OPTIMIZING FRAME STRUCTURE WITH REAL-TIME COMPUTATION FOR INTERACTIVE MULTIVIEW VIDEO STREAMING

Yu Gao[1], *Gene Cheung*[2], *Jie Liang*[1], *and André Kaup*[3]

[1] School of Engineering Science, Simon Fraser University, BC, Canada
[2] National Institute of Informatics, Tokyo, Japan
[3] Multimedia Communications and Signal Processing, University of Erlangen-Nuremberg, Germany

ABSTRACT

Interactive multiview video streaming (IMVS) is an application where a network client requests from server a single video view at a time but can periodically switch to other views as the video is played back uninterrupted. Existing IMVS algorithms output pre-computed frame structures that facilitate permissible view-switching while minimizing the expected transmission rate given a storage constraint. In this paper, we use real-time computation (available at a remote powerful server or media cloud) to assist the pre-computed frame structure to satisfy users' view-switch requests. In particular, we first propose a new frame type called uni-merge frame that is computed in real-time for view-switching from one single view to one target view with low transmission rate and reasonable computation cost. Then, to enable permissible view-switches to a particular target picture, we find the optimal combination of pre-computed frames and real-time computed frames—one that minimizes streaming rate subject to both storage and real-time computation constraints—using a greedy combinatorial algorithm. Experimental results show that with real-time computation, the expected streaming rate of the IMVS system can be further decreased by 50% compared to pre-encoded frame structures without real-time computation.

Index Terms — Multiview videos, video streaming, optimization, real-time computing

1. INTRODUCTION

Multiview videos refer to videos of the same 3D scene captured by multiple closely spaced cameras from different viewpoints. They can enable visually immersive applications such as free viewpoint TV [1], virtual walk-through, etc. However, storage and transmission of multiview video data are very challenging, due to the large amount of visual data involved. As a result, there has been extensive research in multiview video coding (MVC) [2], where the goal is to compress video frames of all captured views across time in a rate-distortion optimal manner.

In many applications, however, not all views are required at the client at the same time. In interactive multiview video streaming (IMVS) [3], a network client only requests from the server one captured view at a time for rendering on conventional 2D display, but can switch to other viewpoints periodically during video playback. The view-switching period is usually set very small to provide smooth view-switching visual experience.

To support frequent view-switching in IMVS without incurring large transmission cost, instead of MVC structures (where inter-view predictions create complicated inter-dependency among frames, limiting random access), previous IMVS studies [3, 4] proposed *redundant frame representation,* so that the expected transmission rate can be reduced at the cost of increased storage. In particular, combinations of *redundant P-frames* (with low transmission rate but large storage cost) and *merge frames* (based on Distributed Source Coding (DSC) [5] to merge multiple decoding paths, with high transmission rate but low storage cost) that optimally trade off between the expected transmission rate and storage required to contain the redundant frame structure are sought in a combinatorial optimization.

The coding structures in the existing IMVS schemes [3, 4] are pre-computed and stored, incurring storage cost. With the advent of parallel and cloud computing, it is now possible to perform a limited amount of video processing tasks in real-time on demand [6, 7, 8] at affordable computation cost. In this paper, we optimize the design of IMVS redundant frame structure with the help of real-time computation. In particular, we first propose a new frame type called *uni-merge* frame[1] that is computed in real-time for view-switching from one single view to one target view with low transmission rate and reasonable computation cost. Then, to enable permissible view-switches to a particular target picture, we find the optimal combination of pre-computed frames and real-time computed frames—one that minimizes expected streaming rate subject to both storage and real-time computation cost constraints—using a greedy combinatorial optimization. Experimental results show that with real-time computation, the expected streaming rate of the IMVS system can be further reduced by 50% compared to pre-encoded frame structure without real-time computation.

The outline of the paper is as follows. We first discuss related work in Section 2. We then describe our IMVS system and optimization in Section 3 and 4. Finally, we present experimental results and conclusion in Section 5 and 6, respectively.

2. RELATED WORK

When a computation-intensive input-to-output information processing task needs to be performed repeatedly over time, instead of computing all possible input-to-output mappings in real-time, mappings corresponding to the more frequently occurring inputs

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada under grants RGPIN312262, STPGP350740 and STPGP380875, and Alexander von Humboldt Foundation. Emails: gaoyug@sfu.ca, cheung@nii.ac.jp, jiel@sfu.ca, kaup@lnt.de.

[1]Previously proposed merge frame [3] will henceforth be called *multi-merge* frame.

can be pre-computed and stored in memory, so that during real-time processing, the pre-computed results can be simply looked up and returned. Finding the optimal mixture of real-time computation and pre-computing partial results in storage for a given processing task is a fundamental problem in algorithm implementation [9], and has been successfully investigated in various problem settings such as IP address lookups in network routers [10] and scalar and vector quantization encoding [9].

With the advent of parallel and cloud computing, real-time computation for some video processing operations can become affordable. We hence revisit the IMVS structure design problem, further optimizing the streaming rate / storage tradeoff with the help of real-time computation. In particular, we study the tradeoff among streaming rate, storage and real-time computation costs for previously proposed view-switching tools [3]—redundant P-frames and multi-merge frame—and a new tools called uni-merge frame, and optimize new frame structures based on our analysis.

3. SYSTEM DESCRIPTION

We consider an IMVS system that offers *dynamic* and *static* view-switching every N frames in time for streaming clients. In other words, after playing back the video of a single view for N temporal frames, a client can either switch to one of the other captured views as video continues playback in time, or freeze in time and switch to other views. To enable this view-switching functionality efficiently, at a particular view-switching point, we find the optimal structure composed of pre-computed frames and real-time computed frames to minimize expected streaming rate subject to storage and real-time computation constraints.

Let $F_{i,j}$ be a *picture group* that includes N pictures[2] of the j-th view with time instants $iN, iN+1, \ldots, (i+1)N-1$. A view switch from group $F_{m,n}$ to $F_{i,j}$ is denoted as $(m,n) \rightarrow (i,j)$. In this paper, only switches within a view distance of K views are supported. Thus, a legal dynamic view-switch can be represented by $(i-1,n) \rightarrow (i,j)$, and a legal static view-switch by $(i,n) \rightarrow (i,j)$, where $j-K \leq n \leq j+K$ in both cases.

Our goal is to design a *frame structure* $\mathcal{S}_{i,j}$ to represent pictures in group $F_{i,j}$, one that facilitates all legal view-switches to $F_{i,j}$, while achieving the optimal tradeoff among transmission rate, storage and real-time computation. In each structure $\mathcal{S}_{i,j}$, the first picture of instant iN can be represented in multiple versions as *redundant P-frames*, or a single version as a *merge frame* [3]. The remaining pictures in the group are each coded as a P-frame, motion-compensated using the previous temporal frame of the same view as predictor. All coded versions of the first picture must reconstruct to exactly the same frame to avoid coding drift in the following differentially coded frames. We accomplish that using DSC frames [5], as discussed below.

3.1. Distributed Source Coding Frames

We employ two types of DSC frames with different tradeoffs between storage and transmission rate [5]. The first type is called DSC0, which includes a set of motion vectors (MV) and low-density parity check (LDPC) code. In particular, for each legal view-switch from a predictor frame to target frame, motion estimation is first performed, resulting in a set of MVs and an estimate of the target frame (*side information* (SI)). LDPC code is then used to remove the difference (noise) between SI and the target frame. Note that only one set of LDPC code is used for all possible predictor frames, so that no matter which SI is available from which predictor frame, the same DSC0 frame can be reconstructed. DSC0 is a *multi-merge* frame, as multiple decoding paths are merged at the target frame so that the same target picture can be reconstructed.

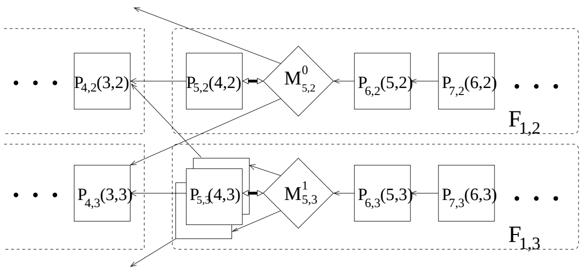

Figure 1. Coding structure examples with P-/DSC0-/DSC1-frames.

The second DSC frame type DSC1 is used together with redundant P-frames. Specifically, motion compensation is first performed from each legal view-switching reference frame, resulting in a set of P-frames. Then, LDPC code is generated to remove the difference between these redundant P-frames and the target frame. Though the coding mechanism is the same, DSC0 acts as a multi-merge frame, while the much smaller DSC1 (redundant P-frames are of the same view and time instant as the target frame, resulting in better quality SI with small noise) acts as a "denoising" frame. Compared to DSC0 frame, the combination of redundant P-frames plus DSC1 frame requires more storage, but it has lower transmission rate, thanks to the smaller DSC1 frame.

Usage of DSC0 and DSC1 is illustrated in Fig. 1 for $N = 5$. $P_{i,j}(m,n)$ denotes a P-frame (square) for instant i and view j using frame of instant m and view n for prediction. $M^0_{i,j}$ and $M^1_{i,j}$ denote DSC0 and DSC1 frame (diamond), respectively. Note that $M^1_{i,j}$ is used in combination with redundant P-frames. Note further that the target frame for DSC frame $M_{i,j}$ is chosen to be a *re-quantized* version[3] of P-frame $P_{i,j}(i-1,j)$; i.e., the DSC frame will reconstruct to be bit-by-bit equivalent to re-quantized $P_{i,j}(i-1,j)$. This is done so that during normal video playback in the same view (the most likely view-switch), only P-frame $P_{i,j}(i-1,j)$ needs to be transmitted.

3.2. Tradeoffs in IMVS View-switching Tools

Having discussed previous view-switching tools for IMVS [3], we now overview the tradeoffs in streaming rate, storage and real-time computation for these tools for intuition. First, consider the case when real-time computation is expensive. In this case, DSC0 frame would offer the most storage-efficient view-switching solution, since no extra P-frames are stored, though it would result in a large transmission rate due to the large DSC0 frame size. Combination of redundant P-frames and DSC1 frame would offer a lower transmission rate solution, due to the smaller DSC1 frame size. However, the redundant P-frames (for large K) would lead to large storage cost.

Consider now the case when real-time computation is affordable. For the combination of redundant P-frames plus DSC1 frame $M^1_{i,j}$, instead of pre-computing and storing redundant P-frames

[2]We will use "picture" to denote the original captured image, and "frame" to denote a coded version of a picture.

[3]Re-quantization means the decoded P-frame is re-encoded as an I-frame. This is done so that LDPC code in the DSC frames only have to match the quantization bin index of each transform coefficient, lowering LDPC encoding rate.

$P_{i,j}(m,n)$'s for all legal view-switches $(m,n) \rightarrow (i,j)$, we can now store only the frequently accessed P-frames, while the less accessed P-frames are computed in real-time. The real-time computed P-frames are discarded after use to avoid storage cost.

In contrast, one can use real-time computation to change the encoding of DSC0: DSC0 can be computed in real-time on demand for a *single* view-switch $(m,n) \rightarrow (i,j)$. That means only one set of MVs for a particular predictor frame plus LDPC code strong enough to overcome the noise in this particular SI is sufficient to perfectly reconstruct the target frame. We denote this real-time computed frame as DSC0-RT $M^0_{i,j}(m,n)$. Note that because it handles only a single view-to-view switch, the size of this DSC0-RT frame is much smaller than pre-computed DSC0 frame, and smaller than $P_{i,j}(m,n)$ plus DSC1 $M^1_{i,j}$. Further, instead of using channel code like LDPC to remove noise, one can alternatively use differential coding techniques like H.264's secondary SP-frames [11] to perfectly reconstruct the target frame. This real-time computed *uni-merge* frame (DSC0-RT)—one with low transmission rate and reasonable real-time computation cost—to switch from a single view to a target view is in stark contrast to the pre-computed *multi-merge* frame (DSC0) that must necessarily merge multiple decoding paths for best tradeoff between transmission rate and storage.

We next discuss how the best combination of DSC0, redundant P-frames plus DSC1, and DSC0-RT can be found through a greedy optimization.

4. FRAME STRUCTURE OPTIMIZATION

4.1. Problem Formulation

We now formulate our objective function. We first assume a user switches from group $F_{m,n}$ to $F_{i,j}$ with view-switch probability $p_{i,j}(m,n)$, where $\sum_{i,j} p_{i,j}(m,n) = 1$, $\forall (m,n)$. The normal playback probability $p_{i,j}(i-1,j)$ will be the largest relative to other switches. Let $\pi_{i,j}$ be the steady-state probability of $F_{i,j}$.

Let $|\mathcal{S}_{i,j}|$ be the size of structure $\mathcal{S}_{i,j}$, and $S^X_{i,j}(m,n)$ and $S^C_{i,j}(m,n)$ be the transmission rate and real-time computation cost associated with the view-switching from $F_{m,n}$ to $F_{i,j}$, respectively. Given storage and computation budget $\bar{S}$ and $\bar{C}$ for the entire multiview video, the constrained optimization is written as:

$$\begin{aligned} \min \quad & \sum_{i,j} \sum_{m,n} \pi_{m,n} p_{i,j}(m,n)\, S^X_{i,j}(m,n) && (1) \\ \text{s.t.} \quad & \sum_{i,j} |\mathcal{S}_{i,j}| \le \bar{S}, \quad \sum_{i,j} \sum_{m,n} \pi_{m,n} p_{i,j}(m,n) S^C_{i,j}(m,n) \le \bar{C} \end{aligned}$$

where the optimization variables are the structures $\mathcal{S}_{i,j}$'s for groups $F_{i,j}$'s in the video.

Instead of solving the constrained problem (1), we solve the unconstrained version problem using Lagrange multipliers λ and μ for the two constraints:

$$\begin{aligned} \min \quad & \sum_{i,j} \sum_{m,n} \pi_{m,n} p_{i,j}(m,n)\, S^X_{i,j}(m,n) + && (2) \\ & \lambda \sum_{i,j} |\mathcal{S}_{i,j}| \; + \; \mu \sum_{i,j} \sum_{m,n} \pi_{m,n} p_{i,j}(m,n) S^C_{i,j}(m,n) \end{aligned}$$

where λ and μ need be adjusted so that the optimal solution to (2) meets the two constraints in original (1).

It is clear that (2) can be solved separately for each group $F_{i,j}$ without losing optimality:

$$\min \; \lambda |\mathcal{S}_{i,j}| \; + \; \sum_{m,n} \pi_{m,n} p_{i,j}(m,n) \left(S^X_{i,j}(m,n) + \; \mu S^C_{i,j}(m,n) \right) \qquad (3)$$

Hence we next describe our algorithm to find the optimal structure $\mathcal{S}_{i,j}$ for each group $F_{i,j}$ in (3).

4.2. Greedy Optimization

Following our discussion in Section 3.2, we see that there are three logical options for structure $\mathcal{S}_{i,j}$. If real-time computation and storage costs are both expensive (weighted by $\pi_{m,n}\; p_{i,j}(m,n)\; \mu$ and λ in (3) respectively), then pre-computing DSC0 frame is optimal—solution with smallest storage size without real-time computation cost. If real-time computation cost is very cheap, then DSC0-RT frame is optimal—solution with smallest streaming rate and no storage cost. For other cases, pre-computing DSC1 frame with redundant P-frames optimally divided between pre-compute and real-time computation would be a good solution. Hence we can devise our optimization strategy as follows to check for the performance of these three basic structures.

1. Pre-compute a DSC0 frame $M^0_{i,j}$ for storage and calculate objective (3) as cost J^0.
2. Pre-compute a DSC1 frame $M^1_{i,j}$ for storage.
3. Incrementally add the *most beneficial* redundant P-frame $P_{i,j}(m,n)$ for a legal view-switch $(m,n) \rightarrow (i,j)$, *i.e.*, one that lowers objective (3) the most. Stop when there are no more beneficial P-frame to add.
4. Calculate objective (3) as cost J^1.
5. Real-time compute DSC0-RT frame $M^0_{i,j}(m,n)$ for all lecal view-switches $(m,n) \rightarrow (i,j)$. Calculate objective (3) as cost J^2.

The structure that corresponds to the smallest of the three costs J^0, J^1 and J^2 would be the optimal structure we choose for $\mathcal{S}_{i,j}$.

5. EXPERIMENTATION

For intuition, we first illustrate when the three basic structures become optimal for different prices of storage and computation. Then we compare our proposed scheme to a competing scheme that does not utilize real-time computation and all frames are pre-computed and stored.

The video playback probability without view-switch $p_{i,j}(i-1,j)$ is denoted by p_{pb} for short. Let $p_1 = (1-p_{\text{pb}})/(K \cdot (K+1))$. The view-switch probability from view j to view $j \pm k$ is $(K+1-k)\,p_1$ for $k \in [1,K]$. This includes both static and dynamic view-switches. In our first experiment, the steady state probability $\pi_{i,j}$ is set to be $0.5/(2K+1)$, where K is the maximal view-switch distance.

We first tune the Lagrange multipliers λ and μ in (3) to show the influence of storage and computation prices on the optimal structure. The picture group $F_{2,3}$ of the multiview video sequence `Kendo` is coded into a structure generated using our proposed scheme, with $K = 3$, $N = 4$ (picture group size). The results are summarized in Table 1 for different p_{pb}.

When storage and real-time computation are both expensive, J^0 is smaller than J^1 and J^2, *i.e.*, the pre-computed DSC0 frame merging all legal view-switches is optimal, since it has the smallest storage and no real-time computation cost. When storage is very cheap, the pre-computed DSC1 with all redundant P-frames also pre-computed is the best choice, where the transmission rate is lower than the pre-computed DSC0 frame and no real-time computation cost is paid. In usual cases, DSC1 with combination of

Table 1. Costs of different structures.

p_{pb}	λ	μ	J_0	J_1 (computation cost)	J_2
0.5	0.1	10	**114212**	128299 (35710)	14296445
	0.0001	0.1	47655	**20593 (0)**	149730
	0.01	0.1	54251	**25821 (714)**	150113
	0.01	0.01	54251	25178 (71)	**21541**
0.9	1	200	**713824**	762087 (285800)	57185054
	0.00001	0.1	47595	**6307 (0)**	32152
	0.01	0.1	54251	**11131 (143)**	32538
	0.01	0.01	54251	11003 (14)	**6824**

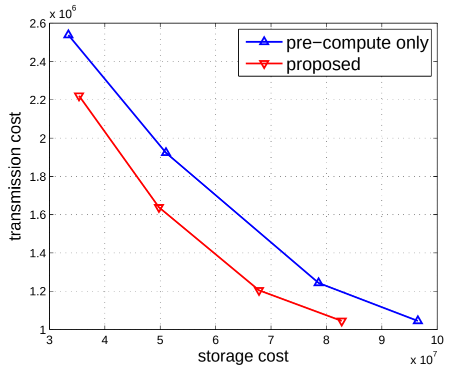

Figure 2. Tradeoff between storage and transmission with $p_{\text{pb}} = 0.5$.

pre-computed and real-time computed P-frames has the lowest total cost (only one typical combination of λ and μ for usual case is shown in Table 1). When the real-time computation is cheap, DSC0-RT is the optimal structure, since it has low transmission rate and no storage cost.

We next consider the optimization of entire multiview video sequence and show the tradeoff between storage and transmission with or without real-time computation. 50 picture groups, each of 4 pictures, from sequence `Kendo` are considered. In this experiment, the steady state probabilities are randomly generated. Note that the computation budget $\bar{C}$ for the proposed scheme is fixed.

The results are plotted in Fig. 2 and 3, showing that the the proposed scheme outperforms the competing scheme without consideration for real-time computation. In particular, the streaming rate can be decreased by approximately 50%, which demonstrates the importance of introducing the real-time computation to IMVS.

6. CONCLUSION

Unlike previous work on interactive multiview video streaming (IMVS) that studied the tradeoff between expected streaming rate and storage cost when optimizing frame structures, we proposed to redesign frame structures with the help of available real-time computation, where frequently used view-switches are handled by pre-computed frames in storage, and infrequently used view-switches are handled by real-time computed frames. In contrast to multi-merge frames previously proposed that offer good tradeoff between transmission rate and storage cost, we proposed a uni-merge frame that is computed in real-time and offers good tradeoff between transmission rate and real-time computation cost. Experimental results show that with real-time computation, the expected streaming rate can be further decreased by 50% compared to pre-encoded structures without real-time computation.

7. REFERENCES

[1] M. Tanimoto, M. P. Tehrani, T. Fujii, and T. Yendo, "Free-viewpoint TV," in *IEEE Signal Processing Magazine*, January 2011, vol. 28, no.1.

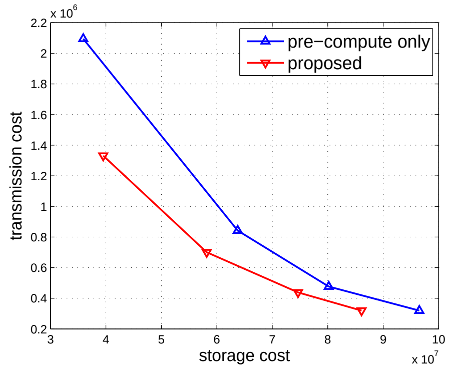

Figure 3. Tradeoff between storage and transmission with $p_{\text{pb}} = 0.9$.

[2] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Efficient prediction structures for multiview video coding," in *IEEE Transactions on Circuits and Systems for Video Technology*, November 2007, vol. 17, no.11, pp. 1461–1473.

[3] G. Cheung, A. Ortega, and N.-M. Cheung, "Interactive streaming of stored multiview video using redundant frame structures," in *IEEE Transactions on Image Processing*, March 2011, vol. 20, no.3, pp. 744–761.

[4] X. Xiu, G. Cheung, and J. Liang, "Delay-cognizant interactive multiview video with free viewpoint synthesis," in *IEEE Transactions on Multimedia*, August 2012, vol. 14, no.4, pp. 1109–1126.

[5] N.-M. Cheung, A. Ortega, and G. Cheung, "Distributed source coding techniques for interactive multiview video streaming," in *27th Picture Coding Symposium*, Chicago, IL, May 2009.

[6] L. Sun, J. Wen, F. Zhang, W. Hu, W. Feng, and S. Yang, "A framework for heuristic scheduling for parallel processing on multicore architecture: A case study with multiview video coding," in *IEEE Trans. Circuits Systems Video Technology*, Nov. 2009, vol. 19, No. 11, pp. 1658–1666.

[7] Y.-C. Li, I-.J. Liao, H.-P. Cheng, and W.-T. Lee, "A cloud computing framework of free view point real-time monitor system working on mobile devices," in *Inter. Symp. Intelligent Sig. Procs. Comm. Sys.*, Chengdu, China, Dec. 2010.

[8] Y. Wu, C. Wu, B. Li, X. Qiu, and F. Lau, "CloudMedia: When cloud on demand meets video on demand," in *Inter. Conf. Distributed Computing Systems*, Minnesota, USA, Jun. 2011, pp. 268–277.

[9] Gene Cheung and Steven McCanne, "A Framework for Computation-Memory Algorithmic Optimization for Signal Processing," *IEEE Transactions on Multimedia*, vol. 5, no. 2, pp. 174–185, Jun. 2003.

[10] V. Srinivasan and G. Varghese, "Fast address lookups using controlled prefix expansion," in *ACM Transactions on Computer Systems*, February 1999, vol. 17, no.1, pp. 1–40.

[11] M. Karczewicz and R. Kurceren, "The SP- and SI-frames design for H.264/AVC," in *IEEE Transactions on Circuits and Systems for Video Technology*, July 2003, vol. 13, no.7, pp. 637–644.