Scalable video coding with managed drift*

Amy R. Reibman
AT&T Labs – Research
Florham Park, NJ 07932
amy@research.att.com

Léon Bottou
NEC Research Institute
Princeton, NJ 08542
leon@bottou.org

Andrea Basso
NMS
Red Bank, NJ 07748
a_basso@att.net

October 15, 2002

Abstract

Traditional scalable video encoders sacrifice coding efficiency to reduce error propagation because they have avoided using enhancement-layer (EL) information to predict the base layer (BL) to prevent the error propagation termed "drift". Drift can produce very poor video quality if left unchecked. In this paper, we propose a video coder with significantly better compression efficiency because it intentionally allows the drift produced by predicting the BL from the EL. Our drift management system balances the trade-off between compression efficiency and error propagation.

The proposed scalable coder uses a spatially adaptive procedure that optimally selects key encoder parameters: the quantizer and the prediction strategy. Our numerical results indicate the encoder is very powerful, and the selection procedure is effective. The video quality of our coder at low rates is only marginally worse than the drift-free case, while its overall compression efficiency is not much worse than a one-layer nonscalable encoder.

1 Introduction

Compressed video, which uses predictive coding algorithms and variable-length coding, is sensitive to network impairments. A single bit error or erasure can cause substantial degradation if no action is taken to stop or limit the extent of error propagation. Motion compensation allows the error to propagate both temporally and spatially. Because of

*This work was done while the authors were at AT&T Labs - Research.

this, there has been extensive effort in the video community to design new techniques that limit the extent of error propagation [1]. However, almost all attempts to limit error propagation decrease the coding efficiency, some dramatically so. To ensure the best operation of the video coder in an error-prone channel, the balance between resilience and efficiency must be managed carefully.

Layered and scalable coding algorithms create a partitioning of the compressed bitstream into more and less important parts. Algorithms with bitstream scalability have the added advantage that the less important enhancement layer (EL) bitstreams are still decodable when truncated at any arbitrary point. These properties allow a natural combination with different mechanisms to prioritize network transport, for example, marking less important parts for early discard [2] [3], applying unequal error protection [4] [5] [6] [7], or facilitating rate-matching between encoder, decoder, and network [8] [9] [10] [11]. When used in conjunction with such techniques, scalable video can be very resilient to network-introduced errors.

One of the primary difficulties in creating a scalable video coder lies in the prediction strategy of an efficient video coder. Past scalable video coders can be characterized into three classes depending on the prediction process used to code the more important (or base) layers (BL) and the less important enhancement layers (EL). In the first class, an EL is compressed without prediction from itself or from other less important layers. For example, Taubman and Zakhor [12] use a prediction structure for a bit-plane encoder which does not use less important bit-planes of previous subbands to predict more important bit-planes of the current subband. This limits error propagation, but can reduce compression efficiency.

In the second class of scalable coders, an EL can be compressed relative to itself, but not relative to other less important layers. For example, in H.263+, the previous EL frames can be used to predict the current EL frame, but the BL frames are predicted only from BL information [13]. This improves compression efficiency, particularly when the rate of the base layer is small relative to the enhancement layer; however, if parts of

the enhancement layer are not received, some error propagation may result.

In the third class of scalable coders, all layers including the base layer can be compressed relative to other less important layers. One example of this is MPEG-2 SNR scalability [2], which uses a single-loop prediction structure in which BL information is predicted from previous base *and* enhancement information. These coders have the highest compression efficiency, but the most error propagation if some information is lost. In this paper, we present a scalable video coder in the third class, which achieves very good trade-off between error propagation and compression efficiency. We explicitly consider a block-based DCT hybrid coder, although the basic ideas can be applied to other hybrid video coders which use some form of motion-compensated temporal prediction.

Each of the three classes of scalable coders described above is represented in MPEG-2, which was the first international standard to define scalable coding [2]. MPEG-2 SNRS is in the third class, while MPEG-2 Spatial Scalability (SS) allows a coder to be in either the first or second class. Neither H.263 nor MPEG-4 version 1 have options for a coder to be in the third class, although they allow both the first or second class as options. However, experiments show that with MPEG-2 SS, MPEG-4 and H.263 scalability, operating as in the second class suffers from 0.5-1.5 dB losses for every layer [2], [14]. When operating with the additional restriction of the first class of scalable coders, with no temporal prediction in the enhancement layer, these coders suffer even more degradation.

Another disadvantage is that each of these earlier standardized scalable coders has monolithic enhancement layers, in the sense that there is no partition *within* an EL which might allow the less important parts of that EL to be discarded prior to more important parts. If part of the EL is lost, the entire EL is lost for that spatial location. Therefore, these coders are not well suited for an application in which a video sequence should be encoded once with the ability to decode to any bandwidth. These coders do allow the partition between the base and enhancement layers to be chosen arbitrarily, but it can not be changed after encoding.

Great strides were made with the introduction of the Fine Grained Scalability (FGS) option of MPEG-4 version 2, which uses bit-plane encoding of the EL [8, 16, 15]. Not only does this major breakthrough facilitate stored video transmission over the Internet (where the transmission rate is variable and not known at the time of encoding), but also the flexible EL structure allows easy design of unequal error protection for wireless networks. Unfortunately, at the time of standardization, MPEG-4 FGS chose to use a temporal prediction structure that places it in the first class of scalable coders above; the EL layer is not used for the temporal prediction of either the base or enhancement layer. As a result, this algorithmic structure can lose 2 dB of compression efficiency when compared to a non-scalable one-layer MPEG-4 coder for the moderately active *Stefan* sequence [16]. (There would be more degradation for a less active sequence, but less for a more active sequence.)

The inefficiency of the MPEG-4 FGS prediction structure has been noted in [17]. The Progressive FGS (PFGS) prediction structure [17] attempts to address this compression inefficiency by predicting EL information from a few bit-planes of EL data. However, it does not allow the possibility of predicting base-layer information from EL information; therefore it still suffers a nontrivial loss in compression efficiency.

Predicting BL information from EL information introduces the possibility of drift, defined here as the mismatch between encoder and decoder frame memories due to partial reception of the less important EL information. The history of avoiding drift started with the introduction by Goodman of Embedded DPCM (EDPCM) in 1980 [18]. There, he showed that for temporal correlation of $\rho = .85$, an infinite time horizon, and quantizers with fixed bit-rates, EDPCM was highly effective when compared to DPCM with drift. The first layered video coder [3] also was configured to avoid drift, and Arnold et. al. [19] present a variety of drift-free prediction structures, after observing that drift continues to accumulate even as the number of times steps gets large. Continued recent effort in developing scalable coders all focus on tolerating absolutely no base-layer drift [20],[21].

However, despite the predominance of arguments in the literature maintaining that

systems should not be designed to allow drift, there is some evidence that drift need not be eliminated completely. In Aravind et. al. [2], the MPEG-2 SNRS with 0.1% cell losses was invisible even when the base bit-rate was only 25% of the total bit-rate. Further, in [22], the loop structure with only partial mismatch control provided the best single-channel reconstruction quality for a given redundancy in a motion-compensated multiple description video coder. Finally, if we reconsider EDPCM using a limited time horizon, entropy coded quantization, and higher temporal correlations of, say, $\rho = .99$, DPCM with drift can become more efficient than EDPCM.

In [23], we showed that drift can indeed be tolerated in a scalable video coder, and that careful management along with macroblock-level mode selection can provide very efficient scalabe hybrid coders. A number of more recent papers have considered systems that control base-layer drift, including [24] and [25].

In this paper, in Section 2, we present our drift management system. In Section 3, we present a DCT-based motion-compensated scalable video coder with two key features. First, the coder allows the BL to be predicted from past EL information, and second, the coder contains mechanisms that allow the resultant drift to be controlled. In [23], we used a simple heuristic decision algorithm to show that this encoder can significantly outperform both the FGS encoder and the one-layer encoder across the range of channel rates. In Section 4 we present a spatially adaptive algorithm for jointly selecting the prediciton and quantization strategies used the scalable encoder. Section 5 demonstrates that our coder performs better than alternative encoders across most of the range of channel bit-rates, even though our coder suffers marginal performance degradation at the lowest bit-rates compared to the no-drift encoder alternative. Our coder has significantly better compression efficiency than FGS for higher bit-rates, with only slightly degraded resilience for the lower bit-rates. Section 6 concludes the paper.

We note that our scalable coder with managed drift has many of the same key advantages of the MPEG-4 FGS that were described in [16]. It allows a server to have minimal real-time processing, it is highly adaptable to unpredictable bandwidth, and

also it is resilient to packet losses. Further, our coder has the additional advantage that it has significantly better compression efficiency than FGS. In particular, our coder is designed to take advantage of the fact that typical video sequences have spatially inconsistent temporal correlation; therefore our coder adapts its coding technique to the local statistics. The FGS prediction structure is a fall-back mode that can always be used by our coder.

2 Managing drift

In this section, we consider structures for managing drift in a scalable DCT-based motion-compensated video coder. For a comprehensive management of drift, five features are necessary. Partial management of drift is possible with different subsets of these five features. Our scalable coder presented in section 3 contains all five features.

- First, there should be a means to introduce drift incrementally. This is straightforward to achieve by bit-plane encoding or by creating an embedded bitstream. This will only be effective, however, if used in conjunction with a mechanism in the transport that provides more reliable delivery of the more important bit-planes to the receiver. Examples can be found in [4] and [9].

- Second, there should be a way for the encoder to measure the drift being potentially introduced, so it knows when drift is becoming significant.

- Third, there should be encoding options that can allow drift (i. e. allow errors in the EL to propagate into the BL), while simultaneously keeping drift under control.

- Fourth, there should be a means to drastically reduce or eliminate drift without using a full I-frame.

- Fifth, there should be a system-level optimization, designed to maximize expected quality across all expected receivers. Inherent to this optimization, there must be some (possibly inaccurate) knowledge on the part of the encoder as to how many

errors the channel will introduce, and how those errors will be introduced (gradually bit-plane by bit-plane, or suddenly when an entire packet of high-priority data is lost).

The recognition that the impact of errors in a one-layer coder can be probabilistically characterized by the encoder has lead to range of methods, including those of Wenger and Côté [26], Zhang et. al. [27], and Wu et. al. [28]. The impact of errors in a two-layer coder has also been probabilistically characterized in the encoding process by Zhang et. al. in [13]. However, while they consider the impact of errors in the enhancement-layer reconstruction, the encoder they optimize does not have the option of using EL information to predict the BL. Hence, their scheme still has limited compression efficiency compared to a one-layer encoder.

The fifth component of our drift management system is based on the observation that if one can effectively manage error propagation in both a one-layer encoder and a two-layer encoder that does not allow the introduction of BL drift, it is possible to extend these techniques to an encoder that does allow the introduction of drift into the base layer. Such an encoder will have greater compression efficiency for higher bit-rates, with only slightly degraded resilience for the lower bit-rates.

3 Scalable DCT coder with drift control

This section presents our encoder and decoder structure which allows managed drift. Our scalable coder balances high compression efficiency with error resilience.

While structures like B-frames or P'-frames[1] naturally reduce drift by having fewer predictions made from partially correct data, we purposely do not consider them here. Instead we focus on ways to manage drift within the predictive framework of P-frames. Both P'-frames and the temporal scalability provided by B-frames [16] can easily be

[1] P'-frames are similar to B-frames without forward prediction, and which are enabled by Reference Picture Selection (RPS) mode of annex N in H.263+ [29].

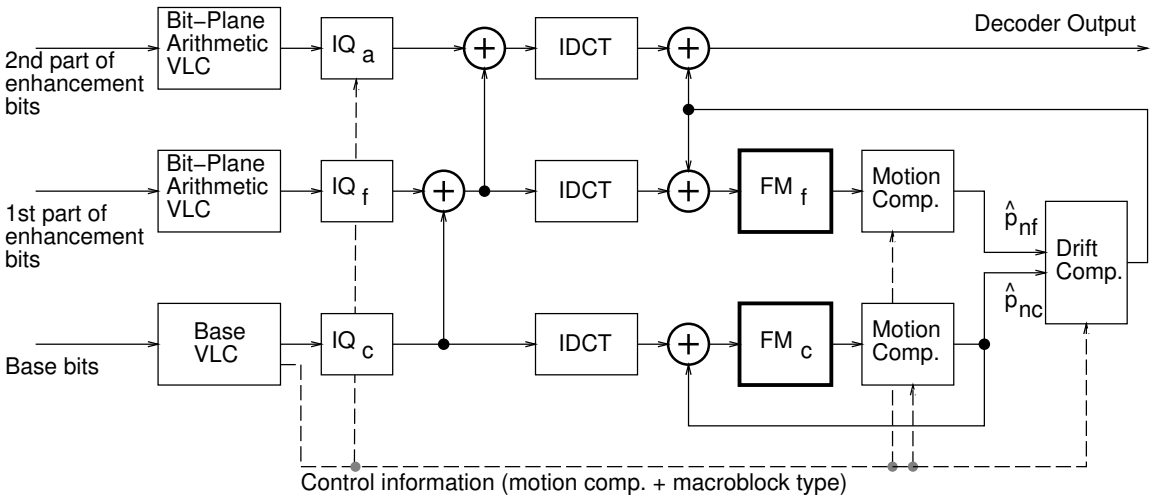

Figure 1: Two-loop decoder with drift control

incorporated into our scalable coder. In fact, a P'-frame is one way to limit the temporal extent of error propagation without an I-frame, even for a one-layer encoder.

3.1 Prediction structure

Figures 1 and 2 show our scalable DCT decoder and encoder with drift control. The encoder incorporates all five components necessary for effective drift management. The decoder (Figure 1) takes three levels of input. The base bits, with bit rate R_{nc}, are assumed to be always available. The first part of the enhancement bits, with bit-rate $R_{nf} - R_{nc}$, may not be received by the decoder, but if received, are used to predict the next frame. The second part of the enhancement bits, with bit-rate $R_{na} - R_{nf}$, may not be received, and is never used to predict the next frame.

Both the decoder and the encoder maintain two frame memories. The *coarse frame memory* depends only on the base bits and never drifts. The *fine frame memory* is updated by first combining both motion-compensated frame memories, and then applying the base bits and the first part of the enhancement bits. The fine memory drifts when some of these enhancement bits are lost.

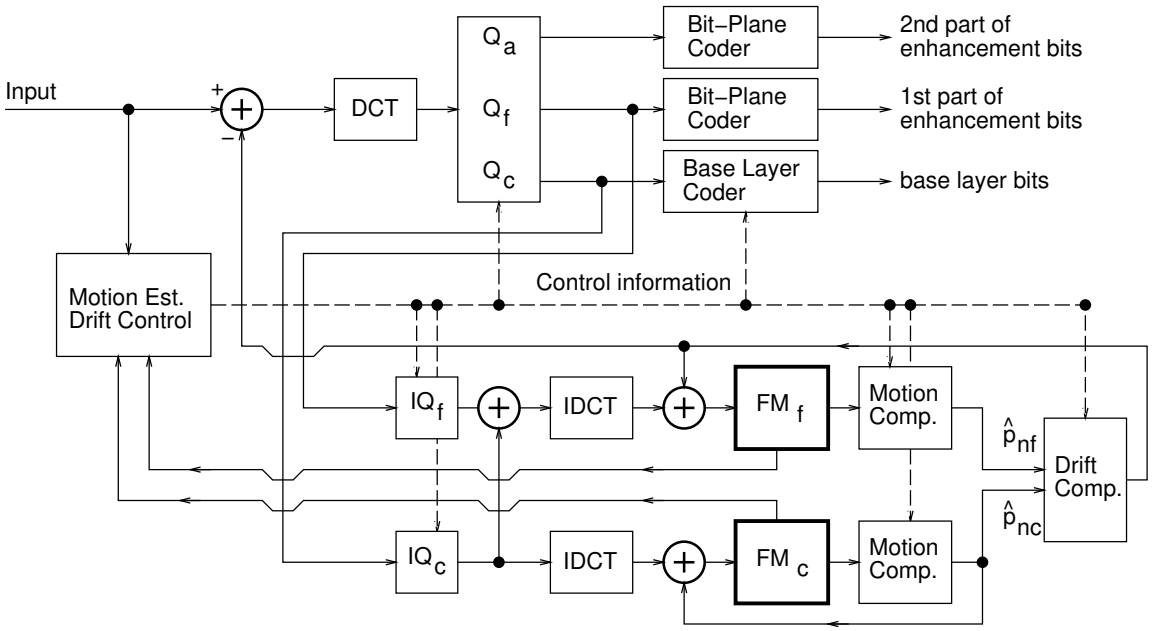

Figure 2: Two-loop encoder with drift control

Let $\hat{p}_{nc}$ and $\hat{p}_{nf}$ be motion-compensated predictions from the coarse and fine memories for macroblock n. For each macroblock (MB), the drift compensation box in Figure 1 combines the coarse and fine predictions according to a MB type information. The first option eliminates drift by taking the coarse prediction $\hat{p}_{nc}$ only (as in FGS). The second option allows drift by taking the fine prediction $\hat{p}_{nf}$ only (as in MPEG-2 SNRS). The third option reduces, but does not eliminate, drift by averaging both predictions. For simplicity, we only consider here these three options; other combinations would naturally extend our scheme.

The scalable DCT encoder (Figure 2) tracks both frame memories under the assumption that all bits are received by the decoder. The encoder makes several decisions that affect the amount of decoder drift in the fine memory. The first decision is the selection of a prediction mode for the drift compensation. The second decision involves the number of bit-planes that might be used in the prediction loop; this is accomplished by adjusting the quantization Q_f relative to the final quantization Q_a. A third technique could be to apply a filter to the prediction from the coarse loop in order to smooth the

discontinuities associated with prediction blocks that straddle macroblock boundaries; we do not explore this latter possibility in this paper.

Different sequences, and different parts of sequences, have different trade-offs between efficiency and resilience as a function of these drift control decisions. In high temporal correlation, it is advantageous to use temporal prediction as much as possible, while in low temporal correlation, temporal prediction can be neglected to reduce error propagation. The encoder must make these decisions and send this information to the decoder. Our encoder makes these choices on a macroblock basis with the goal of optimizing the total system performance as described in Section 4.

To minimize the influence of drift in general, we use an embedded coder (described next) to compress each individual frame. This allows more significant enhancement layer bit-planes to be received and decoded even if the network does not have sufficient bandwidth to send the entire enhancement layer. In our implementation, the base-layer VLC also uses arithmetic bit-plane coding, however it could have been implemented using the usual Huffman method. Macroblock type information and motion vectors are included in the base layer. We use the same motion vectors in both the base and enhancement layers.

3.2 Embedded video bitstream

A scalable video bitstream must indicate the relative importance of bits or groups of bits within the bitstream. This information is necessary to facilitate transport mechanisms that respond to the observed channel characteristics by discarding the least important bits or by applying error correction to the more important bits. In our scalable bitstream, which is produced by a binary adaptive Z-Coder [30], prioritization is imposed by the order in which information is encoded during a *coding run*.

The Z-Coder coder represents probability distributions using context variables. Internal state and context variables are initialized at the beginning of each coding run and are updated after coding each binary decision. If a piece of the bitstream is missing, the

decoder cannot perform identical updates and must stop. Decoding can resume at the beginning of the next coding run. Hence, more important information should be encoded early in the coding run to ensure it is received prior to any information loss.

Shorter coding runs increase robustness to random losses in the channel. However, because we assume prioritized transport, very short coding runs will not be useful in our system. Longer coding runs typically improve the coding efficiency because the contexts do not need to be re-learned at the beginning of every coding run. A typical single frame requires 1.072 or 0.513 bits per pixel when quantized with $Q = 4$ or $Q = 8$ respectively. If the same frame is encoded 256 times without resetting the adaptation contexts after encoding each copy, the resulting bit-rates are 1.069 and 0.509 bits per pixel respectively. This suggests that no more than 1% of the bandwidth could be saved by coding several frames per coding run. Further, latency becomes problematic if we have several frames per coding run. In this paper, the scalable coder currently performs one coding run per frame.

Within each coding run, binary decisions are encoded by decreasing order of importance. We start with the macroblock type, quantization and motion information, followed by the DCT coefficient information for the base layer and the various enhancement layers. The DCT coefficients are encoded as a sequence of binary decisions ordered according to their importance. The coefficients in each 8x8 block are first divided by the finest quantizer Q_a, resulting in an integer quotient with absolute value that can always be represented using twelve bits. Each elementary bitplane coding operation then processes one particular bitplane of one particular 8x8 block of DCT coefficients.

Each elementary coding operation belongs either to the base layer or to one of the two refinement layers, according to the bitplane number and to the values of the Q_c/Q_a and Q_f/Q_a ratios for the corresponding macroblock. First, the base layer is encoded by performing all the base-layer elementary coding operations starting with the most significant bitplanes, and proceeding towards the least significant bitplane. Then each refinement layer is encoded by similarly performing all the elementary coding operations

for that layer in decreasing order of bitplane significance. The Z-Coder internal state is flushed after each layer in order to clearly delimit the bitstream for each layer.

To perform an elementary coding operation, the coder loops over the 64 coefficients and codes whether the binary representation of its absolute value contains a 0 or a 1 for the current bitplane. The sign bit is coded just after coding the first 1 of each coefficient. Such coefficients are named *significant*. Previous bit-plane coders [31, 32] take advantage of the wavelet transform structure by coding decisions addressing the significance of entire coefficient groups. The same result is achieved for zig-zag ordered DCT coefficients by coding a stopping decision after each significant coefficient. A positive stopping decision indicates that none of the remaining coefficients will be significant after processing this bitplane. When it is known that some remaining coefficients are already significant, there is no need to code the stopping decision.

This encoding process produces a high bandwidth bitstream. Lower bandwidth channels are accomodated by simply truncating a fixed proportion of the second refinement layer. Even lower bandwith channels are accomodated by eliminating the second refinement layer altogether and truncating a fixed proportion of the first refinement layer. Extremely low bandwidth channels might be dealt with by eliminating both refinement layers and truncating the base layer. Here, we facilitate the truncation process by dividing the bitstream into "chunks", with one chunk per layer per frame. Each chunk is preceded by a one- to three-byte length indicator, indicating how many bytes are in the chunk. There are three chunks per frame, corresponding to the base layer, the first part of the enhancement layer, and the second part of the enhancement layer.

4 Encoder optimization

The traditional (often implicit) optimization when designing a scalable coder is to minimize the maximum possible distortion at the decoder, subject to the constraint that the channel rate R is in the range $R_{min} \leq R \leq R_{max}$. Typically, both R_{min} and R_{max} are

known, although neither the instantaneous channel rate nor the average channel rate in some time interval is known. The maximum distortion is achieved for the minimum rate R_{min}. Thus, optimizing using this criterion leads to a prediction process that does not tolerate any drift in the base layer. However, this also results in larger than necessary distortions at rates near R_{max}. We explore in Section 4.1 some alternate criteria for optimization, to achieve better compression at high rates without sacrificing too much quality at lower rates. In Section 4.2, we develop an algorithm for macroblock-level selection of encoding parameters based on Lagrange optimization and a simple model, and in Section 4.3, we describe in detail our implementation of that algorithm.

4.1 Alternate optimization strategies

One optimization criterion is to minimize the distortion at the highest rate, subject to constraint that the drift at the lowest rate is kept below some value. This can be expressed as $\min\{D_a\}$ subject to $D_c \leq \bar{D}_c$ and the rate constraints

$$R_c \leq R_{min} \quad \text{and} \quad R_a \leq R_{max}. \tag{1}$$

Here, R_c and R_a are the rates associated with the base bits, and all the bits, respectively, and D_c and D_a are the distortions of the associated reconstructions.

An alternate optimization criterion is to minimize the distortion averaged across all possible channel rates, subject to the rate constraints in (1). Determining the average distortion, however, requires knowledge of the probability distribution of the channel rates. This information is typically not available. However, a simple approximation is to minimize a weighted distortion $w_c D_c + (1 - w_c) D_a$, subject to the rate constraints in (1). The choice of w_c is influenced by the application and by the expected channel rates. For example, if smaller rates are expected to be the norm, then w_c should be set close to 1, while if high channel rates are typically expected, w_c should be set close to zero.

These two constrained optimizations can be solved by converting them to unconstrained Lagrangian optimizations. The unconstrained optimization problems will have

two or three Lagrange parameters, and can be attacked using techniques similar to those in [26, 27, 13]. In this paper, we choose not use the operational rate and distortion, determined by repeated encodings for each macroblock, as in [26, 27, 13]. Instead, we use some very simple models for both rate and distortion. We believe this yields more insight into the solution, and most importantly, our results in Section 5 indicate that this strategy yields very good results for many sequences.

4.2 Our locally adaptive quantizer and prediction selection

In [23] we fixed the quantizers Q_a, Q_f, and Q_c and used a simple heuristic to choose which prediction should be used for each MB. Here, we consider changing both the quantizer Q_f and the prediction, with the goal to minimize $\sum_t R_f(t)$ subject to

$$\sum_t \left[(1-\gamma)\log D_f(t) + \gamma \log D_c(t)\right] \leq \sum_t \log D^*, \tag{2}$$

where $R_f(t)$ and $D_f(t)$ are the rate and distortion produced when using $Q_f(t)$, and $D_c(t)$ is the distortion with only the coarse information, each at time step t. To simplify the notation, we suppress the fact that the summation applies not only over all time steps, but also over all macroblocks in each frame.

The value of γ depends on the application, and should be chosen to create a network-aware encoding [10]. If the channel rate is expected to be generally close to R_{min}, then γ should be set close to 1. Then, the encoder will reduce the drift at the lower rates at the expense of the compression efficiency at higher rates. On the other hand, if the channel rate is expected to be generally greater than R_{min} with occasional dips, then γ should be set close to zero [10].

Applying Lagrangian optimization, we choose the fine quantizer Q_f and the prediction method p such that

$$\min_{\{p,Q_f\}\forall t} \quad \sum_t \left[\mu R_f(t) + (1-\gamma)\log D_f(t) + \gamma \log D_c(t)\right]. \tag{3}$$

We use very simple models for rate and distortion,

$$R = \frac{1}{2}\log(\sigma^2/D), \quad D_f = \frac{Q_f^2}{12}, \quad \text{and} \quad D_c = \frac{Q_c^2}{12} + M_p,$$

where M_p is a measure of the amount of mismatch between a given prediction p and the coarse prediction $\hat{p}_{nc}$, to account for the contribution of drift to D_c. For an I-block or when using $\hat{p}_{nc}$, $M_p = 0$. When using the prediction $\hat{p}_{nf}$, M_p is the mean squared energy between $\hat{p}_f$ and $\hat{p}_{nc}$, while it is one quarter that when using the average prediction[2].

Using these, the optimization equation becomes

$$\min_{\{p,Q_f\}\forall t} \sum_t \left[\mu \log \sigma_p^2(t) + (1-\mu) \log \frac{Q_f(t)^2}{12} \right.$$
$$\left. + \gamma \log \left(\frac{Q_c(t)^2}{Q_f(t)^2} \right) + \gamma \log \left(1 + \frac{12 M_p(t)}{Q_c(t)^2} \right) \right], \tag{4}$$

where p indicates the prediction being evaluated, and we have absorbed the constant multiplier into μ.

In general, the first and fourth terms in equation (4) depend on past predictions and quantizers, as well as on the prediction in the current time step. For example, consider the selection of the best predictor for a particular macroblock in frame t. This macroblock depends on up to four MBs in frame $t-1$. If a finer quantizer Q_f had been used for those four Mbs in frame $t-1$, then the prediction $\hat{p}_f$ in frame t would have had a smaller σ_p^2 but larger M_p. Because of this dependency across time, joint optimization is required; however, a fully joint optimization requires a prohibitive search space.

Here, we choose to ignore some of the dependencies to simplify the optimization procedure. Because the dependence on the previous prediction is generally weaker than the dependence on the previous quantizer, we first consider jointly only the quantizer selection at time $t-1$ with the prediction selection at time t. However, this is generally still too complicated because one MB in frame t depends on four different values of Q_f in frame $t-1$. This is still too large a state space to reasonably consider.

[2]In our experiments, we found using a cumulative sum of the mismatch energy which is reset to zero for an I-block or a block using $\hat{p}_{nc}$ was more effective.

Hence, we consider first the choice of the best quantizer for each MB in frame $t-1$ *assuming the neighboring quantizers are identical*, and consider second the choice of the best prediction using the selected quantizers.

Consider the second step first. Assuming the quantizers Q_f and Q_c are fixed for all time and that the current predictor depends only weakly on the previous predictors, (4) reduces to

$$\min_{p(t)} \quad \mu \log \sigma_p^2(t) + \gamma \log \left(1 + \frac{12 M_p(t)}{Q_c^2}\right) \tag{5}$$

for frame t. Thus, to choose the best predictor given fixed quantizers, we simply evaluate (5) for each of the three predictors ($\hat{p}_{nc}, \hat{p}_{nf}$, and $(\hat{p}_{nc}+\hat{p}_{nf})/2$), and choose the minimizing predictor. Note that in the case when $\gamma = 0$, this simply chooses the predictor which has the minimum prediction error. If $\gamma > 0$, the impact of drift is also considered.

The first step, to choose the best quantizer for a MB in frame $t-1$ assuming the prediction strategies are already fixed, is more complicated. We need to consider the first and fourth terms of equation (4) for frame t, and the second and third terms of equation (4) for frame $t-1$. We can ignore the second and third terms for frame t because the quantizer Q_f at time t is unknown, and we assume it is constant. The first and fourth terms for frame $t-1$ can be ignored because the predictor for $t-1$ is already fixed. Thus, to choose the quantizer $Q_f(t-1)$, we minimize

$$\begin{aligned}(1-\mu) \log Q_f(t-1)^2/12 + \gamma \log \left(\frac{Q_c(t-1)^2}{Q_f(t-1)^2}\right) \\ + \mu \log \sigma_p^2(t) + \gamma \log \left(1 + \frac{12 M_p(t)}{Q_c(t)^2}\right).\end{aligned} \tag{6}$$

For this first step, we consider jointly the effect of the quantizer $Q_f(t-1)$ and the predictor $p(t)$. A particular MB in frame $t-1$ may be used to predict multiple MBs in frame t. Therefore, we first determine all affected MBs in frame t via reverse motion compensation. For every possible quantizer $Q_f(t-1)$, we assume the surrounding MBs use the same quantizer, and determine the best prediction of those affected MBs, as in equation (5). Then we choose the $Q_f(t-1)$ that minimizes the weighted cost of the

affected MBs. To compute the appropriate weight, we use the number of pixels in each MB in frame t that are impacted by the current MB in frame $t-1$.

4.3 Algorithm for selection of Q_f and predictor

To choose effective encoding parameters to encode frame t, we implement the following algorithm.

Preprocessing step 1: Do motion estimation between frames t and $t+1$.

Preprocessing step 2: For each $Q = mQ_a$, where $m = 1, 2, 4, 8$, create a frame memory assuming the entire frame uses quantizer Q. During this process, each macroblock uses the best prediction $p^*(t)$ according to (5).

Step 1: For each macroblock in frame t,

Step 2: For each affected MB in future frame $t+1$,

Step 3: For each $Q = mQ_a$,

Step 4: Find $p^*(t+1)$, the predictor which minimizes (5).

Step 5: Compute incremental cost for this affected MB in future frame $t+1$, $\Delta C(Q)$ according to (6) using $p^*(t+1)$.

Step 6: Add weighted cost to cumulative cost, $C(Q) += w \times \Delta C(Q)$, where w is the percentage of pixels affected in the future MB by the current MB in frame t.

Step 7: Choose Q_f^* to be the $Q = mQ_a$ which has minimum $C(Q)$.

Step 8: Code current macroblock using Q_f^* from step 7 and $p^*(t)$ from preprocessing step 2.

5 Results

In this section, we compare the performance of our drift-controlled coder to alternative coders. To ensure a fair comparison, all encoders in our comparisons are implemented using core components from H.263 baseline, with modifications to obtain the relevant prediction-loop structure. All coders also have the following additional modifications: the H.263 quantizer is replaced by a scalable quantizer [12], and the bitstream encoder is replaced by an embedded DCT coder described in section 3. Using a scalable quantizer instead of the H.263 quantizer loses less than 0.1 dB in quality, while using the context-based arithmetic coder instead of the H.263 Variable Length coder reduces the bit-rate by at least 5%.

In Figures 3 through 8, we compare our optimized coder and our coder with fixed Q_f to a one-layer encoder with no drift control, and to an encoder that uses the FGS prediction structure [8] (i.e., an encoder that always uses the coarse predictor $\hat{p}_c$). We use each encoder to create a single encoding, containing the base layer and 3 bit-planes of enhancement-layer information. Each coder uses a fixed, identical, quantizer in the base layer. In the current implementation, our drift-controlled coder sets $Q_a = 4$ and $Q_c = 8Q_a$. To emphasize the impact of drift, we use one I frame, followed by continuous P frames for each coder.

To obtain the performance comparisons in Figures 3 through 8 for the CIF sequences *Hall*, *News*, *Akiyo*, *Table Tennis*, *Foreman*, and *Mobile*, we truncate the EL bitstreams and decode the remainder. The x-axis shows the decoded bit-rate, and the y-axis shows the PSNR of the resulting decoder reconstruction. Also shown is the performance of the one-layer encoder with no loss, which cannot be generated using a single bitstream but must be obtained by generating a bitstream for each rate of interest. The curves labeled "fixed Q_f" use a fixed quantizer of $Q_f = 2Q_a$ for all MBs, while the curves labeled "optimal Q_f" use the method in Section 4. The curve labeled "FGS" uses the prediction structure of MPEG-4 FGS, where there is no temporal prediction of the enhancement

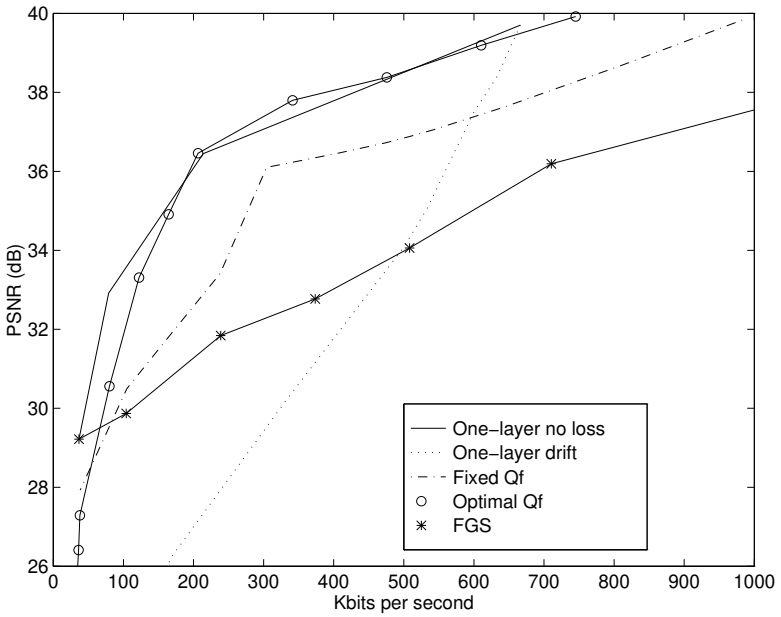

Figure 3: PSNR vs. rate for sequence *Hall* (15 fps).

layer.

The FGS coder performs poorly at the higher rates, especially for the mostly-still *Hall*, *News*, and *Akiyo* sequences. The one-layer decoder with drift suffers more than 5 dB degradation at the lowest bit-rate, compared to the drift-free FGS decoder. Relative to the FGS coder, our drift-controlled coder with fixed Q_f suffers about 1.3-1.4 dB performance degradation at the lowest bit-rate, but significantly outperforms it elsewhere. Our drift-controlled coder with optimized Q_f undergoes slightly more drift for the lowest bit-rates, but performs even better at the higher rates. For the *Hall* sequence, our optimized coder is competitive with the one-layer nonscalable coder with no loss for rates from 200-700 kbps. For *News* and *Akiyo*, our optimized coder is also competitve with the one-layer nonscalable coder in the middle range of bit-rates, losing some efficiency for the higher bit-rates and incurring some drift at the lower bit-rates. For Mobile, our drift-controlled coder loses some efficiency at the highest rates compared to the one-layer coder, but has substantially less drift as bits are discarded.

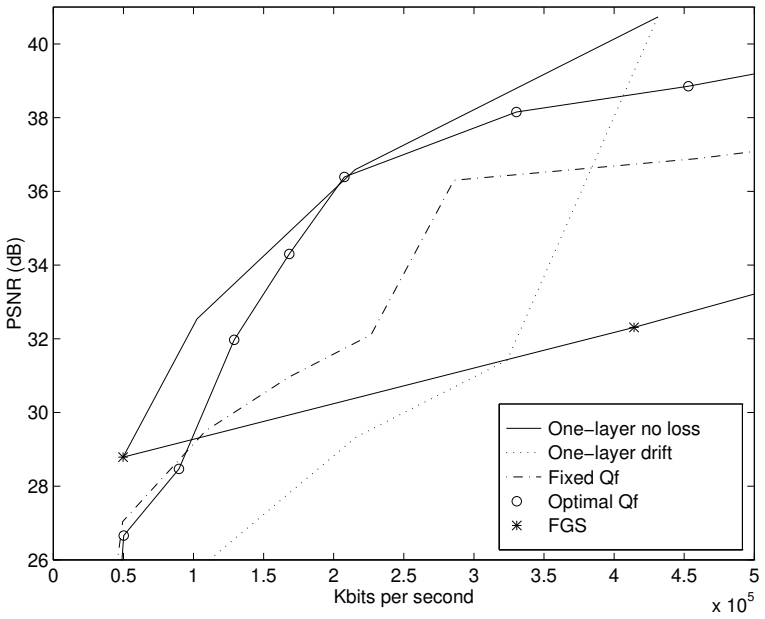

Figure 4: PSNR vs. rate for sequence *News* (15 fps).

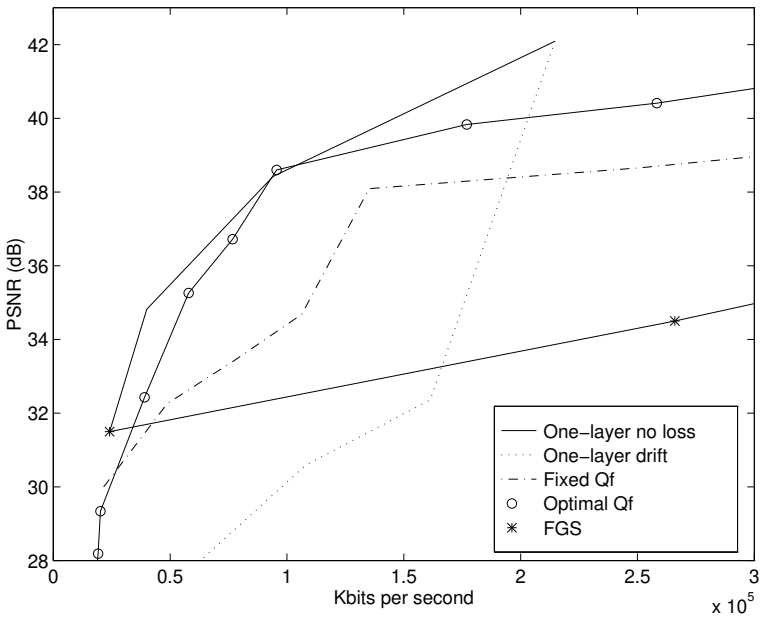

Figure 5: PSNR vs. rate for sequence *Akiyo* (15 fps).

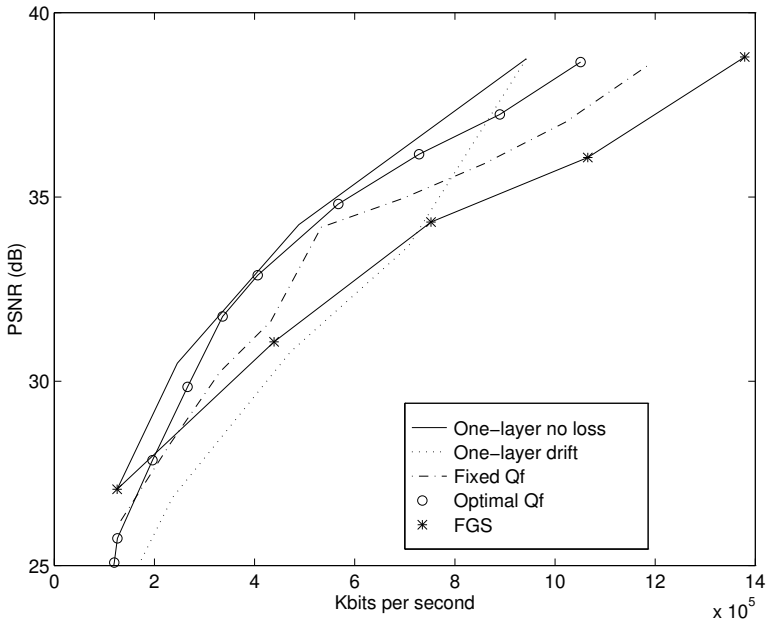

Figure 6: PSNR vs. rate for sequence *Table tennis* (10 fps).

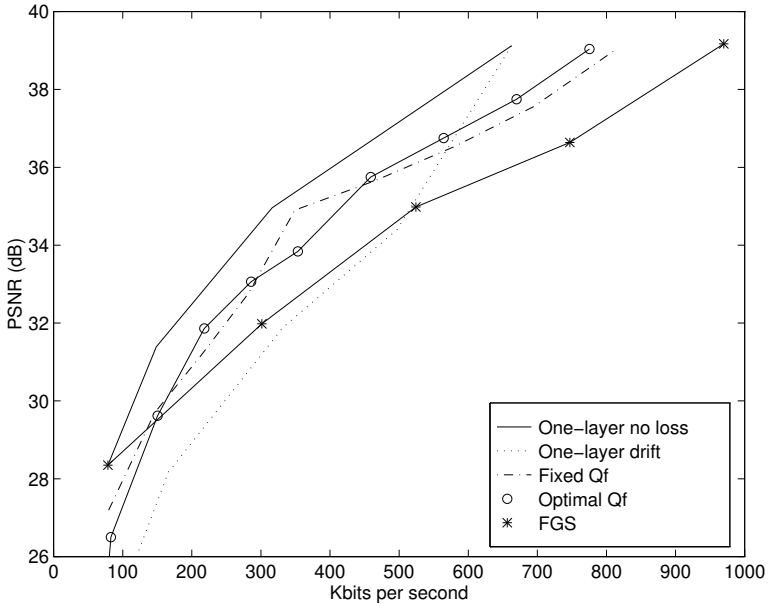

Figure 7: PSNR vs. rate for sequence *Foreman* (7.5 fps).

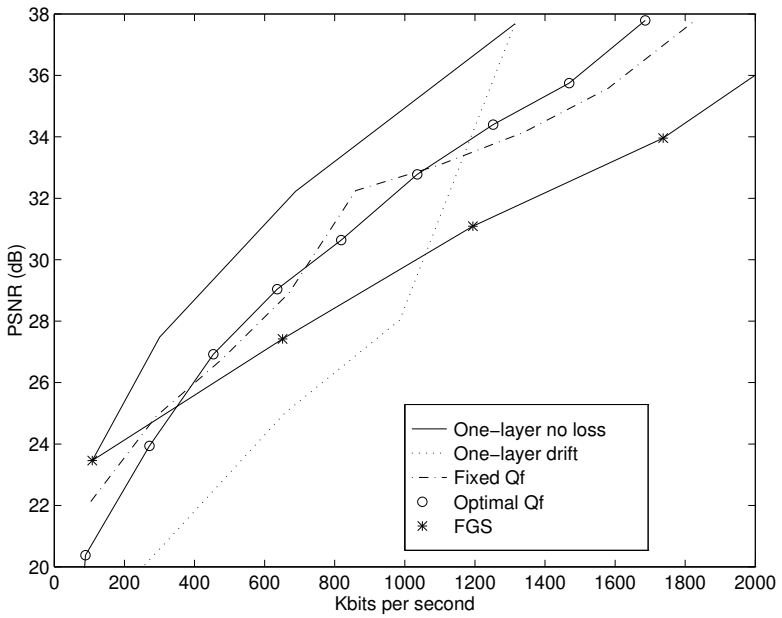

Figure 8: PSNR vs. rate for sequence *Mobile* (15 fps).

Table 1 shows the PSNR for the six sequences averaged across channel rates, assuming a uniform distribution of rates between the smallest and the largest rate of the one-layer encoder. In all cases, both our coders with fixed and optimized Q_f significantly outperform the other coders, beating FGS by 1–5 dB across the range of channel rates. The improvement of our coder relative to the FGS coder is largest for sequences with still backgrounds.

Figures 9 and 10 show individual frames from the *Mobile* sequence, for the one-layer coder with and without loss, and for the FGS and drift-controlled coders, respectively. All encoders use approximately the same average bit-rate as indicated in the captions. The one-layer bitstream for Figure 9(a) is encoded and decoded at the specific rate, while the one-layer bitstream decoded to create Figure 9(b) was truncated from an original bitstream compressed with greater bit-rate. The FGS and drift-controlled pictures were both generated from truncated streams.

Figure 9(b) shows that uncontrolled drift can significantly affect the quality of the

(a)

(b)

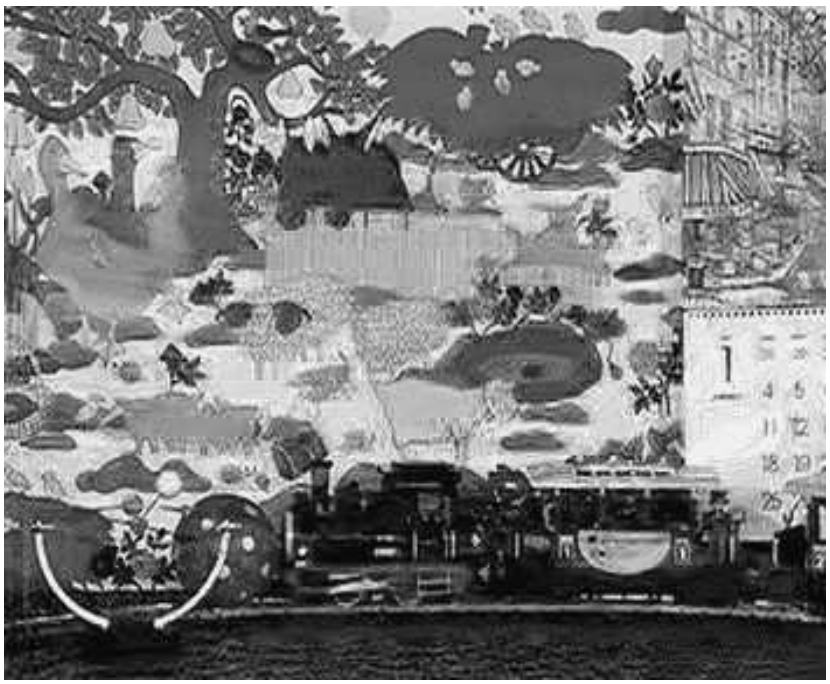

Figure 9: *Mobile*: 50th decoded frame. (a) One-layer no loss (688.6 kbps, 32.23 dB average) (b) One-layer, drift (657.5 kbps, 25.01 dB average)

(a)

(b)

Figure 10: *Mobile*: 50th decoded frame. (a) FGS (651.4 kbps, 27.42 dB average) (b) Controlled-drift (636.0 kbps, 29.04 dB average)

seq.	One-loop	Fixed Q_f	Opt. Q_f	FGS	bound
Hall	30.18	34.75	36.64	32.59	36.86
Akiyo	32.62	35.60	37.69	32.68	38.60
News	30.23	33.13	35.17	30.63	36.30
Tennis	31.45	32.66	33.51	31.80	34.12
Foreman	32.18	33.74	33.69	32.68	35.05
Mobile	26.02	29.27	29.11	27.75	31.78

Table 1: Decoded PSNR averaged across channel assuming uniform distribution.

reconstructed frame, by creating significant blurring throughout. However, the controlled drift in Figure 10(b) shows noticably sharper pictures than FGS in 10(a). The increased clarity can be seen, for example, in the eyes of the horses, the fence beneath the horses, and the rightmost goose. When viewed in real-time, the FGS video also exhibits visibly more flicker, because the artifacts in the enhancement-layer change from frame to frame.

The ability of our coder to keep drift in control is illustrated in Figure 11, which shows the PSNR as a function of time for *Mobile* at rates around 270 kbps, for four decoders. The qualities of the one-layer decoder with drift and of a bitstream which always uses a fine prediction continue to decrease as a function of time. However, the quality for the decoder using optimal drift control is effectively maintained, as the number of decoded frames increases beyond 15.

6 Conclusions

In this paper, we presented an efficient scalable video coder. The temporal prediction structure of the coder intentionally introduces drift by predicting the high-priority base layer from the less-important enhancement layer. To combat the negative effects of drift our scalable coder uses a drift management system that balances compression efficiency and error propagation.

While the FGS prediction structure is effective when the temporal correlation is low

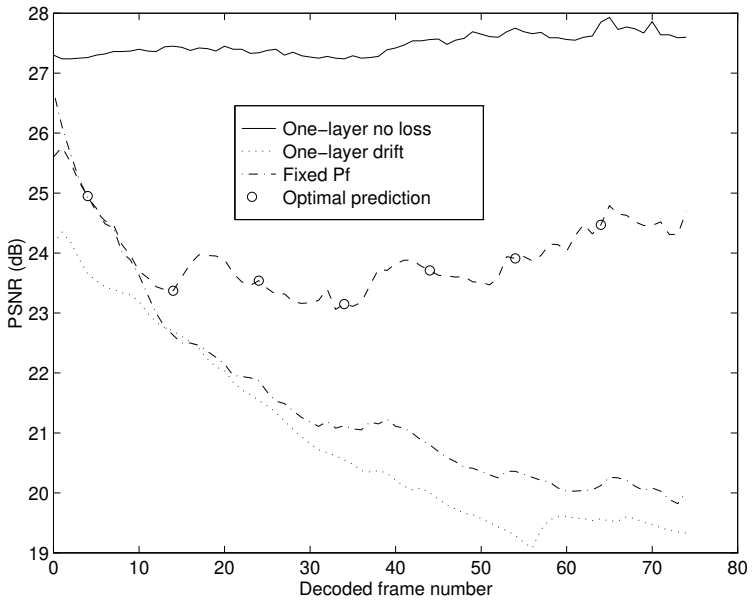

Figure 11: PSNR vs. time for sequence *Mobile*.

[16], our coder takes advantage of the inconsistent temporal correlation within a frame that is typically present in video sequences. With spatially adaptive prediction, we can use FGS-style prediction when the local temporal correlation is low and use more efficient prediction methods when the local temporal correlation is higher. Our results clearly indicate that the locally adaptive selection of quantizer and prediction, in combination with the proposed prediction structure, is highly effective for managing drift in an efficient scalable video coder.

Our current locally adaptive selection algorithm is based on a particular optimization criteria, and uses very simple models to characterize both the rate and distortion. Further improvements might be possible, particularly for the sequences *Mobile* and *Foreman* by using alternate optimization criteria or by replacing the simple models with operational measures of the rate and distortion. Furthermore, the structure of the decoder in Figure 1 uses a specific drift-compensation strategy of switching among three possible predictions. Alternate drift compensation mechanisms may enable additional improvements to the overall performance.

In many video streaming systems, a number of one-layer bitstreams are switched among, depending on an estimate of the channel conditions. Better system performance could be obtained by using instead a number of scalable bitstreams from our coder, each optimized for a different channel condition [10]. Switching among different streams provides long-term adaptation to changing channel conditions, while using a scalable bitstream instead of a one-layer bitstream provides short-term robustness against channel changes.

References

[1] Y. Wang and Q.-F. Zhu, "Error control and concealment for video communication", *Proc. IEEE*, vol. 86, pp. 974-997, May 1998.

[2] R. Aravind, et. al. "Packet loss resilience of MPEG-2 scalable video coding algorithms", *IEEE Trans. Ckts. and Sys. for Video Tech.*, vol. 6, pp. 426-435, Oct. 1996.

[3] M. Ghanbari, "Two-layer coding of video signals for VBR networks", *IEEE J. Sel. Areas Commun.*, vol. 8, pp. 771-781, June 1989.

[4] P. A. Chou, et. al. "FEC and pseudo-ARQ for receiver-driven layered multicast of audio and video", *Data Comp. Conf.*, pp. 440-449, Mar. 2000.

[5] P. A. Chou, et. al. "Error control for receiver-driven layered multicast of audio and video", *IEEE Trans. Multimedia*, vol. 3, pp. 108-122, Mar., 2001.

[6] E. Ayanoglu, et. al. "Forward error control for MPEG-2 video transport in a wireless ATM LAN", *ACM/Baltzer Mobile Net. and Appl.*, vol. 1, pp. 245-258, Dec. 1996.

[7] W. Tan and A. Zakhor, "Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol", *IEEE Trans. Multimedia*, vol. 1, pp. 172-186, June 1999.

[8] H. Radha et. al. "Fine-granular-scalable video for packet networks", *Packet Video Workshop*, New York NY, April 1999.

[9] R. Rejaie, et. al. "Quality adaptation for congestion controlled video playback over the Internet", *Proc. ACM SIGCOMM '99* , Sept. 1999.

[10] R. Rejaie and A. R. Reibman, "Design issues for layered quality-adaptive Internet video playback", Int. Wkshp. Dig. Commun., Sept. 2001.

[11] S. McCanne, et. al. "Receiver-driven layered multicast", *Proc. ACM SIGCOMM '96*, pp. 117-130, Aug. 1996.

[12] D. Taubman and A. Zakhor, "Multirate 3-D subband coding of video", *IEEE Trans. Image Proc.*, vol. 3, pp. 572-588, Sept. 1994.

[13] R. Zhang, et. al. "Switched error concealment and robust coding decisions in scalable video coding", *IEEE Int. Conf. Image Proc.*, Oct. 2000.

[14] K. Rose and S. L. Regunathan, "Towards optimality in scalable predictive coding", *IEEE Trans. Image Proc.*, vol. 10, pp. 965–976, July 2001.

[15] M. van der Schaar and H. Radha, "A novel MPEG-4 based hybrid temporal-SNR scalability for Internet video", *IEEE Int. Conf. Image Proc.*, Oct. 2000.

[16] H. Radha, and M. van der Schaar, "The MPEG-4 Fine-Grained scalable video coding method for multimedia streaming over IP", *IEEE Trans. Multimedia*, vol. 3, pp. 53-68, March, 2001.

[17] F. Wu, et. al. "DCT-prediction based progressive fine granularity scalable coding", *IEEE Int. Conf. Image Proc.*, Oct. 2000.

[18] Goodman, D. J., "Embedded DPCM for variable bit rate transmission", *IEEE Trans. Commun.*, vol. 28, pp. 1040-1046, July 1980.

[19] J. F. Arnold, et. al. "Efficient drift-free Signal-to-Noise Ratio scalability", *IEEE Trans. Ckts. and Sys. Video Tech.*, vol. 10, pp. 70-82, Feb. 2000.

[20] U. Benzler, "Spatial scalable video coding using a combined subband-DCT approach", *IEEE Trans. Ckts. and Sys. Video Tech.*, vol. 10, pp. 1080-1087, Oct. 2000.

[21] M. Domanski, et. al. "Spatio-temporal scalability for MPEG video coding", *IEEE Trans. Ckts. and Sys. Video Tech.*, vol. 10, pp. 1088-1093, Oct. 2000.

[22] A. R. Reibman, et. al. "Multiple description coding for video using motion compensated prediction", *IEEE Int. Conf. Image Proc.*, Oct. 1999.

[23] A. R. Reibman and L. Bottou, "Managing drift in a DCT-based scalable video encoder", *IEEE Data Comp. Conf.*, March 2001.

[24] C. Mayer et. al. "Bit plane quantization for scalable video coding", *SPIE Visual Communications and Image Processing*, January 2002.

[25] H. Yang et. al. "Optimal end-to-end distortion estimation for drift management in scalable video coding", in *Packet Video Workshop '02*, April 2002.

[26] S. Wenger and G. Côté, "Using RFC2429 and H.263+ at low to medium bit-rates for low-latency applications", in *Packet Video Workshop '99*, April 1999.

[27] R. Zhang, et. al. "Video coding with optimal Inter/Intra-mode switching for packet loss resilience", *IEEE J. Sel. Areas Commun.*, vol. 18, pp. 966-976, June 2000.

[28] D. Wu, et. al. "An end-to-end approach for optimal mode selection in Internet video communication: Theory and application", *IEEE J. Sel. Areas Commun.*, vol. 18, pp. 977-995, June 2000.

[29] M.-T. Sun and A. R. Reibman, ed., *Compressed Video over Networks*, Marcel Dekker, Inc, New York, NY, 2001.

[30] Bottou, L. et. al. "The Z-Coder Adaptive Binary Coder", *IEEE Data Comp. Conf.*, March 1998.

[31] J. Shapiro, "Embedded image coding using zerotrees of wavelets coefficients", *IEEE Trans. Signal Proc.*, vol. 41, pp. 3445-3462 Dec., 1993.

[32] A. Said, and W. Pearlman, "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees", *IEEE Trans. Ckts. and Sys. Video Tech.*, vol. 6, pp. 243-250, June, 1996.