Optimized Scalable Video Transmission Based on Conditional Replenishment of JPEG2000 Code-blocks with Motion Compensation

Aous Thabit Naman
School of Electrical Engineering and Telecommunications
University of New South Wales
Sydney, NSW 2052, Australia
aous@student.unsw.edu.au

David Taubman
School of Electrical Engineering and Telecommunications
University of New South Wales
Sydney, NSW 2052, Australia
d.taubman@unsw.edu.au

ABSTRACT

A video stored as a sequence of JPEG2000 images can provide the scalability, flexibility, and accessibility that is lacking in current predictive motion-compensated video coding standards; however, streaming this sequence would consume considerably more bandwidth. This paper presents a new method for optimized streaming of a JPEG2000 video that relies on motion compensation and server-optimized conditional replenishment to reduce temporal redundancy, in collaboration with an intelligent client policy for reconstructing the available content. In particular, we propose transmission of motion vectors and an optimized number of layers, possibly zero, for each code-block of the JPEG2000 representation of each new frame. We also propose the use of a sliding window to optimize a group of frames such that code-blocks of these frames have more than one chance of being enhanced if that is beneficial to subsequent frames. Rate-distortion optimization in the Lagrangian sense is employed to achieve the lowest possible MSE. It is expected that mobile clients with their limited processing powers would benefit from this work in real-time and interactive applications, such as teleconferencing and surveillance. This paper introduces the concept, formulates optimization criteria, and compares the performance with alternative strategies.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms

Keywords

Teleconferencing, video signal processing, image coding, image communication

MV'07, September 28, 2007, Augsburg, Bavaria, Germany.

1. INTRODUCTION

One way of classifying existing digital video coding standards is by their scalability. Standards such as MPEG-1 through MPEG-4 and H.261 through H.264 have been very successful; however, they provide at best limited scalability capabilities. Conversely, a scalable coder aims at producing layered or embedded bit-streams with progressively higher quality, resolution, and frame rate. Although scalable video coding has shown progress in the last few years [2] [5] [4], it still has certain issues and it imposes certain restrictions on streaming. For example, if the client were interest in one particular frame only, the server would have to transmit the group of frames that are needed to invert the motion compensated transform and extract that frame.

In our earlier work [3], we proposed a new paradigm for serving video; in which, we rely on JPEG 2000 to independently compress the original video frames, providing quality scalability and spatial resolution scalability. To exploit the inter-frame redundancy, we rely on a server policy to optimally select the number of quality layer for each code-block transmitted and on a client policy to make the most of the received (distorted) frames and a (possibly client-specific) motion model.

In [3], we restricted our attention to the case where redundancy is exploited only within each pair of frames. By contrast, the current paper extends the approach to the case in which the server optimization policy is run over a sliding window of frames; this potentially allows for the exploitation of redundancy between any two frames in the window.

Some ideas presented in this paper and [3] are new; however, some related concepts can be found in a recent work about interactive browsing of 3D scenes by one of the authors, which has shown promising results [8]. In that work, the server dynamically selects JPEG2000 code-block layers to deliver to the client, so that the client achieves the best quality based on what the server knows about the client cache. In our present work, the transmission history for frames within the server's optimization window plays the same role as the client cache model in [8].

Another recent work [1] investigates a problem similar to the one in this paper. That work focuses on delivering video to a dumb client, lacking the distortion-sensitive client reconstruction policy proposed here. By contrast with our proposed approach, [1] does not take advantage of motion compensation to reduce temporal redundancy. Our pro-

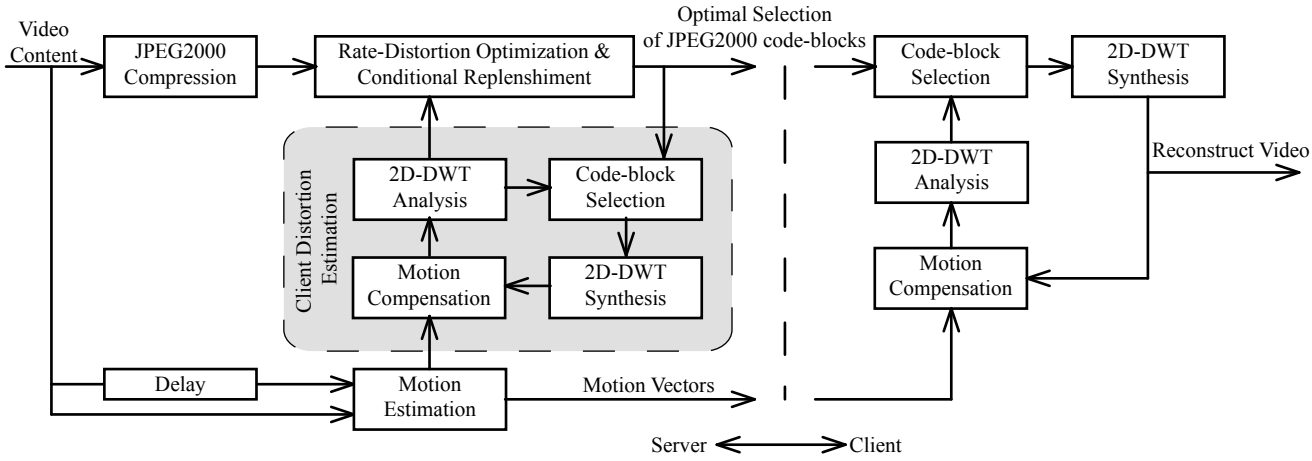


Figure 1: A simplified block diagram of the proposed JPEG2000 video streaming system.

posed server policy has the option to improve the quality of previous frames within the sliding window, wherever this contributes directly or indirectly (e.g., through re-use in subsequent frames) to the expected reconstruction quality. Finally, we allow for code-blocks in a current frame to be predicted from multiple earlier frames, if this proves favorable.

We now briefly consider some benefits of such a delivery system for mobile clients. More advantages of this paradigm are discussed in Section 6. For clients with limited processing power, the proposed paradigm gives the capability of processing and displaying only a desired region of interest within the video. It also enables the server to stream an arbitrary region without the need to re-compress the video. Mobile clients commonly rely on wireless communication links with high loss ratio; predictive coding is less suitable for such clients as it requires the server to re-send all the lost packets. Using the proposed paradigm, however, the server can deal with loss by adjusting its delivery policy for future frames alone, unburdened by the need to correct frames that are no longer current. There is also no need for the server to use the same motion model with every client, which provides more options to handle clients with diverse needs and channel conditions.

The remainder of the paper is structured as follows. Section 2 gives the block diagram of the system under consideration. Sections 3 and 4 elaborate on our client and server policies explaining their theoretical aspects. Section 5 provides preliminary experimental results. Section 6 discusses the benefits of the proposed paradigm further, and elaborates on some contexts where it is applicable. Finally, section 7 states our conclusions.

2. SYSTEM OVERVIEW

Figure 1 depicts a simplified block diagram of the system under consideration; we intentionally leave out the sliding window to simplify the explanation. The first step for each new frame f_n is to apply JPEG2000 compression and find the resulting rate-distortion slopes for the various layers of each code-block. At the same time, this frame together with an earlier frame are used for motion estimation. It is important to note f_{n-1} does not necessarily refer to the frame which immediately precedes f_n in time; the actual identity of f_{n-1} depends on playback direction and frame rate, which can be decided on the fly for each client.

The client distortion estimation block attempts to model the distortion of each code-block in each frame, based on the history of transmitted information. This model could also be adapted to reflect knowledge of the network conditions, client browsing preferences and the client browsing cache. In this paper, our estimation is exact since we ignore packet loss and assume that the client rendering policy is identical to that modeled in the server. In general, this is not the case as the server cannot and need not have an exact nor a very accurate estimate of the client distortion. It is preferable that the server estimates distortion utilizing precomputed lookup tables. These lookup tables can be generated after compression for off-line applications, while for real-time browsing a different server policy might be followed. Our purpose in this paper is to simplify the optimization conditions sufficiently to obtain a reliable upper bound for the performance that can be expected of a more realistic system. We shall have more to say about how meaningful these bounds are in Section 5.

The rate-distortion optimization block of Figure 1 is responsible for Lagrangian rate-distortion optimization, while the code-block selection block selects code-blocks either from the current frame or from motion-compensated previous frames, whichever produces less distortion.

3. CLIENT POLICY

For each code-block C_n^β of each frame f_n, the client receives a number of quality layers q_n^β, possibly 0. We write $\tilde{C}_n^\beta$ for the dequantized samples which can be recovered from these layers and $\tilde{D}_n^\beta = \left\| \tilde{C}_n^\beta - C_n^\beta \right\|^2$ for the corresponding distortion.

As an alternative to using $\tilde{C}_n^\beta$ in frame f_n, the client has the choice of using any of the corresponding sub-band samples, $\tilde{C}_{n-h\to n}^\beta$ to $\tilde{C}_{n-1\to n}^\beta$, obtained from motion compensating the optimized previous h frames; f_{n-h} to f_{n-1}. The tilde-notation here is used to remind the reader that this version also involves quantization distortion. Let us denote the minimum distortion associated with these code-blocks

by

$$\tilde{D}^{\beta}_{\rightarrow n} = \min_{n-h \leqslant s < n} \left\{ \tilde{D}^{\beta}_{s \rightarrow n} \right\}$$

where $\tilde{D}^{\beta}_{s \rightarrow n} = \left\| \tilde{C}^{\beta}_{s \rightarrow n} - C^{\beta}_{n} \right\|^2$. Ideally, a good client policy would select the pieces that result in the least possible distortion; thus, the minimum client distortion $D^{\beta}_{\min,n}$ for code-block C^{β}_{n} would be

$$D^{\beta}_{\min,n}(q^{\beta}_{n}) = \min \left\{ \tilde{D}^{\beta}_{\rightarrow n}, \tilde{D}^{\beta}_{n}(q^{\beta}_{n}) \right\} \quad (1)$$

The method proposed here is not practical since the client needs to have extensive statistics about the frames in order to make the right decisions; however, the results presented here serve as guidelines to the maximum expected benefits of such a system. Realistic implementations need to employ approximations in order to reduce the amount of frame statistics that might need to be sent to the client and reduce the processing time of the client. Indeed, we proposed in [3] an approximation that leads to a simple, effective and realistic implementation for the server and the client. Another approximation that can benefit the client policy is the estimation of the rate-distortion slopes and the total distortion associated with a JPEG2000 stream from the code-blocks themselves without access to the full quality frames or any side information supplied by the server [6].

4. SERVER POLICY

Server optimization is performed at epochs, over a sliding window $\mathcal{W}_k$, which consists of w frames, f_{k-w+1} through f_k. In our current implementation each window has a unique epoch, with the window advancing by one frame between epochs. Each frame f_n within $\mathcal{W}_k$ has a chance to contribute data to the current transmission. Apart from the latest frame f_k, any code-block C^{β}_{n} of any frame within the window might have received some number of quality layers $q^{p,\beta}_{n}$ from an earlier optimization epoch p. For bandwidth control, the server employs a leaky bucket strategy. In the optimization epoch for $\mathcal{W}_k$, the server is able to add $B - F_k$ bytes to the bucket where B is the bucket capacity and F_k is the current bucket fullness. Data is transmitted at a constant rate of L bytes/frame, which empties the bucket, making room for the optimization algorithm to allocate additional data in future epochs.

Using an additive model for distortion and equation (1), the client distortion for the frames within the sliding window can be expressed by

$$D = \sum_{n=k-w+1}^{k} \sum_{\beta \in f_n} D^{\beta}_{\min,n}(q^{\beta}_{n})$$

The minimization of D can be (approximately) recast as the minimization of a family of Lagrangian functionals,

$$J_{\lambda} = \sum_{n=k-w+1}^{k} \sum_{\beta \in f_n} \left(D^{\beta}_{\min,n}(q^{\beta}_{n}) + \lambda \cdot \left| q^{\beta}_{n} \right| \right) \quad (2)$$

where $\left| q^{\beta}_{n} \right|$ denotes the number of bytes in q^{β}_{n} layers of C^{β}_{n} and the Lagrangian parameter λ is adjusted until the solution which minimizes J_{λ} satisfies the length constraint $B - F_k$.

Direct optimization of equation (2) appears difficult, since each $D^{\beta}_{\min,n}(q^{\beta}_{n})$, except when $n = k - w + 1$, depends in a complicated way on earlier frames that are also being optimized. A near optimal solution, however, is possible. Consider the oldest frame in the sliding window at $n = k-w+1$, which depends only on earlier frames that are outside the optimization window. For a given value of λ, q^{β}_{k-w+1} can be easily determined as explained in the following subsections. Once these quality layers are determined, $\tilde{D}^{\beta}_{k-w+1 \rightarrow k-w+2}$ can be calculated, at which point q^{β}_{k-w+2} can be determined, since all of its preceding frames are known. This process is repeated until all q^{β}_{n} within the sliding window are determined.

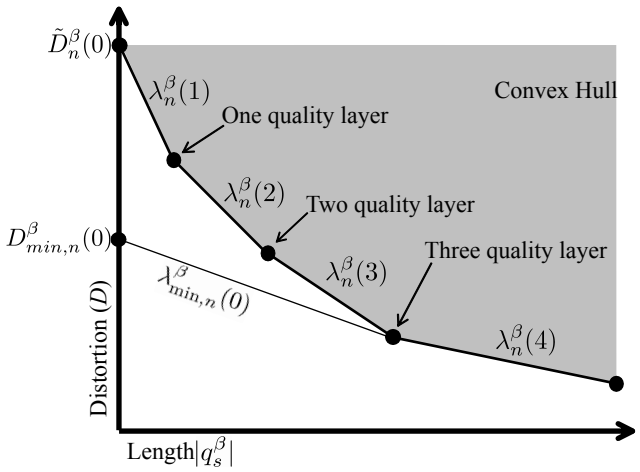


Figure 2: A typical rate-distortion curve for a code-block C^{β}_{p} showing the quality layers and $D^{\beta}_{\min,n}(0)$ when $\tilde{D}^{\beta}_{\rightarrow n} < \tilde{D}^{\beta}_{n}(0)$.

It should be noted that each code-block $\tilde{C}^{\beta}_{n}$ may contribute not only to frame f_n, but also to future frames. These later contributions generally involve motion compensation, which spreads the influence of the distortion in $\tilde{C}^{\beta}_{n}$ over multiple code-blocks in the future frames. In order to properly account for these future frame contributions, the $\tilde{D}^{\beta}_{n}$ terms from highly re-used code-blocks should be weighted more heavily in equation (2). We refer the reader to [3] for more details on how to find these weights. With a window size of $w = 1$, it is not possible to discover the extent to which a code-block influences future frames in the epoch which assigns its quality. For longer windows, however, an iterative multi-pass strategy can take advantage of such information to assign meaningful distortion weights and refine the associated code-block contributions. For the present paper, we choose not to pursue the multi-pass approach mainly for simplicity. Our initial investigations in this direction show that the multi-pass algorithm yields only small improvements with the sequences considered in Section 5.

We turn our attention now to the problem of finding q^{β}_{n}. Initially, we set each $q^{0,\beta}_{n} = 0$. Figure 2 shows a typical rate-distortion curve for a code-block C^{β}_{n}. Each circle on the figure represents one quality layer. Also shown on the figure is the distortion $\tilde{D}^{\beta}_{\min,n}(0)$ when $\tilde{D}^{\beta}_{\rightarrow n} < \tilde{D}^{\beta}_{n}(0)$, which can be achieved by using motion compensated prediction instead of $\tilde{C}^{\beta}_{n}$. If the previous frames are empty, non-existent, or if

$\tilde{D}^{\beta}_{\rightarrow n} \geqslant \tilde{D}^{\beta}_{n}(0)$, the optimal choice of q^{β}_{n} is given by

$$q^{p,\beta}_{n} = \max_{q \geqslant q^{p-1,\beta}_{n}} \left\{ q \mid \lambda^{\beta}_{n}(q) > \lambda \right\} \quad (3)$$

where $\lambda^{\beta}_{n}(q) = (\tilde{D}^{\beta}_{n}(q-1) - \tilde{D}^{\beta}_{n}(q))/(|q| - |q-1|)$ is the distortion length slope associated with layer q. The reason for insisting that $q \geqslant q^{p-1,\beta}_{n}$ is that the server adds all allocated code-block data to the leaky bucket immediately after each optimization epoch. We do not allow the server to withdraw data from the bucket, since that would impose restrictions on the physical embodiment of the leaky bucket abstraction. Note that distortion-length characteristics, $\tilde{D}^{\beta}_{n}(q^{\beta}_{n})$ vs. $|q^{\beta}_{n}|$, of any code-block in any JPEG2000 image is convex by construction [7].

If $\tilde{D}^{\beta}_{\rightarrow n} < \tilde{D}^{\beta}_{n}(0)$, the availability of previous frames reduces the effective distortion associated with the choice $q^{\beta}_{n} = 0$ to $D^{\beta}_{\min,n}(0)$. Figure 2 shows the impact of this option on the effective distortion-length convex hull. The new maximum distortion-length slope is written as $\lambda^{\beta}_{\min,n}(0)$ and the optimal choice for $q^{p,\beta}_{n}$ becomes

$$q^{p,\beta}_{n} = \begin{cases} 0 & \text{if } \lambda > \lambda^{\beta}_{\min,n}(0) \\ \max\limits_{q \geqslant q^{p-1,\beta}_{n}} \{ q \mid \lambda^{\beta}_{n}(q) > \lambda \} & \text{if } \lambda \leqslant \lambda^{\beta}_{\min,n}(0) \end{cases} \quad (4)$$

Obviously, the quality of a frame improves progressively since a frame can only get more quality layers with each optimization pass. Therefore, the client has more than one opportunity for displaying a frame and it would reconstruct a better-quality video if it can afford to wait. As a result, it would be pointless to use long windows for real-time applications.

5. EXPERIMENTAL RESULTS

The results presented here are divided into three subsections. The first part compares the results obtained here to our earlier method [3]. The second part elaborates on the performance for various window sizes and history lengths. The last part investigates the impact of using motion compensation and compares our results with those of [1].

The results are for two different sequences. The first sequence is from the DCI StEM content[1], which is referred to as "Clip 2" during the DCI compression test, starting from frame 6000 up to 6033, with filenames MM_4K_XYZ_06000.tif up to MM_4K_XYZ_06033.tif. The images have a 4096×1714 resolution with a bit-depth of 16-bit per component in the XYZ domain at 24 frames per second, however, only the Y component is used here, after being gamma-corrected and truncated to 8 bits. Also, the images are cropped to 4096×1712 by removing the bottom two rows due to limitations in the available motion compensation subroutines. This sequence is referred to as *sequence 1* in this work.

The second sequence was used in [1][2]. It is a 200 frame CIF surveillance sequence captured with a fixed camera at 25 fps. It is in the YUV space; however, only the Y component is used here, with 8 bit precision.This sequence is referred to as *sequence 2* in this work.

[1]Digital Cinema Initiatives (DCI) and American Society of Cinematographers (ASC), StEM mini-movie access procedure available at http://www.dcimovies.com/.

[2]Speedway sequence available from Wireless Cameras and Audio-Visual Seamless Networking Project website; http://www.ist-wcam.org.

Both sequences are compressed into the JPEG2000 format using the Kakadu implementation[3] employing five levels of wavelet decomposition, 20 quality layers, and a code-block size of 32×32. The motion-compensation routines are mesh-based and applied only at the full frame resolution, in 16×16 patches. Each patch is divided into an upper and lower triangle and affine transformation is used. Sub-pixel values are obtained using the traditional cubic spline filtering and motion is estimated to 1/8 of a pixel accuracy.

It is important to note that the rates given here are only for the encoded sub-band samples; they exclude any headers, motion vectors, and signaling to the client, etc. Also, all the calculations and selections are performed on a code-block basis rather than precincts.

5.1 Comparison with our previous work

In [3], we considered only the special case in which each pair of frames has one motion model, which can be used to produce a motion compensated estimate of the second frame from the first frame. A joint fixed byte budget was allocated to the code-blocks of each pair of frames, so as to minimize the expected distortion.

Figure 3 shows the peak signal-to-noise ratio (PSNR) for sequence 1 at 960kb/s for four methods. The first method, referred to as INTRA, is to optimize each frame individually. The second one is from our previous work [3], referred to as REAL-P. The third one optimizes a pair of frames exactly similar to [3] but using the optimization algorithm proposed in this paper, referred to as IDEAL-P. Finally, the last one applies the general algorithm proposed here to a sequence of frames using a window length of four and a history of one, referred to as IDEAL-WIN. Table 1 shows the PSNR for these methods at various bit rates, which is calculated from the average mean squared error (MSE) of the entire sequence. Figure 4(a) shows a part of the original frame 10, whereas, Figures 4(b) and 4(c) show the reconstruction of the same area at 960kb/s using the INTRA and IDEAL-WIN methods respectively.

Table 1: Comparison between the method presented in [3] and the method presented in this paper at various bit rates. All values are PSNR in dB of the average MSE.

	Bit Rate kb/s				
Method	960	2880	5760	11520	17280
INTRA	29.33	34.11	37.23	39.62	40.60
REAL-P [3]	31.43	35.63	38.03	39.79	40.61
IDEAL-P	31.50	35.58	38.07	39.86	40.72
IDEAL-WIN	32.53	36.20	38.34	39.94	40.75

It is important to remind the reader that REAL-P is a realistic method while IDEAL-P is idealistic and serves as a guideline only. A comparison between their results reveals that the approximations we introduced in [3] are reasonable and perform well. For the IDEAL-WIN method, we observe significantly higher quality than both REAL-P and IDEAL-P indicating that a realistic implementation of IDEAL-WIN would very likely perform better than REAL-P.

In [3], we compared the approach termed REAL-P with a traditional predictive strategy, in which the second frame of

[3]http://www.kakadusoftware.com/, kakadu software, version 5.2.4

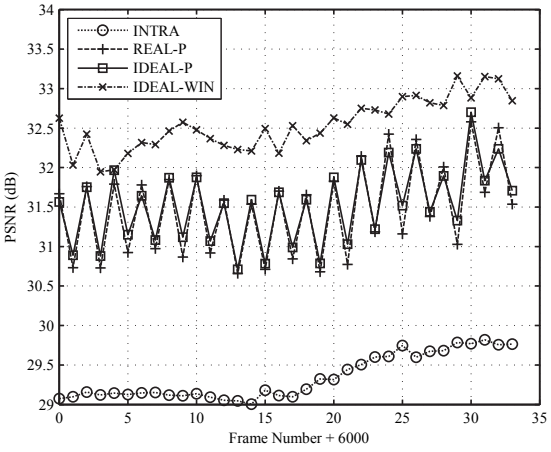
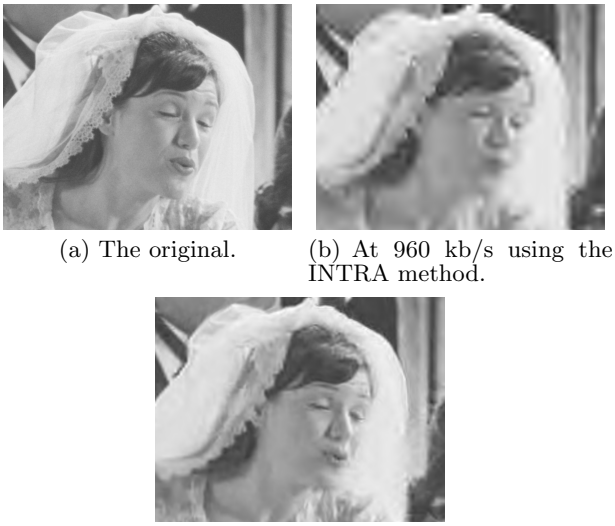


Figure 3: PSNR comparison between the method presented in [3] and the method presented in this paper at 960kb/s bit rate.



(a) The original. (b) At 960 kb/s using the INTRA method. (c) At 960 kb/s using the IDEAL-WIN method.

Figure 4: Part of frame 10 of sequence 1 reconstruct using different methods.

each pair is predicted from the first after motion compensation and only the prediction residual is compressed. Such a comparison cannot meaningfully be made in the case of the moving window, since an equivalent predictive coding scheme would have infinite memory, leading to unbounded drift when reconstructed at reduced quality.

5.2 Performance for various history and window lengths

Window and history lengths of up to four are used in our testing. We find that increasing the history produces only minor improvements in the PSNR for the video sequences examined here. Increasing the window length provides a better management of the allocated byte budget as the algorithm becomes less greedy. Figure 5 shows that as the window length increases, the fluctuations in the PSNR decrease. Figure 5(a) also shows how the PSNR quickly improves once the optimization algorithm is able to exploit the availability of previous frames.

5.3 Comparison for the case of no motion compensation

For sequence 2, Devaux et el. [1] use a different concept than ours; however, their concept is equivalent to window and history lengths of one frame with no motion compensation; we label this configuration as NM. Table 2 shows a comparison of the NM configuration with the full motion compensated approach proposed here, as well as the individual frame optimization strategy, identified as INTRA.

Significantly lower performance results are reported in [1] for both the INTRA and NM cases; the reason, we suspect, is that the JPEG2000 stream used in [1] employed very small code-blocks and only 4 quality layers. Nevertheless, the results presented in Table 2 reveal that motion compensation and non-trivial windows provide a good improvement over the the NM case.

Table 2: Comparison of various methods for sequence 2 at various bit rates. All values are PSNR in dB of the average MSE.

	Bit Rate kb/s			
Method	288	500	1000	2000
INTRA	29.07	31.12	34.13	38.09
$w = 1$, $h = 1$, NM	35.48	37.49	39.88	42.04
$w = 1$, $h = 1$	36.74	38.27	40.13	42.13
$w = 2$, $h = 1$	36.84	38.35	40.21	42.17

6. DISCUSSION

Traditional video encoding algorithms impose certain restriction on how the stream should be decoded. This is true both for predictive schemes and "wavelet-like" scalable video coding schemes. For example, reduced frame-rate video playback can be supported by both hierarchical predictive schemes (e.g., via B-frames in MPEG-1/2) and 3D wavelet-based schemes (e.g., via MCTF). However, these schemes only allow efficient recovery of a pre-determined subset of the frames (e.g., the even indexed frames). The paradigm proposed here imposes no such restrictions. The same can be said of the playback direction. Whereas predictive coders require the video sequence to be decompressed only in the forward direction, our proposed client/server driven conditional replenishment scheme can be used equally efficiently for both forward and reverse playback. The server is free to send only those motion parameters which would be useful to the client. In the same way, the proposed paradigm allows spatio-temporal regions which are of no interest to the client to be skipped, simply by assigning the relevant code-blocks a weight of 0 in the optimization phase. Of course, this changes the way in which motion compensation should be used in predicting future frames, but that is handled dynamically by the server's optimization policy.

For lossy transmission, predictive coding requires the server to retransmit all the lost packets, whereas the proposed paradigm makes it possible for the server to send new code-blocks from subsequent frames instead of the ones that were

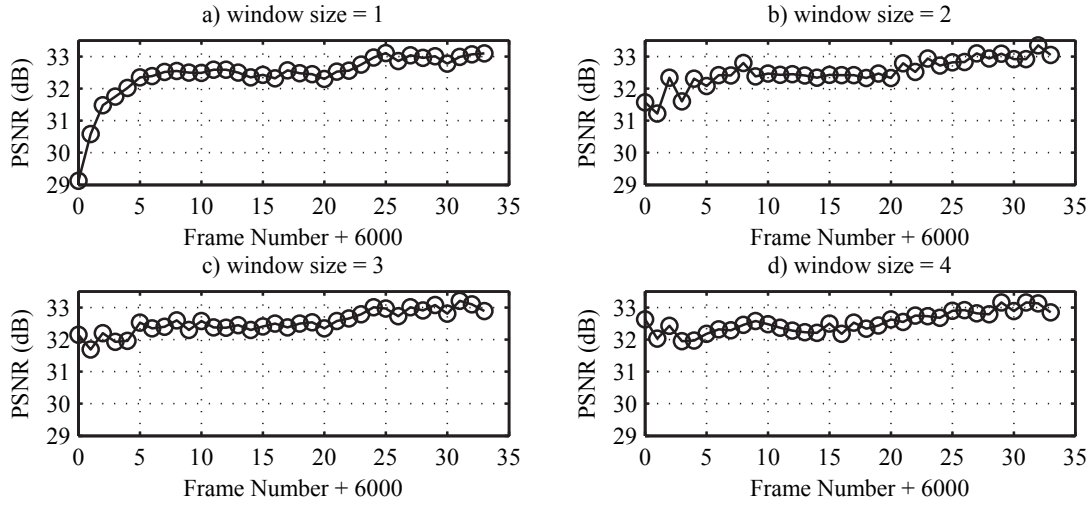


Figure 5: PSNR for sequence 1 at 960 kb/s for various window sizes and a history length of 1.

lost in transmission. Estimated loss probabilities and other statistics from network modeling can be included in the optimization as well. For the same compressed video stream, real-time and off-line clients can be served using different window sizes, allowing delay to be traded against efficiency and the ability to correct transmission errors.

On the server side, it is possible for the server to optimize the delivered video aggressively if it is lightly loaded or use a simpler optimization algorithm when it is serving many clients. For real-time broadcasting, it is possible to split the complete process into three separate stages, possibly running on different machines; one for compression, one for determination of the motion parameters and associated inter-frame distortion statistics, and one for on-line serving of different clients. Depending on the capabilities of the client and the available bandwidth, the server can choose to send approximate motion vectors or no motion information at all, adjusting its optimization algorithm accordingly.

7. CONCLUSIONS

The method present here is an idealistic way of optimizing the streaming of a JPEG2000 based video, which serves as a guideline to the maximum expected benefits of the paradigm we are proposing here and in an earlier paper. The paradigm proposes serving a JPEG2000 video using dynamically optimized conditional replenishment of JPEG2000 code-blocks. It solves issues in scalability and accessibility of the current video encoding algorithms beside being very flexible and performing reasonably well. We found that longer windows provide a better byte budget management and reduce the fluctuations in the PSNR of the reconstructed frames; while improvements produced by using longer history depends on the video content. More research is needed to simplify the calculations in order to make real-time streaming efficient and feasible.

8. REFERENCES

[1] F.-O. Devaux, J. Meessen, C. Parisot, J.-F. Delaigle, B. Macq, and C. De Vleeschouwer. A flexible video transmission system based on JPEG2000 conditional replenishment with multiple references. *Proc. IEEE Int. Conf. Acoust. Speech and Sig. Proc.*, April 2007.

[2] N. Mehrseresht and D. Taubman. An efficient content-adaptive motion compensated 3D-DWT with enhanced spatial and temporal scalability. *IEEE Trans. Image Proc.*, pages 1397–1412, June 2006.

[3] 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*, Septmeber 2007. in press.

[4] J.-R. Ohm. Advances in scalable video coding. *Proc. of the IEEE*, 93, January 2005.

[5] H. Schwarz, D. Marpe, T. Schierl, and T. Wiegand. Combined scalability support for the scalable extension of H.264/AVC. *Proc. Int. Conf. on Multimedia and Expo, 2005*, July 2005.

[6] D. Taubman. Localized distortion estimation from already compressed JPEG2000 images. *Proc. IEEE Int. Conf. Image Proc.*, 2006.

[7] D. Taubman and M. Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice.* Kluwer Academic Publishers, Boston, 2002.

[8] P. Zanuttigh, N. Brusco, D. Taubman, and G. Cortelazzo. Server policies for interactive transmission of 3D scenes. *IEEE Multimedia Signal Processing Workshop*, pages 59–64, October 2006.