically adjusted during the encoding process. The buffer size B_s is set to half of the target rate [1]. According to MPEG-4 core experiments, the PSNR of a skipped VOP is defined by considering that a skipped VOP is represented in the decoded sequence by repeating the last coded VOP of the object [6].

In synchronous MVO RC, MVOs are encoded with the same VOP rate, 30 VOP/s, the Intra period has been set to the half of the VOP rate. The performance results are reported in Table 1, indicating that SAS achieves more accurate target bit rates and the target VOP rate (30 VOP/s) with higher average coding qualities. Also note that for some cases in Table 1, the difference between the PSNR values for VO_1 and VO_2 is much larger with VM8 as compared to SAS. Thus, SAS minimizes the difference in quality to a larger extent.

Figure 2 shows PSNR and buffer curves for the SAS and VM8 algorithm. VM8 RC does not estimate target bits and calculate QPs for I-VOPs. Thus, VM8's performance is not always good for the IPP...IPP... sequence, it has quality fluctuation between inter and intra coded VOPs. In contrast SAS directly estimates QPs for I-VOPs using our method in [7]. Figure 2(a) and (b) show SAS obtains smoother qualities among VOPs. In addition, the buffer occupancy curve of SAS in 2(c) is around the half level of the buffer size with a small variation.

Table 2 shows MVO encoding results with different encoding VOP rates. By default, the encoding VOP rate of the object with higher activity is set to 15 VOP/s, and that of the other object with lower activity is 10 VOP/s, the Intra period for each object has been set to its VOP rate. To evaluate the performance of asynchronous coding, we also give the results of two VOs encoding at the same VOP rate (15 VOP/s) using SAS. Besides accurate target bit rates have been realized without VOP skipping, the results in Table 2 also show that asynchronous coding obtains a better trade-off between spatial and temporal resolutions. For example, for the *News* sequence, VO_1 is Ballet which has faster movement, while VO_2 is the Speakers with slower movement. Asynchronous scheme encodes VO_1 with a higher temporal rate than VO_2, when the target bitrate is 64Kbps, VO_1 obtains a higher average PSNR (34.09 dB) when compared with its PSNR (32.87 dB) in synchronous condition. Meanwhile, although VO_2 gets fewer bits (27.97 Kbps) than its synchronous case (35.63 Kbps), since it has a slower temporal rate, the average visual quality of VO_2 still has 1.00 dB improvement to its PSNR in synchronous case. These results indicate a better trade-off between spatial and temporal qualities can be achieved by SAS. This is especially useful when network bandwidths are very scarce. Figure 3 shows PSNR and buffer curves of *Coastguard* sequence. Since VO_1 and VO_2 have different encoding rates and may appear at different encoding time in asynchronous coding environments, in addition we adopt single joint buffer for MVO RC, we use "encoding time" as x-axis instead of "VOP number" in Fig. 3(b). One can see that buffer curve in Fig. 3 (b) are kept around 50% of the buffer size with a small fluctuation, this indicates our buffer policy is effective for asynchronous RC.

4. CONCLUSION

In this paper, we propose a multiple object rate control scheme for MPEG-4 video coding. Unlike traditional bit allocation methods that first perform bit allocation among scenes at different coding time and then distribute bits among multiple objects in a scene, the proposed algorithm regards each object as one relative independent stream along the coding time, and directly distributes target bits among multiple objects. The algorithm adopts the most effective and direct factor "coding qualities" to control bit allocation among "object streams". The proposed algorithm works well not only for asynchronous multiple objects, but also for synchronous objects. Although this paper only uses the CBR video in the experiments, it is straightforward to apply it to VBR applications as well.

5. REFERENCES

[1] MPEG-4 video verification model V8.0, ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Associated Audio MPEG97/N1796, July 1997, Stockholm, Sweden

[2] P. Nunes and F. Pereira, "Rate Control for Scenes with Multiple Arbitrarily Shaped Video Objects," in Proceedings of the Picture Coding Symposium (PCS'97), Berlin, Germany, Sep. 1997, pp. 303-308.

[3] Paulo Nunes, Fernando Pereira, "Scene Level Rate Control Algorithm for MPEG-4 Video Coding," in *Visual Communications and Image Processing, Proc. SPIE* 4310, pp.194-205, 2001.

[4] Hung-Ju Lee, Tihao Chiang, and Ya-Qin Zhang, "Scalable Rate Control for MPEG-4 Video," *IEEE Trans. On Circuits and Systems for Video Technology*, VOL. 10, pp. 878-894, Sep. 2000.

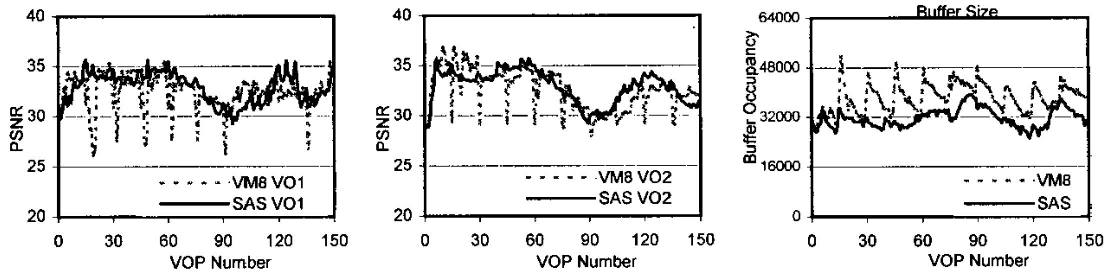
[5] Anthony Vetro, Huifang Sun and Yao Wang, "MPEG-4 Rate Control for Multiple Video Objects," *IEEE Trans. On Circuits and Systems for Video Technology*, VOL. 9, pp. 186-199, Feb. 1999.

[6] Jose I. Ronda, Martina Eckert, Fernando Jaureguizar, and Narciso Garcia, "Rate Control and Bit Allocation for MPEG-4," *IEEE Trans. On Circuits and Systems for Video Technology*, VOL.9, pp.1243-1258, Dec. 1999.

[7] Sun Yu and Ishfaq Ahmad, "A New Rate Control Algorithm for MPEG-4 Video Coding," in *Visual Communications and Image Processing, Proc. SPIE* 4671, pp. 698-709, San Jose, CA. Jan. 2002.

Table 1. Synchronous multiple object rate control (IPP...IPP).

Video Sequence	Algorithm	Bit Rate (kbps)				# Coded VOPs		Average PSNR (dB)	
		Target	Actual	VO1	VO2	VO1	VO2	VO1	VO2
News_1,	VM8	128	130.13	56.09	74.04	140	140	32.42	32.54
News_2	SAS	128	128.74	57.91	70.83	150	150	32.86	32.88
Bream2_1,	VM8	256	260.82	209.02	51.80	145	145	30.71	43.24
Bream2_0	SAS	256	258.14	242.82	15.32	150	150	31.86	38.26
Children2_1,	VM8	256	257.84	186.61	71.23	144	144	28.36	33.84
Children2_2	SAS	256	256.60	191.66	64.94	150	150	28.90	31.64
Coastguard_2	VM8	112	112.50	51.54	60.96	148	148	30.23	30.16
Coastguard_3	SAS	112	112.64	51.78	60.86	150	150	30.38	30.40

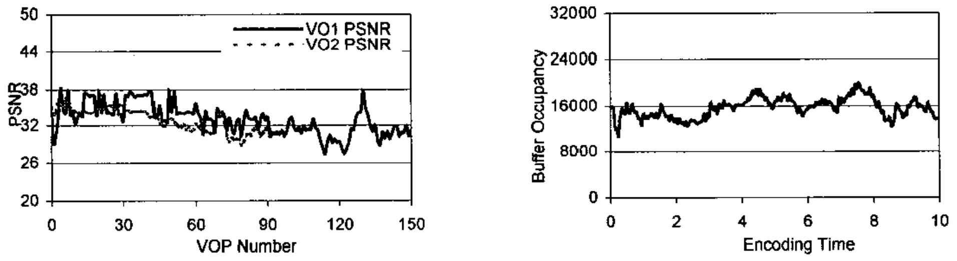


(a) PSNR curves of VO1 (b) PSNR curves of VO2 (c) Buffer Occupancy

Figure 2: *News* sequence in QCIF, 2 synchronous VOs, 30 VOP/s, 128 Kbps.

Table 2. Asynchronous multiple object rate control (IPP…IPP).

Video Sequence	Algorithm	Bit Rate (kbps)				# Coded VOPs		Average PSNR (dB)	
		Target	Actual	VO1	VO2	VO1	VO2	VO1	VO2
News_1,	Synchronous	64	64.42	28.79	35.63	150	150	32.87	32.96
News_2	Asynchronous	64	64.01	36.04	27.97	150	100	34.09	33.96
Bream2_1,	Synchronous	128	128.62	120.52	8.10	150	150	31.82	38.38
Bream2_0	Asynchronous	128	128.95	124.13	4.82	150	100	32.06	38.63
Children2_1,	Synchronous	256	257.16	197.51	59.65	150	150	34.60	38.04
Children2_2	Asynchronous	256	257.13	223.39	33.74	150	100	35.89	36.00
Coastguard_2	Synchronous	64	64.00	28.66	35.34	150	150	31.19	31.31
Coastguard_3	Asynchronous	64	64.05	37.01	27.04	150	100	33.06	32.86



(a) PSNR Curves (b) Buffer curves

Figure 3: *Coastguard* sequence in QCIF, 2 asynchronous VOs, VO1: 15 VOP/s, VO2: 10 VOP/s, 64 Kbps.