

# JPEG2000-Based Scalable Interactive Video (JSIV)

Aous Thabit Naman, *Member, IEEE*, and David Taubman, *Senior Member, IEEE*

**Abstract**—We propose a novel paradigm for interactive video streaming and we coin the term **JPEG2000-Based Scalable Interactive Video (JSIV)** for it. JSIV utilizes JPEG2000 to independently compress the original video sequence frames and provide for quality and spatial resolution scalability. To exploit inter-frame redundancy, JSIV utilizes prediction and conditional replenishment of code-blocks aided by a server policy that optimally selects the number of quality layer for each code-block transmitted and a client policy that makes most of the received (distorted) frames. It is also possible for JSIV to employ motion compensation; however, we leave this topic to future work. To optimally solve the server transmission problem, a Lagrangian-style rate-distortion optimization procedure is employed. In JSIV, a wide variety of frame prediction arrangements can be employed including hierarchical B-frames of the scalable video coding (SVC) extension of the H.264/AVC standard. JSIV provides considerably better interactivity compared to existing schemes and can adapt immediately to interactive changes in client interests, such as forward or backward playback and zooming into individual frames. Experimental results for surveillance footage, which does not suffer from the absence of motion compensation, show that JSIV's performance is comparable to that of SVC in some usage scenarios while JSIV performs better in others.

**Index Terms**—Teleconferencing, video signal processing, image coding, image communication, weighted acyclic directed graphs.

## I. INTRODUCTION

TRADITIONAL video compression techniques, such as MPEG-1 through MPEG-4 and H.261 through H.264, have focused mainly on minimizing the amount of data for a given video quality while offering limited interactivity; for example, temporal random access is usually limited to a predetermined set of points<sup>1</sup> and random access to an arbitrary spatial region is not supported. Limited interactivity and a diversity of client needs have motivated research in the field of scalable video coding.

Scalable video coding can solve some of the existing problems in video storage and streaming as it can accommodate the varying needs of different clients from one base source file; it can also dynamically adapt to available network bandwidth, gracefully degrading the streamed video quality. Research in this area has produced some promising results [1], [2] and recently a scalable video coding (SVC) extension to H.264/AVC [3] has been approved within the ISO working group known as MPEG, to provide improved scalability options.

Even with these improved options, an encoded video for a certain application is not suitable for a different application.

Copyright ©2010 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org

<sup>1</sup>There exist an I-only profile in which temporal access is available to all frames.

For example, an encoded video using I-only frames can provide direct temporal access to each of its frames; however, it is mainly suitable for high bit rate applications as it does not exploit temporal redundancy. To achieve higher compression ratio (e.g., for video streaming applications over networks with limited bandwidth), the encoder must exploit temporal redundancy, which imposes restrictions on the encoded stream that limit temporal accessibility; for example, if a client is interested in one frame only, the server has to send enough data from the group of pictures (GOP) that contains this frame, possibly the whole GOP, and the client has to reconstruct potentially a large number of frames in order to invert the motion compensated transform used during compression and extract the desired frame. More examples are given in the following paragraphs.

For still images, the JPEG2000 Interactive Protocol (JPIP) [4], [5] provides many of the desirable features of scalable interactive browsing. These include resolution scalability, progressive refinement (or quality scalability), spatial random access, and highly efficient compression. JPIP also supports interactive browsing of Motion-JPEG2000<sup>2</sup>, although we note that Motion-JPEG2000 content does not involve any exploitation of inter-frame redundancy.

This work proposes JPEG2000-Based Scalable Interactive Video (JSIV) as a way of providing better flexibility, scalability, and interactivity options for video streaming compared to existing practices. JSIV relies on:

- JPEG2000 to independently compress the individual frames of the video sequence and provide for quality and spatial resolution scalability as well as random accessibility.
- Prediction, with or without motion compensation, and conditional replenishment of JPEG2000 code-blocks to exploit temporal redundancy.
- Loosely-coupled server and client policies. The server policy aims to select the best number of quality layers for each precinct it serves and the client policy attempts to produce the best possible reconstructed frames from the data the client has. Each of these policies may evolve separately without breaking the communication paradigm.

In our preliminary work [7]–[11], we have demonstrated the efficacy of JSIV with motion compensation. In this more rigorous treatment, however, it is convenient to restrict our attention to the case of prediction without motion compensation. This restriction allows us to focus on JSIV concepts, avoiding unnecessary complications in this introductory work; moreover,

<sup>2</sup>Motion-JPEG2000 [6] is a video file format based on JPEG2000. The file contains some video timing information, and each frame is stored independently in its own code-stream.

certain applications (e.g., surveillance) can benefit considerably from JSIV even in the absence of motion compensation. In a later paper, we will explore motion compensated JSIV in greater depth.

Before discussing the simplified block diagram of a JSIV system shown in Figure 1, we find it useful to discuss the underlying philosophy behind JSIV. As mentioned above, JSIV relies on loosely-coupled client and server policies. The philosophy behind JSIV is that the client should not explicitly drive the server's behavior (e.g., it should not request the delivery of specific code-block bit-streams) and the server should not explicitly drive the client's behavior (e.g. it should not tell the client how to synthesize each frame from the delivered content). Instead, the client poses high level requests to the server (e.g., region of interest, playback resolution/frame-rate, etc.) and the server replies with portions of code-block bit-streams which it believes to be helpful in satisfying the client's declared interests.

We postulate that if both the client and the server are intelligent enough to make reasonable decisions, then the decisions made by the server are likely to have the expected impact on the decisions made by the client. For example, if the server sends a reasonable amount of data from a particular code-block bit-stream, judging this to be beneficial to the client, then a reasonable client policy is likely to use the supplied bit-stream rather than predict the code-block in question from neighboring frames. The server may send additional side information to help the client resolve highly ambiguous situations<sup>3</sup>, but this side information should be expressed in a form which is independent of the state of the client-server interaction so that it describes properties of the source which are always true.

There are many advantages to such an approach. One example is when the client has some data that the server is unaware of; for example, the client might have obtained that data from a previous browsing session or from a different server or possibly a proxy. In this case the client uses its knowledge of the stream and its properties to consider both its cache content and the newly received data in order to select the options that, it believes, achieve the best possible quality. There is no need to worry about drifting between the client state and the server state, since the client makes its decisions based on properties of the stream which are always true; we demonstrate this in an example in Section VII. Other examples arise in the case where a client has less data than the server expects, for example due to data loss. For such cases, the client uses the knowledge it has about the stream and its properties to conceal any missing data.

We move our attention to the simplified block diagram of a JSIV system depicted Figure 1; the system has three basic entities: the preprocessing stage, the server, and the client. The preprocessing stage is responsible for compressing each frame individually into JPEG2000 format and preparing side-information for these frames. The side information can include distortion-length slope tables, motion distortions, and

any other side information that might be required during media serving. Side-information can either be generated off-line for pre-recorded media or in real-time for live media. Many of these operations are independent of each other and can be easily delegated to one or more machines in a content delivery network.

The server is composed of two main sub-blocks: the client distortion estimation block (CDEB) and the rate-distortion optimization block (RDOB). The CDEB attempts to model distortions in each code-block or each precinct of each frame, based on its knowledge about transmitted information and an assumed client policy. This model can also be adapted to reflect knowledge of network conditions, client browsing preferences and client browsing cache. It is sufficient for this block to generate approximate estimates of distortions, and therefore it does not have to actually reconstruct the client's view.

The RDOB performs Lagrangian-style rate-distortion optimization to decide the number of quality layers to be sent for each precinct of each frame, which can be zero. It also decides on any side-information needed by the client to best exploit the frame data. All the decisions made by the RDOB take into consideration the estimated distortion provided by the CDEB.

The server communicates with the client employing only JPIP [4], [5]; JSIV stores side-information in additional components in each frame, conceptually known as meta-components or meta-images. This allows the use of JPIP without any modifications to send both code-block data and side-information.

The client receives compressed code-block bit-streams and side information. Using this information and aided by a client policy, the client selects the source of data to use for each code-block. In particular, the client has the option to decode an available code-block bit-stream directly or to predict the code-block from nearby frames (possibly having much higher quality).

The flexibility and accessibility of JSIV is the result of not committing to any predetermined temporal prediction policy. Thus, the server is free to change its policy on the fly and during serve time to respond to changes in client requirements or network conditions; for example, the server can switch from the hierarchical B-frame prediction arrangement to optimizing each frame independently from other frames. The decision to choose a particular prediction arrangement can depend on the amount of data loss on the network (the absence of inter-frame dependency makes the sequence more loss-resilient).

JSIV departs from traditional predictive coding schemes in that side-information in JSIV is only a guide that helps the client make sound decisions while in traditional video compression schemes it totally dictates the client prediction modes, prediction reference frames, and any other client operation modes. Also, JSIV always sends intra-coded frame pieces and never uses residual data. The use of actual data instead of residual data incurs an encoding loss, but at the same time enables the client to make decisions independently of the server and avoid the possibility of drift between server and client states. This independence enables the client to use its cache more effectively and possibly use information obtained

<sup>3</sup>For example, in the absence of any data for some code-block, it is not at all clear whether the client should synthesize the frame using zeros or predicted samples for that code-block.

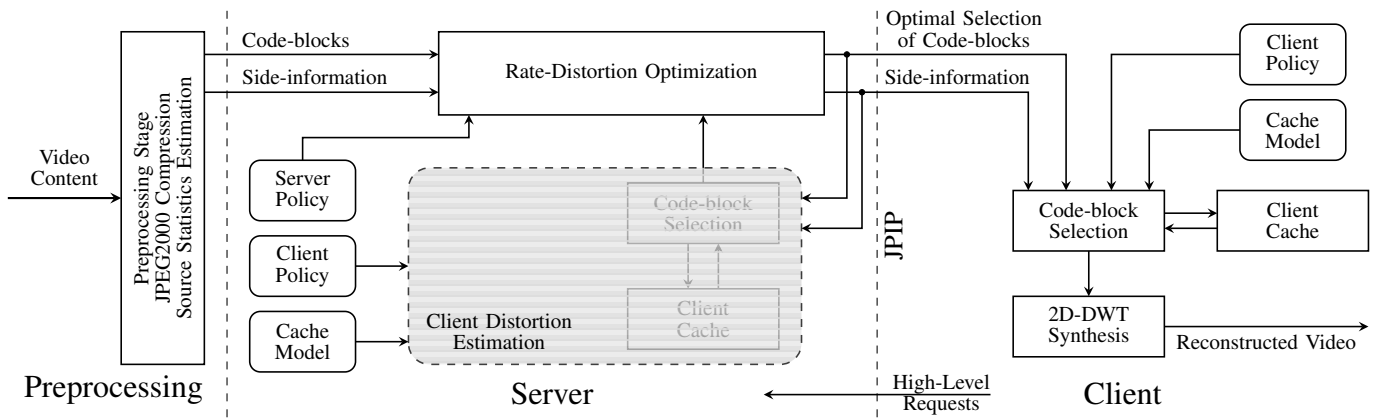


Fig. 1. A simplified block diagram of the proposed JSIV delivery system. The client distortion estimation block, shown in gray, estimates client-side distortions in reconstructed frames without reconstructing them.

from other servers or proxies.

Many researchers have realized the limited interactivity provided by the existing techniques [12]–[17], and have devised different approaches that are favorable in certain situations.

Cheung and Ortega [12], [13] propose *flexible video decoding* that provides forward and backward playback for traditional and multi-view sequences. Their proposed system is similar to JSIV in that the server provides multiple possible predictors and let the client choose the best predictor at decode time; unlike the approach proposed here, theirs is based on distributed video coding. Another work that attempts to improve interactivity is by Mavlankar *et al.* [17]. In that work, they propose a way of dynamically providing pan, tilt, and zoom features in video playback to different clients with varying regions of interest [17]. The proposed method breaks a high resolution video into tiles and streams them simultaneously using H.264 compression and employing some peer-to-peer delivery techniques. They also investigate limited scalability by simultaneously broadcasting two or more streams, with dyadically-spaced spatial resolution. JSIV shares these capabilities and can also provide temporal scalability and quality-optimized region of interest.

Another recent work by Devaux *et al.* [15] that was published around the same time we introduced JSIV [7] investigates a problem that is similar to JSIV in some aspects; however, they stopped short of investigating the flexibility that such a paradigm can provide. For example, the interaction between the client and the server is totally dictated by the server policy with a simple client that is incapable of making its own decisions; moreover, prediction is only possible from the last received frame with no motion compensation. JSIV can employ prediction with or without motion compensation from any frame within the window of the frames being optimized, and can refine previously transmitted frames whenever that is favorable. They extended their work in [16] by employing LPDC to generate parity bits that can potentially improve lightly-distorted predictors. JSIV can also improve prediction as was demonstrated in [11], [18]. The techniques proposed in our work, however, utilizes the client’s knowledge about the quantization bins or intervals of received samples in improving

prediction.

Some concepts in JSIV are adapted from a recent work on interactive browsing of 3D scenes, which has shown promising results [14]. JSIV concepts were introduced progressively by the authors in [7]–[11], [18]; in this work, we elaborate on these concepts and formalize them. In [7], we examined a realistic implementation; however, inter-frame redundancy is exploited only within disjoint pairs of frames. In [8], the server optimization policy was extended to a sliding window of sequential frames, with the potential to exploit the redundancy between any two frames within the window. In [9], the server optimization policy was extended to a hierarchical group of frames, similar to the hierarchical B-frame dyadic prediction structure used by the SVC extension of H.264/AVC. In [10], we proposed a method for approximate distortion estimation rather than reconstructing the video to calculate real distortions. In [11], we introduced a client capable of improving prediction by exploiting its knowledge about quantization bins of the received samples. This client is served by a server that can take advantage of this capability in a realistic implementation context. In [18], we proposed a method to improve prediction by exploiting the quantization intervals of received samples in selecting a more favorable predictor in hierarchical B-frame arrangement without motion compensation. We also demonstrated how the client and server policies can evolve independently with little or no impact on the quality of reconstructed video.

Remote browsing of high-resolution surveillance video is one application where JSIV can provide an improved browsing experience compared to current video coding standards. Usually for such video sequences, most of the changes happen in certain regions, and the background occupies a good percentage of the frames, with little or no changes. For such a case, JSIV works better than conventional JPIP since it effectively reduces to an optimized conditional replenishment scheme while JPIP by itself needs to transmit the full content of each frame. In a typical browsing scenario, a remote client browsing surveillance footage usually can monitor a low-resolution version of the whole scene, as this provides the interactive user with sufficient details to identify any regions of interest. To

have a more detailed look at any of these interesting regions, the interactive user changes his or her window of interest to that region, and starts monitoring it. For monitoring a window of interest JSIV works favorably compared to existing coding standards, as these do not support retrieval of an arbitrary window from the encoded video sequence. Other advantages of JSIV for surveillance video browsing over existing paradigms include the capability of backward playback, reduced temporal rate playback, and lossless retrieval of frames or regions of interest in frames.

We consider here some of the other benefits of JSIV. For lossy transmission environments, the server does not need to retransmit lost packets, instead it can adapt by adjusting its delivery policy for future frames alone. Both the client and the server can easily and dynamically change from video playback mode to individual frame browsing mode and any data in the client cache is readily available for the reconstruction of individual frames. For clients with limited processing capabilities, the server can adapt its policy by reducing frame rate, reducing resolution, or by focusing on the client's window of interest.

The rest of this work is organized as follows. Section II gives a brief overview of the data organization in JPEG2000 code-streams. Sections III and IV describe "oracle" client and server policies that enable us to discuss the basic JSIV optimization algorithm. Section V gives actual client and server policies. In Section VI, we discuss the amount of computations required to deploy JSIV, and we propose a way of significantly reducing it. Section VII gives experimental results spanning many of the interesting use cases mentioned above, and Section VIII states our conclusion and points to future work.

## II. JPEG2000 CODE-STREAM ORGANIZATION

JSIV uses JPEG2000 to store individual frames. For this reason, we find it very important to review the code-stream organization of JPEG2000.

JPEG2000 employs the Discrete Wavelet Transform (DWT);  $D$  stages of DWT, labeled  $d = 1, 2, \dots, D$ , decompose a frame,  $f_n$ , into  $3 \cdot D + 1$  sub-bands, labeled  $HL_d$ ,  $LH_d$ ,  $HH_d$ , and  $LL_D$ . Each sub-band is partitioned into rectangular blocks, known as *code-blocks*. Each code-block is denoted by  $C_n^\beta$ , where  $\beta$  is the code-block index and  $n$  is the index of the frame to which code-block  $C_n^\beta$  belongs. Each code-block  $C_n^\beta$  is independently encoded into a finely embedded bit-stream using a fractional bit-plane arithmetic encoder. We say that a sub-band belongs to resolution  $LL_d$  if it contributes to the reconstruction of that resolution, and not to  $LL_{d+1}$ . So, resolution  $LL_D$  has only one sub-band, while each of the other resolutions has three sub-bands,  $HL_{d+1}$ ,  $LH_{d+1}$ , and  $HH_{d+1}$ .

Although code-blocks are coded independently, they are not explicitly identified within the code-stream. They are collected into larger groupings known as *precincts*. Each precinct is denoted by  $\mathcal{P}_n^\pi$ , where  $\pi$  is the precinct index and  $n$  is the index of the frame to which precinct  $\mathcal{P}_n^\pi$  belongs. Each precinct,  $\mathcal{P}_n^\pi$ , groups code-blocks that contribute to the same spatial region of a single resolution  $LL_d$  from  $HL_{d+1}$ ,  $LH_{d+1}$ , and  $HH_{d+1}$  when  $d < D$ , or from  $LL_D$  when  $d = D$ . For

image browsing/streaming applications it is preferable that the JPEG2000 stream is made up of precincts that have only one code-block from each of their constituent sub-bands since this minimizes the spatial impact of a precinct.

Each precinct is represented as a collection of *packets*, with one packet for each *quality layer*,  $q_n^\pi$ . A complete JPEG2000 code-stream consists of a concatenated list of packets, together with special marker segments which are used principally to signal coding parameters. The standard supports a variety of different packet ordering conventions, including layer-oriented, resolution-oriented, and spatially-oriented sequences.

JPEG2000 arranges quality layers by employing the Embedded Block Coding with Optimized Truncation (EBCOT) algorithm [19]. During the block coding phase, each coding pass of the fractional bit-plane encoder adds data to the embedded bit stream, and thus each coding pass has an associated length and distortion contribution. These coding passes undergo convex hull analysis where suitable truncation points,  $h^m$ , are identified such that their distortion-length slopes,  $S^m$ , given by

$$S^m = \begin{cases} \frac{D_*(h^{m-1}) - D_*(h^m)}{|h^m| - |h^{m-1}|} & m = 1, 2, \dots, H \\ \infty & m = 0 \end{cases} \quad (1)$$

decrease monotonically with  $m$ ; here,  $|h^m|$  and  $D_*(h^m)$  are the length and distortion of the  $m^{\text{th}}$  suitable truncation point.

Each of the  $Q$  quality layers,  $q_n^\pi = 1, 2, \dots, Q$ , is formed by including incremental contributions of  $|h^{m^q}| - |h^{m^{q-1}}|$  code bytes from each code-block within  $\mathcal{P}_n^\pi$ , where  $m^0 = 0$ , and

$$m^q = \max\{m \mid S^m \geq T_q\} \quad (2)$$

The distortion-length slope thresholds  $T_q$  are selected during compression so as to achieve suitably spaced layer bit-rate or quality increments, and are usually fixed for the whole image.

## III. ORACLE CLIENT POLICY

The client policy presented here is termed an "oracle" policy because of the unrealistic underlying assumption that the client can make distortion-based decisions correctly, without actually receiving any information about distortion.

For each code-block,  $C_n^\beta$ , of each frame,  $f_n$ , the client receives zero or more quality layers,  $q_n^\beta$ . Consequently, the de-quantized samples of a code-block,  $C_{*n}^\beta(q_n^\beta)$ , have an associated distortion given by  $D_{*n}^\beta(q_n^\beta) = \|C_{*n}^\beta(q_n^\beta) - \hat{C}_n^\beta\|^2$ , where  $\hat{C}_n^\beta$  are the full quality code-block samples. For frame reconstruction, the client can also use predicted samples,  $C_{\rightarrow n}^\beta$ ; in general, the prediction reference samples of  $C_{\rightarrow n}^\beta$  also suffer from quantization distortion. The techniques and sources used in obtaining these predicted samples depend on the policies employed by the client and the server.

In general, the distortion associated with predicted samples,  $D_{\rightarrow n}^\beta$ , can be approximated by a combination of *motion distortion*,  $D_{\rightarrow n}^{M,\beta}$ , due to motion or other inter-frame changes and *quantization distortion*,  $D_{\rightarrow n}^{Q,\beta}$ , due to quantization in prediction

reference samples. That is,

$$\begin{aligned}
 D_{\rightarrow n}^\beta &= \|\mathcal{C}_{\rightarrow n}^\beta - \hat{\mathcal{C}}_n^\beta\|^2 \\
 &= 2 \cdot \left\langle \hat{\mathcal{C}}_{\rightarrow n}^\beta - \hat{\mathcal{C}}_n^\beta, \mathcal{C}_{\rightarrow n}^\beta - \hat{\mathcal{C}}_{\rightarrow n}^\beta \right\rangle \\
 &\quad + \underbrace{\|\hat{\mathcal{C}}_{\rightarrow n}^\beta - \hat{\mathcal{C}}_n^\beta\|^2}_{D_{\rightarrow n}^{\text{M},\beta}} + \underbrace{\|\mathcal{C}_{\rightarrow n}^\beta - \hat{\mathcal{C}}_{\rightarrow n}^\beta\|^2}_{D_{\rightarrow n}^{\text{Q},\beta}} \\
 &\approx D_{\rightarrow n}^{\text{M},\beta} + D_{\rightarrow n}^{\text{Q},\beta}
 \end{aligned} \tag{3}$$

where  $\hat{\mathcal{C}}_{\rightarrow n}^\beta$  is the predictor obtained from full quality reference samples. We assume that motion and quantization errors are likely to be uncorrelated in practice, allowing us to ignore the cross term. This assumption is supported by experimental results as shown in Section VI.

To improve prediction, it is very common to use some position-dependent linear combination of more than one reference frame. This technique is widely employed in the MPEG1 to MPEG4 standards. Here, we write  $\mathcal{A}(f_n)$  for the set of reference frames that directly contributes to  $f_n$ 's prediction, and we employ a linear combination given by

$$f_{\rightarrow n} = \sum_{r \in \mathcal{A}(f_n)} g_{rn} \cdot f_r \tag{4}$$

We choose to use position-independent scaling factors,  $g_{rn}$ , in this work; that is, we fix  $g_{rn}$  for a given prediction arrangement as is highlighted in Section VII. Space-varying scaling factors, however, can be readily incorporated into the approach.

In view of (4), predicted samples,  $\mathcal{C}_{\rightarrow n}^\beta$ , are given by

$$\mathcal{C}_{\rightarrow n}^\beta = \sum_{r \in \mathcal{C}_r^\beta \in \mathcal{A}(\mathcal{C}_n^\beta)} g_{rn} \cdot \mathcal{C}_{r \rightarrow n}^\beta \tag{5}$$

where  $\mathcal{C}_{r \rightarrow n}^\beta$  are the samples predicted from  $\mathcal{C}_r^\beta$  of frame  $f_r$ , and (3) becomes

$$D_{\rightarrow n}^\beta \approx D_{\rightarrow n}^{\text{M},\beta} + \sum_{r \in \mathcal{C}_r^\beta \in \mathcal{A}(\mathcal{C}_n^\beta)} g_{rn}^2 \cdot D_{r \rightarrow n}^{\text{Q},\beta} \tag{6}$$

This approximation relies upon the same condition as (3); moreover, (6) requires that quantization distortions among the different reference frames in  $\mathcal{A}(f_n)$  be uncorrelated, a commonly employed approximation in the literature.

Using an additive model, precinct distortions,  $D_n^\pi$ , can be approximated by

$$D_n^\pi = \sum_{\beta \in \mathcal{C}^\beta \subset P^\pi} G_{b_\beta} \cdot D_n^\beta \tag{7}$$

where  $G_{b_\beta}$  is the energy gain of sub-band  $b$  to which code-block  $\beta$  belongs. Similar approximations can thus be written for  $D_{*n}^\pi(q_n^\pi)$  and  $D_{\rightarrow n}^\pi$ . These approximations are valid provided that the wavelet transform basis functions are orthogonal or the quantization errors in each of the samples are uncorrelated. Neither of these requirements is strictly satisfied; however, the wavelet kernels used in our experimental investigations in Section VII have nearly orthogonal basis functions.

Ideally, the client chooses the samples that produce lower distortion; that is,

$$D_n^\pi(q_n^\pi) = \min \{ D_{*n}^\pi(q_n^\pi), D_{\rightarrow n}^\pi \} \tag{8}$$

This simple client policy is unrealistic as the client has no access to the actual media and therefore is incapable of calculating distortions, especially for  $D_{\rightarrow n}^\pi$ ; this policy will be revised to one which does not require explicit knowledge of distortions in Section V. Although it is possible for the client to make decisions on a code-block basis rather than on the level of precincts, we choose to work with precincts because the smallest piece a server can send in JPIP is one layer of one precinct (formally known as a packet). This means that the server transmits precinct-optimized data.

#### IV. ORACLE SERVER POLICY

The server policy presented here is termed an ‘‘oracle’’ policy because of the unrealistic underlying assumption that the client can make distortion-based decisions that correctly reflect the server’s intentions without the server sending any information about distortions to the client.

At a JSIV server, rate-distortion optimization is performed over windows of frames. Each frame within the window of frames (WOF) being optimized has a chance of contributing data to the interactive session. We refrain from using the term *group of pictures* (GOP) to describe these frames since WOF refers to the frames being optimized regardless of their prediction arrangement. We solve the rate-distortion optimization problem for a WOF,  $\mathcal{F}_s$ , by utilizing the generalized Lagrange multiplier method [20]; this involves recasting the problem as the minimization of a Lagrangian cost functional,  $J_\lambda$ , given by

$$J_\lambda = \sum_{n \in \mathcal{F}_s} \sum_{\pi \in \mathcal{P}_n} (D_n^\pi + \lambda \cdot |q_n^\pi|) \tag{9}$$

where  $|q_n^\pi|$  is the number of bytes in  $q_n^\pi$  quality layers, and  $\lambda$  is a Lagrangian parameter that is adjusted until the solution which minimizes  $J_\lambda$  satisfies some length constraint.

The minimization of  $J_\lambda$  involves selecting the number of quality layers,  $q_n^\pi$ , for each precinct,  $\mathcal{P}_n^\pi$ , in  $\mathcal{F}_s$ , as well as deciding which precincts are predicted,  $D_{\rightarrow n}^\pi$ , and which are directly decoded (i.e. decoded independently),  $D_{*n}^\pi$ . We denote the state of a precinct by  $\chi_n^\pi$  with  $\chi_n^\pi = 0$  for a predicted precinct and  $\chi_n^\pi = 1$  for a directly decoded precinct. This way, (9) can be written as

$$J_\lambda = \sum_{n \in \mathcal{F}_s} \sum_{\substack{\chi_n^\pi=0 \\ \pi \in \mathcal{P}_n}} D_{\rightarrow n}^\pi + \sum_{n \in \mathcal{F}_s} \sum_{\substack{\chi_n^\pi=1 \\ \pi \in \mathcal{P}_n}} D_{*n}^\pi(q_n^\pi) + \lambda \cdot \sum_{n \in \mathcal{F}_s} \sum_{\substack{\chi_n^\pi=1 \\ \pi \in \mathcal{P}_n}} |q_n^\pi| \tag{10}$$

Direct minimization of this functional is difficult because of the interdependency between predicted precincts and their predictors. For example, the decision to make precinct  $\mathcal{P}_k^\pi$  of Figure 2 predicted,  $\chi_k^\pi = 0$ , depends on the quality of its predictor,  $\mathcal{P}_i^\pi$ , but the quality of  $\mathcal{P}_i^\pi$  depends to some extent on  $\chi_k^\pi$ ; when  $\mathcal{P}_i^\pi$  is used as a prediction reference precinct, its distortion affects multiple precincts, which results in the assignment of more bytes (higher quality) to  $\mathcal{P}_i^\pi$  in the Lagrangian optimization.

This is essentially a dependent bit allocation problem. A few researcher worked on similar problems; in particular, we mention [21]–[23]. By adapting some concepts from [21], it is possible to develop a trellis-based approach to the optimization

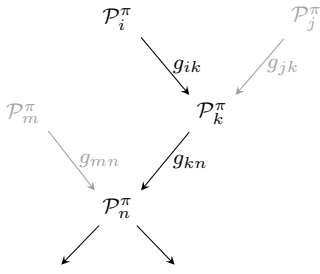


Fig. 2. A WADG,  $\mathcal{D}$ , showing prediction relationship among various precincts with their corresponding scaling factors. The gray part of the graph is for possible precincts that are not used during discussions.

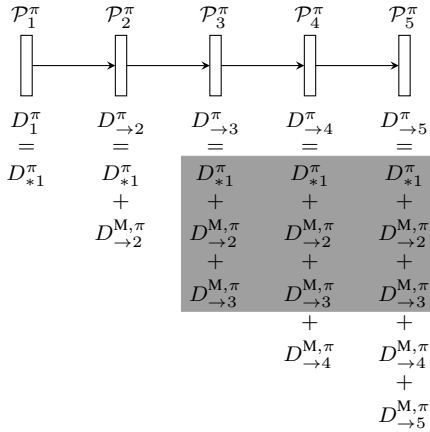


Fig. 3. An illustrative example showing how distortion propagates from one precinct to other precincts. The figure shows five precincts, each from one frame, that are indexed by the same  $\pi$  for a WOF composed of 5 consecutive frames.  $\mathcal{P}_1^\pi$  is a reference precinct while each of the other precincts is predicted from the precinct before it as indicated by the arrows. Underneath each precinct, we show the distortion contribution of that precinct. The gray area is the contribution of the distortion in precinct  $\mathcal{P}_3^\pi$  to the other precincts in the WOF.

of JSIV without motion compensation; however, such an approach becomes highly complex when motion compensation is employed, even with distortion modeling. When motion compensation is employed, the distortion in a given precinct in a reference frame propagates, in general, to multiple precincts in each predicted frame; details about distortion propagation can be found in [24] and [25]. Therefore, we find it more convenient to present an alternate approach that is useful for both JSIV with and without motion compensation.

In this proposed approach, we use of an additive distortion model (an extension of (6)) to simplify (10), together with an iterative approach that attempts to reduce the cost functional in each iteration. The iterative approach is composed of two passes. The first pass evaluates an *additional contribution weight*,  $\theta_n^\pi$ , for each precinct in the WOF; this weight accounts for the additional distortion that a precinct contributes to the WOF when it is used as a prediction reference precinct. The second pass performs rate-distortion optimization for each precinct in the WOF to incrementally optimize both  $\chi_n^\pi$  and  $q_n^\pi$ .

For the additive distortion model, beside the assumptions made in (6), we also assume that motion distortion between one precinct and another is uncorrelated with motion distortion

between any other pair of precincts. Although the validity of this approximation is questionable, it is necessary for guaranteed convergence of the two-pass iterative approach as well as providing significant simplifications. We have more to say about this in Section VI.

Figure 3 shows an application of this additive distortion model; here, each precinct,  $\mathcal{P}_n^\pi$ , is predicted from the precinct before it,  $\mathcal{P}_{n-1}^\pi$ , except for  $\mathcal{P}_1^\pi$  which is directly decoded. The distortion in  $\mathcal{P}_1^\pi$  is  $D_{*1}^\pi$ ; using the additive distortion model, the distortions in  $\mathcal{P}_2^\pi$  is  $D_{*1}^\pi + D_{\rightarrow 2}^{M,\pi}$ , in  $\mathcal{P}_3^\pi$  is  $D_{*1}^\pi + D_{\rightarrow 2}^{M,\pi} + D_{\rightarrow 3}^{M,\pi}$ , and so on. Thus, the effective distortion contribution of precinct  $\mathcal{P}_1^\pi$  to the WOF containing the five precincts shown in Figure 3 is  $(1 + \theta_1^\pi) \cdot D_{*1}^\pi$ , where  $\theta_1^\pi = 4$ . Ultimately, the distortion contribution of these five precincts to the WOF can be written as  $(1 + \theta_1^\pi) \cdot D_{*1}^\pi + \sum_{j=2}^5 (1 + \theta_j^\pi) \cdot D_{\rightarrow j}^{M,\pi}$ , where  $\theta_2^\pi = 3$ ,  $\theta_3^\pi = 2$ ,  $\theta_4^\pi = 1$ , and  $\theta_5^\pi = 0$ .

Utilizing this additive distortion model in (10) and rearranging, we get

$$\begin{aligned}
 J_\lambda = & \sum_{n \in \mathcal{F}_s} \sum_{\substack{\chi_n^\pi=0 \\ \pi \in f_n}} (1 + \theta_n^\pi) \cdot D_{\rightarrow n}^{M,\pi} \\
 & + \sum_{n \in \mathcal{F}_s} \sum_{\substack{\chi_n^\pi=1 \\ \pi \in f_n}} (1 + \theta_n^\pi) \cdot D_{*n}^\pi(q_n^\pi) + \lambda \cdot \sum_{n \in \mathcal{F}_s} \sum_{\substack{\chi_n^\pi=1 \\ \pi \in f_n}} |q_n^\pi|
 \end{aligned} \tag{11}$$

Before discussing the details of the two-pass iterative approach, we find it useful to discuss some concepts from directed graphs [26]. It is obvious that prediction, with or without motion compensation, creates dependency among the frames of one WOF,  $\mathcal{F}_s$ . This dependency can be represented by a weighted acyclic directed graph (WADG) [26]; see Figure 2 for an example. The nodes of the graph, also called vertices, can represent frames, precincts, or code-blocks depending on the context. The links between these nodes, called arcs, are directional links starting from a reference frame and ending in a predicted frame. It is weighted because these arcs have a scaling factor associated with them as in (4). A WADG is a weighted directed graph with no cycles; that is, if  $f_m$  is contributing to  $f_n$ 's prediction, then there is no way, direct or indirect, that  $f_n$  is contributing to  $f_m$ 's prediction. The set of vertices denoted by  $\mathcal{A}(f_n)$  as in (4) are known as the in-neighbors while the set of frames that are directly predicted from frame  $f_n$  are known as out-neighbors and are denoted by  $\mathcal{S}(f_n)$ . Next, we discuss the two passes.

1) *Contribution Weight Pass,  $\Psi_w$* : In  $\Psi_w$ , the additional contribution weights,  $\theta_n^\pi$ , are calculated so that (11) correctly represents (10) so long as  $\{\chi_n^\pi\}_\pi$  and  $\{q_n^\pi\}_\pi$  values remain constant. The weights are updated based on the values of  $\{\chi_n^\pi\}_\pi$  that are obtained from the last rate-distortion optimization pass,  $\Psi_o$ , or from initial conditions. This is achieved by visiting all the frames within the WOF in the acyclic ordering<sup>4</sup> of the converse<sup>5</sup> of the dependency WADG,  $\mathcal{D}^*$ .

<sup>4</sup>It is always possible to arrange the vertices of a WADG in what is called *acyclic ordering* [26]. In such an ordering, each vertex is positioned after all its reference vertices and before any of its dependent vertices.

<sup>5</sup>For every WADG,  $\mathcal{D}$ , there is a converse WADG denoted by  $\mathcal{D}^*$ , which is obtained by reversing all the arcs of  $\mathcal{D}$  [26].

For each precinct in such a frame, the value of  $\theta_n^\pi$  can be easily determined from (10) and (11) and is given by

$$\theta_n^\pi = \sum_{\substack{\chi_j^\pi=0 \\ j \ni \mathcal{P}_j^\pi \in \mathcal{S}(\mathcal{P}_n^\pi)}} g_{n_j}^2 \cdot (1 + \theta_j^\pi) \quad (12)$$

This order guarantees that predicted frames are visited before their reference frames and thus reference frames' contribution to predicted frames is accounted for before visiting these reference frames. In practice, we initialize all  $\{\chi_n^\pi\}_\pi$  to 1; therefore, the first value  $\theta_n^\pi$  takes is 0.

2) *Rate-Distortion Optimization Pass,  $\Psi_o$* : This is the pass where the actual rate-distortion optimization is performed. In  $\Psi_o$ , the values of  $\{\chi_n^\pi\}_\pi$  and  $\{q_n^\pi\}_\pi$  are changed in a way that minimizes the cost functional of (11) while  $\{\theta_n^\pi\}_\pi$  values are kept constant.

Consider the cost contribution of  $\mathcal{P}_3^\pi$  to the cost functional,  $J_\lambda$ , for the WOF containing the five precincts shown in Figure 3. The decision to make  $\mathcal{P}_3^\pi$  predicted ( $\chi_3^\pi = 0$ ) has a cost contribution of  $(1 + \theta_3^\pi) \cdot (D_{*1}^\pi + D_{*2}^{M,\pi} + D_{*3}^{M,\pi})$ , where  $\theta_3^\pi = 2$ ; this cost contribution, which is shown with a gray background in Figure 3, is equivalent to  $(1 + \theta_3^\pi) \cdot D_{*3}^\pi$ . Similarly, we can deduce that the cost contribution of making  $\mathcal{P}_3^\pi$  directly decoded ( $\chi_3^\pi = 1$ ) is  $(1 + \theta_3^\pi) \cdot D_{*3}^\pi + \lambda \cdot |q_3^\pi|$ , since  $D_3^\pi$  equals to  $D_{*3}^\pi$  instead of  $D_{*3}^\pi$  in this case. We conclude that a precinct's contribution to the cost functional of (11), is

$$J_{n,\lambda}^\pi = \begin{cases} (1 + \theta_n^\pi) \cdot D_{*n}^\pi, & \chi_n^\pi = 0 \\ (1 + \theta_n^\pi) \cdot D_{*n}^\pi(q_n^\pi) + \lambda \cdot |q_n^\pi|, & \chi_n^\pi = 1 \end{cases} \quad (13)$$

The  $\Psi_o$  pass involves visiting each frame,  $f_n$ , in the WOF,  $\mathcal{F}_s$ , in the acyclic ordering of the WADG,  $\mathcal{D}$ . For each precinct,  $\mathcal{P}_n^\pi$ , in each visited frame, we first update  $D_{*n}^\pi$  to its latest value then we choose  $\chi_n^\pi$  and  $q_n^\pi$  that produce the lowest  $J_{n,\lambda}^\pi$ . The frame visiting order guarantees that a given frame is processed after its reference frames. This way, we can calculate  $D_{*n}^\pi$  for all the precincts in a given frame before minimizing  $J_{n,\lambda}^\pi$  for these precincts. Changes to  $\{\chi_n^\pi\}_\pi$  during  $\Psi_o$  necessitate another  $\Psi_w$  to update  $\{\theta_n^\pi\}_\pi$  so that (11) correctly models (10). Thus, multiple iterations of  $\Psi_w \Psi_o$  might be needed to achieve the lowest possible cost functional using this method. This iterative process converges when a  $\Psi_o$  pass does not change any of the  $\{\chi_n^\pi\}_\pi$ .

Convergence of this two-pass iterative approach can be shown as follows.  $\Psi_w$  is not part of the rate-distortion optimization; it only updates  $\{\theta_n^\pi\}_\pi$  so that (11) correctly models (10).  $\Psi_o$  either reduces the cost functional  $J_{n,\lambda}^\pi$  for a given precinct or leaves it unchanged. The decisions made for a given precinct,  $\mathcal{P}_n^\pi$ , during  $\Psi_o$  achieve the desired outcome since they are based on correct  $D_{*n}^\pi$  and  $\theta_n^\pi$  at the time that precinct is visited;  $D_{*n}^\pi$  depends on precincts that have already been optimized during this  $\Psi_o$  and  $\theta_n^\pi$  depends on precincts that are yet to be visited so that their associated  $\chi_k^\pi$  values have not been changed since  $\theta_n^\pi$  was computed. Since there must exist a minimum for  $J_\lambda$  and each step of  $\Psi_o$  monotonically reduces  $J_\lambda$ , the process must converge to some minimum albeit not necessarily to a global one.

If actual distortions are used instead of the additive distortion model of (11), convergence is not guaranteed; in fact,

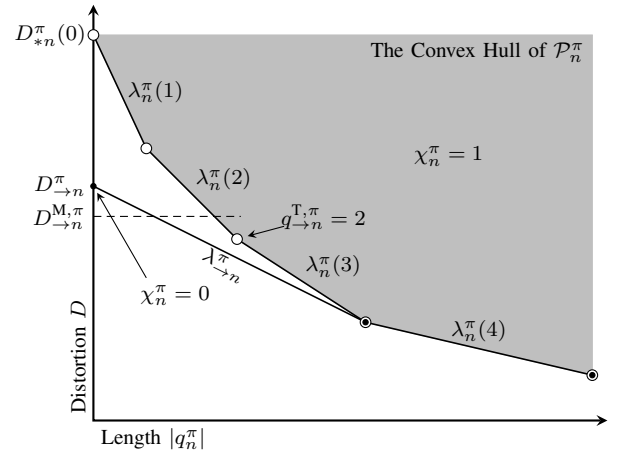


Fig. 4. A typical distortion-length convex hull for a precinct  $\mathcal{P}_n^\pi$ , where each large white circle (O) represents one quality layer. Also shown in the figure is the distortion associated with the predicted version of the precinct,  $D_{*n}^\pi$ , when  $D_{*n}^\pi < D_{*n}^\pi(0)$ ; the small black circles (bullet) represent the modified convex hull.

it can be shown that it does not converge under certain conditions. Despite this, experimental results show that  $\Psi_w \Psi_o$  iterations help in reducing the cost functional even when actual distortions are used.

Next, we give a graphical interpretation and a corresponding solution to (13). Figure 4 depicts a typical rate-distortion curve for a precinct,  $\mathcal{P}_n^\pi$ . It can be easily shown that this curve is a convex since each precinct is made up of convex-by-construction code-blocks using (2). We write  $\lambda_n^\pi(q)$  for the distortion-length slope associated with the quality layers,  $q_n^\pi$ ; that is,

$$\lambda_n^\pi(q) = \frac{D_{*n}^\pi(q-1) - D_{*n}^\pi(q)}{|q| - |q-1|} \quad (14)$$

The figure also shows  $D_{*n}^\pi$  for the case of  $D_{*n}^\pi < D_{*n}^\pi(0)$ . The existence of a predicted version of the precinct,  $\mathcal{P}_{*n}^\pi$ , with distortion  $D_{*n}^\pi$  creates a new distortion-length convex hull. Thus, the distortion-length slopes associated with the first few layers change to  $\hat{\lambda}_{*n}^\pi$ . With this in mind, the complete solution to the minimization of (13) is

$$\chi_n^\pi = \begin{cases} 1, & D_{*n}^\pi > D_{*n}^\pi(0) \text{ or } \lambda \leq \hat{\lambda}_{*n}^\pi \\ 0, & \text{otherwise} \end{cases} \quad (15)$$

$$q_n^\pi = \begin{cases} \max\{q \mid \hat{\lambda}_{*n}^\pi(q) > \lambda\}, & \lambda \leq \hat{\lambda}_{*n}^\pi \\ 0, & \text{otherwise} \end{cases} \quad (16)$$

where  $\hat{\lambda}_{*n}^\pi(q) = (1 + \theta_n^\pi) \cdot \lambda_n^\pi(q)$  and  $\hat{\lambda}_{*n}^\pi = (1 + \theta_n^\pi) \cdot \lambda_{*n}^\pi$ .

## V. ACTUAL CLIENT AND SERVER POLICIES

For the samples of a given precinct, the ultimate objective of the client policy is to achieve (8) by correctly selecting between received samples and predicted samples, possibly from more than one predictor. A good realistic policy should at least be capable of making correct decisions when the difference between the available choices is significant. The main difficulty that faces the client here is how to compare the quality of possibly many candidate precinct samples in order to select the best candidate.

The loose-coupling of client and server policies, first discussed in the introduction, requires any side information that is sent to the client to be universal, by which we mean information that describes some properties of the video sequence being streamed that are always true and independent of the state of the client-server interaction. These properties should allow the client to make reasonably correct decisions under diverse client cache contents.

Here, we propose a client policy and a corresponding server policy that are based on such a universal property, the per-precinct *quality layer threshold*,  $q_{\rightarrow n}^{\text{T},\pi}$ . This threshold, shown in Figure 4, is the first quality layer at which it is better to use received samples than to use predicted samples assuming unquantized prediction source precincts. Specifically,

$$q_{\rightarrow n}^{\text{T},\pi} = \min \{q \mid D_{*n}^{\pi}(q_n^{\pi}) < D_{\rightarrow n}^{\text{M},\pi}\} \quad (17)$$

We remind the reader that  $D_{\rightarrow n}^{\text{M},\pi}$  is obtained from full quality reference frames, and as such,  $D_{\rightarrow n}^{\text{M},\pi}$  represents the best possible result that prediction can produce using this prediction model.

Obviously, this quality layer threshold is related to the prediction model, and therefore each prediction model produces a different threshold. To keep things simple in this work, we choose to limit the possible prediction models for a given precinct to one. Thus, when one frame is predicted from two nearby frames, as in the case of hierarchical B-frames, the only possible predictor is the average of these two frames (that is, we fix  $g_{rn}$  to  $\frac{1}{2}$ ); this also limits the number of thresholds associated with each precinct to one. The efficacy of JSIV when more than one predictor is available have already been demonstrated in our earlier work [8], [18].

Besides using the quality layer threshold, the client policy uses a heuristic to address the fact that reference frames are generally quantized. We write  $q_{R,n}^{\pi}$  for the number of quality layers in the actual reference precinct used in predicting  $\mathcal{P}_n^{\pi}$ ; such a reference precinct could be separated from  $\mathcal{P}_n^{\pi}$  by many frames each with its own  $D_k^{\pi}$  distortions. Thus, each precinct has its own  $q_{R,n}^{\pi}$  and both the client and the server keep track of these values. It is possible that a given precinct is predicted from many precincts using the precinct version of (5); in this case,  $q_{R,n}^{\pi}$  is obtained from

$$q_{R,n}^{\pi} = \left\lceil \sum_{r \in \mathcal{P}_r^{\pi} \in \mathcal{A}(\mathcal{P}_n^{\pi})} g_{rn} \cdot q_{R,r}^{\pi} \right\rceil \quad (18)$$

where  $\lceil \cdot \rceil$  is the ceiling function. Although this approximation is questionable due to the non-linear relation between distortion and number of layers, we find it suitable for the heuristic describe here. With this definition, the complete client policy is

$$P_n^{\pi} = \begin{cases} P_{*n}^{\pi}(q_n^{\pi}), & q_n^{\pi} \geq q_{\rightarrow n}^{\text{T},\pi} \text{ or } q_n^{\pi} \geq \hat{q}_{R,n}^{\pi} \\ P_{\rightarrow n}^{\pi}, & \text{otherwise} \end{cases} \quad (19)$$

where  $\hat{q}_{R,n}^{\pi} = \max\{q_{R,n}^{\pi} - 1, 1\}$ .

The heuristic is explained as follows. If the client receives a number of quality layers equal to the threshold or more,  $q_n^{\pi} \geq q_{\rightarrow n}^{\text{T},\pi}$ , for a given precinct, then the distortion associated with these received samples is smaller than any predicted samples,

in view of (17) and the additive distortion model of (11). If the client receives fewer quality layers than the threshold for a given precinct, then it uses the received samples so long as it has at least  $\hat{q}_{R,n}^{\pi}$  layers. The advantage of using  $\hat{q}_{R,n}^{\pi}$  rather than  $q_{R,n}^{\pi}$  is to allow for graceful degradation in quality over time, while still using transmitted data.

This heuristic is motivated by the practical observation that the condition  $q_n^{\pi} \geq q_{\rightarrow n}^{\text{T},\pi}$  proves problematic at low data rates when the actual reference precinct lies outside the WOF being optimized. Under these conditions the optimization algorithm cannot increase the prediction reference quality, but also cannot usefully send fewer than  $q_{\rightarrow n}^{\text{T},\pi}$  layers for any precinct inside the WOF.

In practice, it is not required for the client to receive all the quality layer thresholds,  $q_{\rightarrow n}^{\text{T},\pi}$ , for all the precincts in each frame; especially when a limited bandwidth is available. Therefore, we send these thresholds for some of the precincts as explained next.

Many ways exist to send the quality layer thresholds to the client, but we propose sending them as one additional JPEG2000 image component per prediction model inside each frame of the video sequence. This choice allows the use of JPIP without any modifications for sending this information to the client. It also allows us to benefit from the features of JPEG2000 such as efficient compression, scalability, and progressive refinement in communicating this information.

Obviously, the quality layer thresholds component is heavily sub-sampled since there is only one threshold per precinct of the regular image components. We use the same number of decomposition levels, quality layers,  $Q$ , and, although not necessary, the same code-block dimensions as those of the original frame. Only one sub-band is needed to store all the thresholds for each resolution level; in practice, we use the HL band leaving the LH and HH bands zero.

The thresholds are encoded using the JPEG2000 block encoder directly. We set the number of coding passes to  $3 \cdot Q - 2$ , and encode  $q_{\rightarrow n}^{\text{T},\pi}$  as  $2^{Q - q_{\rightarrow n}^{\text{T},\pi}}$ . The resulting code-stream is constructed in such a way that each quality layer stores one whole bit-plane.

Side information is delivered to the client using the standard JPIP protocol. We send enough quality layers (or bit-planes) from the thresholds component such that the client is able to deduce  $q_{\rightarrow n}^{\text{T},\pi}$  for all the precincts that have  $q_n^{\pi} \geq q_{\rightarrow n}^{\text{T},\pi}$ ; this is to make sure the the client uses the received samples for these precincts. It is pointless to send the threshold for precincts that have  $q_n^{\pi} < q_{\rightarrow n}^{\text{T},\pi}$  since the client's decision totally depends on  $q_{R,n}^{\pi}$  in this case. Thus, for a code-block,  $C$ , from the quality layer thresholds component, the number of layers,  $\ell_n^C$ , transmitted is

$$\ell_n^C = \max_{\pi \in C} \{1 + q_{\rightarrow n}^{\text{T},\pi} \mid q_n^{\pi} \geq q_{\rightarrow n}^{\text{T},\pi}\} \quad (20)$$

The progressive refinement property of the JPEG2000 format is useful in progressively sending bit-planes (or layers) from the quality layer thresholds component when the need arises. For example, at low bit rate, the server optimization policy selects and sends only a small number of quality layers for the frames of the video sequence, and therefore it is sufficient to



send a small number of layers from the quality layer thresholds component, as given by (20). As the bit rate increases, the server sends more layers for the frames; consequently, it needs to send more layers from the quality layers thresholds component (send the higher  $q_{\rightarrow n}^{T,\pi}$  values).

We turn our attention to the server policy. Server optimization is done in epochs; each epoch corresponds to a fixed time step and a fixed amount of data to be transmitted. In each epoch,  $p$ , all the frames within the corresponding window of frames (WOF) have a chance of contributing data to the transmission. It is possible that one WOF is optimized over more than one consecutive epoch.

In order for the client to use the data it receives from the server for a given precinct, that data must achieve the requirements set out in the first case of (19). Taking note of that, during the rate-distortion optimization pass,  $\Psi_o$ , the number of quality layers in epoch  $p$ ,  $q_n^{p,\pi}$ , can be determined from

$$q_n^{p,\pi} = \begin{cases} \max_{\substack{q \geq q_n^{p-1,\pi} \\ q_n^\pi \geq q_{\rightarrow n}^{T,\pi} \text{ or } q_n^\pi \geq \hat{q}_{R,n}^\pi}} \left\{ q \mid \hat{\lambda}_n^\pi(q) > \lambda \right\}, & \lambda \leq \hat{\lambda}_{\rightarrow n}^\pi \\ 0, & \text{otherwise} \end{cases} \quad (21)$$

This way the server policy works with the client policy to attempt to achieve (8) by making it more favorable to the client to use lower distortion options.

Alternate client and server policies can be obtained by using a pre-determined minimum-quality reference frames for the definition of the quality layer threshold, (17), rather than using full-quality reference frames. We choose not pursue this option here because, with the current definition of the threshold, (17), it is clear that directly decoded samples are better than predicted samples when the client has  $q_{\rightarrow n}^{T,\pi}$  or more quality layers for the precinct in question. With the alternate definition, however, it is possible that predicted samples (with high quality reference frames) are better than directly decoded samples even if the client receives more than the alternate threshold. More importantly, experimental results, shown in Tables I and II under EXACT, reveal that the actual policies presented in this work cause almost no degradation in the quality of reconstructed video compared to the oracle policies of Sections III and IV, which represent an unachievable upper bound on performance. More details about the test sequences and the exact meanings of ‘‘Sequential’’ and ‘‘Hierarchical’’ are available in Section VII.

To achieve a desired rate, a simple rate-control loop is employed. It starts by selecting a value for  $\lambda$  which is used for the  $\Psi_w \Psi_o$  iterations. If the resultant rate for that  $\lambda$  is far-off of the desired rate, another value of  $\lambda$  is selected and the process is repeated until an acceptable resultant rate is achieved. A simple bisection method is employed to find a suitable  $\lambda$  and the side-information overhead is accounted for inside the rate-control loop.

An interesting observation concerning the JSIV optimization algorithm presented here is that it does not generally produce embedded data streams; that is, the optimal representation at a given rate is not necessarily embedded in a higher-rate optimal representation. The lack of embedding has implications for



$\lambda$ (unit <sup>2</sup> /byte)			rate (bytes)								
8000	<table border="1"><tr><td>13</td><td>0</td></tr><tr><td>0</td><td>0</td></tr></table>	13	0	0	0	<table border="1"><tr><td>P</td><td>0</td></tr><tr><td>0</td><td>0</td></tr></table>	P	0	0	0	199
13	0										
0	0										
P	0										
0	0										
7000	<table border="1"><tr><td>12</td><td>0</td></tr><tr><td>0</td><td>0</td></tr></table>	12	0	0	0	<table border="1"><tr><td>13</td><td>0</td></tr><tr><td>0</td><td>0</td></tr></table>	13	0	0	0	344
12	0										
0	0										
13	0										
0	0										

Fig. 5. Lack of embedding in JSIV. The first row shows two frames,  $f_1$  and  $f_2$ , of  $60 \times 60$  pixel resolution compressed using JPEG2000 with  $32 \times 32$  pixel code-blocks and 20 quality layers. The squares in the middle of the second and third rows represent the number of quality layers inside each code-block of the DWT decomposition of each frame; each square is one code-block. The code-blocks belong to the  $LL_1$ ,  $HL_1$ ,  $LH_1$ , and  $HH_1$  sub-bands, arranged from left to right, top to bottom. ‘‘P’’ indicates prediction.

streaming applications in that it requires each epoch to finish its optimization passes before sending any data for that epoch. An embedded stream, on the other hand, can be easily adapted to any desired data rate by simply truncating it at that rate.

We demonstrate this lack of embedding in an example; in this example we show how slightly changing the Lagrange parameter,  $\lambda$ , changes the selected precincts in a non-embedded way. A test sequence of two frames,  $f_1$  and  $f_2$ , with a resolution of  $60 \times 60$  is compressed using JPEG2000 with  $32 \times 32$  pixel code-blocks and 20 quality layers; these frames are shown in the first row of Figure 5. These two frames are jointly optimized subject to the condition that the second frame can be predicted from the first frame. For a Lagrange parameter,  $\lambda$ , of 8000 the  $LL_1$  sub-band of  $f_1$  has 13 quality layers while  $HL_1$ ,  $LH_1$ , and  $HH_1$  sub-bands have zero quality layers. The second frame,  $f_2$ , does not receive any data, and its  $LL_1$  sub-band is predicted from that of  $f_1$ , indicated by ‘‘P’’. For a Lagrange parameter,  $\lambda$ , of 7000 the  $LL_1$  sub-band of  $f_1$  has 12 quality layers while the  $LL_1$  sub-band of  $f_2$  has 13 quality layers. This clearly demonstrates that at a higher  $\lambda$  (lower rate) the optimization algorithm makes a selection that is not embedded in the lower  $\lambda$  (higher rate) selection. The reason behind this is that at the higher  $\lambda$  (lower rate),  $\theta_n^\pi$  of the  $LL_1$  band of  $f_1$  is equal to 1; whereas for lower  $\lambda$  (higher rate), it is equal to 0.

## VI. DISTORTION ESTIMATION AND IMPLEMENTATION COST

In our treatment so far,  $D_{\rightarrow n}^\pi$  is obtained in the server by directly calculating  $\|\mathcal{C}_{\rightarrow n}^\beta - \hat{\mathcal{C}}_n^\beta\|^2$  for all of the code-blocks in that precinct and then employing (7). We refer to this method as EXACT calculations. In the following paragraphs, we introduce an approach based on approximate distortion calculation, which significantly reduces computation in the server.

Consider Figure 2 where  $\mathcal{P}_k^\pi$  is predicted from  $\mathcal{P}_i^\pi$  with a scaling factor of  $g_{ik}$ . We can utilize (3), or its more general form given by (6), to approximate the distortion in  $\mathcal{P}_k^\pi$  without actually calculating it. For the server to be able to approximate this and similar frame arrangements, it must keep tables of distortions,  $D_{*n}^\pi(q_n^\pi)$ , associated with all quality layers, for each

TABLE I  
A COMPARISON BETWEEN DIFFERENT POLICIES FOR THE “SPEEDWAY” SEQUENCE.

Rate <sup>a</sup> (kb/s)	INTRA		Sequential				Hierarchical B-frames			
			Oracle Policy		Actual Policy		Oracle Policy		Actual Policy	
			EXACT	APPEMD	APPROX	EXACT	APPROX	EXACT	APPROX	EXACT
125	25.06	32.80	32.66	30.50	32.72	30.95	32.15	32.23	32.16	32.22
250	27.68	34.91	34.73	32.73	34.89	33.06	34.95	34.98	34.95	34.98
500	30.37	37.18	37.13	35.20	37.21	35.44	37.89	37.87	37.89	37.87
1000	33.27	39.77	39.41	38.02	39.82	38.09	40.82	40.81	40.82	40.81
2000	36.97	41.97	41.59	40.76	41.97	40.78	43.14	43.03	43.14	43.03

<sup>a</sup> To provide a fair comparison, all results reported here are for encoded sub-band samples; they exclude any headers, JPIP, and policy overhead.

TABLE II  
A COMPARISON BETWEEN DIFFERENT POLICIES FOR THE “PROFESSOR” SEQUENCE.

Rate <sup>a</sup> (kb/frame)	INTRA		Sequential				Hierarchical B-frames			
			Oracle Policy		Actual Policy		Oracle Policy		Actual Policy	
			EXACT	APPEMD	APPROX	EXACT	APPROX	EXACT	APPROX	EXACT
40	29.97	34.43	34.22	34.19	34.43	34.27	33.56	33.58	33.56	33.58
80	32.24	37.59	37.39	37.05	37.58	37.17	36.56	36.60	36.55	36.60
160	35.07	40.54	40.47	39.58	40.56	39.69	39.79	39.76	39.78	39.76
240	36.90	42.29	41.93	41.00	42.34	41.07	41.61	41.56	41.61	41.56
320	38.21	43.29	42.85	41.92	43.34	42.05	42.78	42.72	42.80	42.72

<sup>a</sup> To provide a fair comparison, all results reported here are for encoded sub-band samples; they exclude any headers, JPIP, and policy overhead.

precinct of each frame. It must also keep tables for motion distortions,  $D_{\rightarrow n}^{M,\pi}$ , and their associated quality layer thresholds,  $q_{\rightarrow n}^{T,\pi}$ , for all the needed prediction arrangements and perceived playback modes. It is sufficient to keep approximate quantized distortion values, and therefore 2 bytes per entry is more than enough. There is no need to keep tables for the quality layer sizes,  $|q_n^\pi|$ , as these can be easily extracted from code-block headers.

Thus, for each precinct<sup>6</sup>, which usually represents a few thousand samples, we need  $2 \cdot Q$  bytes to store  $D_{*n}^\pi(q_n^\pi)$ , where  $Q$  is the number of quality layers, and 3 bytes for each predictor to store  $D_{\rightarrow n}^{M,\pi}$  and  $q_{\rightarrow n}^{T,\pi}$ . Only some of the  $q_{\rightarrow n}^{T,\pi}$  values are transmitted to the client, as given by (20);  $D_{*n}^\pi(q_n^\pi)$  and  $D_{\rightarrow n}^{M,\pi}$  are used by the server only to estimate distortions (i.e., they are not transmitted).

Next, we investigate the validity of the assumptions we made in (6) and (11). Consider the case of  $\mathcal{P}_n^\pi$  in Figure 2; it is predicted from  $\mathcal{P}_k^\pi$  that is itself predicted from  $\mathcal{P}_i^\pi$ . One approximation, which we refer to as approximate with exact motion distortion (APPEMD), is to always use exact motion distortion; that is,

$$D_{\rightarrow n}^\pi \approx g_{ik}^2 \cdot g_{kn}^2 \cdot D_{*i}^\pi + D_{i \rightarrow n}^{M,\pi} \quad (22)$$

In APPEMD, a reference precinct can occur many frames earlier than the current WOF. We impose a limit on the number of possible reference frames and force a high distortion for prediction sources that are outside this limit. The server policy in this case will replace precincts from frames that are outside

the available reference frames with ones from the current WOF. The limit used, however, is quite large and therefore does not have a measurable impact on the result. Experimental data, shown in Tables I and II under APPEMD, shows only a small degradation in the quality of reconstructed video, less than 0.5dB in the worst case. This supports the earlier claim that quantization distortion and motion distortion are mostly uncorrelated as is assumed in (6).

The other approximation, which we refer to as approximate (APPROX), is used in deriving (11) and is based on the assumption that motion error between  $\mathcal{P}_i^\pi$  and  $\mathcal{P}_k^\pi$  is uncorrelated with the motion error between  $\mathcal{P}_k^\pi$  and  $\mathcal{P}_n^\pi$ , and therefore we can approximate  $\mathcal{P}_n^\pi$ 's distortion by

$$\begin{aligned} D_{\rightarrow n}^\pi &\approx g_{kn}^2 \cdot D_{\rightarrow k}^\pi + D_{k \rightarrow n}^{M,\pi} \\ &\approx g_{kn}^2 \cdot (g_{ik}^2 \cdot D_{*i}^\pi + D_{i \rightarrow k}^{M,\pi}) + D_{k \rightarrow n}^{M,\pi} \end{aligned} \quad (23)$$

Experimental data, shown in Tables I and II under APPROX, shows that there is almost no degradation in the quality of reconstructed video in the case of a hierarchical B-frame prediction arrangement. However, there can be as much as 2dB quality in the case of a sequential prediction arrangement. This suggests that the assumption of uncorrelated motion distortion is not very accurate, especially over an extended sequence of consecutive frames. We believe that it maybe worth sacrificing this reduction in quality, especially when the advantage of JSIV is consistently more than a few dB relative to regular intra-coded video. The main advantages of APPROX are significant storage savings, since APPEMD requires significantly more tables, improved data locality, and lower computational requirements through recursive distortion estimation. Of course, intermediate approaches can be conceived in which exact motion distortions are stored for some

<sup>6</sup>In interactive applications, code-blocks are usually  $32 \times 32$  samples. This gives 2924 precincts in each 4K cinema frame ( $4096 \times 2160$  pixels) when 6 levels of DWT are used, and 45 precincts in each CIF frame ( $352 \times 288$  pixels) when 4 levels of DWT are used.

but not all prediction scenarios.

We turn our attention to the computational requirements of our proposed approach. Using APPROX in the rate-distortion optimization pass,  $\Psi_o$ , to find  $D_{\rightarrow n}^\pi$  from (6) requires  $|\mathcal{A}(\mathcal{P}_n^\pi)|$  additions and  $|\mathcal{A}(\mathcal{P}_n^\pi)|$  multiplications per precinct where  $|\mathcal{A}(\mathcal{P}_n^\pi)|$  is the number of elements in  $\mathcal{A}(\mathcal{P}_n^\pi)$ . To implement (18), we need  $|\mathcal{A}(\mathcal{P}_n^\pi)| - 1$  additions and  $|\mathcal{A}(\mathcal{P}_n^\pi)|$  multiplications per precinct. For the modified convex hull analysis, the Incremental Computation of Convex Hull and Slopes algorithm presented in [27] requires no more than  $2 \cdot Q$  multiplications, where  $Q$  is the number of quality layers. Early termination can also be employed to further reduce computational requirements. For  $\Psi_w$ , finding  $\theta_n^\pi$  from (12) requires  $2 \cdot |\mathcal{S}(\mathcal{P}_n^\pi)| - 1$  additions and  $|\mathcal{S}(\mathcal{P}_n^\pi)|$  multiplications per precinct.

The computational requirements grow linearly with frame size since the number of precincts does so. Obviously, the computational requirements per precinct are small, being on the order of 10 to 20 multiplications and additions where a precinct typically represents several thousand video samples.

### VII. EXPERIMENTAL RESULTS AND USAGE SCENARIOS

In this section we visit a few of the usage scenarios introduced in Section I; we show how JSIV can provide more efficient access to the media than existing approaches. We use two sequences<sup>7</sup>: “Speedway” and “Professor”. These two sequences are chosen because they are surveillance footage and therefore are more suitable for this paper than the standard motion test sequences. The effectiveness of JSIV with standard test sequences have been demonstrated in our earlier work [7]–[11] where JSIV with motion compensation is employed. “Speedway” is a 193 frame sequence<sup>8</sup> that has a resolution of  $352 \times 288$  at 30 frames/s and a bit depth of 8 bits per sample. “Professor” is a 97 frame sequence that has a resolution of  $3008 \times 2000$  captured at one frame every approximately three seconds at a bit depth of 8 bits per sample. Only the Y-component is used for all the tests reported here.

For JSIV, these sequences are converted to JPEG2000 using Kakadu<sup>9</sup>. Three levels of irreversible DWT are employed for “Speedway” and five for “Professor”. A code-block size of  $32 \times 32$  and 20 quality layers are used for both sequences. “Hierarchical” refers to Hierarchical B-frame prediction arrangement, similar to the SVC extension of H.264 [3]; all  $g_{rn}$  values are fixed to  $\frac{1}{2}$  in this case. “Sequential”, on the other hand, refers to arranging the frames such that the WOF = 2; the WOF shifts by one frame after each epoch, and each frame can only be predicted from the frame before it (effectively an “IPP...” arrangement). For the sequential arrangement, we have  $f_{\rightarrow n} = f_{n-1}$ ; that is,  $g_{rn}$  is fixed to 1. All the results reported here use the “Hierarchical” arrangement except Figures 8 and 9, which uses the “Sequential” arrangement. It is important to remember that JSIV never sends residues.

<sup>7</sup>“Speedway” and “Professor” test sequences are available at <http://www.eet.unsw.edu.au/~taubman/sequences.htm>.

<sup>8</sup>It is actually 200 frames but the last 7 frames were dropped to make it more suitable for use with a 3-level hierarchical B-frame prediction arrangement.

<sup>9</sup><http://www.kakadusoftware.com/>, Kakadu software, version 5.2.4.

For INTRA, also known as Motion-JPEG2000, each frame is independently transmitted in an optimal fashion.

For SVC, JSVM<sup>10</sup> is used to compress and reconstruct these sequences. The intra-frame period is set to 8 to match that of JSIV. All the scenarios presented here, except for the scalability scenario, employ three levels of temporal decimation with two enhancement layers. The enhancement layers use two levels of medium-grain scalability (MGS) between them, giving a total of seven quality layers. No spatial scalability option was used for these test; increasing the number of MGS and/or adding spatial scalability would penalize SVC’s performance.

One of the difficulties faced with SVC is the amount of parameters that need to be configured. We started with default options, and we activated all the options that we think improve the coding efficiency such as adaptive inter-layer prediction and context-adaptive binary arithmetic coding (CABAC).

All results are reported in PSNR calculated from the average MSE over the reconstructed sequence. All JSIV results reported use actual policies with 3 passes of  $\Psi_w \Psi_o$  for the hierarchical B-frame prediction arrangement and 2 for the sequential. The rates reported include everything: encoded sub-band samples, JPEG2000 headers, side-information, and JPIP message header overhead.

We compare against SVC because it is considered to be the state of the art compressor with support for scalability. It is important to note that it might be possible to create an SVC-compressed sequence that performs better than JSIV for a given usage scenario, but it is not possible to create an SVC-compressed sequence that performs better than JSIV in all the usage scenarios. The results presented here are biased towards SVC since they do not account for the communication overhead needed to stream SVC, for example from RTP. By contrast, JSIV results include all overhead associated with the highly flexible JPIP protocol.

We start by comparing JSIV performance against that of SVC and INTRA. Figure 6 shows the results for the “Speedway” sequence and Figure 7 shows the result for the “Professor” sequence; in both case, JSIV employs the “Hierarchical” arrangement.

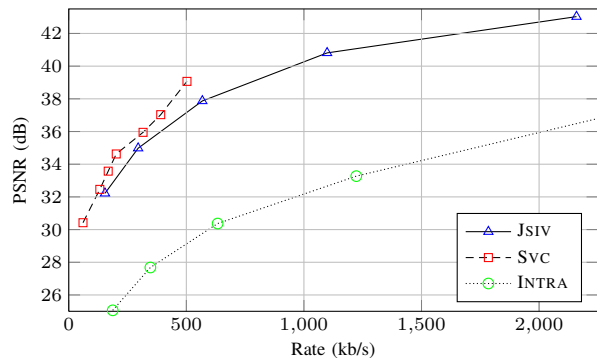


Fig. 6. A comparison of the performance of various schemes for the “Speedway” sequence.

<sup>10</sup>JSVM version 9.18.1 obtained through CVS from its repository at [gacon.ient.rwth-aachen.de](http://gacon.ient.rwth-aachen.de)

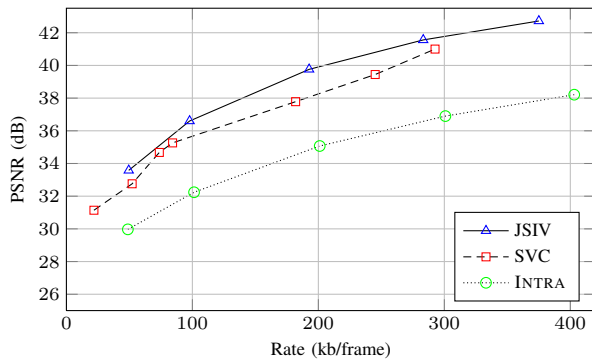


Fig. 7. A comparison of the performance of various schemes for the “Professor” sequence. Note that the x-axis is in (kb/frame).

TABLE III  
OVERHEADS IN JSIV FOR THE “PROFESSOR” SEQUENCE  
IN HIERARCHICAL B-FRAME ARRANGEMENT  
AS A PERCENTAGE OF THE OVERALL RATE.

Rate (kb/frame)	JPIP	Side Information	JPIP for Side Information
49.488	3.305%	4.567%	0.294%
97.819	3.106%	2.770%	0.151%
192.681	2.740%	1.840%	0.082%
283.261	2.403%	1.407%	0.057%
375.028	2.265%	1.200%	0.043%

It can be observed that JSIV, in general, works considerably better than INTRA. Compared to SVC, JSIV works worse for the “Speedway” sequence and better for the “Professor” sequence. The interested reader can refer to [25] for a couple of reconstructed frames from these sequences.

The overheads for hierarchical B-frame prediction arrangement for the “Professor” sequence are shown in Table III, measured against the overall rate. It can be seen that for this arrangement the overhead from JPIP is less than 4% and from side-information is less than 5%. This overhead becomes smaller with increasing data rate. Similar results are obtained for the sequential frame arrangement.

Figures 8 and 9 show the effect of code-block sizes on the quality of the reconstructed video in JSIV for the “Speedway” and “Professor” sequences, respectively, for the sequential frame arrangement; similar results are obtained for the hierarchical B-frame arrangement. The code-block size represents a trade-off between accessibility and compactness of representation; for example, a small block size makes it easier for the server to replace precincts and therefore provides better accessibility at the expense of reduced coding efficiency and increased overhead. It can be seen that the optimal code-block size is  $16 \times 16$  for “Speedway” and  $32 \times 32$  for “Professor”. This difference is expected since the smaller block size provides better interactivity for the lower resolution “Speedway” sequence.

#### A. Individual Frame Retrieval

During an interactive browsing session it is normal for the client to be interested in one frame only; perhaps because that frame shows some event or incident. In a conventional

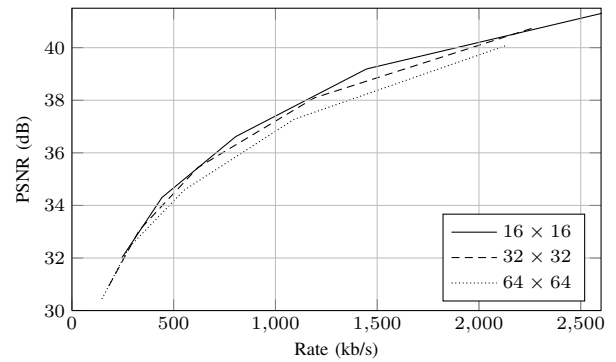


Fig. 8. The effect of code-block size on quality for the “Speedway” sequence when the sequential prediction arrangement is employed.

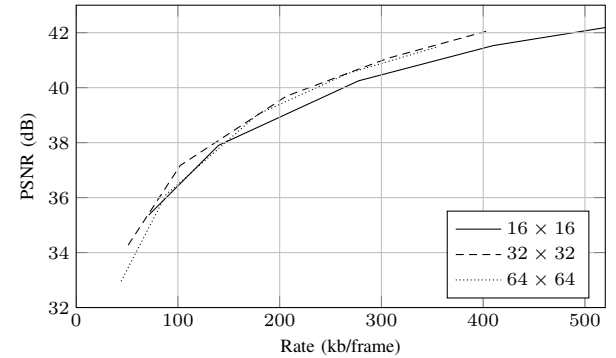


Fig. 9. The effect of code-block size on quality for the “Professor” sequence when the sequential prediction arrangement is employed. Note that the x-axis is in (kb/frame).

predictive coding scheme, the retrieval of a particular frame requires, in general, the retrieval of all of its reference frames, their reference frames, and so on. The number of frames that need to be retrieved depends on the frame arrangement and the position of that particular frame. In JSIV, on the other hand, it is possible to retrieve an individual frame directly.

Figure 10 shows the retrieval of frames 12 and 13 of the “Professor” sequence for both JSIV and SVC cases. For the case of SVC, the rates shown are the sum of the rates for frames 9, 11, 12, 13, and 17 for the case of frame 12; and frames 9, 13, and 17 for the case of frame 13. It should be noted that half the frames in the compressed sequence are in a position similar to that of frame 12. This situation can be much worse for the case of sequential prediction. For JSIV, only the frame of interest is retrieved.

#### B. Spatial and Temporal Scalability

In this subsection we study the performance of JSIV when the server is delivering reduced temporal rate and/or spatial resolution. The SVC encoded sequences here have 4 enhancement layers. The basic layer is at one-sixteenth resolution; the first enhancement layer is at quarter resolution; the second enhancement layer is also at quarter resolution but employs 2 layers of MGS; the third enhancement layer is at full resolution; and the fourth enhancement layer is also at full resolution but employs 2 layers of MGS. For JSIV, the quality of quarter-resolution reconstructed video is measured against

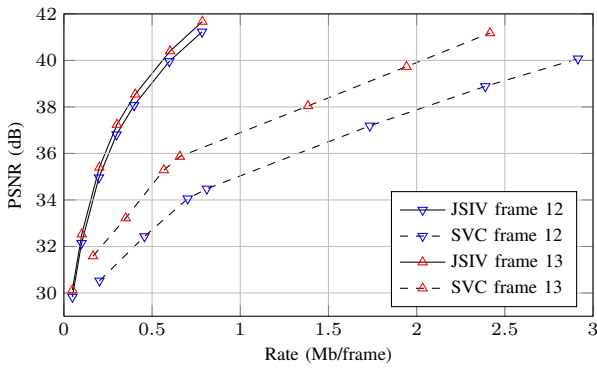


Fig. 10. A comparison between JSIV and SVC when the client is only interested in the retrieval of one frame. Note that the x-axis is in (Mb/frame).

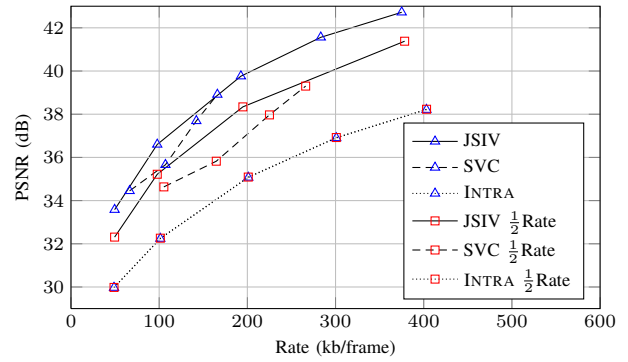


Fig. 12. A comparison of the performance of various schemes at full resolution and full/half temporal frame rate for the “Professor” sequence. Note that the x-axis is in (kb/frame).

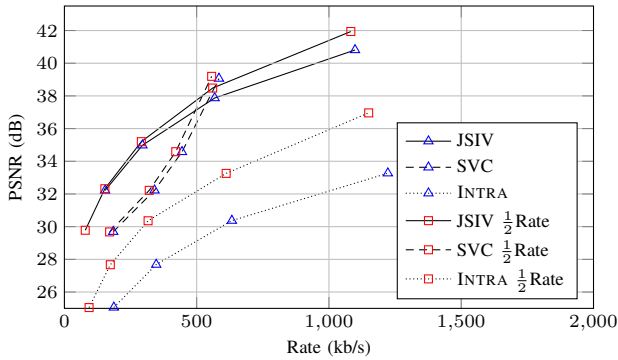


Fig. 11. A comparison of the performance of various schemes at full resolution and full/half temporal frame rate for the “Speedway” sequence.

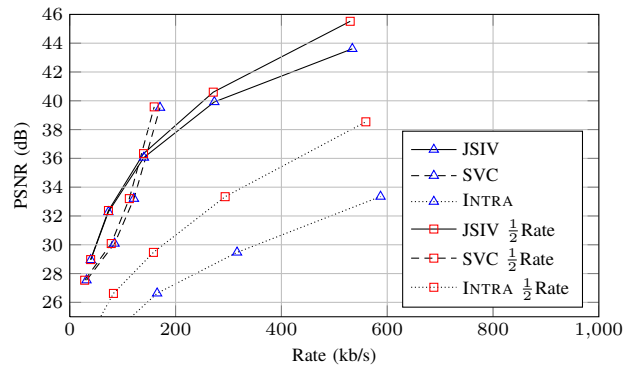


Fig. 13. A comparison of the performance of various schemes at quarter resolution and full/half temporal frame rate for the “Speedway” sequence.

the  $LL_1^{\text{th}}$  resolution of the original sequence while, for SVC, it is measured against reduced-resolution frames generated using JSVM tools.

Figure 11 shows a comparison for the “Speedway” sequence at full resolution with full and half temporal rate. Figure 12 shows the same comparison for the “Professor” sequence. Figure 13 shows a comparison for the “Speedway” sequence at quarter resolution with full and half temporal rate. Figure 14 shows the same comparison for the “Professor” sequence. In all of these case, JSIV employs the “Hierarchical” arrangement.

It can be noticed that SVC’s performance in Figure 11 is lower than that in Figure 6. We suspect that this is due to having more enhancement layers and more resolutions in the case of Figure 11. The same can be said about Figure 12 and Figure 7.

For full resolution, it can be seen that JSIV produces better results than the alternative methods, most of the time. For quarter resolution, however, the result is not consistent, with SVC producing 2-3dB better results for a certain bit-rate range of the “Professor” sequence. This rather big difference can be partially explained by how the spatially-decimated sequences are obtained. For SVC, it is obtained with proper low-pass filtering using one of the tools in the SVC distribution while for JSIV it is obtained by discarding the highest resolution sub-bands. Thus, the JSIV version has more high frequency components and therefore is harder to compress.

### C. Window of Interest

Here, we study the performance of JSIV when the client is interested in an arbitrary spatial region of interest; the region investigated is the left half of each frame<sup>11</sup> from (column 0, row 0) to (column 176, row 288) of the “Speedway” sequence, and from (column 0, row 0) to (column 1504, row 2000) of the “Professor” sequence. Figures 15 shows the results for the “Speedway” sequence, and Figure 16 for the “Professor”

<sup>11</sup>This choice is not aligned to code-block boundaries and therefore is not the most JSIV-favorable option.

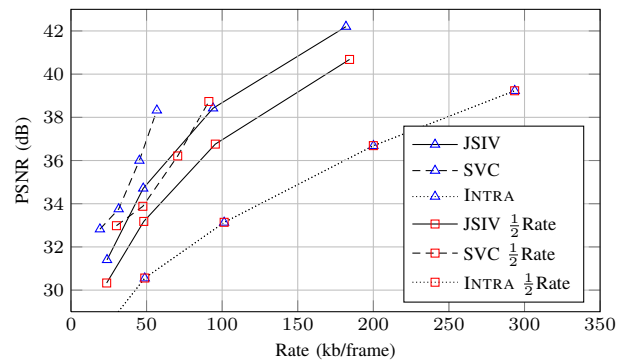


Fig. 14. A comparison of the performance of various schemes at quarter resolution and full/half temporal frame rate for the “Professor” sequence. Note that the x-axis is in (kb/frame).

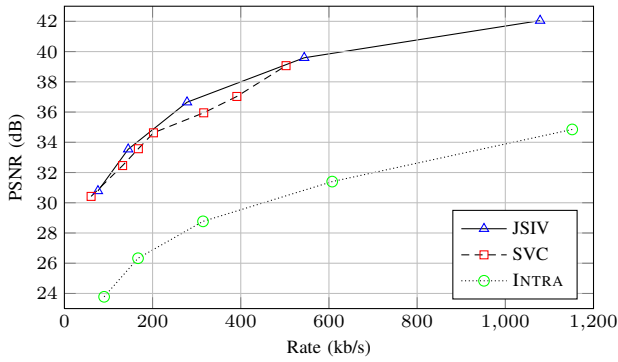


Fig. 15. A comparison between SVC and JSIV when the client is interested in only the left half of each frame of the “Speedway” sequence. SVC does not support random spatial access for pre-compressed sequences.

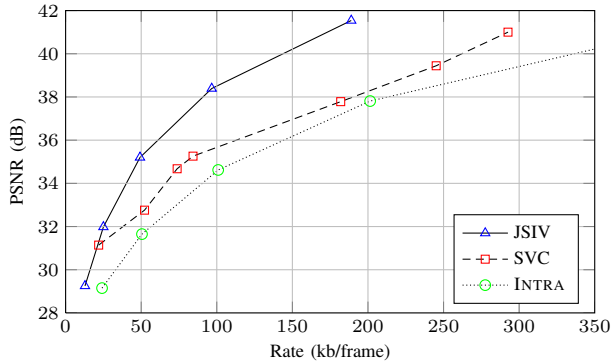


Fig. 16. A comparison between SVC and JSIV when the client is interested in only the left half of each frame of the “Professor” sequence. SVC does not support random spatial access for pre-compressed sequences. Note that the x-axis is in (kb/frame).

sequence; in both cases, JSIV employs the “Hierarchical” arrangement. The results reported for SVC are for full frames as such an option is not available for pre-compressed SVC sequences<sup>12</sup>.

Even with such a large region of interest, experimental data reveals that JSIV performs better than SVC, especially for the “Professor” sequence.

#### D. Use of Available Data

In JSIV, the client can utilize all the data available in its cache for video reconstruction. We consider here the case of a client that has a 48dB quality frame 9 of the “Professor” sequence, being served by a server that is either aware or unaware of this. Frame 9 is one of the independent frames in the hierarchical B-frame prediction arrangement with three levels of temporal decimation. Of course, the aware server tries not to send any information for that particular frame.

The size of the data delivered to the client is around 1.667 Megabits while the file size of frame 9 that is already in the client cache is slightly short of 2.25 Megabits. Figure 17 shows

<sup>12</sup>SVC supports region of interest, but the region has to be known before compressing the sequence; therefore, it cannot be changed after compressing the video. For example, if the SVC sequence is compressed as a left and right halves, it would not be suitable for delivery to a client requesting columns  $W/3$  to  $2 \cdot W/3$ , where  $W$  is the width.

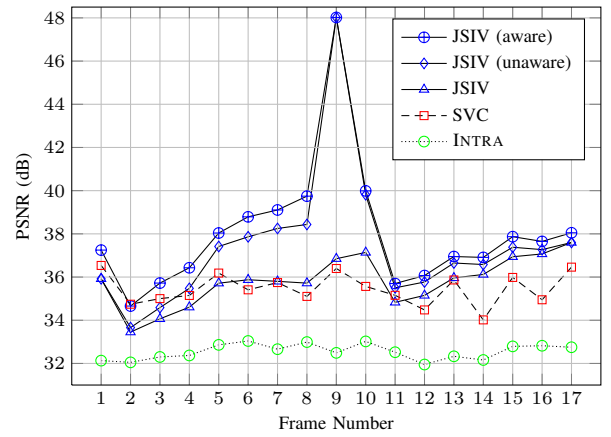


Fig. 17. A comparison among various methods when the client cache has one good quality reference frame (frame 9).

the PSNR of two WOFs around frame 9 for 5 cases: INTRA with a client that has nothing in its cache, SVC, JSIV with a client that has nothing in its cache, JSIV with an unaware server, and JSIV with an aware server.

It can be seen that the availability of a good quality frame can improve nearby frames that are partially predicted from that frame regardless of server awareness. Of course, the best result is obtained when the server is aware of the client’s cache; even independent frames can benefit from the server awareness of the client’s cache, since the server policy allocates the available rate budget differently, as can be seen for frames 1 and 17.

#### E. Data Loss Handling

Providing meaningful experimental results in the case of data loss is hard because they require proper protection for the different parts of the streamed sequence, for all the three schemes being considered. Optimal protection assignment is beyond the scope of this work. We can, however, consider how the system behaves in case of some data loss.

Firstly, we consider the case when a frame reconstruction deadline occurs before all the required data arrives. In this case, concealment techniques must be employed and JSIV should work no worse than SVC, since similar techniques can be employed in both cases.

Secondly, we consider what happens to subsequent dependent frames, whose reconstruction deadline has not yet arrived, so that the server has an opportunity to transmit some data. For SVC, the server has to send some or all the lost data since it is very likely that parts of subsequent frames are predicted from the missing pieces in the corrupted frame. Of course, correct reconstruction of the corrupted frame is useful only for reconstruction of subsequent frames, since its deadline has passed. For JSIV, the server does not need to send the missing parts of the corrupted frame since the client has moved to the subsequent frame and is no longer interested in that frame. The server in this case has to include the effect of the data loss in its client-side distortion estimates and let the optimization algorithm decide how to allocate the available data budget.

### VIII. CONCLUSION AND FUTURE WORK

In this work we have presented JSIV: a novel way of streaming video that provides considerably better interactivity compared to existing streaming schemes. JSIV performance is better than existing schemes in many usage scenarios while having little loss in others.

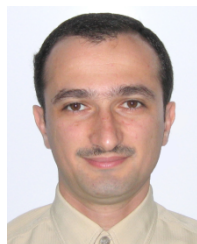
The use of the JPEG2000 format to store individual frames has proven very efficient in providing accessibility and interactivity while Lagrange-style rate-distortion optimization of delivered data exploits a good deal of the temporal redundancy. Storing side-information as meta-images allows the use of the standard JPIP protocol to send this information as well as streaming the video itself.

The proposed scheme's main advantage comes from not committing to a predetermined prediction policy. This allows the server to dynamically and adaptively change its policy to track clients' needs, requiring only modest computation. The use of loosely-coupled policies makes it possible for the client and the server to work independently, which is especially beneficial for cases where the server cannot immediately be aware of the client's cache.

This work is part of an ongoing investigation in this area. In the next work we provide a detailed description of JSIV with motion compensation.

### REFERENCES

- [1] J.-R. Ohm, "Advances in scalable video coding," *Proc. of the IEEE*, vol. 93, January 2005.
- [2] N. Mehrseresht and D. Taubman, "An efficient content-adaptive motion compensated 3D-DWT with enhanced spatial and temporal scalability," *IEEE Trans. Image Proc.*, pp. 1397–1412, June 2006.
- [3] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, September 2007.
- [4] ISO/IEC 15444-9, "Information technology – JPEG 2000 image coding system – Part 9: Interactivity tools, APIs and protocols," 2004.
- [5] D. Taubman and R. Prandolini, "Architecture, philosophy and performance of jpip: internet protocol standard for JPEG 2000," *Int. Symp. Visual Comm. and Image Proc.*, vol. 5150, pp. 649–663, July 2003.
- [6] ISO/IEC 15444-3, "Information technology – JPEG 2000 image coding system – Part 3: Motion JPEG 2000," 2007.
- [7] A. Naman and D. Taubman, "A novel paradigm for optimized scalable video transmission based on JPEG2000 with motion," *Proc. IEEE Int. Conf. Image Proc. 2007*, pp. V–93 – V–96, Septmeber 2007.
- [8] —, "Optimized scalable video transmission based on conditional replenishment of JPEG2000 code-blocks with motion compensation," *MV '07: Proceedings of the International Workshop on Workshop on Mobile Video*, pp. 43–48, Septmeber 2007.
- [9] —, "Rate-distortion optimized delivery of JPEG2000 compressed video with hierarchical motion side information," *Proc. IEEE Int. Conf. Image Proc. 2008*, pp. 2312–2315, October 2008.
- [10] —, "Distortion estimation for optimized delivery of JPEG2000 compressed video with motion," *IEEE 10th Workshop on Multimedia Signal Processing, 2008, MMSP 2008*, pp. 433–438, October 2008.
- [11] —, "Rate-distortion optimized JPEG2000-based scalable interactive video (JSIV) with motion and quantization bin side-information," *Proc. IEEE Int. Conf. Image Proc. 2009*, pp. 3081–3084, November 2009.
- [12] N.-M. Cheung and A. Ortega, "Flexible video decoding: A distributed source coding approach," *IEEE 9th Workshop on Multimedia Signal Processing, 2007, MMSP 2007.*, pp. 103–106, October 2007.
- [13] —, "Compression algorithms for flexible video decoding," *Visual Communications and Image Processing 2008*, vol. 6822, no. 1, p. 68221S, 2008.
- [14] P. Zanuttigh, N. Brusco, D. Taubman, and G. Cortelazzo, "A novel framework for the interactive transmission of 3D scenes," *Signal Processing: Image Communication, Special Issue on Interactive Representation of Still and Dynamic Scenes*, vol. 21, no. 9, pp. 787 – 811, 2006.
- [15] F.-O. Devaux, J. Meessen, C. Parisot, J.-F. Delaigle, B. Macq, and C. De Vleeschouwer, "Remote interactive browsing of video surveillance content based on JPEG 2000," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 8, pp. 1143–1157, August 2009.
- [16] F.-O. Devaux and C. De Vleeschouwer, "Parity bit replenishment for JPEG 2000-based video streaming," *EURASIP Journal on Image and Video Processing*, vol. 2009, 2009.
- [17] A. Mavlankar, J. Noh, P. Baccichet, and B. Girod, "Peer-to-peer multicast live video streaming with interactive virtual pan/tilt/zoom functionality," *Proc. IEEE Int. Conf. Image Proc. 2008*, pp. 2296–2299, October 2008.
- [18] A. Naman and D. Taubman, "Predictor selection using quantization intervals in JPEG2000-based scalable interactive video (JSIV)," *Proc. IEEE Int. Conf. Image Proc. 2010*, September 2010, accepted for publication.
- [19] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Proc.*, vol. 9, no. 7, pp. 1158–1170, July 2000.
- [20] H. Everett, III, "Generalized lagrange multiplier method for solving problems of optimum allocation of resources," *Operations Research*, vol. 11, no. 3, pp. 399–417, 1963.
- [21] K. Ramchandran, A. Ortega, and M. Vetterli, "Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders," *Image Processing, IEEE Transactions on*, vol. 3, no. 5, pp. 533–545, September 1994.
- [22] H.-J. Lee, T. Chiang, and Y.-Q. Zhang, "Scalable rate control for MPEG-4 video," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 10, no. 6, pp. 878–894, September 2000.
- [23] J. H. Kim, J. Garcia, and A. Ortega, "Dependent bit allocation in multiview video coding," *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 2, pp. II – 293–96, September 2005.
- [24] A. Naman and D. Taubman, "JPEG2000-based scalable interactive video (JSIV) with motion compensation," *Image Processing, IEEE Transactions on*, Submitted for publication.
- [25] A. Naman, "JPEG2000-based scalable interactive video (JSIV)," Ph.D. dissertation, School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW 2052, Australia, submitted for examination.
- [26] J. Bang-Jensen and G. Gutin, *Digraphs: Theory, Algorithms and Applications*, 2nd ed., ser. Springer monographs in mathematics. London: Springer-Verlag, 2009.
- [27] D. Taubman and M. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Boston: Kluwer Academic Publishers, 2002.



**Aous Thabit Naman** received B.Sc. degree in Electronics and Telecommunication Engineering from Al-Nahrain University, Baghdad, Iraq, in 1994, M.Eng.Sc. degree in Engineering from University of Malaya, Kuala Lumpur, Malaysia, in 2000. He is currently studying for the Ph.D. degree in Electrical Engineering at the School of Electrical Engineering and Telecommunications, Faculty of Engineering, the University of New South Wales, Sydney, Australia.



**David Taubman** (M'92) received the B.S. and B.Eng. degrees from the University of Sydney, Sydney, Australia, in 1986 and 1988, respectively, and the M.S. and Ph.D. degrees from the University of California, Berkeley, in 1992 and 1994, respectively.

From 1994 to 1998, he was with Hewlett-Packard's Research Laboratories, Palo Alto, CA, joining the University of New South Wales, Sydney, in 1998, where he is a Professor with the School of Electrical Engineering and Telecommunications.

He is the coauthor, with M. Marcellin, of the book *JPEG2000: Image Compression Fundamentals, Standards and Practice* (Boston, MA: Kluwer, 2001). His research interests include highly scalable image and video compression, inverse problems in imaging, perceptual modeling, joint source/channel coding, and multimedia distribution systems.

Dr. Taubman was awarded the University Medal from the University of Sydney; the Institute of Engineers, Australia, Prize; and the Texas Instruments Prize for Digital Signal Processing, all in 1998. He has received two Best Paper awards, one from the IEEE Circuits and Systems Society for the 1996 paper, "A Common Framework for Rate and Distortion Based Scaling of Highly Scalable Compressed Video," and from the IEEE Signal Processing Society for the 2000 paper, "High Performance Scalable Image Compression with EBCOT."