MPEG-4 VIDEO OBJECT-BASED RATE ALLOCATION WITH VARIABLE TEMPORAL RATES

Jeong-Woo Lee[a], *Anthony Vetro*[b], *Yao Wang*[c] *and Yo-Sung Ho*[a]

[a] Kwangju Institute of Science and Technology (K-JIST), Kwangju, KOREA
[b] MERL-Mitsubishi Electric Research Laboratories, Murray Hill, NJ, USA
[c] Polytechnic University, Brooklyn, NY, USA

ABSTRACT

This paper describes a bit allocation algorithm to achieve a constant bit rate when coding multiple video objects (MVO's), while improving the rate-distortion (R-D) performance over the reference method for MPEG-4 object-based rate control [1, 2]. In object -based coding, bit allocation is performed at the object level and temporal rates of different objects may vary. In this paper, we deal with these two issues. We pay particular attention to maintenance of buffer occupancy levels and propose a new method for spatio-temporal trade-offs for object-based coding. The proposed algorithm is able to successfully achieve the target bit rate, effectively code arbitrarily-shaped MVO's with different temporal rates, and maintain a stable buffer level.

1. INTRODUCTION

During the past decade, a number of video coding standards have been developed for visual communications. These standards include MPEG-1 for CD-ROM storage, MPEG-2 for DTV and DVD applications, H.261/H.263 for video conferencing systems, and MPEG-4 for object-based low-rate video applications. In MPEG-4, arbitrarily-shaped objects can be encoded and decoded as separate video object planes (VOP's). At the receiver, video objects (VO's) are combined to form compound objects or complex scenes.

Among the above coding schemes, H.263 and MPEG-4 allow us to encode the sequence with variable frameskip. With this policy, the encoder may choose to skip frames to either satisfy buffer constraints or optimize the video coding process.

For the most part, frame skipping has only been employed to satisfy buffer constraints. In this case, the encoder is forced to drop frames since limitations on the bandwidth do not allow the buffer to drain fast enough. Consequently, bits that would be used to encode the next frame cannot be added to the buffer because they would cause the channel buffer to overflow. We should note that skipping frames could lead to poor reconstruction of the video since frames are skipped according to buffer occupancy and not according to content characteristics. Since MPEG-4 allows coding of arbitrarily-shaped objects, the encoder should allocate a significant amount of bits to code the shape information.

In this paper, we consider the object-based rate-distortion (R-D) encoding of MVO's for MPEG-4 video coding. In a recent paper by Vetro, *et al*. [3], models that estimate the distortion for coded frames as well as non-coded frames were proposed. In addition, a rate control algorithm that makes use of these newly developed distortion models for frame-based coding optimization was also proposed. The frame-based coding optimization considers trade-offs in spatial and temporal quality, i.e., it determines whether it is better to code more frames with lower quality or fewer frames with higher quality. However, this algorithm is not directly applicable to the coding of MVO's, where larger coding gains are expected. Lee, *et al*. [4] proposed an algorithm to achieve a trade-off between spatial and temporal quality when coding MVO's; however, they did not address the possibility to code the objects with varying temporal rates. In other words, every object in the current coding time is either coded or all objects are skipped.

In this paper, we propose a framework that supports object-based coding with different temporal rates, aiming to improve the coding efficiency of an object-based coder. Section 2 presents the general framework for object-based coding with variable frame-skip, including rate and buffer constraints that apply for arbitrary frameskip among objects. In section 3, a rate control algorithm for this new framework is proposed, whose key points are the buffer control strategy and a procedure for selecting an optimal set of VO's to be coded at a particular time instant based on an estimate of the spatio-temporal distortion. Bit allocation among different objects is also discussed. Finally, simulation results are provided in section 4 to demonstrate the coding gains achieved, while keeping stable buffer occupancy, and conclusions of this work are given in section 5.

2. OBJECT CODING WITH VARIABLE FRAMESKIP

In the object-based framework, we have a freedom to choose different frameskip factors and corresponding quantization parameters for each object. There are some common issues between the frame-based and object-based problems. We must deal with varying properties of each video sequence and also consider the impact of rate allocation on a shared buffer. This section introduces the general object-based coding framework and presents a set of constraints on both the rate and the buffer.

Fig. 1 shows an example of object-based coding in which each VO has a different frameskip. Let M denote the set of VO id's, and L denote the complete set of time indices. $M(l)$ denotes the set of coded VO at time index l ($t = t_l$). Also, given $l \in L$, let l_0 equal the previous value of l, except when l is the first element in L; in that case $l_0 = 0$. For example, in Fig. 1, $M = 0, 1, 2$, $L = l_1, l_2, l_3, l_4, l_5, l_6, l_7$, $M(l_1) = 0, 1, 2$, $M(l_2) = 0, 2$, $M(l_3) = 1, 2$, and so on. Then, the constraint on the rate can be written as

This work was supported in part by KOSEF through UFON and in part by MOE through BK21.

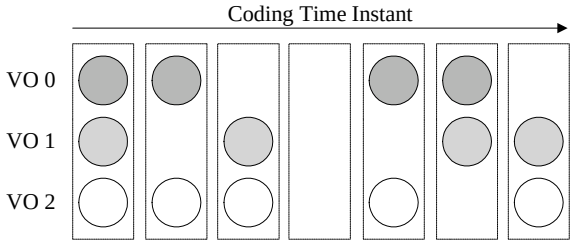


Fig. 1. Object-based Coding with Arbitrary Frameskip

$$\sum_{l\in L}\sum_{j\in M(l)} R_j(t=t_l) \le R_{\text{budget}} \quad (1)$$

where $R_j(t=t_l)$ is the number of bits used for the jth VO at time index l. Eq. (1) essentially says that the sum of rates for all objects at all time instants within the specified time interval must be less than the calculated bit rate budget over that time interval.

In order to ensure that buffer overflow and underflow are avoided at every coded time instant, we have a set of buffer constraints which are given by

$$B_{i+l_0} + \sum_{j\in M(l)} R_j(t=t_{l_0}:t_l) < B_{\max};\ \forall l\in L \quad (2)$$

$$B_{i+l_0} + \sum_{j\in M(l)} R_j(t=t_{l_0}:t_l) - (l-l_0)R_{drain} > 0;\ \forall l\in L \quad (3)$$

where i is the current time index, B_{i+l_0} is the current buffer level in bits, $B_{\max}$ is the maximum buffer size in bits, and R_{drain} is the rate at which the buffer drains per time instant. $R_j(t=t_{l_0}:t_l)$ is the number of bits used for the jth VO at time index from l_0 to l.

The constraints presented above for the rate and the buffer are valid for arbitrary frameskip factors. With these constraints, it is possible to formulate a problem that aims to minimize the overall coding distortion. Such a problem could be solved by searching over all valid combinations of frameskip factors and quantization parameters within a specified period of time. However, the complexity of such a approach is very high. Not only are there many combinations of frameskip factors and quantization for each object, but we must also track the individual time instant that each object is coded. It is also important to avoid composition problems in the decoder. An algorithm for preventing this problem with the shape hints is presented in [5]. In the following, a low-complexity object-based coding framework is considered that aims to improve overall coding efficiency.

3. RATE CONTROL

In this section, we consider the rate control algorithm for a restricted object-based framework. We refer to this framework as being restricted since a decision on the frameskip factor of a particular VO is made locally, i.e., without considering the various combinations of frameskip factors. With this framework, the proposed algorithm first determines the set of objects to be coded at the next coding time, and then considers the bit allocation for each object to be coded. In the remainder of this section, an overview of the algorithm is presented, which mainly includes buffer control, selection of an optimal set of objects to be coded, and bit allocation among these objects.

1 Set $f_s = 1$, $D_{\min} = \infty$
2 Calculate the initial target bits for the current time index, and scale this target based on the current buffer level and the buffer size. See Section 3.4.
3 Compare the target bits with the motion, shape and header bits of the previous coded objects, and determine the coded object set, M_L, based on the target bits. The set M_L includes all possible subsets of M. In the case that every object should be skipped, M_L is defined as an empty set, and we repeat from step 2 with $f_s = f_s + 1$. See Section 3.2.
4 Distribute the target bits according to Eq. (12) for each object included in the subset $M(i+f_s)$ belonging to M_L.
5 Calculate the quantization parameter and the target bits for each object.
6 Estimate the distortion using Eq. (7) and check the buffer condition using Eq. (4).
7 If the current distortion is less than $D_{\min}$, then replace $D_{\min}$ with the current distortion and record encoding parameters.
8 Repeat from step 4 with the next subset $M(i+f_s)$ belonged to M_L.
9 Determine the optimal set of coded objects, $M^*(i+f_s)$, with minimum distortion according to Eq. (7) and encode the objects belonging to this set.
10 Update the next coding time index using f_s and $M^*(i+f_s)$, and repeat from step 1.

Fig. 2. Object-based Rate Control Algorithm

3.1. Algorithm Overview

The purpose of the rate control algorithm is to maximize the coding performance subject to constraints on the overall bit rate and buffer occupancy. The problem is formulated as follows.

$$\min_{M(i+f_s)\subset M} \left| d_{M(i+f_s)}(Q, f_s) \right| \quad (4)$$

$$\text{subject to} \begin{cases} \bar{R} \le R \\ B_i + \sum\limits_{j\in M(i+f_s)} R_j(t_{i+f_s}) < B_{\max} \\ B_i + \sum\limits_{j\in M(i+f_s)} R_j(t_{i+f_s}) - f_s \cdot R_{drain} > 0 \end{cases}$$

where $M(i+f_s)$ denotes the set of coded object. $d_{M(i+f_s)}(Q, f_s)$ represents the distortion of all VO's at time index $(i+f_s)$ and f_s represents the amount of frameskip. $R_j(t_{i+f_s})$ is the number of bits used for the jth VO at time index $(i+f_s)$.

In order to control the rate and select the optimal coded object set, we use the rate control algorithm, shown in Fig. 2. In frame-based rate control, if the target bits are less than the header bits which are used to code the motion, shape and header information, then the encoder is forced to skip all objects. In the proposed object-based rate control, however, the algorithm allows the encoder to code a portion of objects because the proposed bit allocation is performed at the object level. In order to improve the coding efficiency of each object, we distribute the target bits to each object using the size, motion and a variance-like measure. In order to support uniform picture quality from frame to frame, we restrict the current QP to the previous QP of the same VO [2]. In addi-

tion, we set the initial target bits based on the current buffer level; therefore, the buffer check is necessary to prevent buffer overflow and underflow.

3.2. Buffer Control

In addition to the bits used for texture and motion, an object-based encoder must consider the significant amount of bits that are used to code the shape information. In [2], the significance of this problem due to the high percentage of shape bits has been shown for low bit-rate coding conditions. Therefore, we need to develop a buffer control strategy to determine the number of VOP's to be skipped.

The frameskip factor, f_s, is increased, as any object cannot be coded at the current time index. It should be noted that the frame-skip rate for the proposed object-based algorithm is smaller than that for the frame-based algorithm. This is because the part of all objects can be coded at the object-based bit allocation. Let M_L denote all the complete set of partially coded object. If M_L is defined as an empty set, the frameskip factor is increased.

Since the proposed algorithm determines the set of objects to be coded at the next coding time, the buffer constraints are given by Eq. (2) and Eq. (3) with time index $l = (i + f_s)$. We should note that the number of bits used for coding VO during the frameskip rate is 0.

3.3. Selecting an Optimal Set of Video Objects

The set of objects to be coded at a particular time instant is selected based on the total distortion associated with each VOP, which includes both coded distortion due to the quantization error and non-coded distortion due to the skipped VO's. The distortion models presented in [3] for frame-based video coding are used in this work for object-based video coding [4]. For completeness, these models are briefly described below.

The coded distortion for the jth VO at time index i is modeled by

$$D_c(Q_{j,i}) = a \cdot 2^{-2R_j(t_i)} \cdot \sigma^2_{z_{j,i}} \tag{5}$$

where $\sigma^2_{z_{j,i}}$ is the signal variance of the jth coded VO at time index i, $R_j(t_i)$ is the average rate per sample for texture data of the jth VO, $Q_{j,i}$ denotes the quantization parameter, and a is a constant that is dependent on the *pdf* of the input signal and the quantizer characteristics. It is important to note that the average rate, $R_j(t_i)$, is calculated within a region where each object is defined.

The non-coded distortion at time index k for the jth VO that was previously coded at time index i is modeled by

$$D_s(Q_{j,i}, k) = D_c(Q_{j,i}) + E_j\{\Delta^2 z_{i,k}\} \tag{6}$$

where $E_j\{\Delta^2 z_{i,k}\}$ denotes the expected interpolation error between time index i and k.

Therefore, the optimal set of VO's to be coded, $M^*(i + f_s)$, are those that satisfy

$$d_{M^*(i+f_s)}(Q, f_s) = \min_{M(i+f_s)\subset M_L} \left| d_{M(i+f_s)}(Q, f_s) \right| \tag{7}$$

where

$$d_{M(i+f_s)}(Q, f_s) = \sum_{j\in M(i+f_s)} D_{j,c}(Q_{j,i+f_s}) + \sum_{j\notin M(i+f_s)} D_{j,s}(Q_{j,i}, f_s) \tag{8}$$

3.4. Bit Allocation

The initial target bits, T_v, for the current time index is allocated based on the initial assumption that every object is coded at the current time index. Similar to the frame-based bit allocation, the initial target for each object is usually calculated based on the remaining bits, T_r, the number of bits used for coding the previous jth object, $\tilde{T}_{p,j}$, the current value of f_s, the number of VOPs, N_r, which remain to be coded, and the number of objects [2, 4]. If the jth object is not coded at the previous coded time index, $\tilde{T}_{p,j}$ is assigned the coded bits determined from the actually coded time index.

After the initial target has been determined, the target bits are scaled according to

$$T_B = T_V \cdot \frac{\tilde{B}_i + 2(B_{\max} - \tilde{B}_i)}{2\tilde{B}_i + (B_{\max} - \tilde{B}_i)} \tag{9}$$

where

$$\tilde{B}_i = B_i - (f_s - 1) \cdot R_{drain} \tag{10}$$

We should note that $\tilde{B}_i$ emulates a future buffer occupancy based on the frames to be skipped, as determined in section 3.2.

Based on the scaled target, we now consider distributing the available bits to the objects to be coded. Let T_{hdr} be the number of bits used for the shape, motion and header information of the previous objects belonging to the subset $M(i + f_s)$, i.e.,

$$T_{hdr} = \sum_{j\in M(i+f_s)} T_{j,hdr} \tag{11}$$

In order to guarantee that the target for the jth object is always larger than $T_{j,hdr}$, we use the following equation for distributing bits to each object.

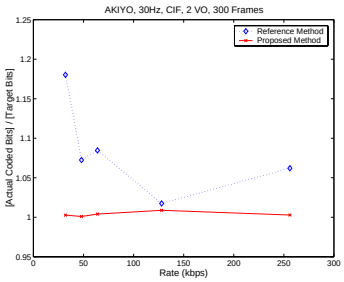
$$T_j = (T - T_{hdr}) \cdot (w_m \mathrm{MOT}_j + w_v \mathrm{VAR}_j) + T_{hdr,j} \tag{12}$$

Given the target number of bits for each object, we calculate the quantization parameter for each object based on a quadratic rate-quantizer model [2].

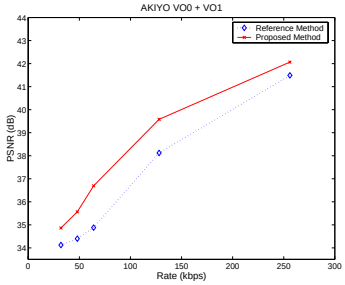
4. SIMULATION RESULTS

In order to evaluate performance of the proposed algorithm, we consider the AKIYO sequence that contains 2 VO's at the CIF format of 300 frames. This sequence is encoded at different bit rates using the standard MPEG-4 rate control algorithm that is implemented as a part of the MPEG-4 reference software [6]. Bit rates that we consider range from 32 kbps to 256 kbps, and the sequence is encoded at the full frame rate of the source sequence on the input. The global buffer size (VBV_SIZE) is just a half of the target bit rate.

Fig. 3(a) plots the R-D curves for the AKIYO sequence. They are calculated over all the frames, including those frames that are skipped and are simply reconstructed by copying from the previous frame. Fig. 3(a) shows that the proposed method outperforms the MPEG-4 reference method. While the MPEG-4 reference method is forced to skip VOP's due to buffer constraints, the proposed method skips VOP's based on buffer constraints as well as the minimum distortion criteria. In the proposed algorithm, it is not necessary for VOP to include every object. We observe that lower QP's are automatically assigned to a more interesting foreground object.

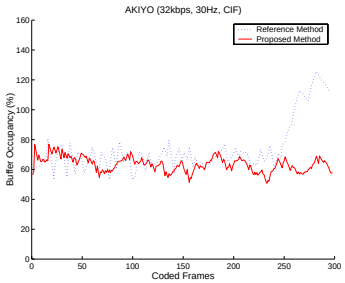


(a) R-D Curves

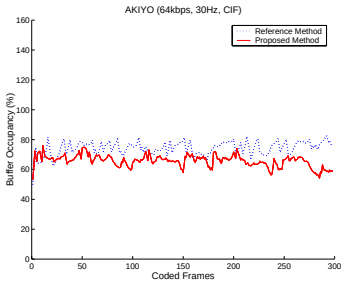


(b) Ratio of Actual Coded Bits to Target Bits

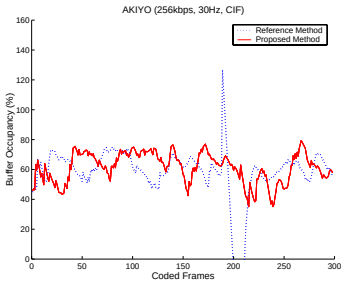
Fig. 3. Performance Comparisons for AKIYO



(a) 32 kbps



(b) 64 kbps



(c) 256 kbps

Fig. 4. Comparison of Buffer Occupancy

One key aspect of the proposed algorithm is that the actual coded bits in the proposed algorithm are similar to the target bits over a wide range of bit rates. Fig. 3(b) shows the ratio of the actual coded bits to the target bits. From Fig. 3(b), we observe that actual coded bits are well matched to the target bits in our proposed algorithm, while there are significant fluctuations in the MPEG-4 reference method.

It is also important to note that we have not experienced any buffer overflow or underflow with the proposed algorithm. Fig. 4 shows the buffer occupancy for the proposed method and the reference method from 32 kbps to 256 kbps. The initial buffer level is set to 0 before coding the first I-VOP. As we can see in Fig. 4, the buffer occupancy for the proposed algorithm is quite stable over the broad range of testing conditions and is always under 100%, although the test sequence has a large frame size. The occupancy has approximately 60% on average and variations of about 20%. From these results, we can say that the buffer has little chance of overflow or underflow, although we do not show other simulation results. As shown in Fig. 4(a) and Fig. 4(c), the buffer in the MPEG-4 reference method experiences at least one overflow. We have observed that this problem is partly because the current QP is forced to vary smoothly from the previous QP.

5. CONCLUSIONS

In this paper, we have proposed an MPEG-4 video object-based rate allocation method where bit allocation is performed at the object level and temporal rates of different objects may vary. The proposed algorithm considers the spatio-temporal trade-offs for object-based coding with variable temporal rates. Simulation results show that the proposed algorithm has an improved performance over the MPEG-4 reference software.

In contrast to the frame-based rate control method, the proposed algorithm allows the encoder to code a subset of VO's because the proposed bit allocation is performed at the object level. Consequently, the proposed algorithm improves the coding efficiency about 1-2 dB than the MPEG-4 reference method. The most important aspect is that the actual coded bits in the proposed algorithm are almost the same as the target bits over a wide range of bit rates, while it enables the encoder to code arbitrarily-shaped MVO's with different temporal rates.

6. REFERENCES

[1] ISO/IEC 14496-2:1998, "Information Technology - Coding of audio/video objects," Part 2:Visual.

[2] A. Vetro, H. Sun, and Y. Wang, "MPEG-4 rate control for multiple video objects," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 186-199, Feb. 1999.

[3] A. Vetro, Y. Wang, and H. Sun, "Rate-Distortion Optimized Video Coding Considering Frameskip," *Proc. ICIP*, pp. 534-537, Oct. 2001.

[4] J.W Lee, A. Vetro, Y. Wang and Y.S. Ho, "Object-based Rate Allocation with Spatio-Temporal Trade-offs," *Proc. VCIP*, pp. 374-384, Feb. 2002.

[5] A. Vetro and H. Sun, "Encoding and transcoding multiple video objects with variable temporal resolution," *Proc. ISCAS*, May 2001.

[6] ISO/IEC 14496-5:2000, "Information Technology - Coding of audio/video objects," Part 5:Reference Software.