Long-Term Memory Motion-Compensated Prediction

Thomas Wiegand, Xiaozheng Zhang, and Bernd Girod

***Abstract*—Long-term memory motion-compensated prediction extends the spatial displacement vector utilized in block-based hybrid video coding by a variable time delay permitting the use of more frames than the previously decoded one for motion-compensated prediction. The long-term memory covers several seconds of decoded frames at the encoder and decoder. The use of multiple frames for motion compensation in most cases provides significantly improved prediction gain. The variable time delay has to be transmitted as side information requiring an additional bit rate which may be prohibitive when the size of the long-term memory becomes too large. Therefore, we control the bit rate of the motion information by employing rate-constrained motion estimation. Simulation results are obtained by integrating long-term memory prediction into an H.263 codec. Reconstruction PSNR improvements up to 2 dB for the Foreman sequence and 1.5 dB for the Mother–Daughter sequence are demonstrated in comparison to the TMN-2.0 H.263 coder. The PSNR improvements correspond to bit-rate savings up to 34 and 30%, respectively. Mathematical inequalities are used to speed up motion estimation while achieving full prediction gain.**

***Index Terms*—H.263, Lagrange methods, motion-compensated prediction, multiple frames, video coding.**

I. Introduction

IN the early 1980's, video compression made the leap from intraframe to interframe algorithms. Significantly lower bit rates were achieved at the expense of memory and computational requirements that were two orders of magnitude larger. Today, with continuously dropping costs of semiconductors, we might soon be able to afford another leap by dramatically increasing the memory in video codecs to possibly hundreds or even thousands of previous frames. Algorithms to take advantage of such large memory capacities, however, are in their infancy today. This has been the motivation for our research into long-term memory motion-compensated prediction (MCP).

In recent years, several standards such as H.261, H.263, MPEG-1, and MPEG-2 have been introduced which mainly address the compression of video data for digital storage and communication services. The most recent standard H.263 [1] initially targeted the very low bit-rate end. But H.263 has emerged as a high-compression standard for moving images, not exclusively focusing on very low bit-rate applications. Moreover, H.263 has been further extended with various features specified in 12 additional annexes (Annexes I–T) that are collected in the H.263 Recommendation Version 2 (H.263+) [2]. H.263(+) as well as the other standards utilize hybrid video coding schemes which consist of block-based MCP and DCT-based transform quantization of the prediction error. It is also highly likely that the future MPEG-4 standard [3] will follow the same video coding approach, but have a different application target from H.263.

These standard algorithms employ intra, inter, and bidirectional picture coding types. In this paper, we restrict our attention to block-based motion compensation (MC) for inter picture coding. In most cases, the MC is carried out by utilizing an immediately preceding frame which is available as a reconstructed frame at the encoder and decoder. Long-term statistical dependencies in the coded video sequence that could be used to improve the efficiency of MCP are not exploited in existing international standards.

An example for the use of long-term dependencies in video sequences can be found in the negotiable H.263+ option called the reference picture selection mode (RPS mode), as specified in Annex N of H.263+ [2]. This mode permits a modified interpicture prediction called "NEWPRED" [4] to stop temporal error propagation due to transmission errors. A similar proposal as NEWPRED has been submitted to the MPEG-4 standardization group [5]. The RPS mode requires the storage of several decoded frames in dedicated picture memories. The encoder may select one of the picture memories to suppress the temporal error propagation due to the interframe coding based on backward channel messages sent from the decoder to inform the encoder which part of which pictures have been correctly decoded at the decoder. A particular picture memory is selected as reference for interframe coding of a complete picture, a "group of blocks," or a "slice." The amount of additional picture memories accommodated in the decoder may be signaled by external means as specified in the H.263+ document. The RPS mode was designed to suppress temporal error propagation, and not to enhance the coding efficiency of MCP. However, we have observed that an architecture very similar to NEWPRED can lead to significant coding gains when omitting the overhead information contained in the syntax of the RPS mode [2].

Techniques which use several reference pictures in order to improve the coding efficiency of video codecs have been considered within the MPEG-4 standardization group, like Dynamic Sprites including global motion compensation (GMC) [6], [7]. Dynamic Sprites and GMC are used to improve prediction efficiency in the case of camera motion by warping a motion-compensated version of the reference frame. However, for Dynamic Sprites, past frames are warped and blended into a sprite memory. In contrast to GMC, where the global MC is applied using the previously decoded frame, Dynamic Sprite uses the Sprite memory buffer to provide the second reference frame. The Dynamic Sprites and GMC in the MPEG-4 verification model use translational, isotropic, affine, and perspective motion models [3]. If the Sprite memory is equal to the frame size and the blending factor equals 1, Dynamic Sprites and GMC are equivalent; thus, Sprites are an extension of GMC.

Another approach, called "short-term frame memory/long-term frame memory" (STFM/LTFM) prediction, was also proposed to the MPEG-4 standardization group [8]. As proposed in [8], the encoder is enabled to use two frame memories to improve prediction efficiency. The STFM stores the most recently decoded frame, while the LTFM stores a frame that has been decoded earlier. In [8], a refresh rule is specified that is based on a detection of scene change. However, the STFM/LTFM approach has shown only small gains when implemented as specified in [8].

In [8], it was also proposed to include frames into the LTFM that are generated by background memory prediction techniques. These algorithms have been around for a while, e.g., see [9]. Generating a background frame is mainly an image segmentation problem, i.e., having all of the problems associated with it. Most of the background memory prediction algorithms work sufficiently well for scenes with stable background, but very often break down if camera motion or background changes occur [9]. An interesting extension to background memory prediction techniques was proposed in [10], wherein the image sequence is represented by layers. The layers are determined by analyzing the motion in a complete image sequence of several seconds, in order to obtain robust segmentation. This introduces a delay problem that cannot be resolved in interactive applications. The layered coding approach improves video compression only for sequences that are easily represented by the layered model [10].

Common to all of the techniques mentioned above is that the video encoder can choose between the immediately preceding reconstructed picture and a second picture either generated by Sprites including GMC, STFM/LTFM, or layered coding, including the background memory technique. These approaches also share the heuristic ideas that MCP can be improved in case certain motion scenarios occur, such as camera motion and camera zoom, as well as covering and uncovering of objects and background.

Higher gains than currently achievable with one of these techniques may be obtained by combining them. However, the selection among the reference pictures has remained a heuristic approach. In contrast, our approach for improving MCP does not rely on heuristic video models. We view block-based MCP as a statistical optimization problem related to vector quantization (VQ). Earlier work of ours on this subject has appeared in [11] and [12].

This paper is organized as follows. In Section II, we explain our approach to utilize long-term statistical dependencies in the coded video sequence for improved MCP. Section III provides results on prediction experiments in order to assess the gains achievable with the proposed technique. Design aspects, when integrating motion-compensated long-term memory prediction into an H.263 codec, are described in Section IV. In Section V, we provide comparisons to the test model TMN-2.0 of the H.263 codec [13]. We also analyze the partitioning of the bit rate within the hybrid video bit stream, and give results on computational complexity.

II. Long-Term Memory Motion-Compensated Prediction

Our approach to improve the efficiency of MCP is to extend the motion vector utilized in hybrid video coding by a variable time delay permitting the use of several decoded frames instead of only the previously decoded one for block-based motion compensation. In this paper, we restrict the maximum number of frames in the long-term memory to 50 corresponding to decoded video frames of 5 s at 10 frames/s sampling rate. The frames inside the long-term memory which is simultaneously built at the encoder and decoder are addressed by a combination of the codes for the spatial displacement vector and the variable time delay. Hence, the transmission of the variable time delay potentially increases the bit rate, which has to be justified by improved MCP. This tradeoff limits the efficiency of the proposed approach.

We can beneficially view long-term memory MCP as a source coding problem with a fidelity criterion. For a certain bit rate required to transmit the predictor parameters' spatial displacement and time delay, long-term memory MCP provides a version of the video signal with a certain distortion. The rate–distortion tradeoff can be controlled by various means. Our approach is to treat MCP as a special case of entropy-constrained vector quantization (ECVQ) [14]. The image blocks to be encoded are quantized using their own code books that consist of image blocks of the same size in the previously decoded frames: the motion search range. A code book entry is addressed by the translational motion parameters which are entropy coded. The criterion for the block motion estimation (ME) is the minimization of a Lagrangian cost function, wherein the distortion represented by the prediction error here is weighted against the rate associated with the translational motion parameters using a Lagrange multiplier. The Lagrange multiplier imposes the rate constraint as for ECVQ, and its value directly controls the rate–distortion tradeoff [14]–[19].

Viewing the motion search as an estimation problem, we can draw the following conclusions. If we drastically increase the number of (possibly very similar) decoded frames, the problem of motion estimation becomes increasingly ill conditioned as it already pertains when estimating motion vectors using the last frame only. The ill conditioning results in increased variance of the estimated motion vectors causing an increased bit rate for motion information. On top of that, the variable time delay needs to be transmitted as well. Hence, the Lagrangian formulation of the ME problem yields a solution to the problem of long-term memory ME not only when viewing it as a source coding problem. It is also a means for introducing a bias in the estimator, thus regularizing the ill-conditioned problem.

The architecture of the long-term memory predictor is depicted in Fig. 1. This figure shows an interframe predictor which uses a number of frame memories ($M, M \geq 1$) that

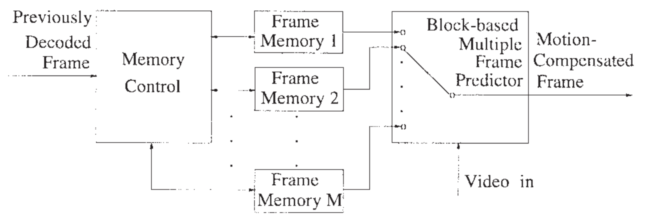

Fig. 1. Long-term memory motion-compensated predictor.

are arranged using the memory control. The memory control may work in several modes of operation. A sliding window over time may be accommodated by the memory control unit as depicted in Fig. 1. For that, past decoded and reconstructed frames, starting with the immediately preceding one, ending with the frame which is decoded M time instants before, are collected in the frame memories 1–M. Alternatively, the set of past decoded and reconstructed frames may be temporally subsampled using a scheme shared by the encoder and decoder. In general, several modes of operation for the memory control may be defined, and the one which is used may be negotiated between the encoder and decoder. In this work, we will use the sliding window approach because of its simplicity.

When permitting the long-term memory buffer to contain only decoded frames, the sliding window approach also minimizes the time at the beginning of the sequence to exploit the full long-term memory size. Note that the long-term memory MCP approach does not introduce any kind of additional transmission delay. Again, the encoder and the decoder might negotiate the number of frames used for long-term memory MCP. If the number of frames maximally accommodated by the long-term memory buffer corresponds to M, the motion estimation when coding frame m with $1 \leq m \leq M$ can utilize m frames. In case $m \geq M$, the maximum number of frames M can be used.

A variation and possibly an extension to long-term memory prediction is presented in [20]. Similar to Dynamic Sprites and GMC, reference frames are warped using an affine polynomial motion model. In contrast to Sprites and GMC, the number of reference frames is not restricted to one frame. In [20], several frames are warped by clustering the dominant motion models in the scene. The number of motion models is determined by the tradeoff sending the affine motion parameters and indexing the reference frames against improved MCP. The solution to the problem is obtained as well using a Lagrangian formulation.

In general, any technique that provides useful image data for MCP may be utilized to generate reference frames. These techniques may include Sprites [6], "layers" from the layered coding scheme [10], or video object planes (VOP's) as defined within MPEG-4 [3]. The decoder just needs to be informed about parameters that are needed to generate the reference frames, and to be given a reference coordinate system to conduct the MC. Based on rate–distortion efficiency, the encoder has to decide whether or not to include a particular frame. However, generating frames by one of the techniques mentioned requires additional computation. Also, the sequences have to lend themselves to representations with Sprites, layers, or VOP's. In contrast, the approach to use several decoded frames as reference frames, as proposed in this paper, does not rely that much on heuristic reflections on video representation.

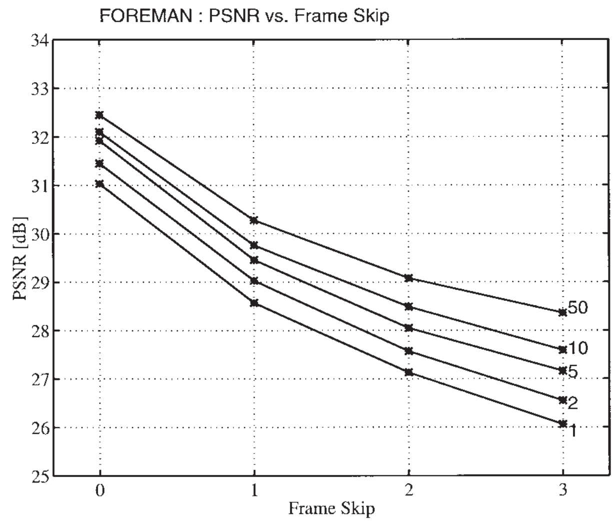

Fig. 2. Prediction gain versus frame skip parameter for the sequence *Foreman* for memory sizes $M = 1, 2, 5, 10$, and 50. The search range is ± 15 samples horizontally and vertically. ME is performed with integer-pel accuracy.

III. Prediction Gain of Long-Term Memory MCP

Improvements when using long-term memory MCP can be expected in the case of repetition of image sequence content. Note that these repetitions may or may not be meaningful in terms of human visual perception. Examples for visually meaningful repetition are moving image contents with repetition in orientation or shape, covered and uncovered objects, shaking of the camera forth and back, etc. Additionally, we obtain improvements if 16×16 or 8×8 blocks in long-term memory are coincidentally similar to the current block. Also, the effect of sampling the video signal at various positions in time may contribute benefits in favor of long-term memory MCP. Hence, it is more appropriate to view block-based MCP as a statistical optimization problem related to ECVQ.

Typically, the motion search range in standard video coding (H.263) for QCIF images is set to ± 15 sampling positions. Permitting half-pel accurate motion compensation, $63^2 = 3969$ different spatial displacement vectors could be transmitted. In our VQ interpretation of MCP, this number coincides with the code book size. For a block of 16×16 samples, which is a vector of 256 elements, the size of the code book is roughly 15.5 times the vector dimension. Extending the code book to several frames in the past increases the code book size (search range) to $15.5 \times M$. In what follows, we show the impact on prediction gain when extending MCP to multiple frames accommodated in the sliding window fashion in a long-term memory buffer.

A. Integer-Pel Accurate Long-Term Memory MCP

Figs. 2 and 3 show results of MCP experiments for the test sequences *Foreman* and *Mother–Daughter*, respectively. The

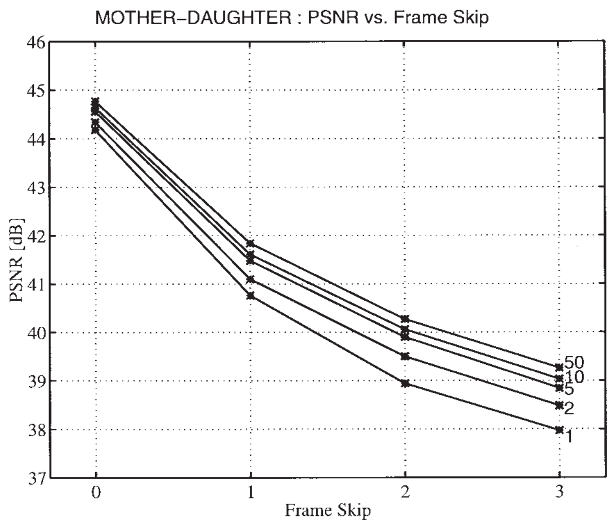

Fig. 3. Prediction gain versus frame skip parameter for the sequence *Mother–Daughter*. Simulation conditions as for Fig. 2.

plots show the motion-compensated prediction gain measured as PSNR in decibels versus frame skip parameter measured over 100 frames. The numbers 1, 2, 5, 10, and 50 relate to the various memory sizes. The frame skip parameter is varied from 0 to 3. In order to obtain consistent results, we predict all frames between 200–299 regardless of what frame skip parameter is chosen, i.e., the frame skip parameter relates to the subsampling of the reference frames. The long-term memory is built up by original frames sampled at the frame rate corresponding to the various frame skip parameters using the sliding window memory control approach as described above. The motion search for the blocks of size 16×16 samples is conducted by full search in the long-term memory buffer containing original frames in the range $M \times [-15 \cdots 15] \times [-15 \cdots 15]$ on integer-pel positions. As a criterion for the block motion search, we use the sum of the squared differences (SSD) between the displaced and original frame. The bit rate is neglected at this point.

We observe that, extending MCP to several frames, the prediction can be significantly improved. The gains due to increased memory size get larger with increased frame skip. We suggest the reason to be that successive frames get more decorrelated as the value of the frame skip parameter increases. Hence, the chance for a frame other than the immediately preceding one to be chosen increases, and with that, the relative gain when increasing the long-term memory. When comparing memories $M = 1$ and $M = 50$ for the case of a frame skip parameter of 2, we obtained a PSNR gain of 2 dB for the *Foreman* sequence and 1.3 dB for our second test sequence *Mother–Daughter*. In practical video codecs, half-pel accurate MC has been recognized to provide large coding gains due to the multihypothesis prediction effect [21], [22]. Thus, we will move on to combining half-pel accurate MCP and our long-term memory approach.

B. Half-Pel Accurate Long-Term Memory MCP

In order to extend our approach to half-pel accurate ME, we have considered the following methods.

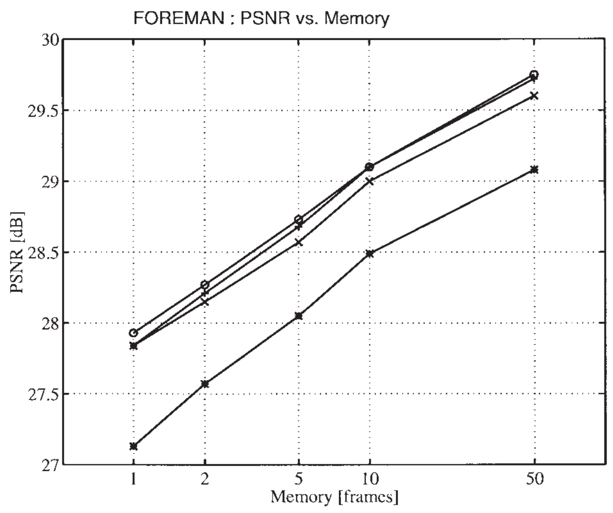

Fig. 4. Prediction gain versus memory for the sequence *Foreman* for frame skip parameter of 2. The search range is ±15 samples horizontally and vertically. The curves marked with "$*$" relate to integer-pel accurate MC, while the curves marked with "$\times$," "$+$," and "$\circ$" correspond to methods 1), 2), and 3) for half-pel accurate MC.

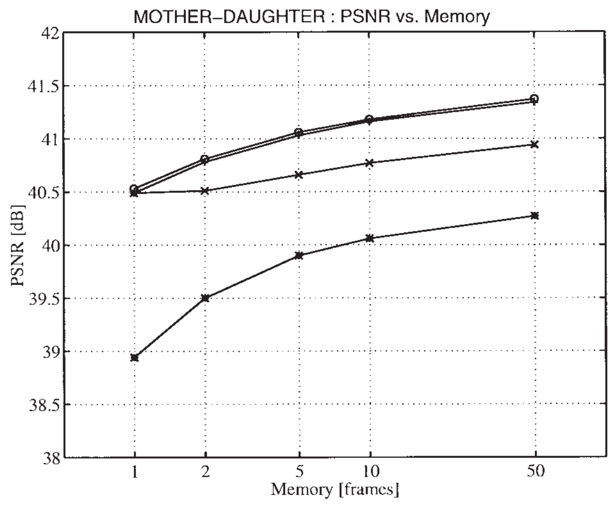

Fig. 5. Prediction gain versus memory for the sequence *Mother–Daughter*. Simulation conditions as for Fig. 4.

1) Find the optimum integer-pel accurate motion vector by full search in the complete long-term memory buffer, and obtain the final motion vector by half-pel refinement.
2) For each frame, find a motion vector by full search on integer-pel positions, followed by half-pel refinement. Determine the final motion vector by choosing among the motion vectors found for each frame.
3) Full search on all half-pel positions in the long-term memory buffer.

Figs. 4 and 5 show the comparisons of the half-pel search methods for our test sequences *Foreman* and *Mother–Daughter*, respectively. As a reference, we also show the result obtained with full-pel accuracy. As a distortion measure, we again used SSD between the displaced and original frame neglecting bit rate. The memory is varied from 1 to 50, and the frame skip parameter is set to 2. The plots show prediction gain versus memory size in logarithmic

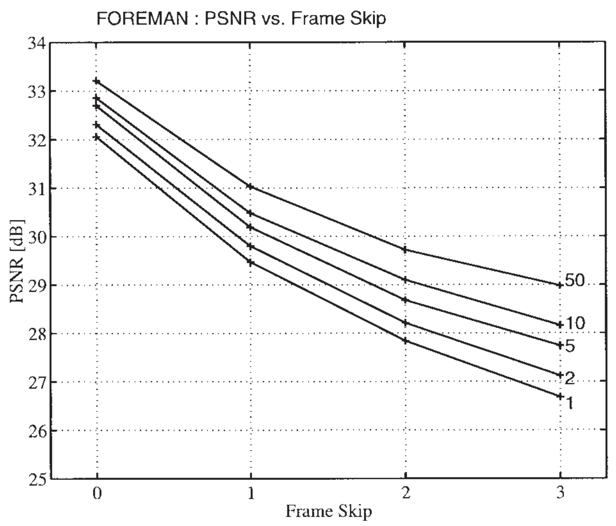

Fig. 6. Prediction gain versus frame skip parameter for the sequence *Foreman* for memory sizes $M = 1, 2, 5, 10, 50$. The search range is ± 15 samples horizontally and vertically. ME is conducted using full search integer-pel followed by half-pel refinement for each frame [method 2)].

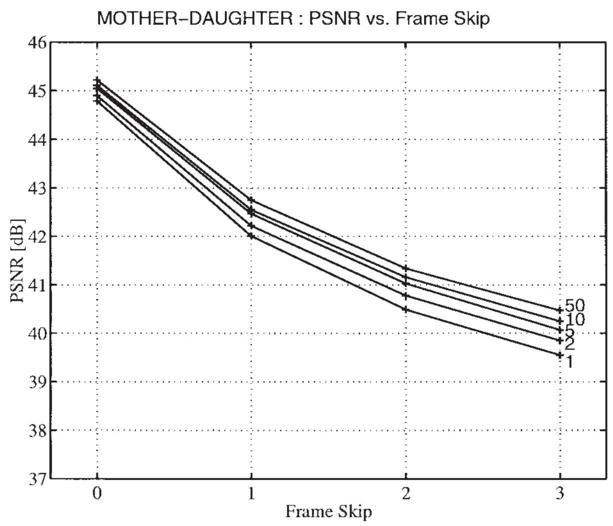

Fig. 7. Prediction gain versus frame skip parameter for the sequence *Mother–Daughter*. Simulation conditions as for Fig. 6.

scale. The curves marked with "$*$" relate to integer-pel accurate MC, while the curves marked with "$\times$," "$+$," and "$\circ$" correspond to methods 1), 2), and 3) for half-pel accurate MC.

First of all, the curves relating to integer-pel accurate MC for both of our test sequences that we consider as two extremes behave differently. The prediction gains for the *Foreman* sequence increase linearly versus logarithmic memory scale, whereas the curve for the *Mother–Daughter* sequence seems to approach a saturation at larger memory sizes, although the average prediction gain should never degrade for increasing memory. Furthermore, comparing the three methods for half-pel MC, the approach labeled 1) ("$\times$") shows significant degradation in PSNR when compared to the upper bound, i.e., the full search half-pel result 3) ("$\circ$"). On the other hand, the results obtained by method 2) ("$+$") show only marginal losses while saving significant computation time compared to 3) ("$\circ$"). Hence, we propose method 2) for half-pel accurate MC. Figs. 6 and 7 demonstrate prediction gain versus frame skip parameter. The plots are obtained under the same conditions as the results in Figs. 2 and 3, besides the integer-pel motion search being exchanged by method 2) for half-pel accurate ME. The PSNR gains in prediction error when comparing MCP with 1 frame to 50 frames in the long-term memory are 2.3 dB for the *Foreman* and 1.1 dB for the *Mother–Daughter* sequence at a frame skip parameter of 2. Hence, we can conclude that long-term memory MCP provides significantly improved prediction gain. However, there are drawbacks:

1) increased computational complexity at the encoder;
2) increased memory requirement at both encoder and decoder;
3) additional bit rate to transmit the variable time delay to the decoder.

We believe that the rapid progress of semiconductor technology will overcome problems 1) and 2) in a short time. Problem 3) can be solved by rate-constrained estimation algorithms that are discussed in the next section.

IV. Integration into H.263

Our motivation for integrating motion-compensated long-term memory prediction into an H.263-based video system is twofold: 1) the algorithm is well defined [1], [2], 2) the test model of the H.263 standard, TMN-2.0, can be used as reference for comparison, and 3) the H.263 Recommendation specifies a state-of-the-art video coding algorithm.

Extending an H.263 codec to multiple frame MCP requires several algorithmic changes:

- rate-constrained multiple frame motion estimation using block sizes 8×8 and 16×16;
- modified motion vector prediction;
- rate-constrained mode decision.

Regarding the bit-stream syntax, we just have to multiplex the codes for the time delay into the bit stream on the macroblock layer [1]. In the following, we will discuss each of these changes in detail.

A. Rate-Constrained Multiple-Frame Motion Estimation

The motion vectors constituting the spatial displacement vectors and the time delay have to be transmitted as side information requiring additional bit rate. In order to control the bit rate for the motion information, the criterion for the block motion search is the minimization of the Lagrangian cost function

$$J(\boldsymbol{d}) = D(\boldsymbol{d}) + \lambda_{\text{motion}} R(\boldsymbol{d} - \boldsymbol{p}) \tag{1}$$

where $D(\boldsymbol{d})$ is a distortion measure for a given motion vector $\boldsymbol{d} = (d_x, d_y, d_t)$, such as the SSD or the sum of the absolute differences (SAD) between the displaced frame from long-term memory and the original, and $R(\boldsymbol{d} - \boldsymbol{p})$ is the bit rate associated with a particular choice of the spatial displacement and time delay given its prediction $\boldsymbol{p} = (p_x, p_y, p_t)$.

In general, by controlling λ_{motion} we can trade off prediction gain and motion vector bit rate. For fixed block sizes, the minimum distortion achievable is limited. On the other hand, the rate–distortion performance of MCP can also be controlled by using variable block sizes or, in general, variable-shaped segments of the image [23], [24]. However, in most cases, the segmentation has to be transmitted as side information, which may be prohibitive for low bit rates. Therefore, the accuracy of the segmentation has to be weighted against the prediction gains of segment-based MCP [25]. H.263 decoders can only handle images segmented either into blocks of 16 $\times$ 16 or 8 $\times$ 8 pels. This very coarse structure of segmentation requires only a small amount of bit rate to be transmitted, and appears to be efficient in terms of rate–distortion performance.

It has been recognized that the performance of video coding strongly depends on the bit allocation in hybrid video coding [16]. In this work, we utilize both tools: rate-constrained ME and the H.263 segmentation architecture by permitting the H.263 INTER-4V mode (Annex F) in addition to the H.263 baseline modes [1].

B. Prediction of the Motion Vector

Since the additional bit rate required for transmission of the motion parameters limits the efficiency of our approach, we are trying to lower the bit rate for transmitting the motion information $R(\boldsymbol{d}-\boldsymbol{p})$ by determining appropriate motion vector predictors $\boldsymbol{p} = (p_x, p_y, p_t)$ and entropy coding of the motion vector prediction error. The prediction approach is based on the underlying assumption that, because of the high correlation of the video source, the code assigned by the compression scheme also shows high correlation. Previous publications on predictive motion vector coding [26], [27] show that the H.263-based median prediction of the spatial displacement vectors [1] appears to be the most efficient method, in terms of computation as well as prediction efficiency as opposed to linear prediction. As main reasons for these results, we suggest that: 1) the prediction efficiency is not very sensitive to the prediction accuracy, and 2) spatial displacement vectors assigned to blocks of sizes 16 $\times$ 16 and 8 $\times$ 8 do not follow the law of Markov random fields. The problem that we are concerned with in this paper is even more extreme with respect to argument 2) in that we have to predict a motion vector containing the spatial displacement vector pointing possibly into a different frame than the adjacent spatial displacement vectors.

In order to apply the median methodology for motion vector prediction, we have separated the problem of vector prediction into three scalar ones. We have found that prediction of the time delay is very difficult. While median and affine prediction of the time delay can cause an even higher bit rate than without prediction, code book switching based on the finite-state method [27] occasionally yields only insignificant gains. Hence, in the following, the time delay is transmitted without predicting it, i.e., $p_t = 0$.

In order to transmit the spatial displacements, we employ the following method. The predictor for the spatial displacement vector (p_x, p_y) is computed using displacement vectors taken from a region of support (ROS). The ROS includes previously coded blocks that are close spatially and temporally as shown in Fig. 8. The block in the center of the five blocks in frame $t-1$ is located in the same position as the block with

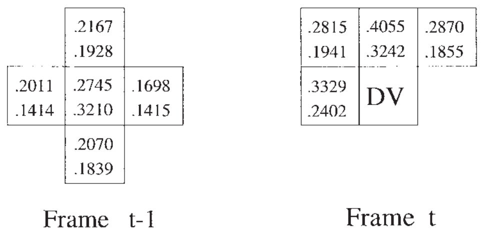

Fig. 8. Region of support for predicting DV. The correlation coefficients between the horizontal (top) as well as the vertical component (bottom) of DV and the displacement vectors of the ROS are obtained by conventional block matching with the immediately preceding frame for a set of ten training sequences with the frame skip parameter set to the value of 2.

displacement vector DV in frame t. We measured correlation coefficients between the horizontal (top) as well as the vertical component (bottom) of DV and the displacement vectors of the ROS for a set of ten training sequences by conventional block matching with the immediately preceding frames with a frame skip parameter of 2.

In order to determine (p_x, p_y), the time delay d_t for the current block is transmitted first. Then, the spatial displacements assigned to blocks in the ROS are selected in case their time delay coincides with the time delay of the current block. The result is sorted in descending order of the correlations between the spatial displacement parameters of the current block and the blocks of the ROS. The predictor is formed by taking the median from the first three of the sorted spatial displacement vectors. In case there are fewer than three displacement vectors available, only the first displacement vector is used as a predictor if it exists. Otherwise, we set the predictor $(p_x, p_y) = (0, 0)$.

C. Rate-Constrained Mode Decision

The idea of rate-constrained mode decision has been published in [28]. In contrast to the work presented in [28], we consider all macroblocks as coded independently, i.e., the current macroblock is coded, given the coding decisions for all of the past macroblocks.[1] The H.263 interprediction modes INTER, INTER-4V, and UNCODED[2]are extended to long-term memory MC. The INTER and UNCODED mode are assigned one codeword representing the variable time delay for the entire macroblock. The INTER-4V mode utilizes four time parameters, each associated with one of the four 8 $\times$ 8 motion vectors.

To run our H.263 as well as our long-term memory coder, we have modified the encoding strategy as utilized by the TMN-2.0 coder. Our encoding strategy differs for the ME and the mode decision, where our scheme is motivated by rate–distortion theory. The problem of optimum bit allocation to the motion vectors and the residual coding in any hybrid

[1] As an aside, we expect the gains possible when utilizing the dependencies between macroblocks for the long-term memory coder to be larger as is possible for the H.263 codec [28] since, for long-term memory coding, there are many more options to encode a macroblock as for H.263. However, for the sake of complexity and reproducibility, we address the independent encoding of macroblocks in this paper.

[2] We call the INTER mode the UNCODED mode when the COD bit indicates copying the macroblock from the previous frame without residual coding [1].

video coder is a nonseparable problem requiring a high amount of computation. To circumvent this joint optimization, we split the problem into two parts: motion estimation and mode decision.

The motion estimation is performed as described above using the minimization of the Lagrangian cost function. For a macroblock, the best motion vector on each frame is found by full search on integer-pel positions followed by half-pel refinement. The integer-pel search is conducted over the range $[-15\cdots15]\times[-15\cdots15]$ pels. The distortion is computed by the sum of the absolute differences (SAD) between the displaced and original frame, and the rate is computed by the bit rate occupied for the motion vector. We have chosen SAD here to make a fair comparison to TMN-2.0, where also the SAD is used for ME. The impact of overlapped block motion compensation is neglected in the motion estimation.

Given the displacements for each particular mode, we are computing the overall rate–distortion costs. The distortion is computed by SSD between the reconstructed and original frame, and the rate is computed including the rates of macroblock headers, motion parameters, and DCT quantization coefficients. The mode with the smallest Lagrangian cost is selected for transmission to the decoder. In case of long-term memory MCP, the ME followed by the mode decision as described is conducted for each frame in the long-term memory buffer.

Since there are now two Lagrangian cost functions to be minimized, we employ two different Lagrange multipliers: one for the motion search $(\lambda_{\rm motion})$, and the other one for the mode decision $(\lambda_{\rm mode})$. Furthermore, the distortion measures differ. Hence, the selection of the Lagrange parameters remains rather difficult in our coder. In this work, we employ the heuristic $\lambda_{\rm motion}=\sqrt{\lambda_{\rm mode}}$, which appears to be sufficient. The parameter $\lambda_{\rm mode}$ itself is derived from the rate–distortion curve that we computed using the TMN-2.0 H.263 coder. Note that there are various ways to obtain the desired Lagrange parameter, and more sophisticated ones than ours, especially when using Lagrangian bit allocation in a practical video coder. However, we have chosen this approach because of its simplicity and its reproducibility.

D. Huffman Codes for the Time Delay

In order to transmit the time delay d_t, we have generated a Huffman code table for each memory size. In [24], the entropy-constrained design of a complete quad-tree video codec is presented. The impact of the rate–distortion optimized bit allocation on the Huffman code design is demonstrated. From the results in [24], we conclude that if we are using rate-constrained bit allocation, we should include it into the design procedure, resulting in an iterative algorithm similar to that of [14]. For further details on iterative Huffman code design, refer to [24].

In our design algorithm, a set of ten QCIF training sequences, each with 10 s of video, is encoded at 10 frames/s. The Lagrange multiplier is set to $\lambda_{\rm motion}=150$ while using SSD as a distortion measure and the overall motion vector bit rate, including the bit rates of the spatial displacement and time delay. During encoding, histograms are gathered on the time delay parameter to design the Huffman codes which are employed in the next iteration step. The loop is performed until convergence is reached, i.e., the changes in the overall Lagrangian costs become small. The spatial displacements (d_x,d_y) are transmitted using the H.263 MVD table [1].

V. Simulation Results

In this section, we demonstrate the performance of the proposed approach. Again, we have chosen the QCIF sequences *Foreman* and *Mother–Daughter* since we consider these as extremes in the spectrum of low bit-rate video applications. These sequences were not part of the training set. The coder is run with various quantizers that are fixed when coding 300 frames of video sampled with 10 frames/s. All results are generated from decoded bit streams.

A. Rate–Distortion Performance

First, let us compare coding efficiency permitting only modes INTRA, INTER, and UNCODED for the TMN-2.0 codec, our rate–distortion-optimized H.263 codec, and the long-term memory MCP codec. In this "baseline" mode, the motion estimation for all codecs is performed very similarly in that the SAD of the displaced frame is used for the full search on integer-pel positions in the range $[-15\cdots15]\times[-15\cdots15]$ followed by half-pel refinement. In addition to the SAD, the TMN-2.0 uses a bias of -100 applied to the motion vector (0, 0) in order to prefer that candidate [13]. In our coders, we bias all positions by their costs in terms of $\lambda_{\rm motion}R(\boldsymbol{d}-\boldsymbol{p})$. In our H.263 coder, the rate of the spatial displacement vector is counted, whereas in the long-term memory case, the additional bit rate of the time delay is incorporated. Furthermore, for the prediction of the motion vector, we use the H.263-like median prediction of the spatial displacement conditioned on the time delay as described in Section IV-B. Also, the mode decision part differs as described above, in that the TMN-2.0 employs thresholds, while our mode decision is based on the Lagrangian cost function. Otherwise, both codecs are equivalent in terms of the residual coding and bit-stream syntax, except that we use a Huffman code table for the time delay that is multiplexed into the bit stream.

Figs. 9 and 10 show the average PSNR from reconstructed frames produced by the TMN-2.0 codec, our rate–distortion-optimized H.263 codec, and the long-term memory prediction codec versus overall bit rate. The size of the long-term memory is selected as 2, 5, 10, and 50 frames. The curve is generated by varying the Lagrange parameter and the DCT quantization parameter accordingly, whereas for the TMN-2.0 coder, the quantizer is varied only. Hence, the points marked with "$+$" in the plots relate to values computed from the entire sequence. The long-term memory buffer is built up simultaneously at the encoder and decoder by reconstructed frames. The results are obtained by measuring over frames $50\ \cdots\ 100$ in order to avoid the effects at the beginning of the sequence. The PSNR gains yielded by our rate–distortion-motivated encoding strategy are rather small, about 0.2 dB when compared against the results produced by the TMN-2.0 coder. The PSNR gains

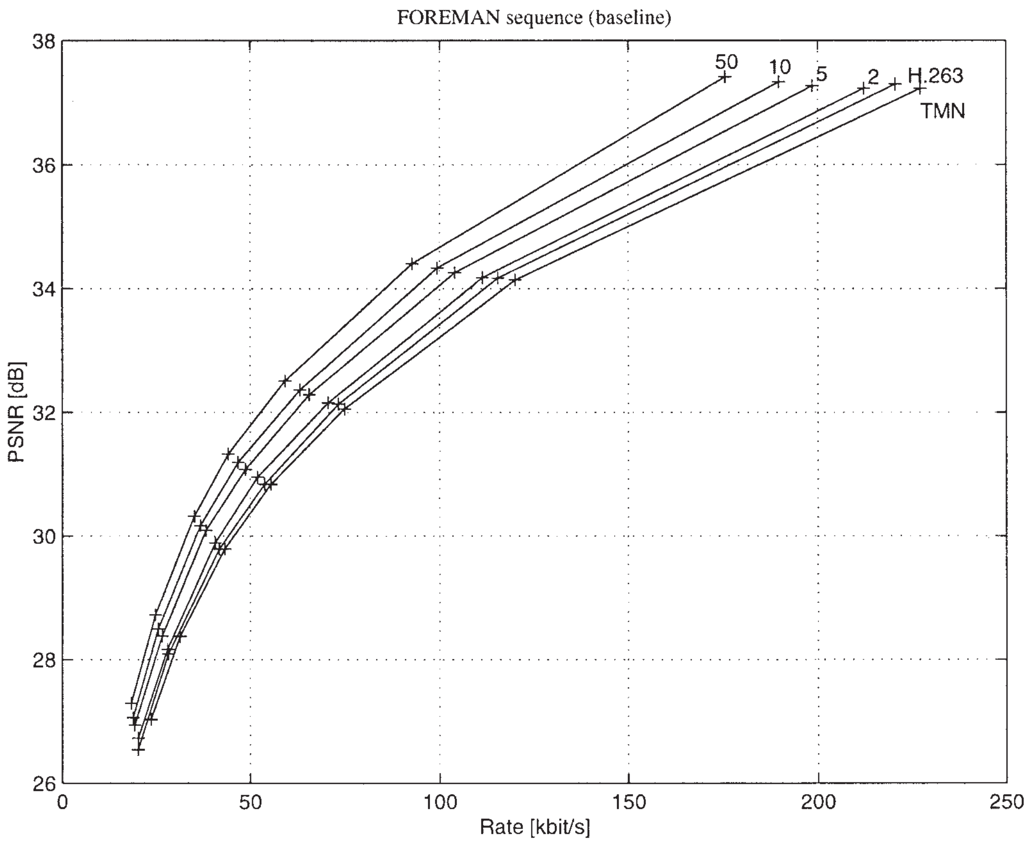

Fig. 9. PSNR versus overall bit-rate for the sequence *Foreman* for frame skip parameter of 2 when running all codecs in baseline mode. The quantization parameter is varied over the values 7, 10, 13, 16, 22, 31.

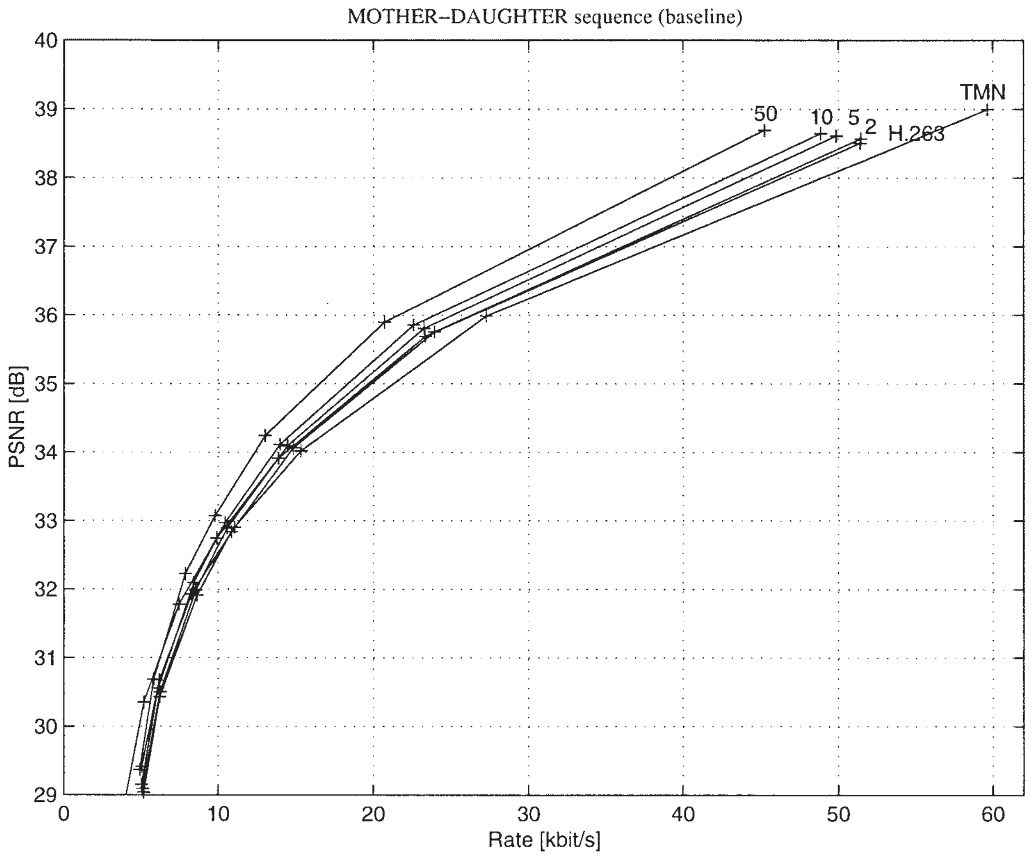

Fig. 10. PSNR versus overall bit-rate for the sequence *Mother–Daughter*. Simulation conditions as for Fig. 9.

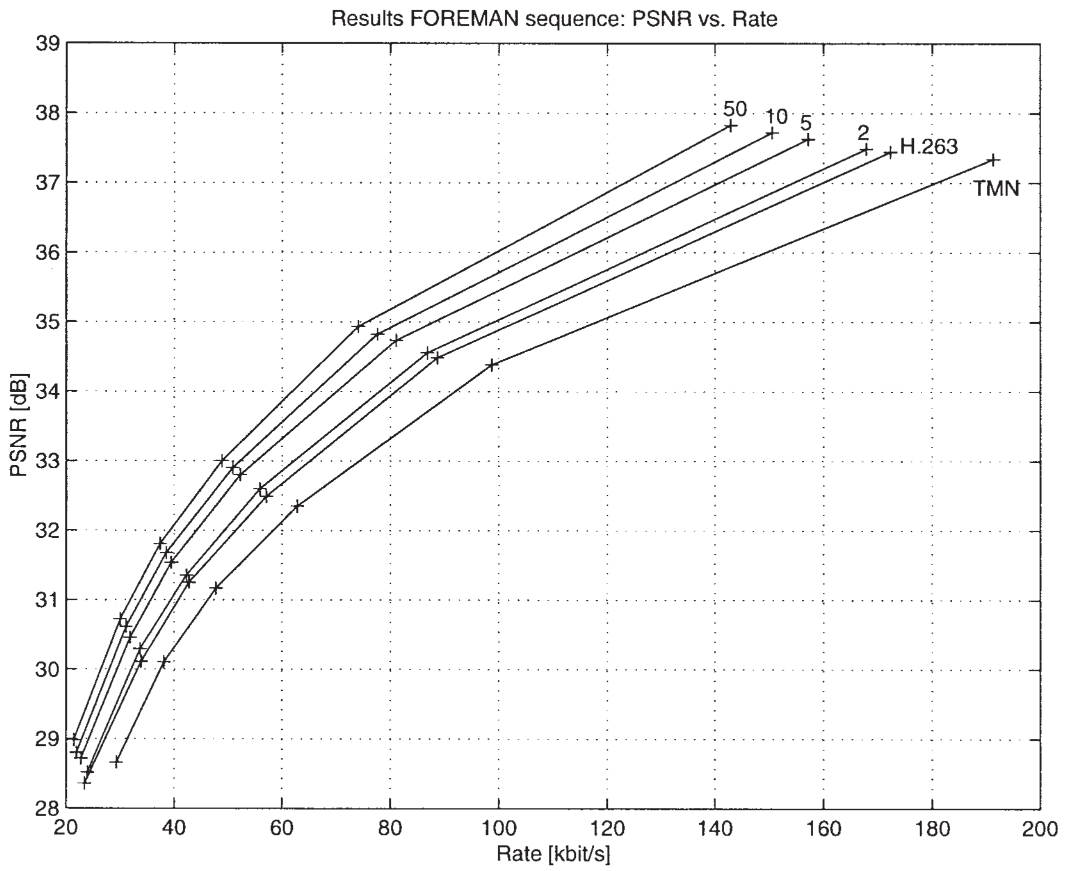

Fig. 11. PSNR versus overall bit rate for the sequence *Foreman* for frame skip parameter of 2 when permitting for all codecs the use of the INTER-4V macroblock mode in addition to the baseline mode. The quantization parameter is varied over the values 7, 10, 13, 16, 22, 31.

obtained when comparing the long-term memory MCP codec with memory $M = 50$ to our H.263 codec are 1.5 dB for the *Foreman* and 0.8 dB for the *Mother–Daughter* sequence at the higher bit-rate end. Equivalently, these PSNR gains relate to bit-rate savings of 22 and 15% for those sequences, respectively. Note that the bit-rate savings are roughly constant over the entire range of bit rates.

Figs. 11 and 12 show the average PSNR from reconstructed frames versus overall bit rate for the same settings as for Figs. 9 and 10, except that the use of the INTER-4V mode is permitted for all coders in our comparisons. Here, the impact of the rate-constrained encoding strategy is more visible when comparing our H.263 codec with TMN-2.0. For both sequences, a PSNR gain of 0.6 dB is due to our rate–distortion optimization relating to a bit-rate savings of about 11% for the *Foreman* sequence and 13% for the *Mother–Daughter* sequence at the high bit-rate end. We noticed that the usage of the full motion estimation search range $[-15 \cdots 15] \times [-15 \cdots 15]$ for the 8 × 8 block displacement vectors in the INTER-4V mode provides most of the gain for our H.263 codec. The TMN-2.0 coder only permits the use of half-pel positions for the 8 × 8 block displacement vectors that surround the previously found 16 × 16 block displacement vector, which is searched in the range $[-15 \cdots 15] \times [-15 \cdots 15]$. We have observed that using the full search range for the 8 × 8 block displacement vectors leads to improved coding performance for our rate-constrained motion estimation, whereas for the TMN-2.0, we get worse results since no rate constraint is employed. This effect is even stronger in the case of long-term memory MCP where we have many more search positions: $M \times [-15 \cdots 15] \times [-15 \cdots 15]$.

In additon to that, we obtain significant gains by using long-term memory MCP. When comparing long-term memory MCP to our own rate–distortion-optimized H.263 coder, the PSNR gains achieved are about 1.4 dB for the *Foreman* sequence and 0.9 dB for the *Mother–Daughter* sequence at the high bit-rate end when using a memory of 50 frames. The bit-rate savings related to that are 23 and 17% for those sequences.

These results demonstrate that, utilizing the long-term memory MCP in combination with rate-constrained motion estimation and mode decision, we get an improved motion-compensating prediction scheme in terms of rate–distortion performance. When permitting all modes, we obtain PSNR gains up to 2 and 1.5 dB for the *Foreman* and *Mother–Daughter* sequence, corresponding to overall bit-rate savings of 34 and 30% when comparing the long-term memory MCP coder to TMN-2.0. The gains tend to vanish for very low bit rates, i.e., for bit rates below 10 kbits/s, which is in line with our interpretation of MCP as ECVQ, where the motion search range is related to the code book size in ECVQ. As in ECVQ, the "effective" code book size reduces with bit rate [14].

In order to confirm the performance of long-term memory MCP, we have also run the coder on eight self-recorded natural

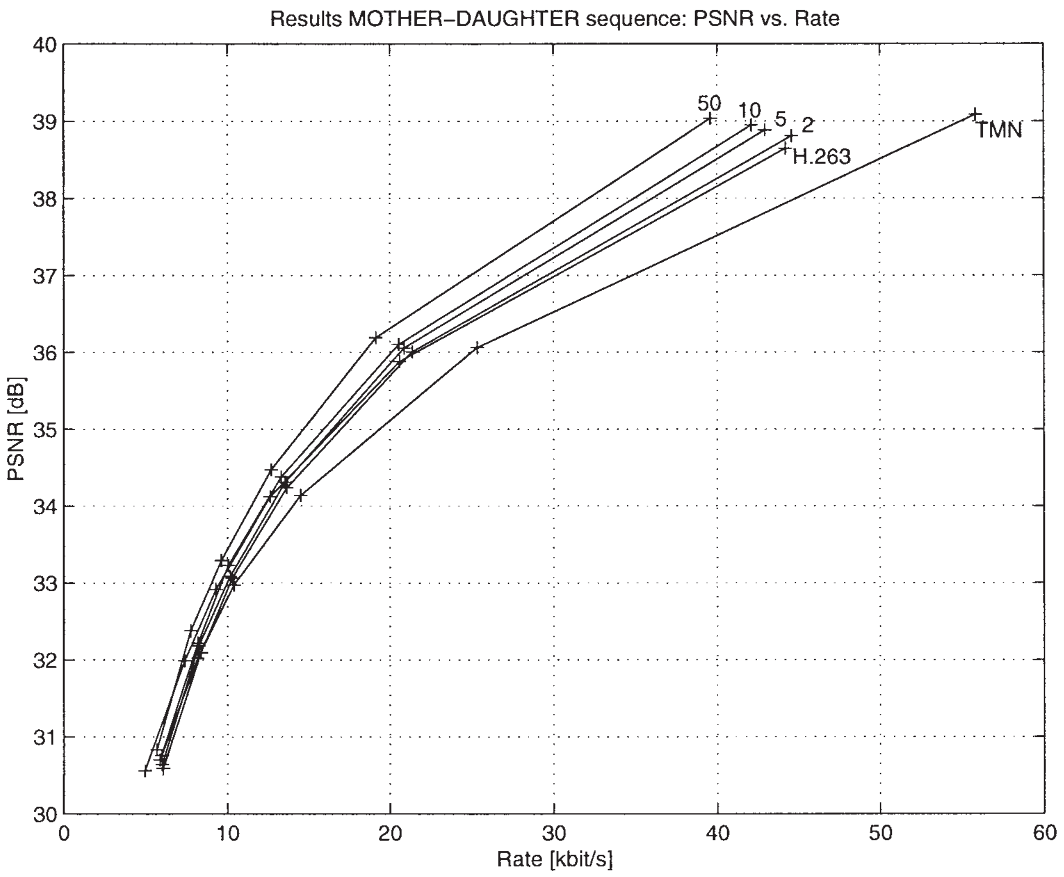

Fig. 12. PSNR versus overall bit rate for the sequence *Mother–Daughter* when permitting INTER-4V mode. Simulation conditions as for Fig. 11.

sequences. These sequences show typical interactive video phone contents. We have obtained average bit-rate savings from 12.5 up to 30%.

Finally, we should mention the exceptional bit-rate savings that we found when coding the MPEG-4 sequence *News*. We used the same settings as for the sequence *Foreman*. The result is given in Fig. 13. The MPEG-4 test sequence *News* is an artificial sequence where, in the background, two distinct sequences of dancers are displayed. These sequences, however, are repeated every 2.5 s, corresponding to 25 frames in the long-term memory buffer. Hence, our long-term memory coder obtains extremely large gains for memory $M = 50$ when compared to the TMN-2.0 curve since, in that case, the old dancer sequence is still available in the long-term memory buffer. The PSNR gains for memory $M = 50$ compared to the TMN-2.0 coder are more than 6 dB or correspond to bit-rate savings of more than 60%. We do not consider this scenario as representative, but we include the results to demonstrate the potential and the soundness of our approach.

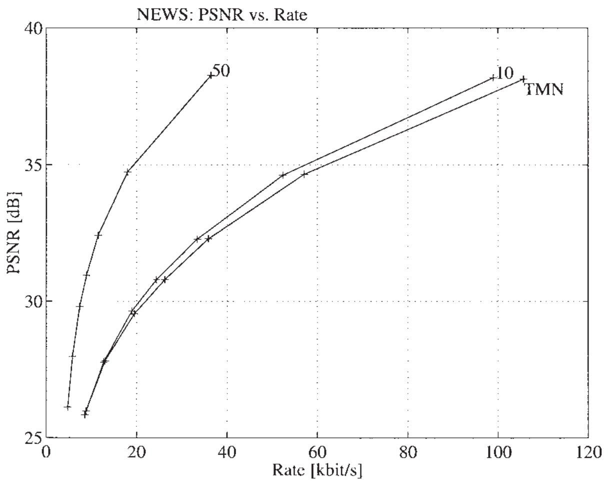

Fig. 13. PSNR versus overall bit rate for the sequence *News*. Simulation conditions as for Fig. 11.

B. Bit-Rate Partitions

In the following, we analyze the gains for long-term memory MCP in further detail. For that, we take a closer look at the experiments plotted in Fig. 11.

Fig. 14 shows the bit rate for the motion vectors including the bit rates for the spatial displacements and the time delay. Two tendencies can be observed for our H.263 and long-term memory coder: the motion vector bit rate increases as the overall bit rate increases, and the curves merge at the very low bit-rate end for the various memory sizes.

However, the curve for the TMN-2.0 coder shows a completely different behavior. For the sequence *Foreman*, the motion vector bit rate decreases as the overall bit rate increases. This results from the facts that the TMN-2.0 does not employ a rate constraint, and motion estimation is performed using the reconstructed frames (for TMN-2.0 as well as for our

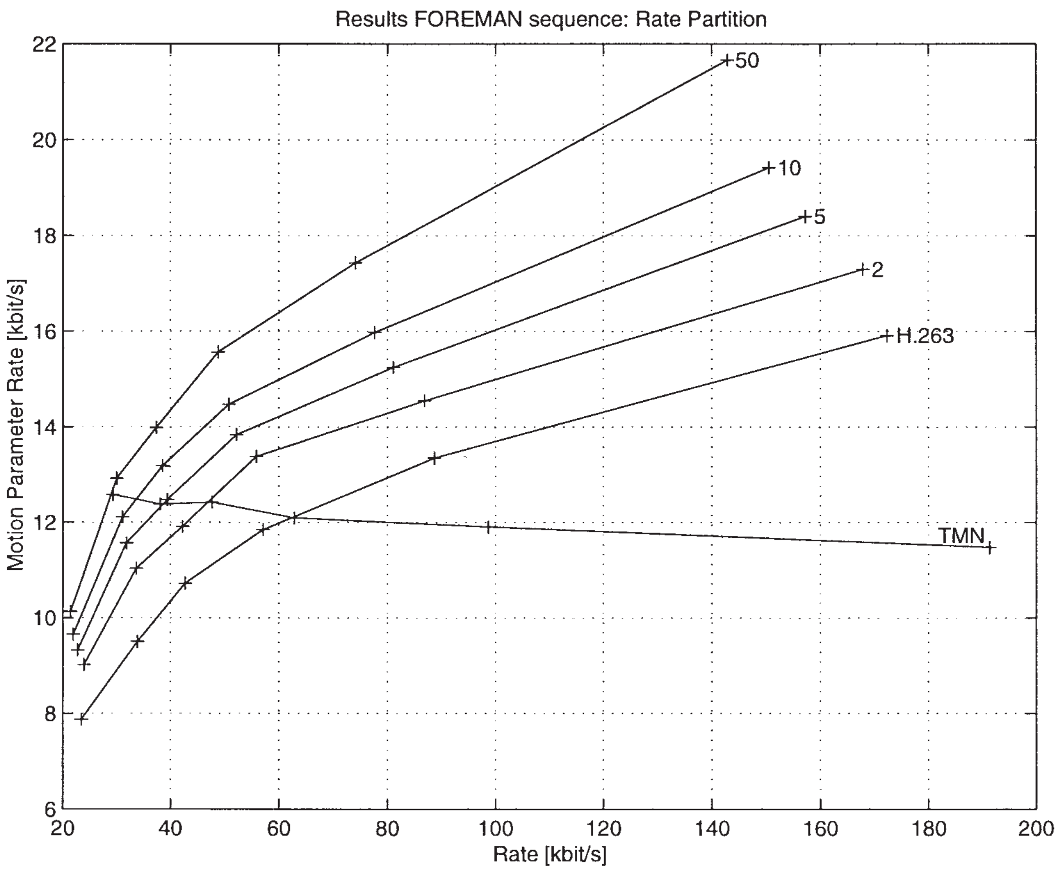

Fig. 14. Motion rate versus overall bit rate for the sequence *Foreman*. Simulation conditions are the same as for Fig. 11.

coder). As the bit rate decreases, these reconstructed frames get noisier, and since the regularization by the rate constraint is missing for the TMN-2.0, the estimates for the motion data get noisier, requiring a higher bit-rate.

Fig. 15 shows the amount of bit rate used for the spatial displacement vectors. Note that for our rate-constrained motion estimation, the bit rate occupied by the spatial displacement vectors is roughly independent of the size of the long-term memory. This indicates that our prediction scheme for the spatial displacement vectors works sufficiently well for all memory sizes.

Finally, Fig. 16 depicts the bit rate required to transmit the time delay. As the long-term memory size increases, the bit rate for the time delay increases. But this increase is well compensated by the reduction in bit rate that is used for coding of the MCP residual.

The bit-rate partition for *Mother–Daughter* as well as for other sequences shows a similar qualitative behavior to what we computed for the *Foreman* sequence.

C. Computational Complexity at the Encoder

In this section, we analyze the computational complexity associated with the ME at the encoder. Note that the long-term memory MCP decoder's computational complexity is only affected by fetches into a larger memory when compared to the H.263 algorithm. This is important to notice because, in certain applications with off-line encoding of video sequences, such as video-on-demand, the decoding complexity is the main issue.

In practical video coders, ME based on the immediately preceding frame is very often performed as full search on the integer-pel grid followed by a half-pel refinement step. In Section III-B, we compared several approaches to half-pel accurate long-term memory ME. From our comparisons, we determined method 2) in Section III-B to be included into our coder.

The computation time of the minimization task related to ME can be significantly reduced by fast search techniques, where mathematical inequalities are utilized that give lower bounds on norms of vector differences. One example for these mathematical inequalities is the triangle inequality. Incorporating the triangle inequality into the SAD or SSD yields the following inequality for the distortion:

$$
\begin{aligned}
D(\boldsymbol{s},\boldsymbol{c}) &= \sum_{[x,y]\in\mathcal{B}} |s[x,y]-c[x,y]|^p \geq \hat{D}(\boldsymbol{s},\boldsymbol{c}) \\
&= \left| \left(\sum_{[x,y]\in\mathcal{B}} |s[x,y]|^p \right)^{1/p} \right. \\
&\qquad \left. - \left(\sum_{[x,y]\in\mathcal{B}} |c[x,y]|^p \right)^{1/p} \right|^p
\end{aligned} \tag{2}
$$

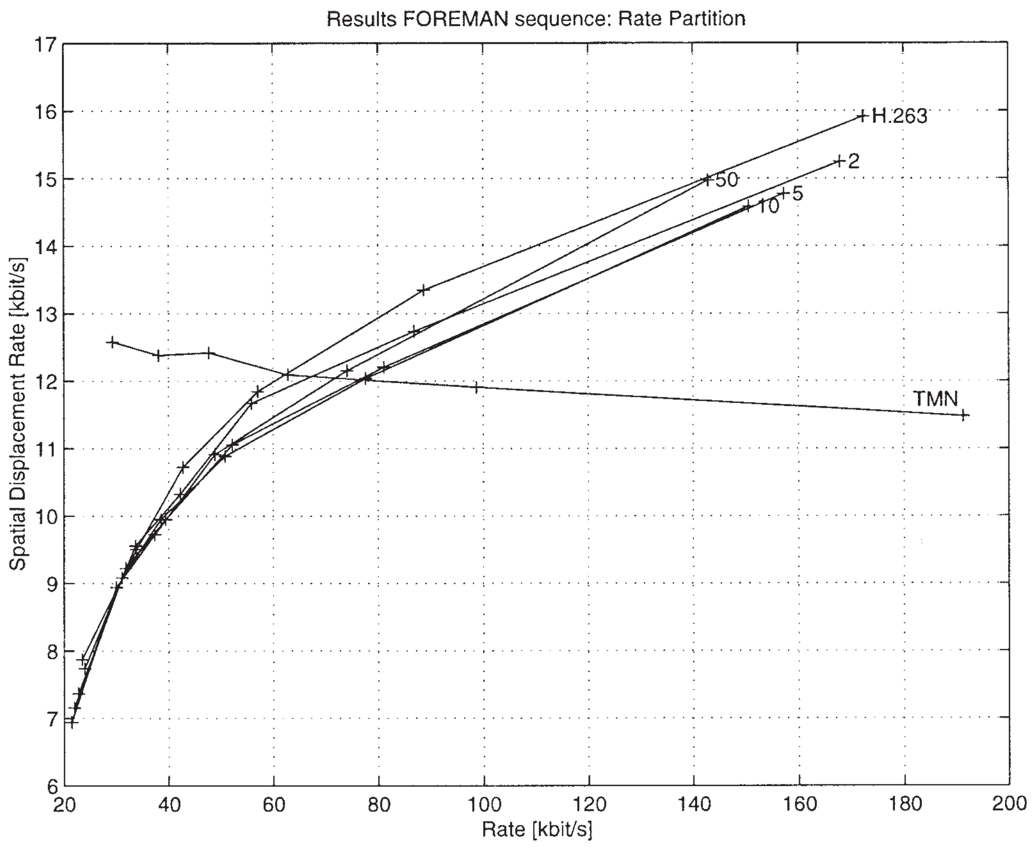

Fig. 15. Spatial displacement vector rate versus overall bit rate for the sequence *Foreman*. Simulation conditions are the same as used to produce the plots in Figs. 11 and 14.

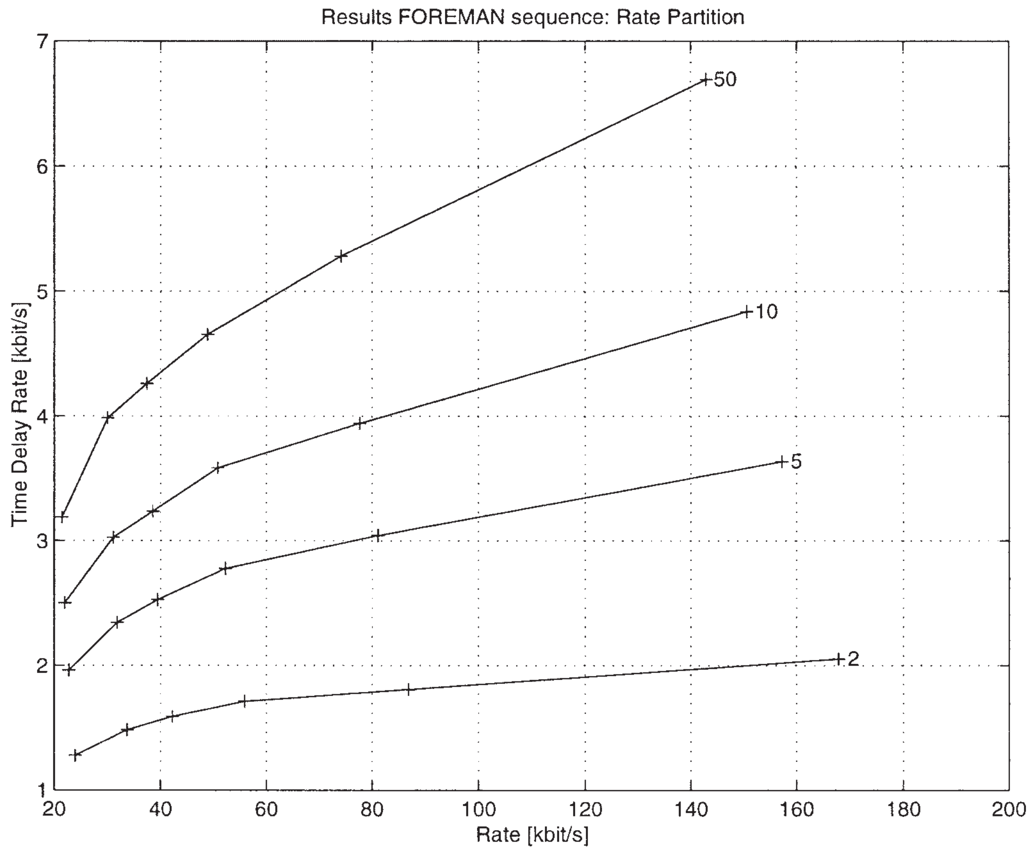

Fig. 16. Time delay rate versus overall bit rate for the sequence *Foreman*. Simulation conditions are the same as used to produce the plots in Figs. 11, 14, and 15.

by varying the parameter $p = 1$ for SAD and $p = 2$ for SSD. The set $\mathcal{B}$ comprises the sampling positions of the blocks considered, e.g., a block of 16 × 16 samples.

Assume $D_{\min}$ to be the smallest distortion value previously computed in the block motion search. Then, the distortion $D(\boldsymbol{s}, \boldsymbol{c})$ of another block $\boldsymbol{c}$ in our search range is guaranteed to exceed $D_{\min}$ if the lower bound of $D(\boldsymbol{s}, \boldsymbol{c})$ exceeds $D_{\min}$. More precisely, reject block $\boldsymbol{c}$ if

$$\hat{D}(\boldsymbol{s}, \boldsymbol{c}) \geq D_{\min}. \tag{3}$$

The special structure of the motion estimation problem permits a fast method to compute the norm values of all blocks $\boldsymbol{c}$ in the previously decoded frames [29]. In [30], a rate constraint is also incorporated.

In the sliding window long-term memory approach as described in Section II, a frame remains M time instants in the buffer when the memory size is set to M. If we compute, for each decoded image, the norms of the blocks that are candidates for the ME, we can keep these "norm" images in a second long-term memory buffer. This way, we only have to compute the norms of the most recently decoded image when predicting the current one. In addition, we also employ a hierarchical ME strategy. For each frame, we search in the order of decreasing probability of search positions, i.e., those with the lowest spatial displacement vector bit rate first, and terminate the search as soon as the cost of a candidate exceeds the previously determined minimal costs. At the cost of an additional long-term memory buffer, we obtain a system operating at reduced encoding complexity, as we will demonstrate at the end of this section.

Using the methodology delineated above, we may be able to even further reduce the encoding complexity. Assume a partition of $\mathcal{B}$ into subsets $\mathcal{B}_n$ so that

$$\mathcal{B} = \bigcup_n \mathcal{B}_n \quad \text{and} \quad \bigcap_n \mathcal{B}_n = \emptyset. \tag{4}$$

The triangle inequality (2) holds for all possible subsets $\mathcal{B}_n$. Rewriting the formula for $D(\boldsymbol{s}, \boldsymbol{c})$, we get

$$\sum_{[x,y]\in\mathcal{B}} |s[x,y] - c[x,y]|^p = \sum_n \sum_{[x,y]\in\mathcal{B}_n} |s[x,y] - c[x,y]|^p \tag{5}$$

and applying the triangle inequality for all $\mathcal{B}_n$ yields

$$\begin{aligned} D(\boldsymbol{s}, \boldsymbol{c}) &= \sum_{[x,y]\in\mathcal{B}} |s[x,y] - c[x,y]|^p \\ &\geq \sum_n \left| \left(\sum_{[x,y]\in\mathcal{B}_n} |s[x,y]|^p \right)^{1/p} \right. \\ &\qquad \left. - \left(\sum_{[x,y]\in\mathcal{B}_n} |c[x,y]|^p \right)^{1/p} \right|^p. \end{aligned} \tag{6}$$

Note that (6) is a tighter lower bound than (2); however, it requires more computation (if $\mathcal{B}_n \neq \mathcal{B}$). Hence, at this point, we can trade off the sharpness of the lower bound against computational complexity.

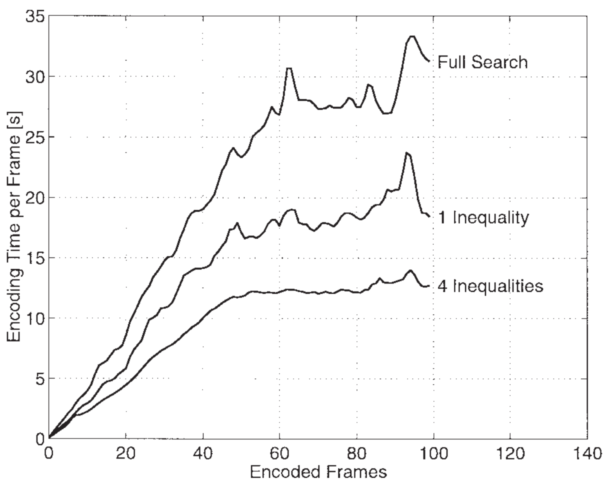

Fig. 17. Encoding time per frame versus frame numbers when coding the sequence Foreman using $M = 50$ frames in the long-term memory, a quantizer value of 10, and a frame skip parameter of 2. The curves marked with "full search" relate to no speed up applied, "1 inequality" relates to the use of one triangle inequality on the 16 × 16 block size, and the curve marked with "4 inequalities" relates to four triangle inequalities for block sizes 16 × 16, 8 × 8, 4 × 4, and 2 × 2. The curves relate to five different runs where, for each frame, the median encoding time per frame is given.

An important issue within this context remains to be the choice of the partitions $\mathcal{B}_n$. Of course, (6) works for all possible subsets that satisfy (4). However, since the norm values of all blocks in our search space have to be precomputed, we want to take advantage of the fast method described in [29]. Therefore, a random subdivision of $\mathcal{B}$ into n subsets may not be the appropriate choice. Instead, for sake of computation, a symmetric subdivision of $\mathcal{B}$ may be more desirable. In [31], it is proposed to divide a square 16 × 16 block into two different partitions. The first partitioning produces 16 subsets $\mathcal{B}_n$, each being one of 16 lines containing 16 samples. The second partition consists of 16 subsets $\mathcal{B}_n$, each being one of 16 columns containing 16 samples.

Note that the H.263 video coding standard permits blocks of size 16 × 16 and blocks of size 8 × 8 in the advanced prediction mode. Hence, we follow the approach proposed in [32] where a 16 × 16 block is decomposed into four different partitions. The 16 × 16 block is partitioned into one set of 16 × 16 samples, into four subsets of 8 × 8 samples, into 16 subsets of 4 × 4 samples, and into 64 subsets of size 2 × 2 samples. The various (subset) triangle inequalities are successively applied in the order of the computation time to evaluate them, i.e., first the 16 × 16 triangle inequality is checked, and then the inequalities relating to blocks of size 8 × 8, 4 × 4, and 2 × 2 samples are computed using (6).

Fig. 17 shows the encoding time per frame versus frame numbers when coding the sequence Foreman using $M = 50$ frames in the long-term memory using a quantizer value of 10 and a frame skip parameter of 2. The plot compares the cases of: 1) "full search" which is related to no speed up applied, 2) "1 inequality" which relates to the use of one triangle inequality on the 16 × 16 block size, and 3) the case "4 inequalities" relating to the use of four triangle inequalities for block sizes 16 × 16, 8 × 8, 4 × 4, and 2 × 2 as described in the previous paragraph. The experiments are conducted on a 300 MHz Sun UltraSPARC, single processor, 1 Gbyte RAM, Solaris 2.5. No VIS instructions are used. The curves shown relate to five different runs, where for each frame, the median encoding time per frame is depicted. The usage of one triangle inequality cuts down the encoding time per frame by 30%, while using four triangle inequalities yields a 55% saving in computation time.

VI. Conclusions

In this paper, we have presented a new technique for motion-compensated video coding that exploits long-term statistical dependencies in video sequences. We extend the spatial displacement vector by a variable time delay that is transmitted as side information. We control the bit rate by viewing MCP as an ECVQ problem. Experimental results are presented with up to 50 previous frames in long-term memory.

The prediction gains achievable with long-term memory MCP are significant. Especially, when using integer-pel accurate MC only, large improvements in MCP can be observed. This is also due to the fact that various subpel-shifted versions of video content may be available in the long-term memory buffer. Extending the MC to half-pel accuracy, we have observed that the approach of full search integer-pel accurate ME followed by half-pel refinement should be performed for each frame in the long-term memory buffer. Then, the motion vector for the entire long-term memory buffer is determined by choosing among the candidates optimized for each frame. By using this method, the computational burden associated with full search half-pel accurate ME is avoided while being only slightly suboptimal.

When extending an H.263-based codec to long-term memory MCP, we consider rate-constrained motion estimation important since it regularizes the motion estimation, lowering the variance of the estimates, and therefore lowering their bit rate. We also utilize a modified motion vector prediction technique by extending the H.263-based median to prediction of spatial displacements from multiple frames. Rate-constrained mode decision has been implemented in order to run our hybrid video codec without the heuristic thresholds of the TMN-2.0 coder.

All of these modifications lead to bit-rate savings of up to 34 and 30% for our test sequences *Foreman* and *Mother–Daughter* when permitting all modes and 50 frames in the long-term memory in comparison to TMN-2.0. These bit-rate savings partition into 11% (*Foreman*) and 13% (*Mother–Daughter*) due to our modifications to the encoding strategy and into 23% (*Foreman*) and 17% (*Mother–Daughter*) due to the impact of long-term memory MCP.

We also delineated how to lower the computation time required for the ME by using well-known techniques that are based on the triangle inequality for vector norms and by employing a hierarchical search strategy. The long-term memory approach permits speed-up methods that are operating at the cost of increased memory requirement. These methods precompute data that are attached to the decoded frames in order to evaluate whether or not a particular position may be better than a previously found one. Nevertheless, speeding up long-term memory ME is subject to future research [33].

Acknowledgment

The authors thank E. Steinbach, U. Horn, N. Färber, K. Stuhlmüller, S. Chaudhuri, M. Flierl, B. Andrews, and B. Lincoln for useful discussions and their suggestions. They would also like to thank the anonymous reviewers for their comments and suggestions.

References

[1] ITU-T Recommendation H.263, “Video coding for low bitrate communication,” ver. 1, Nov. 1995.
[2] ITU-T Recommendation H.263 Version 2 (“H.263+”), “Video coding for low bitrate communication,” Jan. 1998.
[3] ISO/IEC JTC1/SC29/WG11, “MPEG-4 video verification model,” Draft, July 1997.
[4] ITU-T, SG15/WP15/1, LBC-95-033, Telenor R&D, “An error resilience method based on back channel signaling and FEC,” also submitted to ISO/IEC JTC1/SC29/WG11 as contribution MPEG96/M0616, Jan. 1996.
[5] ISO/IEC JTC1/SC29/WG11 MPEG96/M0768, Iterated Systems Inc., “An error recovery strategy for videophone applications,” Mar. 1996.
[6] F. Dufaux and F. Moscheni, “Background mosaicking for low bit rate video coding,” in *Proc. IEEE Int. Conf. Image Processing*, Lausanne, Switzerland, Sept. 1996.
[7] ISO/IEC JTC1/SC29/WG11 MPEG96/N1648, “Core experiment on sprites and GMC,” Apr. 1997.
[8] ISO/IEC JTC1/SC29/WG11 MPEG96/M0654, “Core experiment of video coding with block-partitioning and adaptive selection of two frame memories (STFM/LTFM),” Dec. 1996.
[9] D. Hepper, “Efficiency analysis and application of uncovered background prediction in a low bit rate image coder,” *IEEE Trans. Commun.*, vol. 38, pp. 1578–1584, Sept. 1990.
[10] J. Y. A. Wang and E. H. Adelson, “Representing moving images with layers,” *IEEE Trans. Image Processing*, vol. 3, pp. 625–638, Sept. 1994.
[11] T. Wiegand, X. Zhang, and B. Girod, “Motion-compensating long-term memory prediction,” in *Proc. IEEE Int. Conf. Image Processing*, Santa Barbara, CA, Oct. 1997.
[12] ——, “Block-based hybrid video coding using motion-compensated long-term memory prediction,” in *Proc. Picture Coding Symp.*, Berlin, Germany, Sept. 1997.
[13] Telenor Research, “TMN (H.263) encoder/decoder, version 2.0,” [Online]. Available FTP: bonde.nta.no.
[14] P. A. Chou, T. Lookabaugh, and R. M. Gray, “Entropy-constrained vector quantization,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 31–42, Jan. 1989.
[15] Y. Shoham and A. Gersho, “Efficient bit allocation for an arbitrary set of quantizers,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1445–1453, Sept. 1988.
[16] B. Girod, “Rate-constrained motion estimation,” in *Proc. SPIE Conf. Visual Commun. Image Processing*, Chicago, IL, Sept. 1994, pp. 1026–1034.
[17] J. Lee, “Optimal quadtree for variable block size motion estimation,” in *Proc. IEEE Int. Conf. Image Processing*, Washington, DC, vol. II, Oct. 1995, pp. 480–483.
[18] W. C. Chung, F. Kossentini, and M. J. T. Smith, “An efficient motion estimation technique based on a rate-distortion criterion,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Atlanta, GA, May 1996.
[19] M. C. Chen and A. N. Willson, “Rate-distortion optimal motion estimation algorithm for video coding,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Atlanta, GA, May 1996, pp. 2096–2099.
[20] T. Wiegand, E. Steinbach, A. Stensrud, and B. Girod, “Multiple reference picture coding using polynomial motion models,” in *Proc. SPIE Conf. Visual Commun. Image Processing*, San Jose, CA, Feb. 1998.
[21] B. Girod, “Efficiency analysis of multi-hypothesis motion-compensated prediction for video coding,” *IEEE Trans. Image Processing*, 1997.
[22] B. Girod, E. Steinbach, and N. Färber, “Performance of the H.263 video compression standard,” *J. VLSI Signal Processing: Syst. Signal, Image, Video Technol.*, vol. 17, pp. 101–111, 1997.
[23] G. J. Sullivan and R. L. Baker, “Efficient quadtree coding of images and video,” *IEEE Trans. Image Processing*, vol. 3, pp. 327–331, May 1994.

[24] T. Wiegand, M. Flierl, and B. Girod, “Entropy-constrained design of quadtree video coding schemes,” in *Proc. IEE Int. Conf. Image Processing Appl.*, Dublin, Ireland, July 1997, pp. 36–40.

[25] K. W. Stuhlmüller, A. Salai, and B. Girod, “Rate-constrained contour-representation for region-based motion compensation,” in *Proc. SPIE Conf. Visual Commun. Image Processing*, Orlando, FL, Mar. 1996.

[26] R. Armitano, R. W. Schafer, F. L. Kitson, and V. Bhaskaran, “Linear predictive coding of motion vectors,” in *Proc. Picture Coding Symp.*, Sacramento, CA, Sept. 1994, pp. 233–236.

[27] F. Kossentini, Y.-W. Lee, M. J. T. Smith, and R. Ward, “Predictive RD-constrained motion estimation for very low bit rate video coding,” *IEEE J. Select. Areas Commun.*, 1997.

[28] T. Wiegand, M. Lightstone, D. Mukherjee, T. G. Campbell, and S. K. Mitra, “Rate-distortion optimized mode selection for very low bit rate video coding and the emerging H.263 standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 182–190, Apr. 1996.

[29] W. Li and E. Salari, “Successive elimination algorithm for motion estimation,” *IEEE Trans. Image Processing*, vol. 4, pp. 105–107, Jan. 1995.

[30] M. H. Johnson, R. Ladner, and E. A. Riskin, “Fast nearest neighbor search for ECVQ and other modified distortion measures,” in *Proc. IEEE Int. Conf. Image Processing*, Lausanne, Switzerland, Sept. 1996, pp. 423–426.

[31] Y.-C. Lin and S.-C. Tai, “Fast full-search block-matching algorithm for motion-compensated video compression,” *IEEE Trans. Commun.*, vol. 45, pp. 527–531, May 1997.

[32] C.-H. Lee and L.-H. Chen, “A fast search algorithm for vector quantization using mean pyramids of codewords,” *IEEE Trans. Commun.*, vol. 43, pp. 604–612, Feb./Mar./Apr. 1995.

[33] T. Wiegand, B. Lincoln, and B. Girod, “Fast search for long-term memory motion-compensated prediction,” in *Proc. IEEE Int. Conf. Image Processing*, Chicago, IL, 1998, vol. 3, pp. 619–622.