Coding Structure Optimization for Interactive Multiview Streaming in Virtual World Observation

Gene Cheung [#1], Antonio Ortega [*2], Takashi Sakamoto [#3]

[#] *Hewlett-Packard Laboratories Japan*
3-8-13 Takaido-higashi, Suginami-ku, Tokyo, 168-0072, Japan
[1] gene-cs.cheung@hp.com [3] takashi.sakamoto@hp.com

[*] *Signal and Image Processing Institute, University of Southern California*
Los Angeles, CA 90089-2564
[2] ortega@sipi.usc.edu

***Abstract*—While most multiview coding techniques focus on compressing all frames in a multiview video sequence in a rate-distortion optimal manner, in this paper we address the problem of interactive multiview streaming, where we minimize the expected transmission rate of an interactive multiview video stream, where the observer can select the view of the next frame, subject to a storage constraint. We show that gains can be achieved by optimizing the trade-off between overall storage and transmission rate, i.e., by storing a more redundant multiview representation (where some frames are encoded more than once, each time using a different reference frame) it is possible to reduce the overall bandwidth needed for online interactive viewing. We show that our proposed redundant representation can reduce the transmission cost of interactive multiview streaming by up to 65% as compared to a good non-redundant representation for the same storage constraint.**

I. Introduction

Emerging video applications are being developed where multiple views of a scene are captured, encoded and delivered to users, and where new levels of interactivity are added by letting users switch in real-time among multiple view points. Such applications are being considered for both real video data and for computer generated content.

Recent standard activities on multiview video coding (MVC) [1], [2] have studied efficient techniques to compress video sequences captured by multiple cameras exploiting both cross-view and temporal redundancy. Among applications for MVC tools are those where users are allowed to select for playback only a subset of those views or, potentially, virtual views generated from the actual video data captured.

Moreover, increased popularity and realism of computer graphics has led to considering virtual worlds as applications not only for active play but also for observing. Examples of popular virtual worlds include `Second Life` [3] and online games like `Warsow` [4]. Applications are built to enable users to become spectators, so that they can follow the actions of skilled semi-pro players in the virtual worlds. These application have proven popular, to the extent that game tournaments are now broadcasted as network videos at Half-Life TV [5] and other portals. This has led to increase interest in providing network streaming services for 3D world observation beyond active participation, where "observers" do not need access to powerful rendering engines, since virtual world data is compressed into standard-compliant video streams.

Note that while current motivation for viewing video captured from virtual worlds has come mostly from entertainment applications, video captures can also be used in training and education applications in virtual worlds, commonly known as *serious games* [6]. For example, this would allow later viewing of captured training sessions by instructors or students.

An important motivation for the work in this paper comes from the observation that, with this computer generated data, multiview information can be easily captured by slightly modifying the source code and then grabbing frames from frame buffers (as opposed to having multiple physical cameras). Thus, the multiview experience envisioned for video generated from physical cameras can potentially be extended to virtual worlds. In this paper we will consider a generic multiview interactive viewing problem; our experimental results show that our algorithms apply equally well to virtual multiview video data obtained from a game and to multiview video data captured by physical cameras.

We consider multiview streaming scenarios where multiple views are stored as encoded video in a server of finite storage. Our goal is to allow a viewer maximum flexibility in playing back the content, so that a video stream can be requested from the server and the viewpoint can be changed interactively.

Work to date in both MVC and game observing video has focused on capture and compression. For example, the MVC standardization process has concentrated on developing new compression algorithms to encode all frames in the multiview sequence in a rate-distortion optimal manner. For game-generated data, in our previous work [7] we have optimized the video quality of regions of interest (ROI) during encoding of single game view in H.263 using per-pixel depth information, which can be extracted from the depth buffer during frame rendering [8]. We have also developed *community streaming* [9], where a group of game observers can interact visually via overlaid talking heads or personalized avatars in a shared view. Most recently, we have developed fast mode selection algorithms for H.264 video encoding of 3D virtual worlds

978-1-4244-2295-1/08/$25.00 © 2008 IEEE

using per-pixel depth information [10].

Instead, in this paper we focus on how users may interact with multiview content. More specifically we consider the likely scenario where each user only views a subset of available content, e.g., users are allowed to display one frame from one view at a time, so that the interaction with the content can be seen as a view "traversal" over time, where at each time users can switch views or continue playing back the current one. In this context, we consider the problem of minimizing streaming rates, while not exceeding a given constraint on the total storage required to store the multiview dataset.

A key observation in our work is that there is a trade-off between these two requirements. Consider a multiview video dataset: many proposed MVC algorithms extend conventional video coding schemes by combining cross-view predictive coding with existing temporal prediction. Thus, in order to minimize *total* storage, complex dependencies can be created in the data: decoding a specific frame may require having access to multiple other frames that act as predictors, both in the current view and in neighboring views. Note that these predictors will have to be transmitted whether or not the corresponding frames will be displayed. Thus, when the goal is to stream and decode the dataset *partially* only, e.g., at any point in time only one view is being displayed, this minimal storage solution leads to high transmission cost (the total number of frames transmitted exceeds the total number of frames displayed). On the other hand, for a given traversal of views, one can prepare *a priori* a sequence of prediction-coded frames with dependency path that mimics the traversal. This ensures minimal transmission cost, but the required storage would be too large if one were to prepare such predicted frame sequences for all possible view traversals. Our goal is then to find the coding structure that provides the optimal trade-off between storage cost and transmission cost.

We use a simple abstraction of the encoding structure of video sequences, representing them as a set of dependency trees, where a given frame corresponds to a node in the tree and can have both parents (reference frames) and children (frames that use it as a reference). Each of these trees is rooted by an Intra frame. With this abstraction it is clear that decoding a given frame requires having access to all its ascendants in the tree. Thus, the worst case scenario is when different frames corresponding to different views belong to different dependency trees and thus switching would require retrieving a number of frames in the new tree: none of these will be displayed, they will be needed simply to represent the desired frame in the new view.

Our philosophy to tackle this problem is *to allow frames to belong to more than one dependency tree*, i.e., to encode certain frames more than once (each time with a different set of reference frames.) This will increase the flexibility in decoding, but at the cost of increases in overall storage. This is the key objective of our optimization.

Our proposed algorithms do not require development of new encoding algorithms, but instead rely on existing H.264 codecs. We are trying to enable interactive multiview streaming from a standard-compliant H.264 streaming client, where RTSP [11] commands "fast-forward" and "rewind" are mapped to view-switching commands. Note that alternative tools for efficient view switching have been recently proposed based on distributed source coding (DSC) [12], but our primary focus here is that of enabling switching with existing coding techniques. Analysis of this problem in the context of DSC encoding is left for future work. Note also that SP-frames [13] have been proposed to facilitate switching between streams without creating drift. As compared to our approach (where drift is avoided by storing multiple sets of descendants for frames that can be obtained on multiple different predictors) an approach based on SP-frames would require increases in transmission cost (due to the inefficiency of SP-frames relative to P-frames) but require lower overall storage. While the analysis in this paper does not include SP-frames, our framework can be generalized to incorporate SP-frames so the best combination of SP-frames and redundant P-frames can be chosen. This is also left for future work.

To the best of our knowledge, this formulation of interactive multiview streaming, which has practical significance, has not been performed in the literature. We show that our proposed redundant representation can reduce the transmission cost of interactive multiview streaming by up to 65% compared to a good non-redundant representation for the same storage constraint.

The outline of this paper is as follows. We first formally define the optimization problem for interactive multiview streaming in Section II. Given the problem is NP-hard (shown in Appendix A), we derive approximation algorithms that produce locally optimal solutions in Section III. We discuss results in Section IV and conclude in Section V.

II. Problem Formulation

Fig. 1. Example Multiview Screen-shots of `Warsow`.

A. Feasible Search Space

To support interactive multiview streaming, a server compresses, stores and streams individual frames of a multiview video upon client requests. The compressed multiview representation stored at the server can be one of many possibilities in a large feasible space; we first discuss this space in abstract.

Let the number of captured views at a given frame capturing instant be a constant K; i.e., K different views of a scene are simultaneously captured periodically. See Figure 1 for an example of simultaneous frame capture of three views for first-person-shooter game `Warsow`. Let us assume that the first frame-capturing instance of time index 0 is of a single view

$\lceil \frac{K}{2} \rceil$, and there are N frame-capturing instances in total. See Figure 2 for an example.

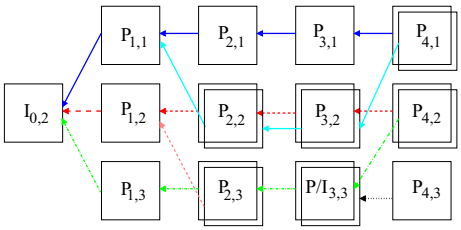


Fig. 2. Example of Representation of multiview video for $K = 3$ and $N = 5$. There are two basic dependency trees with roots at $I_{0,2}$ and $I_{3,3}$.

The server compresses the multiview video into a redundant representation format as follows. An original frame $F^o_{i,j}$ of time index i and view j can be encoded into numerous versions $F_{i,j}$'s (and at least one version) as an I-frame $I_{i,j}$, or a P-frame $P_{i,j}(F)$ motioned-compensated using different references F's. We do not consider B-frames in this formulation. For simplicity, we assume that a P-frame $P_{i,j}(F)$ is only motion-compensated using as reference an encoded frame F of time index $i-1$ and of view between $\max(1, j-1)$ and $\min(K, j+1)$.

All encoded frames of a particular representation can be organized into a set of S *basic dependency trees* $\mathcal{T} = \{T^1, \ldots, T^S\}$ of different root frames. A tree $T^s(I^s_{i,j})$, with unique root I-frame $I^s_{i,j}$, is recursively defined as follows:

$$T(F) = F \cup \{T(F') | F \leftarrow F'\} \tag{1}$$

(1) states that a tree $T(F)$ rooted at a frame F is a frame set composed of F and (sub-)trees stemming from F. A basic dependency tree is a tree with root frame encoded as an I-frame. Using tree set $\mathcal{T}$, we can define a *dependency path* of a frame F, $\mathbf{p}(F)$, as the ordered set of all frames that proceed from the root I-frame to F in the basic dependency tree that F belongs to. More precisely, dependency path $\mathbf{p}(F) = \{F^{\mathbf{p}}_1, \ldots, F^{\mathbf{p}}_{|\mathbf{p}|} = F\}$ is a series of motion-compensated frames, where inside path $\mathbf{p}$ frame $F^{\mathbf{p}}_i$ is motion-compensated using $F^{\mathbf{p}}_{i-1}$ for $i \geq 2$ and $F^{\mathbf{p}}_1$ is an I-frame.

Note that in general, a set of basic dependency trees can have an exponentially large number of nodes as function of original frames NK. For practical purposes, we will assume each original frame $F^o_{i,j}$ cannot be encoded in more than M versions. That means the maximum number of nodes in a set of basic dependency trees is bounded by MNK.

The feasible space for the representation of the multiview video, Θ, can now be formally defined as the set of basic dependency trees $\mathcal{T}$, as described in (1), such that each original frame $F^o_{i,j}$ is encoded no more than M times but at least once, either as an I-frame or as a P-frame using an encoded $F_{i-1,k}$, for $\max(1, j-1) \leq k \leq \min(K, j+1)$. Each chosen tree set $\mathcal{T} \in \Theta$ implies both a storage and transmission cost, which we will discuss next.

B. Tree Set Storage Cost

For I-frames, let $|I_{i,j}|$ denote the byte count of encoding original frame $F^o_{i,j}$ as I-frame. $|I_{i,j}| = \infty$ will denote the case when $F^o_{i,j}$ was not encoded as an I-frame. Similarly, for P-frames, let $|P_{i,j}(F)|$ denote the byte count required to encode $F^o_{i,j}$ as P-frame using frame F for motion compensation. $|P_{i,j}(F)| = \infty$ will denote the case when $F^o_{i,j}$ was not encoded as a P-frame using F for motion compensation. We write the *storage cost* of the representation $\mathcal{T}$, $B(\mathcal{T})$, as:

$$\begin{aligned} B(\mathcal{T}) &= \sum_{T^s \in \mathcal{T}} b(I^s_{i,j}) \\ b(F) &= |F| + \sum_{F' | F \leftarrow F'} b(F') \end{aligned} \tag{2}$$

In words, the storage cost of a tree $T(F)$ rooted at F is the size of root frame $|F|$ plus trees stemming from F.

C. Frame-to-frame Transmission Cost

Suppose after observing an encoded frame $F_{i,j}$, with dependency path $\mathbf{p} = \mathbf{p}(F_{i,j})$, an observer chooses for viewing next frame at time $i+1$ of view k. The server makes a deterministic decision on which encoded version of $F^o_{i+1,k}$ to send to the observer based on $F_{i,j}$ and k as follows. First, if either encoded I-frame $I_{i+1,k}$ or P-frame $P_{i+1,k}(F_{i,j})$ is available, the server can send either one for client to decode and display.

If neither is available, the server finds an *alternative P-frame* $P_{i+1,k}(F')$, $F' \neq F_{i,j}$—a P-frame whose reference frame is not available at the client—with alternative path $\mathbf{q} = \mathbf{p}(F')$, where either: i) paths $\mathbf{p}$ and $\mathbf{q}$ are non-overlapping paths; or, ii) paths $\mathbf{p}$ and $\mathbf{q}$ overlap and first diverge after frame $F^{\mathbf{q}}_d$ of path $\mathbf{q}$. "non-overlapping" here means motion-compensated $F_{i,j}$ and F share no common decoding history, and "overlapping" means $F_{i,j}$ and F share common decoding history up till $F^{\mathbf{q}}_d$. In the first case, the server needs to send all the frames in dependency path $\mathbf{q} = \{F^{\mathbf{q}}_1, \ldots, F^{\mathbf{q}}_{|\mathbf{q}|} = F'\}$ and the P-frame $P_{i+1,k}(F')$ itself for correct decoding of frame with time index $i+1$ and view k; decoder at the observer will of course display only the decoded P-frame $P_{i+1,k}(F')$. In the second case, the server needs to send sub-path $\{F^{\mathbf{q}}_{d+1}, \ldots, F^{\mathbf{q}}_{|\mathbf{q}|} = F'\}$ of path $\mathbf{q}$ and the P-frame $P_{i+1,k}(F')$. Thus the total transmission cost of re-routing dependency path from $\mathbf{p}$ to $\mathbf{q}$, $r(\mathbf{p}, \mathbf{q})$, for each of these two cases is as follows:

$$r(\mathbf{p}, \mathbf{q}) = \begin{cases} |F^{\mathbf{q}}_{d+1}| + \ldots + |F^{\mathbf{q}}_{|\mathbf{q}|}| & \text{if } \mathbf{p}, \mathbf{q} \text{ overlap till } F^{\mathbf{q}}_d \\ |F^{\mathbf{q}}_1| + \ldots + |F^{\mathbf{q}}_{|\mathbf{q}|}| & \text{o.w.} \end{cases} \tag{3}$$

As an example, consider path $\mathbf{p} = \{I_{0,2}, P_{1,2}, P_{2,3}\}$ in Figure 2. At frame $P_{2,3}$ observer requests frame $F_{3,2}$. Server will need to send frame $P^{\mathbf{q}}_{3,2}$ together with frame $P_{2,2}$ of path $\mathbf{q} = \{I_{0,2}, P_{1,2}, P_{2,2}\}$, overlapping $\mathbf{p}$ up to frame $P_{1,2}$.

There can be multiple alternative P-frames $P_{i+1,k}(F')$'s for different references F''s and alternative paths $\mathbf{p}(F')$'s, so the server needs to find one with the lowest transmission cost $\phi(\mathbf{p}, k)$ given dependency path $\mathbf{p}(F_{i,j})$ and desired view k:

$$\phi(\mathbf{p}(F_{i,j}), k) = \min_{F'} \left\{ |P_{i+1,k}(F')| + r(\mathbf{p}(F_{i,j}), \mathbf{p}(F')) \right\} \tag{4}$$

The transmission cost for observer to choose view k after observing encoded frame $F_{i,j}$, $\Phi(F_{i,j},k)$, is then the minimum transmission cost of the possibly available I-frame, P-frame, and alternative P-frame(s):

$$\Phi(F_{i,j},k)=\min\left\{|I_{i+1,k}|,\ |P_{i+1,k}(F_{i,j})|,\ \phi(\mathbf{p}(F_{i,j}),k)\right\} \quad (5)$$

Note that $\Phi(F_{i,j},k)$ in (5) will never return ∞ since there is at least one encoded version of original frame $F^o_{i+1,k}$. Finally, for ease of derivation in later sections, let $\psi(F_{i,j},k)$ be the server-selected compressed version of original frame $F^o_{i+1,k}$ minimizing (5).

D. Optimization Definition

Having defined the storage cost of a tree set $\mathcal{T}$ and transmission cost of switching from observed frame $F_{i,j}$ to some encoded version of the next frame $F^o_{i+1,k}$, we are ready to define formally the optimization problem. Let $C(\mathcal{T})$ denote the expected transmission cost of N-frame interactive multiview streaming given tree set $\mathcal{T}$. After observing an encoded version of original frame $F^o_{i,j}$, we will assume an observer watches the next view k at the next time index $i+1$ with probability $\alpha_{i,j}(k)$, where $\sum_k \alpha_{i,j}(k)=1$. Using derivation of frame-to-frame transmission cost (5), we write $C(\mathcal{T})$ as:

$$\begin{aligned} C(\mathcal{T}) &= |F_{0,\lceil\frac{K}{2}\rceil}| \ + \ c(F_{0,\lceil\frac{K}{2}\rceil}) \quad (6)\\ c(F_{i,j}) &= \sum_{k=\max(1,j-1)}^{\min(K,j+1)} \alpha_{i,j}(k)\left[\Phi(F_{i,j},k)+c(\psi(F_{i,j},k))\right] \end{aligned}$$

(6) can be calculated efficiently in a recursive manner. First, $c(F_{i,j})$ is a sum of at most three terms. Second, $\phi(\mathbf{p}(F_{i,j}),k)$ in $\Phi(F_{i,j},k)$ has at most M references F''s (maximum M versions of $F_{i+1,k}$) to test. For each reference F', rerouting cost $r(\mathbf{p}(F_{i,j}),\mathbf{p}(F'))$ has at most N additions. First time computed $c(F_{i,j})$ is stored in a table so that a future recursive call to $c(F_{i,j})$ can simply return the calculated value. The computation complexity of (6) is therfore $M*N$ times the maximum number of nodes in $\mathcal{T}$, or $O(M^2N^2K)$.

To simplify the calculation of the storage cost, instead of finding exact encoding costs of P-frames for all possible encoded versions of an original frame, we assume $I_{i,j}=r^I_{i,j}$ and $P_{i,j}(F_{i-1,k})=r^P_{i,j}(k)$ for any encoded version $F_{i-1,k}$ of original frame $F^o_{i-1,k}$. The interactive multiview streaming optimization, denoted as IMVS, can now be formalized as follows. Given transition probabilities $\alpha_{i,j}(k)$'s and encoding rates $r^I_{i,j}$'s and $r^P_{i,j}(k)$'s for N frames of multiview video of K views, find the optimal tree set $\mathcal{T}$ in feasible space Θ that minimizes expected transmission cost $C(\mathcal{T})$ subject to a storage constraint $\bar{B}$. Mathematically, we write:

$$\min_{\mathcal{T}\in\Theta} C(\mathcal{T}) \quad \text{s.t.} \ \ B(\mathcal{T})\le\bar{B} \quad (7)$$

We outline an NP-hardness proof in Appendix A for IMVS. Given IMVS is NP-hard, we next derive approximation algorithms that find locally optimal solutions.

III. Approximation Algorithms

Given the computation of expected transmission cost in (6) is fairly expensive itself—though in practice M is likely small—we derive approximation algorithms for IMVS using a computation-efficient greedy approach. As an initial solution $\mathcal{T}^i$, we first find a minimum-storage solution—one that requires minimum storage space for all frames of all views, where each frame of each view is encoded only once. Assuming the size of an I-frame $|I_{i,j}|$ is larger than its P-frame counterpart $|P_{i,j}(F_{i-1,k})|$, it is easy to see that the minimum-storage solution $\mathcal{T}^i$ is I-frame followed by all P-frames. In particular, we can find $\mathcal{T}^i$ mathematically as follows:

$$\begin{aligned} B_{\min}(\mathcal{T}) &= I_{0,\lceil\frac{K}{2}\rceil}+b_{\min}(1) \quad (8)\\ b_{\min}(i) &= \sum_{j=1}^{K}\min_{k=\max(1,j-1)}^{\min(K,j+1)} r^P_{i,j}(k)+U(i<N-1)b_{\min}(i+1) \end{aligned}$$

where $U(\mathtt{c})=1$ if clause $\mathtt{c}$ is true and 0 otherwise. (8) basically finds the smallest P-frame $P_{i,j}(F_{i-1,k})$ for each original frame $F^o_{i,j}$.

Given a minimum-storage solution $\mathcal{T}^i$, our next step is to find locally optimal solution from $\mathcal{T}^i$. This is done iteratively by defining a series of augmentations and selecting among those the one that provides the greater decrease in a chosen cost function. The augmentations we use are:

1) Change a P-frame $P_{i,j}(F_{i-1,k})$ to I-frame $I_{i,j}$.
2) Select a different reference F' for a P-frame $P_{i,j}(F_{i-1,k})$.
3) Add a new I-frame $I_{i,j}$.
4) Add a new P-frame $P_{i,j}(F_{i-1,k})$.

The first two augmentations do not increase the number of representations of a given frame, while each of the next two increases by one (only performed if number of versions of that frame is $<M$). That means the resulting solution will always have at least one representation of each frame of each view. When a new I-frame $I_{i,j}$ is added to "complement" existing P-frame(s), we determine which children of existing P-frame(s) should switch parent to the newly added I-frame. This is done greedily: a child of existing P-frame is switched if by switching, the transmission cost goes down. Similarly optimum parent and children node selections are also performed greedily when adding a new P-frame.

Given the augmentations, we propose two algorithms with two cost functions. First is Lagrangian cost:

$$J(\mathcal{T})=C(\mathcal{T})+\lambda B(\mathcal{T}) \quad (9)$$

where $\lambda\ge 0$ is the Lagrange multiplier. At each iteration we select the augmentation providing the greatest decrease in Lagrangian cost. Algorithm stops when no further cost reductions are possible given λ. Using different λ's, we can trade off storage and transmission bandwidth differently.

Alternatively, at each iteration we select the augmentation of all frames $F_{i,j}$'s in current solution $\mathcal{T}$ such that the ratio of the decrease in transmission cost $\Delta C(\mathcal{T})$ to increase in storage cost $\Delta B(\mathcal{T})$ is the largest. Algorithm stops when the next such beneficial augmentation will exceed storage budget $\bar{B}$. This alternative algorithm has the advantage that it directly targets

a given storage. Note that both approaches are sub-optimal in that they are greedy: choosing the best augmentation at a given iteration does not guarantee that the globally optimal solution is achieved for given storage.

IV. Experimentation

A. Experimental Setup

To collect virtual data for experimentation, we captured two 100-frame multiview sequences of the online game `Warsow` [4] at 10 frames per second (fps) during regular game play. At each frame-capturing moment, in addition to the regular view, two side views rotated 30^o to the left and to the right were captured. Figure 1 shows example screen-shots of such multiview captures. For real video data, we used the `ballroom` sequence from [15] at 25 fps.

Given these raw frames, scaled to QCIF (176×144), we used H.264 JM reference software version 12.4 [14], with quantization parameters for I- and P-frames set constant at 30, to generate encoding rates $r^I_{i,j}$'s and $r^P_{i,j}(k)$'s as follows. For I-frame rates $r^I_{i,j}$'s, we simply encoded all frames of all three views as I-frames and recorded the byte count. For P-frame rates $r^P_{i,j}(k)$'s, we first generated four zigzagged streams z's as follows:

1) $z_{lc} = \{I_{0,0},\ P_{1,1},\ P_{2,0},\ P_{3,1},\ \ldots\}$.
2) $z_{cr} = \{I_{0,1},\ P_{1,2},\ P_{2,1},\ P_{3,2},\ \ldots\}$.
3) $z_{cl} = \{I_{0,1},\ P_{1,0},\ P_{2,1},\ P_{3,0},\ \ldots\}$.
4) $z_{rc} = \{I_{0,2},\ P_{1,1},\ P_{2,2},\ P_{3,1},\ \ldots\}$.

For P-frame rate $r^P_{i,j}(k)$, we simply located the zigzagged stream z that contains the sub-sequence $\{F_{i-1,k}, P_{i,j}\}$ and assigned the coding rate of $P_{i,j}$ in z to $r^P_{i,j}(k)$.

For transition probabilities $\alpha_{i,j}(k)$'s, in the experiments we assumed the following evaluation for simplicity:

$$\alpha_{i,j}(k) = \begin{cases} 1-\alpha & \text{if } j = k \\ \alpha & \text{else if } j = 1 \text{ or } j = K \\ \alpha/2 & \text{o.w.} \end{cases} \tag{10}$$

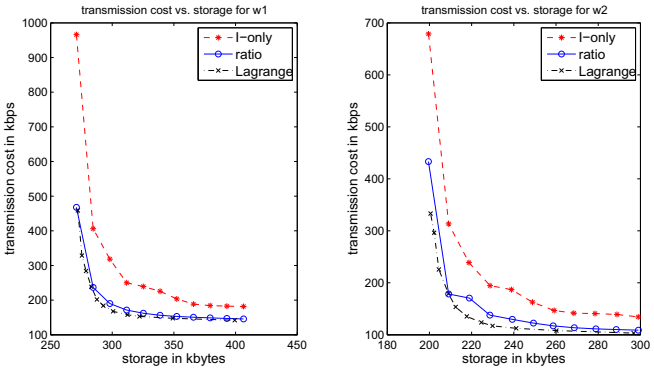
We assume $\alpha = 0.4$ throughout the experiment.

Using I- and all P-frame prediction structure for same-view motion compensation, this resulted in $34.9dB$ and $36.4dB$ in PSNR at bit-rate $74kbps$ and $55kbps$ for the center-view of two `Warsow` sequences `w1` and `w2`, respectively. This rate-distortion disparity indicates that sequence `w1` is much more active than `w2`. Using camera 2, `ballroom` was encoded at $33.47dB$ at $172.4kbps$.

B. Experimental Results

For comparison, we constructed a non-redundant multiview representation we call `I-only`, where frames of each of the K available views are first encoded as I- plus all P-frames independent of other views, then P-frames are converted to I-frames at evenly spaced intervals (at $N/2$, $N/4$, $3N/4$, $N/8$, $3N/8$, etc) at all K views until the storage budget has been expended. `I-only` is non-redundant in that each frame is encoded only once.

Using the two `Warsow` sequences `w1` and `w2`, we applied our two approximation algorithms `ratio` and `Lagrange` discussed in Section III with $M = 5$ and `I-only` to find the minimum expected transmission cost for varying storage budget. Note that both `ratio` and `Lagrange` represent greedy search strategies and so we make no claims of optimality.



a) Trans.-Storage Tradeoff for `w1` b) Trans.-Storage Tradeoff for `w2`

Fig. 3. Tradeoffs of Expected Transmission Cost and Storage Expenditure for `Warsow` Sequences `w1` and `w2`

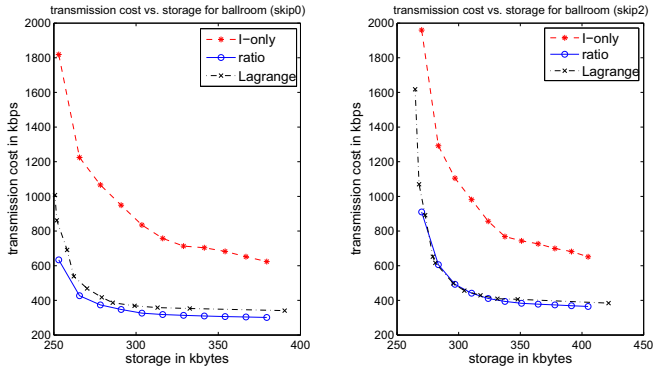
The tradeoffs of storage versus expected transmission cost is shown in Figure 3, where storage $\bar{B}$ ranged from three I-frames above minimum storage required to store all frames to 1.5 times the minimum storage. First, we see in general an inverse proportional relationship between expected transmission cost and storage as we anticipated. Comparing the three schemes, we see that for both `w1` and `w2`, `Lagrange` and `ratio` had lower expected transmission cost than `I-only` for the entire range.

Numerically, `ratio` reduced transmission cost of `I-only` by up to 52% for `w1` and 43% for `w2`, and `Lagrange` performed similarly. Moreover, though the performance gap decreased as storage budget increased, at 1.5 times the minimum storage, `ratio` still outperformed `I-only` by 20% for `w1` and 19% for `w2`. This shows that our proposed redundant representation of multiview video can indeed outperform significantly a good non-redundant representation in terms of transmission cost for a large range of storage budgets.

For the `ballroom` sequence, Figure 4 shows the performance of `I-only`, `ratio` and `Lagrange` when different subsets of capturing cameras were used to form the multiview sequence. We see again in Figure 4 that `ratio` and `Lagrange` outperformed `I-only` for all ranges of storage budget. `Lagrange` performed similarly to `ratio` at most points and slightly worse at some points.

Numerically, `ratio` reduced transmission cost of `I-only` by up to 65% and 55% for camera subset $\{0, 1, 2\}$ and subset $\{0, 3, 6\}$ respectively. More so than the `Warsow` sequences, `ratio` and `Lagrange` continued to outperform `I-only` by a wide margin even at high storage budget: at 1.5 times the minimum storage, `ratio` reduced transmission cost of `I-only` by 52% and 44% for the two camera subsets respectively. We conjecture that this is because `ballroom` was captured using closely spaced cameras, meaning P-frames predicted from neighboring views are significantly smaller than I-frames. Hence even at large storage budget, `I-only`

remained significantly more costly than `ratio`, `Lagrange` utilizing redundant P-frames.



a) Multiview from camera 0, 1, 2 b) Multiview from camera 0, 3, 6

Fig. 4. Tradeoffs of Expected Transmission Cost and Storage Expenditure for `ballroom` Sequence from Different Subsets of Cameras

Note that because the cost functions are different for `Lagrange` and `ratio`, there is no reason to expect them to yield the same solutions. Indeed, in our experiments we observe that the best algorithm choice depends on the specific data being encoded. In future work we plan to further study the optimization process.

V. Conclusion

In this paper, we addressed the problem of interactive multiview streaming, where we minimize the expected transmission rate of an interactive multiview video stream, where the observer can select the view of the next frame, subject to a storage constraint. We showed the problem is NP-hard. We derived approximation algorithms that find well performing locally optimal solutions. Using captured multiview game sequence `Warsow` and real video sequence `ballroom`, we showed in our results that indeed expected transmission cost can be gracefully traded off with storage expenditure.

Appendix

A. NP-Hardness Proof of `IMVS`

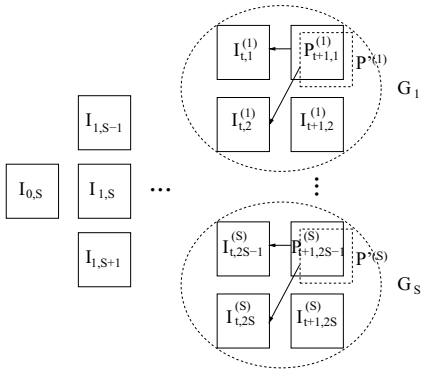


Fig. 5. Construct used for NP-hardness Proof of `IMVS`

For brevity we only outline an NP-hardness proof for optimization `IMVS`. First, it can be easily shown that storage cost $B(\mathcal{T})$ in (3) and transmission cost $C(\mathcal{T})$ in (6) can be calculated in polynomial time, given solution instance $\mathcal{T}$. Hence `IMVS` is obviously in NP.

Second, we show that `IMVS` contains the well-known NP-hard knapsack (`KS`) problem as a special case, so `IMVS` is at least as hard as `KS`, and `IMVS` is NP-hard. We restate `KS` as follows: given set $\mathcal{S}$ of S items, where each item $s \in \mathcal{S}$ is of value $v(s)$ and of weight $w(s)$, find a subset $\mathcal{S}' \subset \mathcal{S}$, such that the total values $\sum_{s\in\mathcal{S}'} v(s)$ is maximized subject to a weight constraint $\sum_{s\in\mathcal{S}'} w(s) \leq W$. For each instance of `KS`, we construct a corresponding instance of `IMVS` as shown in Figure 5. We set small rates $r^I_{i,j}$'s for I-frames such that an optimal solution $\mathcal{T}$ must contain I-frames for first frame $F_{0,S}$ to frames of time index $t-1$. At time indices t and $t+1$, there are $2S$ views, and we set $\alpha_{i,j}$'s such that the probability of entering each of $2S$ views at time index t is $\frac{1}{2S}$.

We construct a group G_s of four frames for each item $s \in \mathcal{S}$ as follows. We set the sizes of the top-left I-frame and top-right P-frame predicting from top-left I-frame to be $u(s)+1$ and $\epsilon \approx 0$ respectively. We set the size of an additional top-right P-frame predicting from bottom-left I-frame to be $u(s)$, and the size of an additional top-right I-frame to be so large that it is not selectable. Finally, we set the transition probability from bottom-left I-frame to top-right frame to be $\frac{v(s)}{v(s_{\max})}$, where $s_{\max}$ is the item in $\mathcal{S}$ with maximum value. The extra storage cost and reduction in expected transmission cost for adding top-right P-frame are therefore $u(s)$ and $\frac{v(s)}{2Sv(s_{\max})}$. If extra storage available for these additional P-frames, beyond necessary I-frames, is W, then the optimal set of additional P-frames selected in `IMVS` corresponds to the optimal subset $\mathcal{S}'$ in `KS`.

References

[1] 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*, vol. 17, no.11, November 2007, pp. 1461–1473.

[2] M. Flierl, A. Mavlankar, and B. Girod, "Motion and disparity compensated coding for multiview video," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no.11, November 2007, pp. 1474–1484.

[3] "Second Life: Official site of the 3D online virtual world," http://secondlife.com.

[4] "Warsow: a fast paced first person shooter game," http://www.warsow.net.

[5] "Half-Life TV," http://www.hltv.org.

[6] "Serious Games Initiative," http://www.seriousgames.org.

[7] G. Cheung, T. Sakamoto, and W.-T. Tan, "Graphics-to-video encoding for 3G mobile game viewer multicast using depth values," in *IEEE International Conference on Image Processing*, Singapore, October 2004.

[8] T. Akenine-Moller and E. Haines, *Real-time Rendering*. AK Peters, 2002.

[9] G. Cheung, W.-T. Tan, B. Shen, and A. Ortega, "ECHO: A community video streaming system with interactive visual overlays," in *IS&T/SPIE 15th Aunnual Multimedia Computing and Networking (MMCN'08)*, San Jose, CA, January 2008.

[10] G. Cheung, A. Ortega, and T. Sakamoto, "Fast H.264 mode selection using depth information for distributed game viewing," in *IS&T/SPIE Visual Communications and Image Processing (VCIP'08)*, San Jose, CA, January 2008.

[11] H. Schulzrine, A. Rao, and R. Lanphier, "Real time streaming protcol (RTSP)," April 1998, IETF RFC 2326.

[12] N. Cheung and A. Ortega, "Distributed source coding application to low-delay free viewpoint switching in multiview video compression," in *Proc. of Picture Coding Symposium, PCS'07*, Lisbon, Portugal, Nov. 2007.

[13] 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*, vol. 13, no.7, July 2003.

[14] "The TML project web-page and archive," http://kbc.cs.tu-berlin.de/ stewe/vceg/.

[15] "Test sequences at MERL," ftp://ftp.merl.com/pub/avetro/mvc-testseq.