GENERATION OF REDUNDANT FRAME STRUCTURE FOR INTERACTIVE MULTIVIEW STREAMING

Gene Cheung

Hewlett-Packard Laboratories Japan

Antonio Ortega, Ngai-Man Cheung

University of Southern California

ABSTRACT

While multiview video coding focuses on the rate-distortion performance of compressing all frames of all views, we address the problem of designing a frame structure to enable interactive multiview streaming, where clients can interactively switch views during video playback. Thus, as a client is playing back successive frames (in time) for a given view, it can send a request to the server to switch to a different view while continuing uninterrupted temporal playback. Noting that standard tools for random access (i.e., I-frame insertion) can be inefficient for this application, we propose a technique where redundant representations of some frames can be stored to facilitate view switching. We first present an optimal algorithm with exponential running time that generates such a frame structure so that the expected transmission rate is optimally traded off with total storage. We then present methods to reduce the algorithm complexity for practical use. We show in our experiments that we can generate redundant frame structures offering a range of tradeoff points with transmission and storage, including ones that outperform simple I-frame insertion structures by up to 48% in terms of bandwidth efficiency for similar storage costs.

Index Terms— Multiview Video, Video Streaming, Interaction, Optimization Methods

1. INTRODUCTION

Multiview video consists of sequences of spatially related pictures captured simultaneously and periodically by multiple closely spaced cameras. Applications of multiview video include free-viewpoint TV [1], 3D display, immersive video conferencing, etc. Much of the previous research on multiview video focuses on compression: to design advanced motion- and disparity-compensated coding techniques to encode all frames and all views of a multiview sequence in a rate-distortion optimal manner [2, 3, 4].

In this paper, we focus instead on the problem of *interactive multiview video streaming* (IMVS). In this problem, after one pre-encoded representation of a multiview sequence is chosen and stored at the server, streaming clients *interactively* request desired views for successive video frames in time. Each client watches and requests one single view at a time out of possibly many available views, meaning that the requested data corresponds to only a small subset out of a large set of available multiview data at the server. The encoding is done once at the server for a possibly large group of clients, each of which can navigate the content by playing it back (in time) while switching views, thus resulting in a different traversal of views across time for each user. Our goal is to provide a desired level of view interactivity with minimum expected transmission bandwidth cost. The extent of view interactivity is determined by the *view switching period* M, i.e., view switching can only take place at multiples of M frames.

A natural approach to enable this kind of interactive view switching is to make use of standard random access tools, i.e., making every M-th frame (in all views) an I-frame. Our work is based on the observation that *random access and view switching are fundamentally different functionalities*, and thus efficient tools for one problem may not provide the best solution for the other. For random access to a frame, one can make no assumptions about which frames are available at the decoder; independently coded I-frames are therefore well suited for this purpose. View switching, on the other hand, arises when temporal playback is not interrupted, i.e., successive frames are displayed, but one wishes to switch point of view. The key difference with respect to random access is that the decoder has access to some of the frames immediately preceding the requested frame (albeit from a different view) in time. Thus, since consecutive frames in different views tend to be correlated, using an I-frame for switching can be inefficient in terms of bandwidth.

The main focus of our work is then to study alternatives for view switching that are more bandwidth-efficient than simple I-frame insertions. Note that our proposed tools *do not* support random access, and thus we are not advocating using these tools *instead* of random access tools such as periodic I-frames insertion. Rather, we propose to consider view switching and random access as two explicitly different functionalities, supported with different tools. It will then be up to the system designer to select the appropriate setting for a given application. For example, one may select a parameter M for view switching and separately allow random access at every M'-th frame, where typically $M \ll M'$.

Since in the case of view switching we know that only one of a *few* previous frames could have been decoded, it is

possible to use specific source coding tools such as redundant P-frames [5] or distributed source coded frames [6, 7]. These tools can lead to lower bandwidth costs (e.g., as compared to inserting I-frames) but can require additional storage (e.g., the total storage required for insertion of multiple P-frame representations).

Thus, the design of an optimal multiview representation to permit interactive view switching involves a tradeoff between the expected transmission rate and the storage space required to store the representation. For simplicity consider the $M = 1$ case, i.e., we require the ability to switch views at any time. One extreme case would be to encode all frames of all views as I-frames, so that the server can simply send the requested I-frame with no concern for inter-frame dependencies. As an alternative, we can allow each picture to be encoded and stored more than once (leading to an overall increase in storage), and then use a starting I-frame plus successive P-frames to encode every possible frame traversal in time by the client. (Encoding different versions of a frame depending on the decoding path is required to eliminate decoding drift.) While this results in minimum transmission cost, the storage required is prohibitive.

Clearly, more practical multiview representations lie between these two extremes that optimally trade off transmission and storage costs. In this paper, we develop an optimization algorithm with exponential running time that generates an optimal frame structure trading off these quantities. We then present methods to reduce the algorithm complexity for practical use. In our experiments with several multiview video datasets, we show that our algorithm can offer a range of tradeoff points between transmission and storage costs, and that the generated frame structures outperform the simple I-frame insertion approach. In particular, we show that in some cases our algorithm generates frame structures reducing expected transmission rate by up to 48% compared to I-frame insertion approach, for similar storage costs.

Note that though we focus exclusively on the use of I- and P-frames in designing our multiview representation in this paper, other coding tools such as SP-frames [8] and distributed source coding (DSC) [6] can also be effective for IMVS. Investigation of these tools for IMVS is currently underway [7] and will be integrated into our optimization framework as future work.

The outline of the paper is as follows. We first review related work in Section 2. We then overview our IMVS system and models in Section 3. We formulate the problem of optimally generating redundant frame structure for IMVS in Section 4 and present our algorithm in Section 5. We discuss our experiments and conclude in Section 6 and 7, respectively.

2. RELATED WORK

As mentioned, much of the previous research in multiview video has focused on efficient compression of all frames of all views using motion- and disparity-compensated techniques [2, 3, 4]. As in standard video, I-frames can be periodically inserted, say every M'-frame interval, to permit some level of random access. Consider as an example the frame structure proposed in [2] and shown in Fig. 1, where every M'-th frame becomes a "key frame", providing temporal random access. In order to facilitate interactive view switching every M frames, the structure in Fig. 1 could be generated with $M' = M$. Clearly, for small M this leads to high bandwidth usage, which is not desirable.

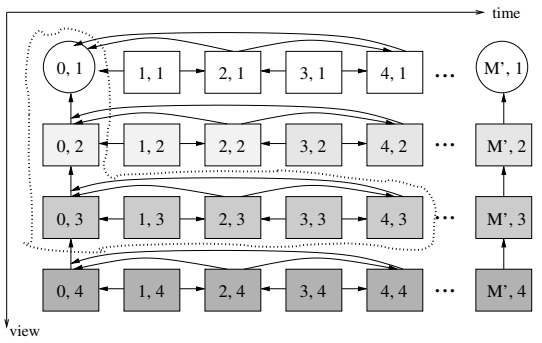


Fig. 1. A 4-view, 5-instant Example of Multiview Frame Structure using Inter-view Prediction for Key Frames in [2].

An alternative strategy is to select a compression-optimized frame structure with $M' \gg M$, and send to the decoder all necessary frames for a specific view switching request (all frames needed to reconstruct the requested frame). For example, in Fig. 1, in order to switch from frame (3,3) to frame (4,2), a server would send frames (1,2) through (4,2) to the decoder, but only frame (4,2) would be displayed. We call this strategy *rerouting*. Rerouting causes an increase in transmission rate during a view switch. This is particularly problematic when using structures optimized for coding efficiency only, such as Fig. 1, since the number of frames that have to be transmitted but are not displayed is potentially large.

In contrast, in this paper we explicitly study the optimal design of frame structures when limited rerouting is permitted. More specifically, our formulation seeks to minimize the expected transmission rate during an IMVS session for given desired view interactivity and limited rerouting, at the expense of a modest and controlled increase in storage. Given the rate of increase in storage byte per dollar continues to outpace bandwidth byte per dollar, offering improved interactivity to users using more storage is a sensible endeavor.

Study of the conflicting requirements of interactivity and compression is not new and has been considered in the context of light fields [9] and virtual walkthroughs [10, 11], where Intra- and Inter-coding are used to provide the said tradeoffs given specified interaction models. Our IMVS work differs in that we *allow multiple representations of a single original picture* (at the expense of increased in storage), so that multi-

ple decoding paths demanding the same view frame can each have their own versions of inter-coded frames, without resorting to a more transmission-expensive intra-coded frame.

We formally posed the IMVS problem as a combinatorial optimization in our earlier work [5], proved its NP-hardness, and provided two heuristics-based algorithms to find good frame structures without enforcing rerouting limits for IMVS. This paper is a more thorough and analytical treatment of the same problem with rerouting limits. We have also developed two novel DSC techniques [7] for IMVS. Integration of these tools into our framework is left for future work.

We note that the IMVS optimization to be defined in Section 4 demands optimal constructions of *both* a coding structure *and* a transmission schedule for the structure—there lies the intrinsic difficulty of the IMVS problem. IMVS fundamentally differs from previous scheduling works [12, 13, 14] where a fixed pre-encoded structure is first defined based on one criteria (e.g. compression efficiency), then an optimal schedule is sought based on another criteria (e.g. expected distortion at the streaming client) afterwards. A notable exception is [15], where P-frames' reference pictures (and hence inter-dependencies) and their transmission schedules are searched simultaneously within a short optimization window for single-view video. We differ in that we focus on redundant representation of multiview video.

3. SYSTEM & INTERACTION MODELS

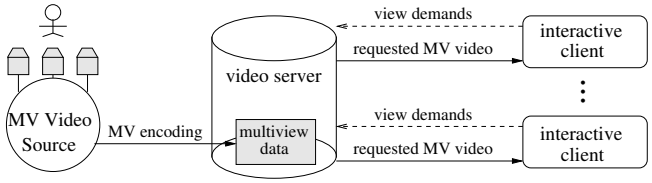


Fig. 2. Interactive Multiview Video Streaming System

3.1. System Model

The system model we consider for our IMVS problem is shown in Fig. 2. A *Multiview Video Source* simultaneously captures multiple pictures of different views at regular intervals. An example of a multiview sequence of two views across four time instants is shown in Fig. 3(a). A *Video Server* sequentially grabs the captured uncompressed pictures from Multiview Video Source and encodes them either offline or online. For example, in the online case, the encoder could operate on non-overlapping windows of M' frames. After selecting an optimized frame structure $\mathcal{T}$, the Video Server stores a single version of the sequence, with which the server can serve multiple streaming clients. (An alternative approach of live encoding a path traversal tailor-made for each streaming client's interactivity is computationally prohibitive if the number of clients is large.) Once a frame window of M' time instants has been encoded, interactive clients can begin requesting views of that window of consecutive time instants, so that even in a live streaming scenario, clients only experience a delay of M' frames from the original source.

3.2. View Interaction Model

In what follows, we will use the term *frame* to denote a specific coded version of a picture and use the term *picture* for the corresponding original captured image. Thus, our system will have redundant storage, in the sense that there may be multiple frames representing a given picture. We assume a view interaction model where, upon watching any decoded version of the picture $F^o_{i,j}$, corresponding to time instant i and view j, an interactive client will request a coded version of picture $F^o_{i+1,k}$ of view k and *next* time instant $i+1$, where view k is between $j-1$ and $j+1$, with *view transition probability* $\alpha_{i,j}(k)$; we call this interactivity *forward view switching*[1]. Another possible interactivity for multiview video is to freeze video in time and switch view (*static view switching*); we conjecture that this interactivity can be efficiently supported by novel usage of DSC [7] and is left for future work.

Note that a significant difference between our setting and that of virtual walkthroughs [10, 11] is that in the latter case the user is free to explore a static scene in all directions, while here we play forward in time with only limited switching possibilities (i.e., among neighboring views).

Though our interactive model presumes a client's desire to switch view at single-frame level ($M = 1$), our model encompasses the more general case of a view switching period $M \neq 1$. In the more general case, a "frame" $F_{i,j}$ in our model can represent M consecutive frames of view j (a carefully chosen I- or P-frame determined by our optimization followed by $M-1$ consecutive P-frames of the same view).

4. PROBLEM FORMULATION

We now formulate our IMVS problem as a combinatorial optimization problem. We first present necessary definitions in Section 4.1. We then define IMVS formally in Section 4.2.

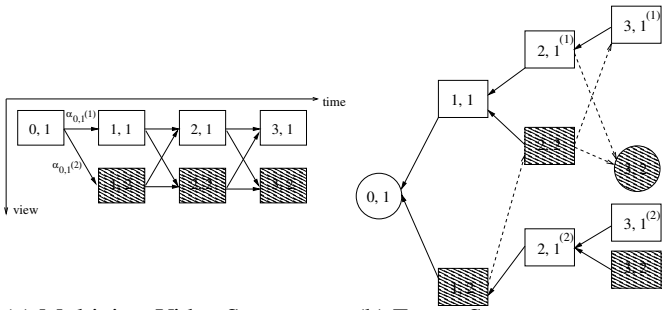
4.1. Definitions

4.1.1. Redundant Frame Structure

Suppose we are given a multiview sequence of K views and N time switching instants[2], and corresponding view transition probabilities $\alpha_{i,j}(k)$'s as discussed in Section 3.2. We assume video starts with a single view K^o, and frame $F_{i,j}$ can

[1] All video streaming systems today offer forward playback of video, hence forward view switching is a natural extension.

[2] Given random access and view switching intervals M' and M, respectively, where $M' > M$, we have $N = \left\lfloor \frac{M'}{M} \right\rfloor - 1$.



(a) Multiview Video Sequence (b) Frame Structure

Fig. 3. Example of Redundant Frame Structure. I- and P-frames are drawn as circles and boxes, respectively. A solid edge from $F_{i+1,k}$ to $F_{i,j}$ in (b) means a P-frame $F_{i+1,k}$ is predictively coded using reference frame $F_{i,j}$. A dotted edge from $F_{i,j}$ to $F_{i+1,k}$ in (b) means a schedule $\mathbf{G}$ dictates transition to $F_{i+1,k}$ if view k is requested after viewing $F_{i,j}$.

only transition to neighboring views $F_{i+1,k}$, $\max(1, j-1) \leq k \leq \min(K, j+1)$. Fig. 3(a) shows an example multiview sequence where $K = 2$, $N = 3$ and $K^o = 1$.

Given a multiview sequence, one can construct a *redundant frame structure* $\mathcal{T}$ comprised of coded I- and P-frames, denoted as $I_{i,j}$'s and $P_{i,j}$'s, to represent the sequence and enable IMVS. Here a P-frame $P_{i,j}$ is a predictively coded frame using a selected $F_{i-1,k}$ as predictor. Note that we do not specify whether the prediction is motion-based, disparity-based or both; our framework aims only to capture the dependencies among frames and not the particular encoding tool used. A structure representing the example multiview sequence in Fig. 3(a) is shown in Fig. 3(b).

A frame structure $\mathcal{T}$ forms a set of dependency trees with I-frames as root nodes. In Fig. 3(b), there are two root nodes, $I_{0,1}$ and $I_{3,2}$. $\mathcal{T}$ is redundant in that an original picture $F^o_{i,j}$ can be represented by multiple frames, i.e., different $F_{i,j}$'s. In Fig. 3(b), $F^o_{2,1}$ is represented by two P-frames $P^{(1)}_{2,1}$ and $P^{(2)}_{2,1}$. Multiple $F_{i,j}$'s are used to avoid coding drift.

4.1.2. Frame-to-frame Schedule

Associated with a frame structure $\mathcal{T}$ is a *frame-to-frame schedule* $\mathbf{G}$, which determines which frame $F_{i+1,k}$ should be sent by the streaming server, given that the viewer has just observed $F_{i,j}$ and requested view k. We denote a scheduled frame-to-frame transition as $F_{i,j} \overset{\mathbf{G}}{\Rightarrow} F_{i+1,k}$. A *feasible schedule* is one where each possible requested view k from a frame $F_{i,j}$ is mapped to a frame $F_{i+1,k}$ satisfying the request. If there exists a P-frame $P_{i+1,k}$ predicted from $F_{i,j}$, then schedule $\mathbf{G}$ will instruct server to send $P_{i+1,k}$ since it is for this transition that $P_{i+1,k}$ was constructed in $\mathcal{T}$.

If $P_{i+1,k}$ predicted from $F_{i,j}$ does not exist, then schedule $\mathbf{G}$ identifies either an I-frame $I_{i+1,k}$ or an *alternative P-frame* $P_{i+1,k}$, one whose predictor is not $F_{i,j}$, to send. For example, in Fig. 3(b), if viewer requests view 1 after observing $P_{2,2}$, schedule $\mathbf{G}$ will instruct server to send $P^{(1)}_{3,1}$. For alternative P-frames, *rerouting costs* are incurred; in this example, for the viewer to correctly decode $P^{(1)}_{3,1}$ after observing $P_{2,2}$, it will be necessary to send $P^{(1)}_{2,1}$, which will not have to be displayed.

Given a fixed frame structure $\mathcal{T}$ and view transition probabilities $\alpha_{i,j}(k)$'s, an optimal schedule $\mathbf{G}^*$ using $\mathcal{T}$ that minimizes the expected transmission cost can be derived. We discuss this towards the end of this section.

4.1.3. Storage Cost

For a given frame structure $\mathcal{T}$, we can define the corresponding *storage cost*, $B(\mathcal{T})$, by simply adding the storage required by all encoded frames in $\mathcal{T}$:

$$B(\mathcal{T}) = \sum_{F_{i,j} \in \mathcal{T}} |F_{i,j}|, \tag{1}$$

where $|F_{i,j}|$ denotes the storage required by frame $F_{i,j}$. For I-frames, the rate only depends on the frame itself (for a given quantization choice) and so we denote $|I_{i,j}| = r^I_{i,j}$. In contrast, the rate for P-frames depends also on the frame used for prediction. Assuming $P_{i,j}$ is encoded using as a predictor $F_{i-1,k}$, the corresponding rate will be $|P_{i,j}| = r^P_{i,j}(k)$, that is, we assume that $|P_{i,j}|$ depends only on the view of the predictor $F_{i-1,k}$ but not on other characteristics of $F_{i-1,k}$ such as size $|F_{i-1,k}|$. For example, we expect that a more accurate prediction can be obtained if $j = k$, and so in general $r^P_{i,j}(j) \leq r^P_{i,j}(k)$, for $k \neq j$. We will discuss how $r^I_{i,j}$'s and $r^P_{i,j}(k)$'s are generated incrementally in Section 6.

4.1.4. Transmission Cost

Given a structure $\mathcal{T}$, we can define a corresponding transmission cost for $\mathcal{T}$ as follows. Any feasible frame-to-frame schedule $\mathbf{G}$ associated with $\mathcal{T}$, together with frame transition probabilities $\alpha_{i,j}(k)$'s, leads to an expected transmission cost for $\mathcal{T}$; we define $C(\mathcal{T})$ for given $\mathcal{T}$ to be the *minimum* expected transmission cost possible for *all* feasible schedules $\mathbf{G}$'s using $\mathcal{T}$. The optimal schedule $\mathbf{G}^*$ given $\mathcal{T}$ can be found using dynamic programming, which we discuss next.

The minimum expected transmission cost $C(\mathcal{T})$ is the sum of the size of the starting I-frame $|I_{0,K^o}|$ and a *recursive transmission cost* $c(I_{0,K^o})$, where $c(F_{i,j})$ is the minimum expected future transmission cost given that a client has just viewed frame $F_{i,j}$. $c(F_{i,j})$ in turn can be written as a weighted sum of *recursive transition costs* $\Phi(F_{i,j}, k)$'s of possible transitions to other views k's in time instant $i+1$:

$$\begin{aligned} C(\mathcal{T}) &= r^I_{0,K^o} + c(I_{0,K^o}) \\ c(F_{i,j}) &= \sum_{k=\max(1,j-1)}^{\min(K,j+1)} \alpha_{i,j}(k)\Phi(F_{i,j}, k) \end{aligned} \tag{2}$$

The recursive transition cost $\Phi(F_{i,j}, k)$ is the smallest possible expected transmission cost from frame $F_{i,j}$ to any suitable frame $F_{i+1,k}$ and beyond; optimal schedule $\mathbf{G}^*$ records this optimal selection of $F_{i+1,k}$ given $F_{i,j}$ to instruct server during IMVS. For given $F_{i+1,k}$, it is a sum of *rerouting cost*, $\phi(F_{i,j}, F_{i+1,k})$, size of $F_{i+1,k}$, and subsequent recursive transmission cost $c(F_{i+1,k})$ after transition:

$$\Phi(F_{i,j}, k) = \min_{F_{i+1,k}} \{\phi(F_{i,j}, F_{i+1,k}) + |F_{i+1,k}| + c(F_{i+1,k})\} \tag{3}$$

The rerouting cost $\phi(F_{i,j}, F_{i+1,k})$ is zero if $F_{i+1,k}$ is an I-frame or a P-frame predictively coded using $F_{i,j}$. Otherwise, $F_{i+1,k}$ is an alternative P-frame, and $\phi(F_{i,j}, F_{i+1,k})$ is the cost of sending enough additional frames for viewer to correctly decode $F_{i+1,k}$. Given that the viewer has just observed $F_{i,j}$, we can assume viewer's decoder buffer has all frames along dependency path from $F_{i,j}$ to root I-frame in $\mathcal{T}$; we denote such path as $w(F_{i,j})$. Let $r(F_{i,j}, F_{i+1,k})$ be the sub-path necessary to decode $F_{i+1,k}$ given frames on path $w(F_{i,j})$ are available at decoder. We can now write $\phi(F_{i,j}, F_{i+1,k})$ as:

$$\phi(F_{i,j}, F_{i+1,k}) = \begin{cases} 0 & \text{if } F_{i+1,k} \text{ is I-frame} \\ 0 & \text{if } F_{i+1,k} \rightarrow F_{i,j} \\ |r(F_{i,j}, F_{i+1,k})| & \text{o.w.} \end{cases} \tag{4}$$

$|r(F_{i,j}, F_{i+1,k})|$ can be determined in one of two ways: i) if paths $w(F_{i,j})$ and $w(F_{i+1,k})$ overlap from the same root I-frame up to a diverging frame $F_{m,n}$ (i.e., $F_{i,j}$ and $F_{i+1,k}$ are in the same dependency tree), then $r(F_{i,j}, F_{i+1,k})$ is the sub-path from $F_{m,n}$ to $F_{i+1,k}$, excluding sub-path end nodes $F_{m,n}$ and $F_{i+1,k}$; or, ii) if paths $w(F_{i,j})$ and $w(F_{i+1,k})$ do not overlap, then $r(F_{i,j}, F_{i+1,k})$ is the entire path $w(F_{i+1,k})$, excluding $F_{i+1,k}$.

Continuing our earlier example, if a viewer requests view 1 after viewing $P_{2,2}$, $P_{2,2}$ can transition to $P_{3,1}^{(1)}$ with sub-path $r(P_{2,2}, P_{3,1}^{(1)}) = \{P_{2,1}^{(1)}\}$, or transition to $P_{3,1}^{(2)}$ with sub-path $r(P_{2,2}, P_{3,1}^{(2)}) = \{P_{1,2}, P_{2,1}^{(2)}\}$. In this case, $\Phi(F_{2,2}, 1)$ selects the first option with rerouting cost $\phi(P_{2,2}, P_{3,1}^{(1)}) = r_{2,1}^{P}(1)$.

4.1.5. Reroute Limit

In addition, for practical purposes we can restrict the number of frames required for rerouting $||\ r(F_{i,j}, F_{i+1,k})\ ||$—*frame reroute limit*—to be no larger than R frames:

$$||\ r(F_{i,j}, F_{i+1,k})\ || \leq R \tag{5}$$

R is the number of overhead frames a decoder must decode in addition to the frame being displayed, and hence clearly a small R is desired. If a particular reroute $r(F_{i,j}, F_{i+1,k})$ exceeds R, it will return ∞ to signal violation of frame reroute limit to $\phi(F_{i,j}, F_{i+1,k})$.

It is now easy to see now that $c(I_{0,K^o})$ can be efficiently evaluated using *dynamic programming* (DP): each time a recursive call $c(F_{i,j})$ is made, the solution is stored (also known as *memoization* [16]), so that subsequent calls to $c(F_{i,j})$ can simply return the previously computed value. The sequence of frame-to-frame transitions in (3) that minimizes $c(I_{0,K^o})$ is the optimal schedule $\mathbf{G}^*$ for structure $\mathcal{T}$.

4.2. Optimization Problem Defined

We can now formally define the optimal generation of redundant frame structure for IMVS as a combinatorial optimization problem: find a structure $\mathcal{T}$ in feasible space[3] Θ that possesses the smallest possible minimum expected transmission cost $C(\mathcal{T})$ while a storage constraint $\bar{B}$ is observed. We denote this optimization problem as IMVS*:

$$\min_{\mathcal{T} \in \Theta} C(\mathcal{T}) \text{ s.t. } B(\mathcal{T}) \leq \bar{B} \tag{6}$$

Though (6) slightly differs from the definition in [5], a similar proof to [5] can be easily constructed to show that IMVS* is NP-hard. We see why the optimization (6) is difficult: the computation of minimum expected transmission cost $C(\mathcal{T})$ of a given structure $\mathcal{T}$ involves finding the optimal schedule $\mathbf{G}^*$, where the best frame-to-frame transition from $F_{i,j}$ to a selected $F_{i+1,k}$ is determined only after examining future transitions from $F_{i+1,k}$'s onwards to the end of structure. This makes the problem tightly coupled, and hence it is difficult to devise "divide-and-conquer" type strategies to first divide and then combine locally optimal structures (and associated schedules) to form globally optimal ones.

Given the computational difficulty of (6), we focus next on solving the corresponding Lagrangian optimization for given Lagrange multiplier λ instead:

$$\min_{\mathcal{T} \in \Theta} J(\mathcal{T}) = C(\mathcal{T}) + \lambda B(\mathcal{T}) \tag{7}$$

4.2.1. Alternative Expression for Lagrangian Cost

For ease of discussion in Section 5, the Lagrangian cost of structure $\mathcal{T}$ in (7) will be written in terms of *display probabilities* $q(F_{i,j})$'s—the probabilities that frames $F_{i,j}$'s are sent by server to be displayed at the viewer. We can compute $q(F_{i,j})$'s from front to back as follows, given the view transition probabilities, $\alpha_{i,j}(k)$. Given optimal schedule $\mathbf{G}^*$, we know the frame $F_{i+1,k}$ into which frame $F_{i,j}$ will transition, denoted as $F_{i,j} \overset{\mathbf{G}^*}{\Rightarrow} F_{i+1,k}$, if viewer selects view k after $F_{i,j}$. $q(F_{i,j})$ can be computed iteratively from front of the structure to the back as:

[3]The feasible space is the set of structures that enable all permissible transitions from frame $F_{i,j}$ to frame $F_{i+1,k}$, where transitions have been constrained depending on desirable application characteristics, e.g., j and k may be constrained to be neighboring views.

$$q(F_{0,K^o}) = 1 \quad (8)$$
$$q(F_{i+1,k}) = \sum_{F_{i,j} \overset{\mathbf{G}^*}{\Rightarrow} F_{i+1,k}} q(F_{i,j})\alpha_{i,j}(k)$$

Given the computed $q(F_{i,j})$'s, the Lagrangian cost can be written as a sum of frame display Lagrangian costs and rerouting costs:

$$J(\mathcal{T}) = \sum_{F_{i,j} \in \mathcal{T}} (q(F_{i,j}) + \lambda)\,|F_{i,j}| + \quad (9)$$
$$+ \sum_{F_{i-1,k} \overset{\mathbf{G}^*}{\Rightarrow} F_{i,j} | F_{i,j} \in \mathcal{T}} q(F_{i-1,k})\alpha_{i-1,k}(j)\ \phi(F_{i-1,k}, F_{i,j})$$

To impart intuition in (9), consider the case when there is no rerouting. A picture $F^o_{i,j}$ can be represented by a large number X of P-frames, each having a small transmission cost $q(P^{(x)}_{i,j})|P^{(x)}_{i,j}|$, but together constitute large storage $\sum_x |P^{(x)}_{i,j}|$. When λ is small, the penalty $\lambda \sum_x |P^{(x)}_{i,j}|$ is negligible and multiple P-frames are attractive. When λ is large, the penalty of large storage becomes costly, and a single I-frame $I_{i,j}$ representing picture $F^o_{i,j}$ with larger transmission cost $q(I_{i,j})|I_{i,j}|$ but smaller storage $|I_{i,j}|$ is preferable.

5. ALGORITHM DEVELOPMENT

We now develop algorithms to generate frame structure for IMVS*. We first focus on the simpler but nevertheless useful case when frame reroute limit $R = 0$. We then generalize the algorithm to the case when $R \geq 1$.

5.1. IMVS* Algorithm: $R = 0$

Setting R to 0 in (5) means there is no rerouting, and a frame $F_{i,j}$ must transition to either an I-frame $I_{i+1,k}$ or a P-frame $P_{i+1,k}$ predictively coded from $F_{i,j}$. This case is appropriate for thin client devices that cannot first decode multiple frames before displaying a single frame. We first define an optimal algorithm for IMVS* with exponential running time in Section 5.1.1. We then discuss methods to reduce its complexity for practical use in Section 5.1.2.

5.1.1. Optimal Algorithm

We first overview the algorithm to provide intuition. We exhaustively search for an optimal structure *and* associated schedule simultaneously one instant at a time for all instants from front to back; we call a structure $\mathcal{T}_i(\mathbf{G}_i)$ constructed up to instant i, with schedule $\mathbf{G}_i$ attached, a *scheduled structure*. For a given scheduled structure $\mathcal{T}_{i-1}(\mathbf{G}_{i-1})$ built up to instant $i-1$, each frame $F_{i-1,k}$ in $\mathcal{T}_{i-1}$ has a display probability $q(F_{i-1,k})$ computable using (8).

For $R = 0$, each frame $F_{i-1,k}$ will transition to view j with probability $q(F_{i-1,k})\alpha_{i-1,k}(j)$ either via a P-frame $P_{i,j}(k)$ predicting from $F_{i-1,k}$, or via an I-frame $I_{i,j}$, each with different local Lagrangian costs at instant i. An optimal scheduled structure $\mathcal{T}_i(\mathbf{G}_i)$ at instant i given $\mathcal{T}_{i-1}(\mathbf{G}_{i-1})$ has the smallest sum of: i) local Lagrangian costs at instant i, and ii) future Lagrangian costs stemming from $\mathcal{T}_i(\mathbf{G}_i)$.

We define the minimum Lagrangian cost $L_i(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}))$ from instant i onwards, given scheduled structure $\mathcal{T}_{i-1}(\mathbf{G}_{i-1})$ constructed up to instant $i-1$, as the sum of *local Lagrangian costs* $l_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}_{i,j})$'s for all views j's using *local schedules* $\mathbf{g}_{i,j}$'s at instant i (each dictates the transitions from frames $F_{i-1,k}$'s in $\mathcal{T}_{i-1}(\mathbf{G}_{i-1})$ to frame $F_{i,j}$ of view j), and future recursive cost $L_{i+1}(\mathcal{T}_i(\mathbf{G}_i))$:

$$L_i(\mathcal{T}_{i-1}(\mathbf{G}_{i-1})) = \min_{\mathbf{g}_{i,j}} \left\{ \sum_{j=1}^{K} l_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}_{i,j}) + L_{i+1}(\mathcal{T}_i(\mathbf{G}_i)) \right\} \quad (10)$$

Note that local schedules $\mathbf{g}_{i,j}$'s must be searched exhaustively to find the global minimum in (10).

Local Lagrangian cost $l_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}_{i,j}))$ for view j is itself a sum of costs of P-frames and of I-frames scheduled to handle transitions to view j from frames in instant $i-1$:

$$l_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}_{i,j}) = l^I_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}^I_{i,j}) + l^P_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}^P_{i,j}) \quad (11)$$

where $\mathbf{g}^I_{i,j}$ and $\mathbf{g}^I_{i,j}$ are the I- and P-frame local schedules from frames $F_{i-1,k}$'s in $\mathcal{T}_{i-1}(\mathbf{G}_{i-1})$ to I-frame $I_{i,j}$ and P-frame(s) $P^{(x)}_{i,j}$'s, respectively.

The local Lagrangian cost $l^I_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}^I_{i,j})$ for I-frame $I_{i,j}$ is the sum of transition probabilities into $I_{i,j}$ plus λ times the size of $I_{i,j}$:

$$l^I_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}^I_{i,j}) = \left[\sum_{F_{i-1,k} \overset{\mathbf{g}^I_{i,j}}{\Rightarrow} I_{i,j}} q(F_{i-1,k})\alpha_{i-1,k}(j) + \lambda \right] r^I_{i,j} \quad (12)$$

The local Lagrangian cost $l^P_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}^P_{i,j})$ for P-frames $P_{i,j}$'s, on the other hand, is the sum of each transition probability into a unique $P^{(x)}_{i,j}$ plus λ times the size of $P^{(x)}_{i,j}$:

$$l^P_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}^P_{i,j}) = \sum_{F_{i-1,k} \overset{\mathbf{g}^P_{i,j}}{\Rightarrow} P^{(x)}_{i,j}(k), \forall x} \left[q(F_{i-1,k})\alpha_{i-1,k}(j) + \lambda \right] r^P_{i,j}(k) \quad (13)$$

Display probabilities for I-frame $\mathcal{T}^I_{i,j}(\mathbf{G}_{i,j})$ and P-frames $\mathcal{T}^P_{i,j}(\mathbf{G}_{i,j})$ for view j in scheduled structure $\mathcal{T}_i(\mathbf{G}_i)$ of instant i in (10) can be derived using local schedules $\mathbf{g}^I_{i,j}$'s and $\mathbf{g}^P_{i,j}$'s:

$$\mathcal{T}^I_{i,j}(\mathbf{G}_{i,j}) = \left\{ \sum_{F_{i-1,k} \overset{\mathbf{g}^I_{i,j}}{\Rightarrow} I_{i,j}} q(F_{i-1,k})\alpha_{i-1,k}(j) \right\} \quad (14)$$

$$\mathcal{T}^P_{i,j}(\mathbf{G}_{i,j}) = \left\{ \bigcup_{F_{i-1,k} \overset{\mathbf{g}^P_{i,j}}{\Rightarrow} P^{(x)}_{i,j}(k), \forall x} q(F_{i-1,k})\alpha_{i-1,k}(j) \right\}$$

where $\bigcup(\ )$ denotes the enumeration of terms in $(\ )$, and each term in $\mathcal{T}^I_{i,j}(\mathbf{G}_{i,j})$ and $\mathcal{T}^P_{i,j}(\mathbf{G}_{i,j})$ is the display probability $q(F_{i,j})$ of a unique frame $F_{i,j}$ in scheduled structure $\mathcal{T}_i(\mathbf{G}_i)$.

We make two observations here. First, it is clear from (14) that there is at most one I-frame $I_{i,j}$, but possibly multiple P-frames $P_{i,j}(k)$'s, for view j in $\mathcal{T}_i(\mathbf{G}_i)$. Second, only frames and their associated display probabilities in $\mathcal{T}_i(\mathbf{G}_i)$ at instant i are needed as arguments for recursive calls to the next instant $i+1$, instead of the entire scheduled structure $\mathcal{T}_i(\mathbf{G}_i)$.

We claim that a call $L_0(I_{0,K^o})$ using (10) solves the Lagrangian (7) optimally. We state this result formally as a Theorem below and then provide an intuitive proof.

Theorem 1 *Initial call $L_0(I_{0,K^o})$ using (10) returns an optimal solution to IMVS* for $R = 0$.*

Proof of Theorem 1 *All local schedules $\mathbf{g}_{i,j}$'s at each instant i are searched in (10). Since scheduled structure $\mathcal{T}_i(\mathbf{G}_i)$ is constructed from $\mathcal{T}_{i-1}(\mathbf{G}_{i-1})$ and $\mathbf{g}_{i,j}$'s using (14), this implies that (10) searches all scheduled structures. All scheduled structures include the optimal structure $\mathcal{T}$ with optimal schedule $\mathbf{G}^*$. Hence (10) returns the optimal solution.* □

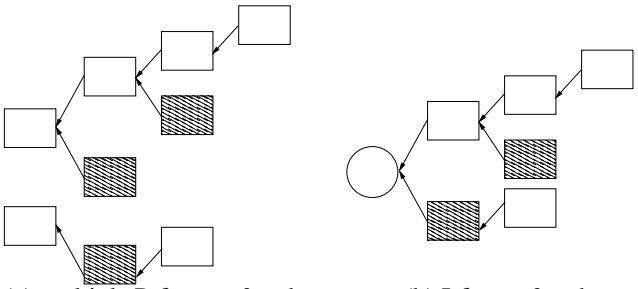
Though optimal, (10) is nevertheless exponential in running time; for n potential transitions into instant i and view j, there are $O(2^n)$ local schedules alone—each transition can be served either with an I-frame $I_{i,j}$ or a P-frame $P_{i,j}$. We hence discuss strategies to reduce its complexity next.

5.1.2. Complexity Reduction 1

To reduce complexity in (10), one must drastically reduce the exponential number of local scheduled structures that have to be tested in order to find the optimal one. To aid us towards that goal, we have the following lemma:

Lemma 1 *If local Lagrangian cost of transitioning from X frames in instant $i-1$ to a single I-frame $I_{i,j}$ at instant i and view j is no larger than corresponding local Lagrangian costs of transitioning to X P-frames $P^{(x)}_{i,j}$'s, $x = 1, \ldots, X$, at instant i, then resulting global Lagrangian cost of transitioning to $I_{i,j}$ is also no larger than resulting global Lagrangian costs[4] of transitioning to $P^{(x)}_{i,j}$'s.*

Proof of Lemma 1 *We show that for any set of subtrees stemming from $P^{(x)}_{i,j}$'s, one can construct a corresponding subtree stemming from $I_{i,j}$ such that the resulting global Lagrangian cost is no larger. For given set of subtrees $\mathcal{T}^{(x)}$'s below $P^{(x)}_{i,j}$'s, we construct corresponding subtree $\mathcal{S}$ for $I_{i,j}$ by taking the union of $\mathcal{T}^{(x)}$'s; i.e., for each P-frame $P_{y,z}$ in $\mathcal{T}^{(x)}$, we add a corresponding $P_{y,z}$ of the same size to $\mathcal{S}$ if $\mathcal{S}$ does not already have $P_{y,z}$ constructed. (See Fig. 4 for an example.) First, we know $|\mathcal{S}| \leq |\mathcal{T}^{(1)}| + \ldots + |\mathcal{T}^{(X)}|$ since union of sets is no larger then sum of sets. Further, transmission cost from $P^{(x)}_{i,j}$ to any frame $P_{y,z}$ in $\mathcal{T}^{(x)}$ is the same from $I_{i,j}$ to its corresponding $P_{y,z}$ in $\mathcal{S}$, hence the transmission cost of using $\mathcal{S}$ over $\mathcal{T}^{(x)}$'s can be no worse. Since by assumption $I_{i,j}$ alone induces no larger local Lagrangian cost than $P^{(x)}_{i,j}$'s, we conclude $I_{i,j}$ and $\mathcal{S}$ also result no larger global cost than $P^{(x)}_{i,j}$'s and $\mathcal{T}^{(x)}$'s.* □



(a) multiple P-frames & subtrees (b) I-frame & subtree

Fig. 4. Example of Union Construction of I-frame Subtree

The corollary of Lemma 1 is that if transitioning from a set of frames in $\mathcal{T}_{i-1}(\mathbf{G}_{i-1})$ to an I-frame $I_{i,j}$ at instant i results in no larger local Lagrangian cost than cost of transitioning to corresponding multiple P-frames $P_{i,j}$'s, then recursive calls to later instants for multiple P-frames $P_{i,j}$'s are not needed. To reduce the number of searches, we combine a greedy procedure with Lemma 1 as follows for transitions to view j:

1. Assign all feasible transitions from frames in $\mathcal{T}_{i-1}(\mathbf{G}_{i-1})$ to $\mathbf{g}^I_{i,j}$ and compute global Lagrangian cost using (10).
2. Repeatedly move a frame transition from $\mathbf{g}^I_{i,j}$ to $\mathbf{g}^P_{i,j}$ that yields the largest decrease in local Lagrangian cost and compute global cost using (10). Stop when no such transition is found.
3. Assign all remaining transitions in $\mathbf{g}^I_{i,j}$ to $\mathbf{g}^P_{i,j}$ and compare local Lagrangian cost to step 2. If smaller, compute global cost using (10).

The last step is needed since termination of step 2 does not entail that the local Lagrangian cost of only P-frames $P_{i,j}$'s and no I-frame $I_{i,j}$ is larger than I-frame and some P-frames. This procedure is applied to transitions to all views j's at instant i. Using this procedure for selecting local schedules $\mathbf{g}_{i,j}$'s for scheduled structure $\mathcal{T}_{i-1}(\mathbf{G}_{i-1})$, the number of next-level recursive calls for n transitions is now $O(n)$.

5.1.3. Complexity Reduction 2

Though the number of next-level recursive calls from each call instant is now linear due to procedure in Section 5.1.2, the number of total calls is still exponential in the depth of

[4]Lemma assumes each P-frame $P_{i,j}$ encoded using predictor $F_{i-1,k}$ is of size $r^P_{i,j}(k)$ as described in Section 4.1.3.

the recursion N. If N is large, then the algorithm is nonetheless computationally infeasible. Hence we propose a *sliding-window* strategy of lookahead depth $h < N$ as follows:

1. Let sliding index $s = 0$.
2. Perform algorithm IMVS* for window of depth h; i.e., optimize frames in instants $i = 0 + s, \ldots, h - 1 + s$.
3. Commit frames of the first instant in solution in step 2 to frame structure $\mathcal{T}$.
4. Increment s by 1. Goto step 2.

We will show in Section 6.2.1 that the sliding-window strategy produces good approximated results.

5.2. IMVS* Algorithm: $R \geq 1$

When $R \geq 1$, a feasible transition from $F_{i-1,k}$ in $\mathcal{T}_{i-1}(\mathbf{G}_{i-1})$ to view j can now be mapped to an alternative P-frame $P_{i,j}$ not predictively coded from $F_{i-1,k}$, incurring a multi-frame rerouting cost $\phi(F_{i-1,k}, P_{i,j})$. As a result, multiple transitions from multiple frames in $\mathcal{T}_{i-1}(\mathbf{G}_{i-1})$ can now be mapped to a single P-frame $P_{i,j}$.

To properly account for rerouting to alternative P-frames, P-frame local schedule $\mathbf{g}^P_{i,j}$ is now divided into a set of *sub-schedules* $\mathbf{g}^P_{i,j}(x)$'s, where $P^{(x)}_{i,j}$ is the key P-frame where frames in each sub-schedule $\mathbf{g}^P_{i,j}(x)$ are rerouted, and $F^{(x)}_{i-1,v(x)}$ of view $v(x)$ is the predictor of $P^{(x)}_{i,j}$. The local Lagrangian cost $l_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}^P_{i,j})$ is now:

$$l^P_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}^P_{i,j}) = \sum_x l_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}^P_{i,j}(x)) \quad (15)$$

where the local Lagrangian cost for sub-schedule $\mathbf{g}^P_{i,j}(x)$, $l_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}^P_{i,j}(x))$, is:

$$= \left[\sum_{F_{i-1,k} \overset{\mathbf{g}^P_{i,j}(x)}{\Rightarrow} P^{(x)}_{i,j}} q(F_{i-1,k})\alpha_{i-1,k}(j) + \lambda \right] r^P_{i,j}(v(x)) \; + \quad (16)$$
$$+ \sum_{F_{i-1,k} \overset{\mathbf{g}^P_{i,j}(x)}{\Rightarrow} P^{(x)}_{i,j}} q(F_{i-1,k})\alpha_{i-1,k}(j)\, \phi(F_{i-1,k}, P^{(x)}_{i,j})$$

Note that the sum of frame display Lagrangian costs and rerouting costs in (16) is analogous to (9).

It can be shown that given the new definition of local P-frame Lagrangian cost $l_{i,j}(\mathcal{T}_{i-1}(\mathbf{G}_{i-1}), \mathbf{g}^P_{i,j})$ in (15), one can still use (10) to exhaustively search through all local schedules $\mathbf{g}_{i,j}$'s at each instant i and find the optimal scheduled structure for $R \geq 1$ as well. But given the complexity of searching all scheduled structures as previously discussed, we instead generalize our previous fast IMVS* algorithm for $R = 0$ to the case when $R \geq 1$ as follows. For each view j:

1. Assign all feasible transitions from frames in $\mathcal{T}_{i-1}(\mathbf{G}_{i-1})$ to $\mathbf{g}^I_{i,j}$ and compute global Lagrangian cost using (10).
2. Repeatedly move a transition from $\mathbf{g}^I_{i,j}$ either to an existing sub-schedule $\mathbf{g}^P_{i,j}(x)$, or to a new sub-schedule, such that the resulting decrease in local Lagrangian cost is maximized. Compute the corresponding global cost in (10). Stop when no such transition is found.
3. Assign all remaining transitions in $\mathbf{g}^I_{i,j}$ to existing or new sub-schedules $\mathbf{g}^P_{i,j}(x)$'s while minimizing Lagrangian cost locally, and compare to step 2. If smaller, compute global cost using (10).

The sliding-window strategy described in Section 5.1.3 for $R = 0$ is used here as well for $R \geq 1$ when the depth of the recursion N is large.

6. EXPERIMENTATION

6.1. Experimental Setup

Fig. 5. Example Multiview Screen-shots of warsow.

To gather multiview video data for our experiments, we downloaded three consecutive views of the 100-frame ballroom sequence from [17] captured at 25fps and downscaled to QCIF (176×144). In addition, we modified the game client of Internet game warsow [18] to render and capture a 100-frame sequence of three views simultaneously at 10fps, each displaced by 30^o. Example screen-shots of warsow is shown in Fig. 5. The motivation is to supply two very different data sets: the former is real data with closely spaced cameras at high frame rate, while the latter is synthetic data with virtual cameras far apart at low frame rate.

For each sequence, we generated encoding rates $r^I_{i,j}$'s and $r^P_{i,j}(k)$'s as inputs to our IMVS* algorithm using H.264 JM version 12.4 [19] as follows. Each $r^I_{i,j}$ was obtained when we encoded picture $F^o_{i,j}$ as I-frame. $r^P_{i,j}(j)$'s were obtained when we encoded picture $F^o_{i,j}$ in a single-view sequence where each $F_{i,j}$ was motion-compensated from $F_{i-1,j}$. For $r^P_{i,j}(k)$'s, $k \neq j$, we first generated four zigzagged sequences as follows:

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

For each $r^P_{i,j}(k)$, we simply located the zigzagged stream z that contained the sub-sequence $\{F_{i-1,k}, P_{i,j}\}$ and assigned the corresponding coding rate.

For transition probabilities $\alpha_{i,j}(k)$'s, we assume frame $F_{i,1}$ remains at the same view $F_{i+1,1}$ with probability $1-\alpha$, and transitions to neighboring views $F_{i+1,0}$ and $F_{i+1,2}$ with probability $\alpha/2$ each. Frame $F_{i,0}$ ($F_{i,2}$) transition to the single neighboring view $F_{i,1}$ with the same probability α. We assume $\alpha = 0.1$ throughout our experiment.

6.2. Experimental Results

6.2.1. *Approximation Using Sliding-Window Strategy*

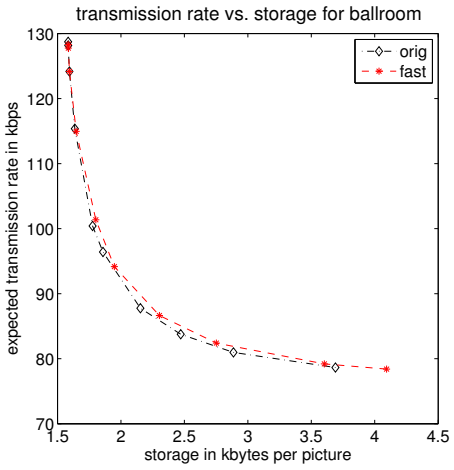
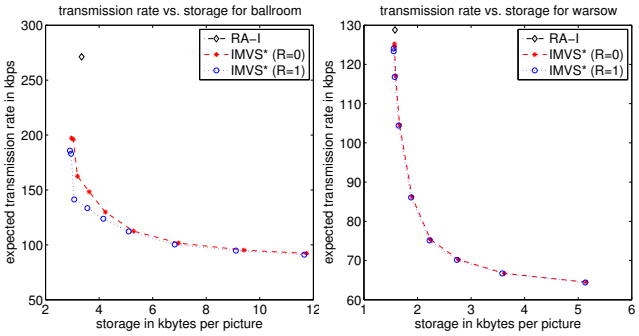


Fig. 6. Tradeoffs between Expected Transmission Rate and Storage Per Picture for `ballroom` using Sliding-window Strategy: $M' = 27$, $M = 3$.

We first examine how closely the fast sliding-window strategy (`fast`) discussed in Section 5.1.3 approximates the `IMVS`* algorithm without the sliding window (`orig`). Using `ballroom` as a test sequence, for random access period of $M' = 27$ and switch period of $M = 3$—hence optimization window depth of $N = \lfloor \frac{27}{3} \rfloor - 1 = 8$—we generated tradeoff points between expected transmission rate and storage required per picture using `orig`, shown in Fig. 6. Lagrangian multiplier λ was swept from 0.01 to 10.24 to induce different tradeoffs. We also generated tradeoff points for `fast` when the lookahead depth was $h = 5$. We see that the convex hull of `fast` closely resembled that of `orig`, demonstrating that the sliding-window strategy performs numerically close to the original in practice.

6.2.2. *Algorithm Performance Comparison*

For algorithm comparison, using first a random access period of $M' = 30$ frames and switching period of $M = 1$ frames, we plotted the tradeoff points for our `IMVS`* algorithm for the `ballroom` and `warsow` sequences in Fig. 7(a) and 7(b), respectively. In time, this corresponds to random access periods of 1.2s and 3.0s and switching periods of 40ms



(a) `ballroom` (b) `warsow`

Fig. 7. Tradeoffs between Expected Transmission Rate and Storage Per Picture for `IMVS`*: $M' = 30$, $M = 1$.

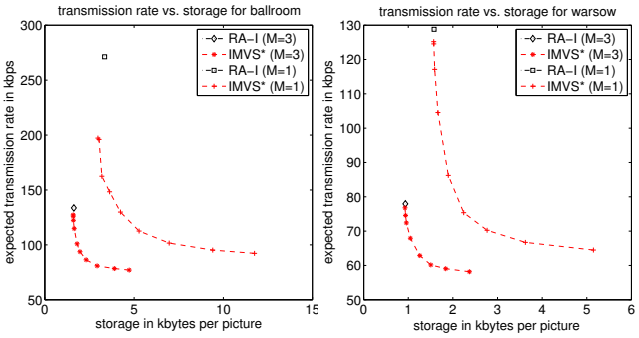
and 100ms for `ballroom` and `warsow`, respectively. We also plotted the performance of the random access approach (`RA-I`) where I-frames were inserted at all switching points for view switching. We first see that `RA-I` was represented by a single point; because the I-frame insertion algorithm was fixed, it had one corresponding fixed transmission and storage cost, and therefore could not take advantage of extra storage capacity if available to lower transmission cost.

Second, we see that our algorithm found tradeoff points that were to the lower left of `RA-I`; i.e., our `IMVS`* algorithm generated frame structures that offered lower transmission rates than `RA-I` *and* required smaller storage. This is particularly noticeable for `ballroom`, where our algorithm generated a frame structure with 48% smaller expected transmission rate while requiring smaller storage.

Third, unlike `RA-I` our algorithm offered a range of tradeoff points to take advantage of extra storage when available to further decrease expected transmission rate. In particular, at twice the storage of `RA-I`, our algorithm generated frame structures with 63% and 46% smaller expected transmission rate than `RA-I` for `ballroom` and `warsow`, respectively. The performance improvement of `IMVS`* over `RA-I` was more dramatic for `ballroom` than `warsow`; we conjecture that this was due to relatively smaller sizes of P-frames compared to I-frames in `ballroom`, as a result of the higher capturing frame rate and closely spaced cameras.

Fourth, we see that as the rerouting limit R increased from 0 to 1, the performance of our algorithm improved. The improvement was more noticeable in `ballroom`, and when storage required was small. This is intuitive because when storage is abundant, redundant P-frames will tend to be selected over rerouting due to their smaller transmission costs.

We repeated the experiments when the switching period was increased to $M = 3$. The random access periods of 1.2s and 3.0s for `ballroom` and `warsow` remained the same, while the switching periods were now 120ms and 300ms, respectively. The corresponding results for the two sequences



(a) `ballroom` (b) `warsow`

Fig. 8. Tradeoffs between Expected Transmission Rate and Storage Per Picture for `IMVS*`: $M' = 30$, $R = 0$.

when $M = 3$, as well as when $M = 1$, are shown in Fig. 8(a) and 8(b) when no rerouting is permitted. While the trends we observed earlier also hold here, we can make a couple of additional observations. First, the transmission and storage costs overall were lower than results for $M = 1$; this is expected since larger switching period means P-frames predictively coded using previous frames of the same view, which are very coding-efficient, are used until the next switching point.

Second, the performance gain over `RA-I` is less dramatic (though still significant) compared to the case when $M = 1$. In particular, at twice the storage of `RA-I`, `IMVS*` generated structures lowering expected transmission rate by 40% and 24% compared to `RA-I` for `ballroom` and `warsow`, respectively. This is because both `RA-I` and our generated structures needed to send $M - 1$ P-frames predictively coded in the same view until the next switching point, diluting the benefit of our view switching optimization.

We conjecture that other coding tools beyond I- and P-frames, like SP-frames [8] and Distributed Source Coding [6] would be very useful for IMVS for large values of M. Investigation and integration of these coding tools into our optimization framework is ongoing [7].

7. CONCLUSIONS

In this paper, we argued the important functional difference between view switching and random access in interactive multiview video streaming, and presented an algorithm that generates good redundant frame structures to enable bandwidth-efficient view switching. Our algorithm trades off expected transmission cost with total storage using different Lagrange multiplier values, and can impose frame reroute limit to limit the decoding complexity of a streaming client. Our results show that the generated frame structure outperformed the periodical I-frame insertion strategy commonly used for random access.

8. REFERENCES

[1] T. Fujii and M. Tanimoto, "Free viewpoint TV system based on ray-space representation," in *Proceedings of SPIE*, 2002, vol. 4864, pp. 175–189.

[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] 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*, November 2007, vol. 17, no.11, pp. 1474–1484.

[4] S. Shimizu, M. Kitahara, H. Kimata, K. Kamikura, and Y. Yashima, "View scalable multiview coding using 3-D warping with depth map," in *IEEE Transactions on Circuits and Systems for Video Technology*, November 2007, vol. 17, no.11, pp. 1485–1495.

[5] G. Cheung, A. Ortega, and T. Sakamoto, "Coding structure optimization for interactive multiview streaming in virtual world observation," in *IEEE International Workshop on Multimedia Signal Processing*, Cairns, Queensland, Australia, October 2008.

[6] 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.

[7] 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.

[8] 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.

[9] P. Ramanathan, M. Kalman, and B. Girod, "Rate-distortion optimized interactive light field streaming," in *IEEE Transactions on Multimedia*, June 2007, vol. 9, no.4, pp. 813–825.

[10] I. Bauermann and E. Steinbach, "RDTC optimized compression of image-based scene representation (part I): Modeling and theoretical analysis," in *IEEE Transactions on Image Processing*, May 2008, vol. 17, no.5, pp. 709–723.

[11] I. Bauermann and E. Steinbach, "RDTC optimized compression of image-based scene representation (part II): Practical coding," in *IEEE Transactions on Image Processing*, May 2008, vol. 17, no.5, pp. 724–736.

[12] P. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," in *IEEE Transactions on Multimedia*, April 2006, vol. 8, no.2, pp. 390–404.

[13] C. De Vleeschouwer, J. Chakareski, and P. Frossard, "The virtue of patience in low-complexity scheduling of packetized media with feedback," in *IEEE Transactions on Multimedia*, February 2007, vol. 9, no.2, pp. 348–265.

[14] H. Wang and A. Ortega, "Rate-distortion optimized scheduling for redundant video representations," in *IEEE Transactions on Image Processing*, February 2009, vol. 18, no.2, pp. 225–140.

[15] G. Cheung, W.-T. Tan, and C. Chan, "Reference frame optimization for multiple-path video streaming with complexity scaling," in *IEEE Transactions on Circuits and Systems for Video Technology*, June 2007, vol. 17, no.6, pp. 649–662.

[16] Cormen, Leiserson, and Rivest, *Introduction to Algorithms*, McGraw Hill, 1986.

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

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

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