Adaptive Linear Prediction for Resource Estimation of Video Decoding

Yiannis Andreopoulos, *Member, IEEE*, and Mihaela van der Schaar, *Senior Member, IEEE*

***Abstract*—Current systems often assume "worst case" resource utilization for the design and implementation of compression techniques and standards, thereby neglecting the fact that multimedia coding algorithms require time-varying resources, which differ significantly from the "worst case" requirements. To enable adaptive resource management for multimedia systems, resource-estimation mechanisms are needed. Previous research demonstrated that online adaptive linear prediction techniques typically exhibit superior efficiency to other alternatives for resource prediction of multimedia systems. In this paper, we formulate the problem of adaptive linear prediction of video decoding resources by analytically expressing the possible adaptation parameters for a broad class of video decoders. The resources are measured in terms of the time required for a particular operation of each decoding unit (e.g., motion compensation or entropy decoding of a video frame). Unlike prior research that mainly focuses on estimation of execution time based on previous measurements (i.e., based on autoregressive prediction or platform and decoder-specific off-line training), we propose the use of generic complexity metrics (GCMs) as the input for the adaptive predictor. GCMs represent the number of times the basic building blocks are executed by the decoder and depend on the source characteristics, decoding bit rate, and the specific algorithm implementation. Different GCM granularities (e.g., per video frame or macroblock) are explored. Our previous research indicated that GCMs can be measured or modeled at the encoder or the video server side and they can be streamed to the decoder along with the compressed bitstream. A comparison of GCM-based versus autoregressive adaptive prediction over a large range of adaptation parameters is performed. Our results indicate that GCM-based prediction is significantly superior to the autoregressive approach and also requires less computational resources at the decoder. As a result, a novel resource–prediction tradeoff is explored between: 1) the communication overhead for GCMs and/or the implementation overhead for the realization of the predictor and 2) the improvement of the prediction performance. Since this tradeoff can be significant for the decoder platform (either from the communication or the implementation perspective), we propose complexity (or communication)-bounded adaptive linear prediction in order to derive the best resource estimation under the given implementation (or GCM-communication) bound.**

Manuscript received March 29, 2006; revised August 1, 2006. This work was supported by the National Science Foundation under Grant NSF CCF 0541867 (Career), Grant NSF CCF 0541453, and Grant NSF CNS 0509522. This paper was recommended by Associate Editor Z. He.

Y. Andreopoulos was with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA. He is now with the Electronic Engineering Department, Queen Mary University of London, London, E1 4NS, U.K.

M. van der Schaar is with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: mihaela@ee.ucla.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2007.896662

***Index Terms*—Complexity modeling, linear prediction, resource estimation, video decoding.**

I. Introduction

THE proliferation of media-enabled portable devices based on low-cost processors makes resource-constrained video decoding an important area of research. However, state-of-the-art video coders tend to be highly content-adaptive in order to achieve the best compression performance [21]–[24]. Therefore, they exhibit a highly variable complexity and rate profile [1]–[3]. This makes efficient resource-prediction techniques for video decoding a particularly important and challenging research topic.

A. Related Work

Previous work can be broadly separated in two categories. The first category encompasses coder-specific or system-specific frameworks for resource prediction. For example, Horowitz *et al.* [1] and Lappalainen *et al.* [2] analyze the timing complexity of an H.264/AVC baseline decoder on two popular platforms using reference software implementations. Valentim *et al.* [3] analyze the visual part of MPEG-4 decoding with the use of descriptors from the encoding parameters and the definition of worst case performance results. Saponara *et al.* [4] analyze in a tool-by-tool basis the H.264/AVC encoding and decoding complexity in terms of memory usage and relative execution time using profiling tools and compare it to MPEG-4 Simple Profile. Ravasi and Mattavelli [5] propose a tool-based methodology for complexity analysis of video encoders or decoders using a typical software description. Most of these approaches are inspired by profiling-based methodologies for execution time prediction [6] and specialize their application to media decoding algorithms.

The second category of approaches attempts to capture the dynamic resources of video decoders following online adaptation or offline training approaches that "learn" the features of a particular system following a generic methodology. For example, Bavier *et al.* [7] predict MPEG decoding execution time using offline measurements for typical video sequences and regression analysis. Dinda and Hallaron [8] experiment with different autoregressive models for host load prediction. Yuan and Nahrstedt [9] predict video encoding complexity using a histogram-based estimation. Liang *et al.* [10] analyze processor utilization for media decoding and compared various autoregressive linear predictors. Similarly, Sachs *et al.* [11] model H.263 encoding time using a variety of linear predictors. Stolberg *et al.* [12] use a mixed methodology based on prediction techniques and offline tool-based analysis and, in more recent work [13], propose a generic approach for offline

or online linear mapping of critical function executions to the observed timing complexity. He *et al.* [14] proposed an analytic model for power/rate/distortion estimation of hybrid video encoding and investigated several tradeoffs using dynamic voltage-scaling techniques. Mattavelli and Brunetton [15] derive a broad framework for estimation of media decoding execution-time based on a linear mapping of encoding-modes (extracted from the video bitstream) to the execution time of the critical functions of the video decoder. Offline measurements were used to define the parameters of the linear mapping. Our previous work [16]–[18] follows a similar approach but, instead of encoding decisions (which are both algorithm and implementation dependent), it uses compressed-content features, such as motion vectors and the number of nonzero transform coefficients, which apply in a broader category of video compression algorithms and are also easily extracted during encoding. In addition, instead of a linear estimation of decoder functions, we recently proposed a decomposition framework [17] that estimates platform-independent generic complexity metrics (GCMs), such as the number of motion-compensation operations, the number of nonzero filter-kernel applications for fractional-pixel interpolation and spatial reconstruction of a video frame.

B. Contribution of the Paper

In this paper we focus on online prediction-based approaches for video decoding resource estimation. Following previous work [10], [15], [17], we attempt to systematically address the following two basic problems for state-of-the-art video decoding algorithms with rapidly time-varying complexity profile:

- a general framework for resource-optimized prediction of video decoding execution time under a given computational bound for the predictor;
- a comparison of autoregressive prediction methods with adaptive linear prediction methods based on encoder-generated GCMs.

Instead of focusing on worst case complexity estimation, as it is commonly done in recent video complexity verifier (VCV) models for MPEG video [3], we follow a recently proposed VCV model [19] that focuses on average decoding complexity by introducing variable-size frame buffers and controlling the decoding delay. This type of VCV model was shown to be significantly better than the conventional VCV models for streaming video and soft real-time decoding applications, where additional decoding delay is permitted. This is due to the fact that the model targets systems with flexible resource utilization rather than systems that are designed based on worst case performance guarantees.

For both autoregressive and GCM-based prediction, we examine different configurations for the adaptive predictor depending on the granularity of the decoder measurement feedback, the complexity requirements imposed at the decoding and the granularity of the input GCMs (if available). In addition, an optimization framework is proposed that selects the prediction parameters under prediction-accuracy constraints using offline statistical estimates of the prediction error such that decoder or transmission resources are utilized in the best possible way.

II. Video Decoding Architectures—GCMs

A. Motion-Compensated Video Decoding

In MPEG video coding standards, after the motion-compensation step, the output frames are categorized as intra-frames (I-frames—denoted as L frames in this paper), predicted frames (P-frames) from past pictures, and bi-directionally predicted frames (B-frames). Both P- and B-frames are denoted as H frames in this paper. All output frames are spatially transformed and entropy-coded. The decoder receives the compressed bitstream and, after decoding and the inverse spatial transform, inverts the prediction relationships in order to reconstruct the decoded video frames A. In order to allow efficient synchronization and random access capabilities to the decoded bitstream, typical coding structures follow a temporal prediction pattern for a group of pictures (GOP). An example of the decoding process for a "IBPBPBPB" GOP structure is pictorially shown in Fig. 1(a), where we also highlight the analogy between L and H frames and the conventional I-, P- and B-frame notation used in MPEG [20]. The superscripts (t, n) indicate for each frame the temporal-decomposition level it belongs to (decreased levels t indicate dyadically increased frame-rate, $0 \leq t \leq T+1$), as well as the frame index within each level (n). Notice that all of the H frames (and $L^{T+1,0}$) are first entropy-decoded and then reconstructed spatially by performing the inverse spatial transform prior to their use in the inverse temporal reconstruction example of Fig. 1(a).

Although the example of Fig. 1(a) corresponds to a temporal analysis framework with three P-frames $(H^{2,0}, H^{2,1}, H^{2,2})$ and four B-frames $(H^{1,0}, H^{1,1}, H^{1,2}, H^{1,3})$, other combinations may utilize multiple reference frames from the past or the future [21]. In addition, in recent state-of-the-art coding frameworks, such as the MPEG/ITU AVC/H.264 codec [21], macroblocks of the same video frame may utilize motion-compensated prediction with variable block sizes and multiple motion vectors from several reference frames in order to adapt better to the video content characteristics. Although these techniques increase coding efficiency by a significant amount, they pose new challenges for efficient resource-prediction approaches since a rapidly varying complexity profile is generated. Consequently, this demands a generic treatment of the resource-prediction issue that can encapsulate a variety of encoding parameter settings in an automatic manner.

Scalable video coding architectures based on motion-compensated temporal filtering (MCTF) have been shown to be a generalization of the previous predictive coding architectures of MPEG [22]. In particular, at the encoder, video frames are filtered into L (low-frequency or average) and H (high-frequency of difference) frames by performing a series of motion-compensated predict and motion-compensated update steps [22]. The process is applied in a hierarchy of T temporal levels. An example instantiation of the inverse MCTF that reconstructs the input video at the decoder side is shown in Fig. 1(b) (with $T = 3$), where, based on the decompressed $L^{4,0}$ frame and the hierarchy of H frames, the temporal filtering is inverted (by performing inverse predict and update steps) in order to reconstruct the decoded video frames A (shown at the top of the figure).

Similarly as before, in state-of-the-art MCTF-based video coding [23], [24], the example of inverse MCTF illustrated in

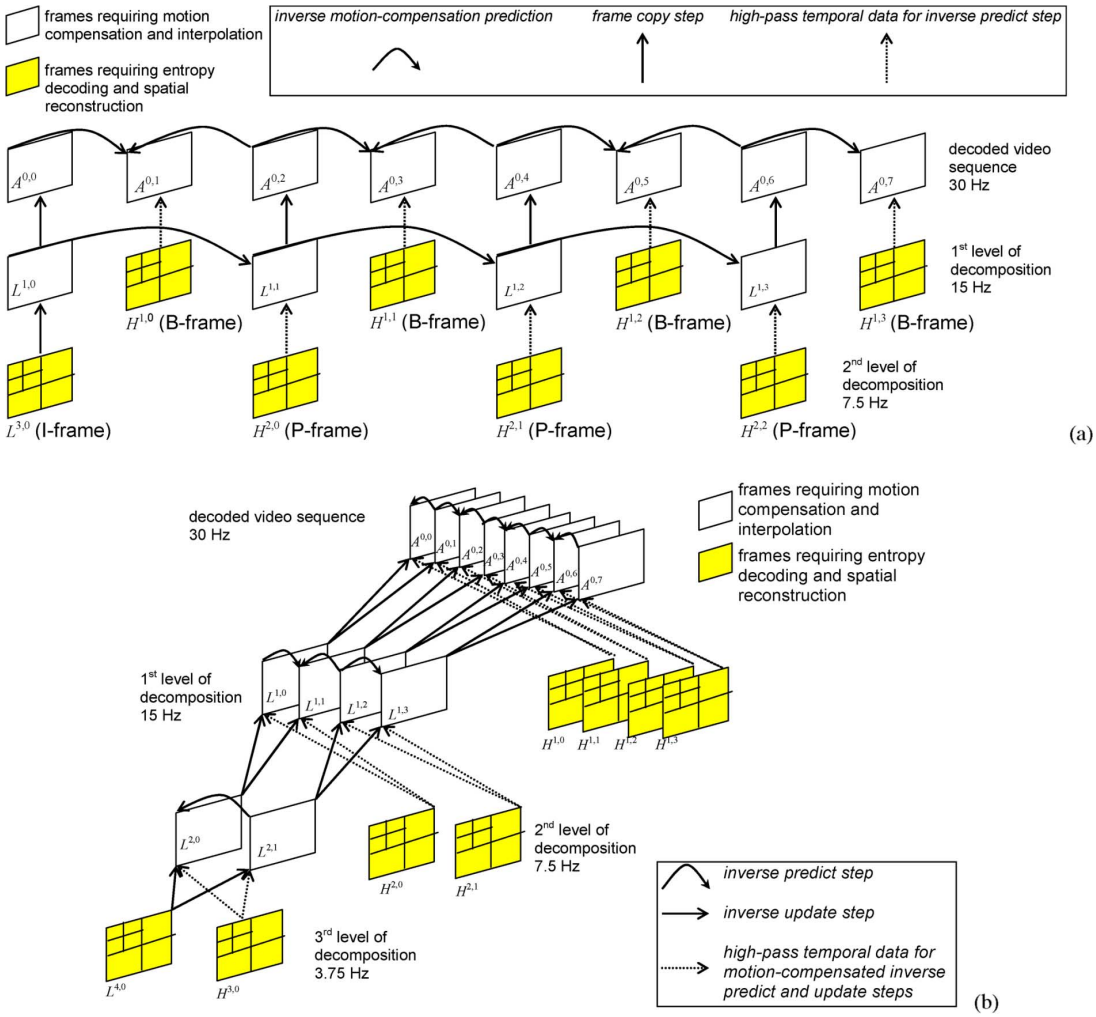

Fig. 1. (a) Example of the temporal reconstruction of conventional MPEG coding using a GOP structure with one I-frame ($L^{3,0}$), three P-frames ($H^{2,0}, H^{2,1}, H^{2,2}$), and four B-frames ($H^{1,0}, H^{1,1}, H^{1,2}, H^{1,3}$). The decoded frames are indicated by a pyramidal (two-level) spatial wavelet decomposition, although different transforms such as DCT could be employed. (b) Example of MCTF reconstruction using the bidirectional Haar filter [24].

Fig. 1(b) is generalized to motion-compensated temporal filtering using multiple block sizes with different temporal filters for each macroblock. Finally, it is important to notice that, when motion-compensated update is disabled in the MCTF process, the decoder structure becomes equivalent to a conventional predictive framework. Therefore, MCTF-based frameworks can be seen as a generalization of conventional motion-compensated video decoders.

B. Generic Complexity Metrics: A Framework for Abstract Complexity Description

In order to represent at the encoder (server) side different decoder (receiver) architectures in a generic manner, in our recent work [17], [18], we have deployed a concept that has been successful in the area of computer systems, namely, a virtual machine. We assume an abstract decoder referred to as a generic reference machine (GRM), which represents the vast majority of video decoding architectures. The key idea of the proposed paradigm is that the same bitstream will require/involve different resources/complexities on various decoders. However, given the number of factors that influence the complexity of the decoder, it is impractical to determine at the encoder side the specific (real) complexity for every possible decoder architecture. Consequently, we adopt a generic complexity model that captures the abstract/generic complexity metrics of the employed decoding or streaming algorithm depending on the content characteristics and transmission bit rate. GCMs are derived by computing the average number of times the different GRM operations are executed. Assuming that the GRM is the target receiver, we developed abstract complexity measures to quantify the decoding complexity of multimedia bitstreams [17], [18].

The GRM framework for the majority of transform-based motion-compensated video decoders can be summarized by the modules illustrated in Fig. 2 [17]. The decoder modules, depicted as nodes connected by edges, communicate via decoded data that are determined by the received bitstream. For a particular realization of a video decoding algorithm (i.e., using a programmable processor, an application-specific media processor, or custom hardware), the decoding complexity associated with the information exchange between the modules of Fig. 2 may be expressed in execution cycles, processor functional-unit utilization, or energy consumption. Based on preliminary experiments that analyzed several such costs [16], we assume generic

operations for each module that are indicated by the connecting arrows.

- Entropy decoding: the number of iterations of the "Read Symbol" (RS) function is considered. This function encapsulates the (context-based) entropy-decoding of a symbol from the compressed bitstream.
- Inverse transform: the number of multiply-accumulate (MAC) operations with nonzero entries is considered, by counting the number of times a MAC operation occurs between a filter-coefficient (FC) and a nonzero decoded pixel or transform coefficient.[1]
- Motion compensation: the basic motion-compensation (MC) operation[2] per pixel or transform coefficient is the dominant factor in this module's complexity.
- Fractional-pixel interpolation: the MAC operations corresponding to horizontal or vertical interpolation (IO_H), and the MAC operations corresponding to diagonal interpolation (IO_D), can characterize the complexity of this module. Notice that diagonal interpolation is more complex since it includes both horizontal and vertical interpolation. If a deblocking filter is used at the decoder, IO_H, IO_D could incorporate the additional MAC operations for the deblocking filter realization, since both the interpolation and the deblocking filter depend on the values of the decoded motion vectors.

In total, we define the set of operations

$$\mathfrak{G} = \{\mathrm{MC}, \mathrm{IO}_H, \mathrm{IO}_D, \mathrm{FC}, \mathrm{RS}\}. \tag{1}$$

The value of each $g \in \mathfrak{G}$ may be determined at encoding time for each adaptation unit (AU) n (e.g., the nth video frame or the nth macroblock) following experimental or modeling approaches [17]. We define the cardinality of set $\mathfrak{G}$ as $N_{\text{input}}^{\max} = |\mathfrak{G}|$; based on (1), $N_{\text{input}}^{\max} = 5$. In this way, the GCMs[3] $g(n)$ are generated, with n bounded by the total AUs of the input video. Any discrete data entity at the decoder can be selected as an AU, such as a macroblock within a frame, a spatial resolution level in the decoded L and H frames, or an entire L, H, or A frame. Finally, notice that the chosen operations of (1), as well as the subsequent adaptive linear complexity prediction methods in this paper are in no way restricted to a particular coding structure (predictive or MCTF-based) and hence are applicable for a broad class of motion-compensated video decoders.

The GCMs, or model parameters via which the GCMs can be derived [17], are encapsulated in the compressed bitstream. GCM-based complexity prediction at the decoder aims at translating the GCMs into real complexity. In our case, real complexity is the time required for the completion of the AU processing for each module of Fig. 2. The summation of the execution time for all operations of all AUs participating in the reconstruction of a video frame represents the decoding complexity for the particular frame. A resource monitor can be implemented at the receiver that measures the execution time after the completion of each task in an AU, in order for them to be used in updating the linear prediction parameters for the subsequent complexity estimation. In this paper, we follow the generic approach of assuming that the required time for the processing of every AU in every module of Fig. 2 can be measured in real-time. The vast majority of general-purpose processors or DSPs have built-in registers that can accurately monitor the timing based on the processor built-in clock circuitry [25]. This process is very low-complex and provides accurate measurements. Moreover, the instrumentation required in the software implementation is minimal and can be done in a similar fashion for a variety of underlying architectures and software implementations.

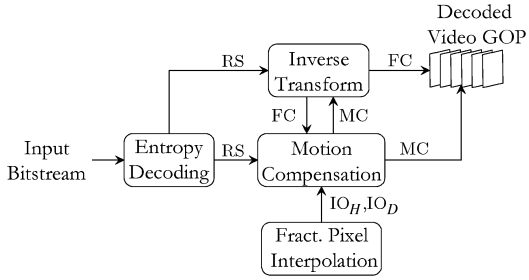

Fig. 2. Basic modules of the proposed GRM for our video-decoding complexity modeling framework. The complexity functions (with the corresponding complexity variables) are indicated in each module; the GCMs of each module are indicated at the connecting arrows.

III. Adaptive Linear Prediction of Video Decoding Complexity

A. Basic Framework

The complexity mapping problem can be formulated as an adaptive linear mapping between the input GCMs and the real complexity (execution time) or between previous and current complexity. The k-step linear mapping problem estimates complexity $c_g(n+k)$, which corresponds to the execution time of operation g for AU $n+k$, using either the current GCM sequence $g(n)$ or previous values of $c_g(n)$. Specifically

$$\widehat{c_g}(n+k) = \sum_{p=0}^{P[g]-1} w_g(p) \cdot u_g(n-p) \tag{2}$$

where

$$\begin{aligned} u_g(n-p) &= c_g(n-p) \\ k &> 0 \end{aligned} \tag{3}$$

for the case of adaptive linear prediction of the complexity of AU $n+k$ from previous complexity measurements, while

$$\begin{aligned} u_g(n-p) &= g(n-p) \\ k &> 0 \end{aligned} \tag{4}$$

[1]Notice that, regardless of the use of convolution or faster algorithms (e.g., the lifting-scheme), the inverse transform always consists of filtering operations (similar rules apply for the case of the discrete cosine transform or other linear transforms) or at least of accumulated bit-shift-and-addition operations (for the case of simplified integer-to-integer transforms). Consequently, nonzero MAC operations are a good metric for a large variety of transform realizations. Notice that GCMs are not expected to capture the precise implementation complexity, which is affected regardless by the implementation architecture specifics, memory accesses, and the software/compiler optimizations. They are rather meant as metrics that represent the content-dependent complexity variation for a generic reference machine.

[2]In this operation we also encapsulate the decoding of the motion vectors, which, in the vast majority of cases, requires negligible complexity in comparison to motion compensation.

[3]Notice that g indicates the different operations shown in (1), while $g(n)$ represents the particular GCM value for AU n.

for the case of adaptive linear mapping of GCMs to the complexity of AU $n+k$. The maximum order of the predictor is indicated by $P[g]$ and the predictor coefficients are

$$\mathbf{w}_g = [w_g(0) \quad w_g(1) \quad \cdots \quad w_g(P[g]-1)]^T. \tag{5}$$

We define the utilized measurements vector or GCM vector as

$$\mathbf{u}_g(n) = [u_g(n) \quad u_g(n-1) \quad \cdots \quad u_g(n-P[g]+1)]^T. \tag{6}$$

We additionally define the prediction error for each AU n as

$$e_g(n) = c_g(n+k) - \widehat{c_g}(n+k). \tag{7}$$

Based on (2), the last equation becomes

$$e_g(n) = c_g(n+k) - \mathbf{w}_g^T \mathbf{u}_g(n). \tag{8}$$

For each operation $g \in \mathfrak{G}$, the optimal linear predictor in the mean-square error sense is minimizing $E\{e_g^2(n)\}$. We can identify the (unique) predictor which achieves this by taking the gradient of (8) for $\mathbf{w}_g$ and setting it to zero, thereby deriving

$$\nabla_{\mathbf{w}_g} E\left\{e_g^2(n)\right\} = 0 \Leftrightarrow -2E\left\{e_g(n)\mathbf{u}_g(n)\right\} = 0 \tag{9}$$

which can also be expressed as

$$\mathbf{R}\{u_g\} \cdot \mathbf{w}_g = \mathbf{r}_g(k) \tag{10}$$

where $\mathbf{R}\{u_g\} = E\{\mathbf{u}_g(n)\mathbf{u}_g(n)^T\}$ us the input's autocorrelation matrix and $\mathbf{r}_g(k) = E\{\mathbf{u}_g(n)u_g(n+k)\}$. If (3) holds, then (10) corresponds to the $P[g]$th-order autoregressive process with an additional delay of $k-1$ samples.

Due to the fact that the optimal solution of (10) requires the knowledge of the autocorrelation of the input, as well as the inversion of $\mathbf{R}\{u_g\}$, we use the popular normalized least-mean square (NLMS) adaptation [26], which periodically updates the coefficients of the predictor based on the normalized gradient of (9) to yield

$$\mathbf{w}_g(n+l[g]) = \mathbf{w}_g(n) + \frac{\mu[g]}{\|\mathbf{u}_g(n)\|^2} 2e_g(n)\mathbf{u}_g(n) \tag{11}$$

where $0 < \mu[g] < 1$ is the predictor step-size, $l[g] > 0$ is the predictor adaptation interval, and

$$\|\mathbf{u}_g(n)\|^2 = \mathbf{u}_g(n)^T \mathbf{u}_g(n) \tag{12}$$

is the square of the vector norm of the input. The value of the scaling (step-size) $\mu[g]/\|\mathbf{u}_g(n)\|^2$ controls the convergence speed. Higher values lead to faster convergence to the optimal solution (which is achieved under wide-sense stationarity) and faster adaptation to input changes; however they also lead to higher predictor-coefficient fluctuations after convergence [26]. If we arbitrarily set

$$\|\mathbf{u}_g(n)\|^2 \equiv 1 \tag{13}$$

then (11) corresponds to the conventional LMS algorithm, which does not require the computation of the square of the norm of the input, but may exhibit worse convergence properties than NLMS. The value of $l[g]$ of (11) is the predictor update period, where larger values lead to lower complexity due to less frequent updates but also to less accuracy in the adaptation.

Although there are several alternatives to this algorithm [26], (2) and (11) consist of a good approach for our problem since LMS combines good adaptation properties, stability of convergence with nonstationary inputs, and low implementation complexity compared with other approaches such as recursive least-squares estimation [26]. Moreover, it was also proven that LMS is H^∞ optimal [27], hence, under proper step-size ($\mu[g]$) selection, the adaptation minimizes the maximum energy gain from the disturbances to the prediction error. This is an important aspect for our study, since, under bounded disturbances in the steady state of the prediction, it guarantees that the energy of the prediction error for the resource estimation of each decoded video frame will be upper bounded, which means that, depending on the LMS parameters, the complexity prediction over a time interval (e.g., one GOP) should be within a certain accuracy range.

B. Extensions of Adaptive Prediction

Apart from the order of the predictor ($P[g]$), the LMS step-size ($\mu[g]$) selection, and the predictor-update period ($l[g]$), other parameters can also influence the prediction efficiency and the complexity of the adaptation. The most important is the selection of the appropriate AU granularity for the input and output of the adaptive prediction. For the complexity prediction of motion compensation and interpolation operations ($c_{\mathrm{MC}}(n)$ and $c_{\mathrm{IO}_H}(n)$, $c_{\mathrm{IO}_D}(n)$), we are experimenting with the following three AUs: 1) a video macroblock; 2) one macroblock row; or 3) an entire video frame. For the inverse transform and entropy decoding ($c_{\mathrm{RS}}(n)$ and $c_{\mathrm{FC}}(n)$), we are experimenting with the following two AUs: 1) one spatial resolution of the discrete wavelet decomposition and 2) an entire decompressed frame. Notice that, in all cases, higher granularity of the utilized measurements should lead to better prediction results, albeit at the expense of additional complexity at the decoder. We also investigate whether increased granularity in the provided GCMs assists the linear adaptation. Since the prediction is done at the decoder side, this comes at a cost of higher communication overhead for the transmission of these metrics. It is important to mention that in our experiments we always predict separately for each temporal decomposition level and each frame type. Therefore, complexity prediction of the L or H frames of each level of the MCTF reconstruction of Fig. 1 is performed separately for each level with a different adaptive linear predictor, leading to $N_{\max,\mathrm{predict}} = T+2$ separate linear predictors for T temporal decomposition levels, where the increase by two is due to the additional separate predictor for the L frames ("intra" frames), as well as the adaptive prediction performed at the decoded output video frames (temporal level zero in Fig. 1). In addition, we always perform the adaptive prediction over an interval of frames (typically one GOP). Therefore, for each temporal level t of Fig. 1 and each operation g, we have $k = \{1, 2, \ldots, 2^{T-\min\{t,T\}}\}$ for autoregressive prediction. On the other hand, for the GCM-based prediction, we have $k = 0$ since the GCMs of each interval of frames (e.g., current GOP) are available from the video bitstream. In both cases, in order to have causality, the utilized coefficients for the realization of the prediction are updated only after the frames of the prediction interval have been decoded and the complexity (execution-time) measurements have been obtained, even though multiple predictor-coefficient updates may occur within a prediction interval. It is also important to mention that we are always measuring the prediction error for

each individual decoded video frame, even though the prediction interval can be larger (e.g., the duration of one GOP).

Finally, we would like to remark that general combinations of autoregressive prediction [under (3) and GCM-based prediction [under (4)] could also be envisaged for different time intervals, or for varying video content characteristics. Alternatively, combinations of autoregressive and GCM-based predictors for the various temporal decomposition levels could be devised. However, we do not investigate these mixed estimations in this work.

C. Generalization of Adaptive Prediction Parameter Set

Let $q = (g,t)$ be a set-element indicating the prediction operation-type (g) and the temporal level (t) of the prediction out of the entire set $\mathfrak{Q} = \{\mathfrak{G}, \{0,\ldots,T+1\}\}$ of possible choices. For each set element $q = (g,t)$, $q \in \mathfrak{Q}$, we represent the AU granularity (sampling) of the input and output of the adaptive prediction by $n_{i[q]}$, with $i[q] = \{1,\ldots,N^{\max}_{\text{sample},q}\}$, i.e., the subscript i of AU n (for operation g at temporal level t) increases with the increase in the AU granularity. In the presented experiments, following the previous description in the beginning of Section III-B, we have $N^{\max}_{\text{sample},q} = 3$ for $q = \{(\text{MC},t),(\text{IO}_H,t),(\text{IO}_D,t)\}$, and $N^{\max}_{\text{sample},q} = 2$ for $q = \{(\text{RS},t),(\text{FC},t)\}$. Notice that, for all cases, n_1 corresponds to the coarsest AU granularity, which is one entire decompressed video frame A^{0,n_1}. Moreover, if the provided granularity for the input GCMs is lower than that of the desired granularity for the predicted complexity, bilinear interpolation or sample-and-hold techniques are used to increase the sampling rate of the input GCMs. The generalized adaptive complexity prediction of AU $n_{i[q]} + k_{i[q]}$ is now formulated as

$$\widehat{c}_q\left(n_{i[q]} + k_{i[q]}\right) = \sum_{p_{i[q]}=0}^{P[q]-1} w_q\left(p_{i[q]}\right) \cdot u_q\left(n_{i[q]} - p_{i[q]}\right). \quad (14)$$

The adaptation of the predictor is performed as

$$\mathbf{w}_q\left(n_{i[q]} + l[q]\right) = \mathbf{w}_q\left(n_{i[q]}\right) + \frac{\mu[q]}{\left\|\mathbf{u}_q\left(n_{i[q]}\right)\right\|^2} 2e_q\left(n_{i[q]}\right)\mathbf{u}_q\left(n_{i[q]}\right). \quad (15)$$

The last two equations represent a very large number of possibilities since the cardinality of the set of choices is[4]

$$\left|\mathfrak{E}\right| = N^{\max}_{\text{input}} \cdot (T+1) \quad (16)$$

and the parameters that can be adjusted are $i[q]$, $l[q]$, and $P[q]$.

For example, if the prediction is performed for an interval of one GOP, for $T = 4$ (typical parameter for MCTF-based coders), and $N^{\max}_{\text{input}} = 5$ [following (1)], we get $\left|\mathfrak{E}\right| = 25$. Then, even for a fixed predictor order $P[q]$, for two possible values for each of the $i[q]$, $l[q]$ parameters, we get over 10^{13} possible combinations of predictor assignments for each GOP of decoded video. Each parameter assignment requires certain computation for the prediction realization and can lead to different expected prediction efficiency. Hence, under a general optimization framework for complexity prediction, all possible assignments should be considered.

[4]In (16), the second term is $T+1$ and not $T+2$ since some operations do not occur for $T = 0$ or $T = T+2$ as it will be explained in Section III-E.

D. Generalization of the Adaptive Prediction Time Interval

Depending on the particular application, the proposed adaptive prediction of video decoding complexity may be applicable for the derivation of the expected complexity for one frame, or for a group of decoded frames. This can be expressed as

$$\widehat{c_{\text{out}}}(n_1) = \sum_{\forall q \in \mathfrak{Q}} \sum_{n^{\text{depend}}_{i[q]} \in \mathfrak{d}_q} \widehat{c}_q\left(n^{\text{depend}}_{i[q]}\right) \quad (17)$$

where $q = (g,t)$ $(q \in \mathfrak{Q})$ was defined previously, $\widehat{c_{\text{out}}}(n_1)$ is the output complexity estimate for AU (decoded video frame)[5] A^{0,n_1}, $\mathfrak{d}_q$ is the set containing the indexes to the AUs, denoted by $n^{\text{depend}}_{i[q]}$, which are utilized for the construction of the output AU n_1 (i.e., the temporal dependencies of AU n_1), and the complexity of each individual AU $n^{\text{depend}}_{i[q]}$, denoted by $\widehat{c}_q(n^{\text{depend}}_{i[q]})$, is defined as in (14). Notice that the cardinality of each set $\mathfrak{d}_q$, denoted by $|\mathfrak{d}_q|$, depends on the AU sampling choice $i[q]$.

As an illustrative example, if we assume that $\forall q \in \mathfrak{Q}$, we select $i[q] = 1$ for the instantiation of the MCTF decoding depicted in Fig. 1(b), for the complexity for the reconstruction of frame $A^{0,0}$, i.e., $\widehat{c_{\text{out}}}(0)$, we have $\mathfrak{d}_{(g,0)} = \{1\}$ and $\mathfrak{d}_{(g,\{1,2,3\})} = \{0\}$ for $g = \{\text{MC},\text{IO}_H,\text{IO}_D\}$ and $\mathfrak{d}_{(g,\{1,2,3,4\})} = \{0\}$ for $g = \{\text{RS},\text{FC}\}$. In addition, for the complexity of the reconstruction of the entire GOP, $\widehat{c_{\text{out}}}(\mathbf{n_1})$ with $\mathbf{n_1} = \{0,\ldots,7\}$, we have $\mathfrak{d}_{(g,1)} = \{0,1,2,3\}$, $\mathfrak{d}_{(g,2)} = \{0,1\}$, and $\mathfrak{d}_{(g,3)} = \{0\}$ for $g = \{\text{MC},\text{IO}_H,\text{IO}_D\}$ and $\mathfrak{d}_{(g,4)} = \{0\}$ for $g = \{\text{RS},\text{FC}\}$.

It is important to emphasize that, $\forall q \in \mathfrak{Q}$, the complexity $\widehat{c}_q(n^{\text{depend}}_{i[q]})$ in (17) has a different granularity expressed by $i[q]$. As a result, depending on the choices for $i[q]$ in (14), the output of (17) changes and the cost (in terms of implementation or communication overhead) for deriving this complexity estimate varies. The analytical formulation of this cost is the topic of the following subsection.

E. Complexity and Information Requirements for Adaptive Prediction

Following the generalized adaptive prediction of (14) and (17), for every decoded video frame A^{0,n_1} we need the following average number of operations (MACs):

$$\text{O}_{\text{compute}}(n_1) = \sum_{\forall q \in \mathfrak{Q}} \sum_{n^{\text{depend}}_{i[q]} \in \mathfrak{d}_q} \left(P[q] + \text{O}_{\text{coeff}}[q] \cdot \frac{P[q]}{l[q]}\right) \quad (18)$$

where $\mathfrak{d}_q$ is defined previously for (17) and

$$\text{O}_{\text{coeff}}[q] = \begin{cases} 2, & \text{if NLMS is used} \\ 1, & \text{if LMS is used} \end{cases} \quad (19)$$

where the case of NLMS corresponds to (12), while LMS is the case where the simplification of (13) is used. In (18), we account for $P[q]$ MAC operations for (14) and $P[q]$ operations for the predictor update every $l[q]$ samples. In addition, we account for $P[q]$ operations for the term $\|\mathbf{u}_q(n_{i[q]})\|^2$ in (15) for the case of NLMS. Notice that $\mu[q]$ is predefined and fixed for a particular predictor (as discussed in the experimental results section of the

[5]Notice that in $\widehat{c_{\text{out}}}$ the subscript of the AU n is always one since in this paper we are interested in predicting the complexity at the coarsest AU granularity, which is at the video frame granularity.

paper). Hence, the division of $\mu[q]$ by $||\mathbf{u}_q(n_{i[q]})||^2$ in (15) is assumed to be performed by rounding and a lookup table (LUT) and is not included in the complexity estimation of (18), since the comparison of division and multiplication operations is machine dependent. However, even under the inclusion of the division operation, the predictor complexity results are simply offset by the machine-specific division overhead.

Notice that the outer summation of (18) operates $T+2$ times for each operation g. However, depending on the operation, $t=0$ or $t=T+1$ may entail no complexity. This is because we have $\mathfrak{d}_{(\mathrm{MC},T+1)} \triangleq \varnothing$, $\mathfrak{d}_{(\mathrm{IO}_H,T+1)} \triangleq \varnothing$, and $\mathfrak{d}_{(\mathrm{IO}_D,T+1)} \triangleq \varnothing$, i.e., there are no motion-compensation or interpolation operations for the construction of the $L^{T+1,0}$ frame, and $\mathfrak{d}_{(\mathrm{RS},0)} \triangleq \varnothing$, $\mathfrak{d}_{(\mathrm{FC},0)} \triangleq \varnothing$ due to the fact that the output video frames A^{0,n_1} do not require entropy-decoding operations or inverse spatial transform operations.

For complexity prediction with GCMs at the decoder side, the server must transmit

$$\mathrm{I}_{\mathrm{GCM}}(n_1) = \sum_{\forall q \in \mathfrak{Q}} \frac{|\mathfrak{d}_q|}{r_q} \tag{20}$$

GCMs, where r_q is the ratio of complexity measurements to GCMs for each operation g of each temporal level t. Typical values are $r_q = 1$, or we may set $r_{(\mathrm{MC},t)}$, $r_{(\mathrm{IO}_H,t)}$, and $r_{(\mathrm{IO}_D,t)}$ equal to the number of macroblocks within a video frame and $r_{(\mathrm{RS},t)}$, $r_{(\mathrm{FC},t)}$ equal to the number of spatial resolutions of a video frame and bilinearly interpolate (or sample-and-hold) in order to produce the required GCM granularity at the receiver side. Although the calculations of this subsection were presented under the assumption of a total of T temporal reconstructions, low frame-rate decoding may be accommodated by excluding the calculations for $t = 0, \ldots, t_{\min} - 1$, where $t_{\min}$ is the lowest reconstructed temporal level. Low-resolution decoding (for scalable coders) can be accommodated in a similar manner.

Finally, concerning the expected prediction error, similar to (7), we can formulate the error of the generalized complexity prediction for decoded frame A^{0,n_1} as

$$e_{\mathrm{out}}(n_1) = \sum_{\forall q \in \mathfrak{Q}} \sum_{n_{i[q]}^{\mathrm{depend}} \in \mathfrak{d}_q} \left| c_q\left(n_{i[q]}^{\mathrm{depend}}\right) - \widehat{c_q}\left(n_{i[q]}^{\mathrm{depend}}\right) \right| \tag{21}$$

which becomes

$$e_{\mathrm{out}}(n_1) = \sum_{\forall q \in \mathfrak{Q}} \sum_{n_{i[q]}^{\mathrm{depend}} \in_q \mathfrak{d}} \left| c_q\left(n_{i[q]}^{\mathrm{depend}}\right) - \sum_{p_{i[q]}=0}^{P[q]-1} w_q\left(p_{i[q]}\right) \cdot u_q\left(n_{i[q]}^{\mathrm{depend}} - p_{i[q]} - k_{i[q]}\right) \right|. \tag{22}$$

The last equation indicates that the contribution of each operation g of each temporal level t to the output prediction error $e_{\mathrm{out}}(n_1)$ may differ. In fact, the intermediate prediction errors of (22), along with the required complexity or information overhead for the realization of the prediction [(18)–(20)] may be used as the driving force of a low-complexity optimization mechanism for the adaptive selection of parameters of the adaptive complexity prediction formulated by (17). This is the topic of the following subsection.

F. Resource-Constrained Adaptive Linear Complexity Prediction

Based on the previous definition of the adaptive prediction, for each $q = (g,t)$, we can modify the AU granularity ($i[q]$), the prediction update period ($l[q]$), the predictor order ($P[q]$) and the predictor step-size ($\mu[q]$). In this paper, the optimal predictor order $P[q]$ and step-size $\mu[q]$ were selected based on exhaustive experimentation with a large set of possible values, as it will be explained in the experimental section of the paper. Consequently, we are focusing on the $i[q]$ and $l[q]$ parameters here. In principle, we would like to select the parameters that would minimize the output prediction error $e_{\mathrm{out}}(n_1)$ given in (22). However, this may entail high computational complexity or high information requirements (under the calculations of (18) and (20), respectively). Moreover, under the flexible VCV model [19] adopted in this study, the complexity (timing) bound for each group of output frames (AUs) may be satisfied without the best-possible prediction accuracy and therefore the resource predictor may be realized with less computation. After making this key observation, we would like to obtain the best possible prediction accuracy under a certain predictor-complexity bound

$$\begin{aligned} &(i^*[q], l^*[q])_{\forall q \in \mathfrak{Q}} = \arg\min \widehat{e_{\mathrm{out}}}(\mathbf{n}_1) \\ &\text{s.t. } \mathrm{O}_{\mathrm{compute}}(\mathbf{n}_1) \le \mathrm{O}_{\mathrm{bound}}(\mathbf{n}_1) \end{aligned} \tag{23}$$

where the optimal solution $(i^*[q], l^*[q])$ is a point in the $|\mathfrak{Q}|$-dimensional hyper-set of possible parameters $\mathfrak{B}^{|\mathfrak{Q}|}$, which is bounded by

$$q \in \mathfrak{Q},\ 1 \le i[q] \le N_{\mathrm{sample},q}^{\max},\ 1 \le l[q] \le |\mathfrak{d}_q|. \tag{24}$$

In the last equation, we selected $|\mathfrak{d}_q|$ as the maximum predictor update period ($l[q]$) in order to ensure that the predictor is updated at least once during each group of AUs $|\mathfrak{d}_q|$.

Notice that (23) represents the most general form of the problem that, under a given predictor complexity $\mathrm{O}_{\mathrm{compute}}(\mathbf{n}_1)$, simultaneously adapts all of the available prediction parameters in order to minimize the expected prediction error over a finite interval of future decoded video frames $\mathbf{n}_1$. The problem can be similarly defined for the minimization of the information requirements $\mathrm{I}_{\mathrm{GCM}}$

$$(i^*[q])_{\forall q \in \mathfrak{Q}} = \arg\min \widehat{e_{\mathrm{out}}}(\mathbf{n}_1) \text{ s.t. } \mathrm{I}_{\mathrm{GCM}}(\mathbf{n}_1) \le \mathrm{I}_{\mathrm{bound}}(\mathbf{n}_1). \tag{25}$$

The optimization of (25) is simpler than that of (23) as, from the definition of $\mathrm{I}_{\mathrm{GCM}}$ of (20), it only involves the selection of the GCM granularity $i[q]$. As a result, we shall focus on (23) in the remainder of this paper, since simplified solutions to (23) can be applied for the case of (25) as well.

In order to solve (23), we follow an offline estimation and pruning approach and then during the online estimation we select the appropriate prediction parameters based on the offline estimates. In this way, the offline optimization part can entail high complexity while the complexity of the online estimation is only stemming from the adaptive filtering.

We separate the prediction error into smaller parts

$$\widehat{e_{\mathrm{out}}}(\mathbf{n}_1) = \sum_{\forall q \in \mathfrak{Q}} \widehat{e_{\mathrm{out}}}(\mathfrak{d}_q) \tag{26}$$

where each intermediate error $\widehat{e_{\text{out}}}(\mathfrak{d}_q)$ is defined based on the correspondence of (26) with (21). As seen by (26), each of these intermediate errors is equally important in the output error metric. Moreover, for each choice of $\{i[q], l[q]\}_{\forall q \in \mathfrak{Q}}$, based on (18), we can analytically establish the resulting computational impact in the adaptive complexity prediction realization. It is straightforward to observe from (18) that the required computation monotonically increases with increasing $i[q]$ and with decreasing $l[q]$. Under the assumption[6] that the expected error can be reliably estimated offline for the possible choices of $i[q]$ and $l[q]$, the optimization of (23) can be performed with dynamic programming techniques, as explained in the following.

Each point $(i[q], l[q]) \in \mathfrak{B}^{|\mathfrak{Q}|}$ (where $\mathfrak{B}^{|\mathfrak{Q}|}$ is the initial parameter hyper-set) is associated with its corresponding computation/prediction-error (C-P) point $(\mathrm{O}_{\text{compute}}, e)_{(i[q],l[q])}$. Each $(i[q], l[q])$ is selected within the bounds set in (24). Assuming that all possible alternatives are to be utilized in our complexity prediction system, for each $q \in \mathfrak{Q}$, we initially obtain $N^{\max}_{\text{sample},q} \cdot |\mathfrak{d}_q|$ C-P points, which form the initial set of valid C-P points, denoted by $\mathfrak{B}_q$. During a pruning stage, for $q \in \mathfrak{Q}$ each we remove from $\mathfrak{B}_q$ all C-P points for which there exists another C-P point in $\mathfrak{B}_q$ with lower or equal computational requirements and lower prediction error.

We then group together all admitted points in sets $|\mathfrak{B}_q|_{\forall q \in \mathfrak{Q}}$ into the sets $\mathfrak{B}_{(\mathfrak{G},t)}$, by summing up the admitted points of $\mathfrak{B}_q$ that correspond to the same temporal level, i.e.,

$$(\mathrm{O}_{\text{compute}}, e)_{(i[\mathfrak{G},t], l[\mathfrak{G},t])} = \sum_{\forall g \in \mathfrak{G}} (\mathrm{O}_{\text{compute}}, e)_{(i[g,t], l[g,t])}. \tag{27}$$

The pruning stage is repeated for all C-P points of $\mathfrak{B}_{(\mathfrak{G},t)}$. Finally, in the third stage, we group all admitted points of $\mathfrak{B}_{(\mathfrak{G},t)}$ for all levels

$$(\mathrm{O}_{\text{compute}}(\mathbf{n}_1, \widehat{e_{\text{out}}}(n_1))_{(i[\mathfrak{Q}], l[\mathfrak{Q}])} = \sum_{t=0}^{T+1} (\mathrm{O}_{\text{compute}}, e)_{(i[\mathfrak{G},t], l[\mathfrak{G},t])} \tag{28}$$

where the C-P points $(\mathrm{O}_{\text{compute}}(\mathbf{n}_1, \widehat{e_{\text{out}}}(\mathbf{n}_1))_{(i|\mathfrak{Q}|, l|\mathfrak{Q}|)}$ are pruned as before in order to form the set of admissible points $\mathfrak{B}_{\mathfrak{Q}}$. Every C-P point that is finally admitted to $\mathfrak{B}_{\mathfrak{Q}}$ forms a potential (optimal) solution, depending on the bound set on the predictor complexity by (23). Notice that the grouping and pruning process described previously can be performed offline if estimates for all C-P points are available. In addition, all the admitted points can be offline sorted in ascending order with respect to the expected prediction error (and, correspondingly, in descending order with respect to the predictor complexity), thereby creating the sorted list of admissible points $\mathfrak{B}^{\text{sorted}}_{\mathfrak{Q}}$. As a result, the online optimization simply amounts to selecting the C-P point in $\mathfrak{B}^{\text{sorted}}_{\mathfrak{Q}}$ with the complexity that is closest to $\mathrm{O}_{\text{bound}}(\mathbf{n}_1)$. This ensures negligible complexity overhead for the optimization of (23).

G. Offline Estimation of the Expected Prediction Error Under Predictor-Parameter Adaptation

Having a reliable offline estimation of the expected error for a particular set of predictor parameters is a crucial aspect of the optimization framework of the previous subsection, because, under wrong error estimates, a suboptimal point of the parameter hyper-set $\mathfrak{B}^{|\mathfrak{Q}|}$ may be selected. Our expected error estimation is performed offline based on a collection of training data and the bootstrap principle [28]. Since the statistics of the prediction errors are dependent on the prediction parameters and the utilized video coding scheme, conventional statistical modeling approaches can be very complicated and application-specific. The bootstrap approach calculates confidence intervals for parameters where such standard methods cannot be applied [28]. It is essentially a computerized method that replaces statistical analysis with computation, as it will be explained in the following.

We collect offline experimental measurements using seven typical video sequences decoded at 13 equidistant average bit rates between the values of interest (200–1500 kbps). We also fix our experimentation in the prediction of the decoding time of each frame of an entire GOP. Our test case uses $T = 4$ temporal levels of an MCTF-based coder, a separate adaptive predictor per temporal level and per operation type [set $\mathfrak{G}$ of (1)], as well as a separate predictor for the L frame of each GOP and for the decoded video frames (temporal level zero). We applied the AU sampling granularities described in introduction of Section II-B and, for each sampling granularity, three different predictor update periods were tested, i.e., $l[q] = \{1, 2, 4\}$ samples. The entire parameter space explored is presented in Fig. 3. Notice that, depending on the chosen application, a different (even larger) parameter space could be used with the same methodology for offline prediction error estimation. Under the selected parameter space of Fig. 3, we decoded the test video material at the bit rates of interest and generated offline decoding time measurements for each operation g, as well as their prediction with all possible predictor configurations. For each temporal level and each $g \in \mathfrak{G}$, $\widehat{e_{\text{out}}}(\mathfrak{d}_q)$ was estimated as follows.

Each measurement set $\mathfrak{M}_{\{q,(i[g],l[g])\}}$ consists of m pairs of measurements of timing complexity and predicted values under the particular choice $(i[q], l[q])$, indicated by c_m and $\widehat{c}_m$, respectively. In total, for each $\{q, (i[q], l[q])\}$, we have $1 \leq m \leq N_{\text{measure}}$ pairs of measurements, with $N_{\text{measure}} = (13 \cdot 7 \cdot F)/2^{\min\{t,T\}}$ under the chosen settings (seven sequences, each consisting of $F = 256$ total video frames, decoded at 13 different bit rates). We calculate the mean absolute prediction error as

$$e^0_{(i[q],l[q])} = \frac{1}{N_{\text{measure}}} \sum_{m=1}^{N_{\text{measure}}} |c_m - \widehat{c}_m|. \tag{29}$$

Subsequently, based on a pseudorandom number generator, we resample $\mathfrak{M}_{\{q,(i[g],l[g])\}}$ with replacement and produce a resampled set $\mathfrak{M}^1_{\{q,(i[g],l[g])\}}$ containing N_{measure} pairs of points. The new mean prediction error, denoted as $e^1_{(i[q],l[q])}$, is calculated as in (29), with $(c_m, c_m) \in \mathfrak{M}^1_{\{q,(i[g],l[g])\}}$. This process is repeated for a total of $N_{\text{bootstrap}}$ iterations, thereby generating the set

$$\mathfrak{B}_{(i[q],l[q])} = \left\{ e^1_{(i[q],l[q])}, \ldots, e^{N_{\text{bootstrap}}}_{(i[q],l[q])} \right\} \tag{30}$$

offline generated measurements of the mean prediction error for the particular parameter choice $(i[q], l[q])$. Notice that several pairs may be duplicated during the resampling process,

[6]This will be analyzed in Section III-G.

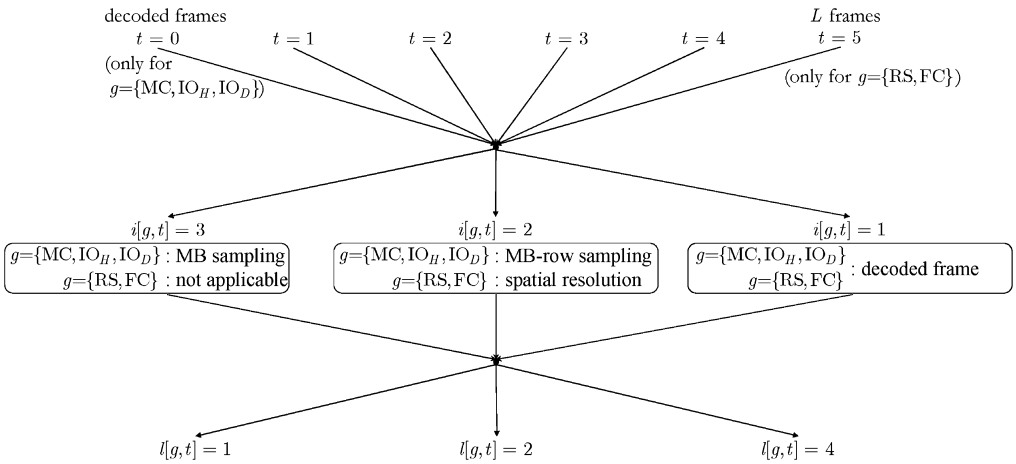

Fig. 3. Explored parameter space.

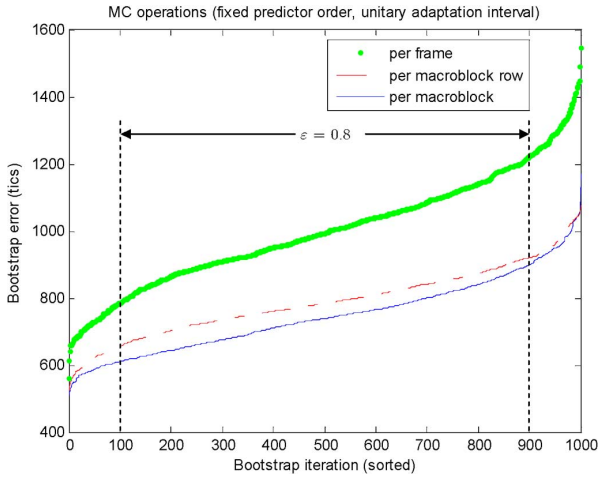

Fig. 4. Example of the derived offline bootstrap estimates for the MC operations at temporal level $t = 2$ (under a given predictor order and predictor adaptation interval of one sample, with $N_{\mathrm{bootstrap}} = 1000$). An example of a confidence interval is indicated as well.

thereby creating a different bias for the resampled set of measurements, which is a key aspect for the utilized bootstrap parameter estimation technique. These measurements are sorted in increasing order $\widehat{e}^{1}_{(i[g],l[g])} \leq \widehat{e}^{2}_{(i[g],l[g])} \leq \cdots \leq \widehat{e}^{N_{\mathrm{bootstrap}}}_{(i[g],l[g])}$, where $\widehat{e}^{m}_{(i[g],l[g])}$ is the mth smallest element of $\mathfrak{B}_{(i[g],l[g])}$. We define the sorted measurement set as $\widehat{\mathfrak{B}}_{(i[g],l[g])}$.

If we desire $\varepsilon \cdot 100$ bootstrap confidence interval, then the expected mean prediction error is bounded by $(\widehat{e}^{\alpha}_{(i[g],l[g])}, \widehat{e}^{\beta}_{(i[g],l[g])})$, with the indexes α, β defined as

$$\alpha = \lfloor 0.5 N_{\mathrm{bootstrap}}(1-\varepsilon) \rfloor \tag{31}$$

$$\beta = N_{\mathrm{bootstrap}} - \alpha + 1. \tag{32}$$

Fig. 4 shows an example of the (sorted) bootstrap-estimated error for the MC operations of temporal level one under various granularities for the prediction. As shown there, we performed $N_{\mathrm{bootstrap}} = 1000$ iterations for each case. In our experiments, we adapt the confidence interval dynamically based on the previously observed errors. In this way, although the error estimation is performed offline, our "confidence" on the estimate is adapted online by matching the offline (bootstrap) estimate with the observed error estimate of the previously predicted frames. Since different confidence intervals have to be considered and this may alter the points excluded from the pruning process of the previous subsection, we selected ten values for the confidence interval, defined by $\varepsilon = \{0.1, 0.2, \ldots, 1.0\}$, and performed the pruning process under each one. Once the derived indexes α, β are derived for each interval ε based on (31) and (32), we define the bootstrap mean error for this interval (for each operation g at each temporal level t and each predictor parameter choice $\{i[g,t], l[g,t]\}$) as

$$\widehat{e_{\mathrm{out}}}(\mathfrak{d}_q) = \frac{1}{\alpha - \beta} \sum_{i=\alpha}^{\beta} \hat{e}^{i}_{(i[q],l[q])}. \tag{33}$$

In total, during the online complexity prediction process, the optimal predictor is selected based on (23), with the expected error $\widehat{e_{\mathrm{out}}}(\mathbf{n}_1)$ found from the bootstrap estimates under the appropriate confidence interval for every q, denoted by $\varepsilon[q]$. After the video frames are decoded and the complexity prediction error is calculated based on the experimental measurements, the confidence interval is readjusted according to the mismatch between $\widehat{e_{\mathrm{out}}}(\mathfrak{d}_{(g,t)})$ and the measured error $e_{\mathrm{out}}(\mathfrak{d}_q)$. Although we used the first sample moment for our optimization process [mean estimate of (33)], higher sample moments such as the expected error variance could be used in the proposed framework.

Finally, it is important to emphasize that, although we selected the uniform distribution for the resampling of the bootstrap process (via the use of a pseudorandom number generator), if explicit knowledge is given for the probability distribution function (pdf) of the errors of a particular parameter choice $(i[q], l[q])$, this pdf can be used for the resampling process in order to increase the bootstrapping estimation performance.

IV. Experimental Evaluation

A. GCM-Based Prediction Versus Autoregressive Prediction

We first indicate the achieved prediction performance and average complexity requirements for the predictor (in terms of MAC operations per video frame) for different configurations. The results are seen in Table I for the GCM-based adaptive prediction and in Table II for the autoregressive adaptive prediction. The numbers in parentheses indicate the required number of

TABLE I
ERROR-OVER-MEASUREMENT POWER RATIO (IN PERCENTAGE) AND COMPUTATIONAL REQUIREMENTS PER FRAME (IN NUMBER OF MACS—IN PARENTHESIS) OF GCM-BASED PREDICTION FOR ALL THE DIFFERENT OPERATIONS. SEVEN CIF VIDEO SEQUENCES DECODED AT 512 kbps WERE USED

Operation Type (g)	MC			FC	
Granularity ($i[q]$)	MB	MB-row	Frame	Spatial Resolution	Frame
Adaptation Interval ($l[q]$)	(i[MC,t]=3)	(i[MC,t]=2)	(i[MC,t]=1)	(i[FC,t]=2)	(i[FC,t]=1)
1	4.07% (188)	4.58% (38)	5.82% (10)	8.56% (15)	9.19% (15)
2	3.47% (169)	3.90% (34)	5.40% (9)	9.17% (14)	10.22% (14)
4	4.45% (160)	6.47% (32)	6.16% (8)	10.22% (13)	14.98% (13)
Operation Type (g)	IO = IO_H + IO_D			RS	
Granularity ($i[q]$)	MB	MB-row	Frame	Spatial Resolution	Frame
Adaptation Interval ($l[q]$)	(i[IO,t]=3)	(i[IO,t]=2)	(i[IO,t]=1)	(i[RS,t]=2)	(i[RS,t]=1)
1	3.70% (188)	4.69% (38)	5.71% (10)	8.86% (15)	8.04% (15)
2	3.40% (169)	4.52% (34)	6.40% (9)	6.77% (14)	9.41% (14)
4	5.56% (160)	13.51% (32)	6.50% (8)	10.84% (13)	8.25% (13)

Average EM power ratio and Total Predictor Computation

Granularity ($i[q]$)	High	Medium	Low
Adaptation Interval ($l[q]$)	($i[q]$ = 3)	($i[q]$ = 2)	($i[q]$ = 1)
1	6.29% (406)	6.67% (106)	7.19% (50)
2	5.70% (366)	6.09% (96)	7.86% (46)
4	7.77% (346)	10.26% (90)	8.97% (42)

TABLE II
ERROR-OVER-MEASUREMENT POWER RATIO (IN PERCENTAGE) AND COMPUTATIONAL REQUIREMENTS PER FRAME (IN NUMBER OF MACS—IN PARENTHESIS) OF AUTOREGRESSIVE PREDICTION FOR ALL THE OPERATIONS. SEVEN CIF VIDEO SEQUENCES DECODED AT 512 kbps WERE USED

Operation Type (g)	MC			FC	
Granularity ($i[q]$)	MB	MB-row	Frame	Spatial Resolution	Frame
Adaptation Interval ($l[q]$)	(i[MC,t]=3)	(i[MC,t]=2)	(i[MC,t]=1)	(i[FC,t]=2)	(i[FC,t]=1)
1	12.67%(1613)	11.06% (83)	8.43% (10)	10.22% (129)	13.79% (43)
2	10.51%(1238)	12.25% (68)	8.15% (9)	13.29% (99)	12.98% (33)
4	10.37%(1050)	19.21% (60)	8.34% (8)	9.73% (84)	22.38% (28)
Operation Type (g)	IO = IO_H + IO_D			RS	
Granularity ($i[q]$)	MB	MB-row	Frame	Spatial Resolution	Frame
Adaptation Interval ($l[q]$)	(i[IO,t]=3)	(i[IO,t]=2)	(i[IO,t]=1)	(i[RS,t]=2)	(i[RS,t]=1)
1	13.53%(1613)	13.53% (83)	10.98% (10)	15.82% (129)	11.64% (43)
2	13.35%(1238)	14.81% (68)	11.33% (9)	12.56% (99)	11.35% (33)
4	13.23%(1050)	18.75% (60)	11.90% (8)	14.25% (84)	10.36% (28)

Average EM power ratio and Total Predictor Computation

Granularity ($i[q]$)	High	Medium	Low
Adaptation Interval ($l[q]$)	($i[q]$ = 3)	($i[q]$ = 2)	($i[q]$ = 1)
1	17.01%(3484)	12.66%(424)	11.21%(106)
2	12.43%(2674)	13.23%(334)	10.95% (75)
4	11.90%(2268)	15.49%(288)	13.25% (72)

MAC operations for the realization of the complexity predictor, calculated based on (18) for each predictor granularity and adaptation interval. We used seven MPEG CIF sequences ("City," "Football," "Raven," "Sailormen," "Tempete," "Mobile," and "Foreman" of 300 frames each) with a replay rate of 30 Hz, decompressed at an average of 512 kbps (variable bit rate) with the advanced MCTF-based video decoder [24] used in our previous experiments [18], [24], which combines multiframe advanced motion-compensated prediction with variable block sizes and context-based entropy coding. Our choice of the particular decoder stems from three factors: 1) complexity prediction within an MCTF-based framework is more challenging than in predictive coding due to the more complex interactions required for the decoding of each GOP; 2) the decoder of [24] performs advanced multiframe multihypothesis motion compensation with variable block sizes and in this way it resembles state-of-the-art video coding systems such as the MPEG AVC/ITU H.264; and 3) by using the decoder of [24], this paper is directly linked to our previous related work [17].

Notice that, since the GCM framework of Fig. 2 is applicable to essentially all transform-based motion-compensated video coding schemes, the proposed framework is only tied to a particular decoder in terms of the GCM granularity and the number of different frame types chosen. Both of these features must be selected based on the decoder features. For example, for a block-based transform system with predictive coding using I, B, and P frames (i.e., an MPEG-1/2/4 coder or AVC/H.264), the high granularity for the FC operations would be at the macroblock level instead of the spatial resolution chosen for the discrete wavelet transform. In addition, we would set $T = 2$ as shown in the example of Fig. 1(a). Hence, customization to any decoder can be performed by appropriately selecting the GCM granularity and the number of different complexity predictors based on the decoder characteristics, and the proposed complexity prediction methodology follows.

The chosen decoding algorithm [24] for the validation experiments of this paper was implemented in C++, compiled with Microsoft Visual Studio 6.0 in release mode, and executed in the Microsoft Windows XP environment in a Pentium-IV system. With all the advanced coding options enabled (e.g., long temporal filters, multihypothesis motion-compensated prediction, and update), real-time decoding was not possible without additional software optimization. Hence, similar to prior work [3], [19], we resorted to simulation-based results. In Tables I and II, we present the average error-over-measurement (EM) power ratio for each of the generic operations presented in Section V-B as well as the computational complexity of each predictor in terms of average MAC operations per decoded video frame.

All measurements of prediction errors are obtained per decoded video frame. The prediction estimates the complexity of the following GOP based on the GCMs of the GOP (GCM-based prediction) or the previous GOP's complexity (autoregressive prediction) as seen in (14). In particular, for every temporal level t in (14) and (15), we have $u_{(g,t)}(n_{i[g,t]} - p_{i[g,t]}) = g(n_{i[g,t]} - p_{i[g,t]})$ and $k_{i[g,t]} = 0, l_{i[g,t]} = 1$ for the GCM-based prediction, and $u_{(g,t)}(n_{i[g,t]} - p_{i[g,t]}) = c_g(n_{i[g,t]} - p_{i[g,t]})$ and $k_{i[g,t]} = 1$, $l_{i[g,t]} = 1$ for the autoregressive prediction. During the GOP decoding, in both cases the predictor is adapted following (15), and, at the end of the GOP, the predictor to be used for the following GOP is updated. Obviously, more frequent updates of the predictor lead to better prediction accuracy, but the number of future frames whose complexity can be predicted is decreased.

Different sampling granularities are considered in the experiments reported in Tables I and II, as explained in Section IV-B. In addition, we considered different predictor adaptation intervals. The reported experiments used the best predictor order not greater than 64. The best predictor order for each case was found under exhaustive offline experimentation with the video sequences following a variation of the Akaike information criterion [26], which jointly minimizes the variance of the prediction error and the order of the predictor. During each step of this exhaustive testing, the step size was varied from $\mu = 0.005$ to $\mu = 0.8$ in order to jointly derive the best order and step-size. For all of our experiments, we focused on the NLMS case since it was found to provide significantly higher stability and faster convergence properties. It is interesting to remark that, in the majority of cases, GCM-based prediction required small predictor orders; even $P[q] = 1$ worked sufficiently well for the majority of q. This indicates that there is a strong statistical correlation between the utilized GCMs and the measured complexity. On the other hand, in the autoregressive prediction case, higher predictor orders gave better results.

For each case, the total prediction error and complexity reported at the bottom of Tables I and II are calculated by adding the intermediate results of all four generic metrics for the corresponding predictor settings. Notice, however, that, in our optimization framework, we consider all possible cross combinations of predictors; therefore, for the same computational complexity for the adaptive prediction, the optimized resource prediction is expected to achieve better performance than what is reported in Tables I and II.

The results indicate that GCM-based prediction outperforms autoregressive prediction by a significant margin of EM power ratio. Although higher granularity and a small adaptation interval tend to improve the complexity-prediction accuracy, this is not always the case, especially in the autoregressive prediction case. For example, for the GCM-based prediction of MC operations, the best result is obtained under an adaptation interval of 2 measurements at the macroblock granularity. This means that other combinations with higher complexity and higher expected error should be excluded. This is automatically performed during the offline pruning process of the proposed optimization framework, as described in the previous section. Consequently, it can be said that the proposed offline optimization framework automatically provides the subset of predictor configurations that always provide lower (expected) error for higher complexity without incurring any additional online computation penalty.

Examples of GCM-based and autoregressive complexity prediction across time are given in Fig. 5, where we report the prediction estimates versus the time measured in units ("tics") of the internal processor counter [25]. As shown in the figure, the autoregressive-based prediction tends to "overshoot" or "undershoot" during the complexity estimation in comparison to the GCM-based prediction that appears to follow the experimental results more accurately. An additional aspect of the GCM-based approach is that the prediction with a higher granularity (i.e., at the macroblock or spatial-decomposition level) tends to follow the actual measurements with significantly higher accuracy than that at the decoded frame granularity even though the two methods may happen to be comparable when the prediction is averaged over a time interval (e.g., corresponding to a GOP). This is not necessarily true for autoregressive complexity prediction as shown by the relevant experiments in Fig. 5. However, notice that GCM-based prediction also requires communication overhead for the transmission of the GCM values or the modeling parameters via which GCMs may be derived [17].

The communication overhead is proportional to the granularity and the adaptation interval at which complexity is predicted [see (20)]. Hence, the proposed optimization framework that operates with bounded predictor complexity implicitly bounds the communication overhead for the GCM transmission to the receiver as well.

B. Resource-Constrained Complexity Prediction

The results of Tables I and II demonstrate that GCM-based prediction is significantly better than autoregressive prediction and requires on average less complexity for the predictor realization, we focus on this category for the remainder of this section. For the optimization framework of Section V, the process is performed as follows.

1) *Algorithm Summary*
   - We apply the offline bootstrapping process of Section III-G to the seven video sequences used previously for the estimation of the expected error for each predictor configuration under the ten confidence intervals mentioned in Section III-G.
   - The offline pruning process of Section III-F is subsequently applied for each confidence interval. All of the admitted predictor configurations from the explored parameter space of Fig. 3 are then sorted (offline) in decreasing complexity (and hence increased estimation error). In this way, for each temporal level, we obtain estimations of predictor complexity and expected error such as the ones seen in Fig. 6. The results of the figure were generated with two confidence intervals for the bootstrap process. Similar results were obtained with other confidence intervals and for the remaining temporal levels.
   - During the online complexity estimation, we use (14) and (15), and, for every temporal level t, we have $u_{(g,t)}(n_{i[g,t]} - p_{i[g,t]}) = g(n_{i[g,t]} - p_{i[g,t]})$ and $k_{i[g,t]} = 0$ for the GCM-based prediction. The complexity predictor parameters $i[q]$ and $l[q]$ are obtained by the given predictor complexity bound $\mathrm{O}_{\mathrm{bound}}(\mathbf{n}_1)$ and the current confidence parameter $\varepsilon[q]$ based on the

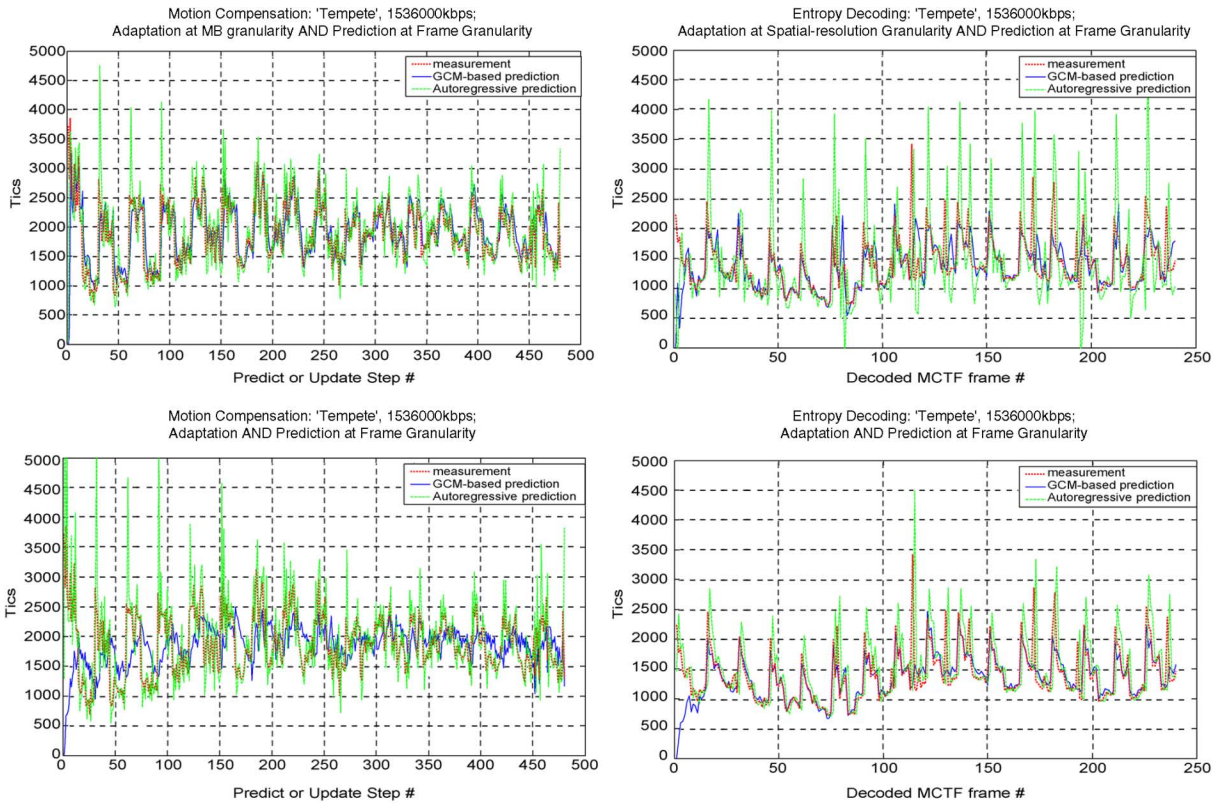

Fig. 5. Examples of GCM-based and autoregressive-based execution time prediction. The time required for the motion-compensation and entropy=decoding operations for the reconstruction of all MCTF frames of each GOP of one video sequence are presented. Top row: predictor adaptation with macroblock or spatial-decomposition level granularity. Bottom row: predictor adaptation with frame granularity. "Tics" indicate the value measured by the internal processor counter and the prediction error is always measured in a frame-by-frame basis.

offline sorted list of admitted predictor configurations from the previous step.

V. Experimental Results

Since only the last step from the above is performed online, the online complexity prediction is guaranteed to have complexity equal to $\mathrm{O}_{\mathrm{bound}}(\mathbf{n}_1)$. Table III presents the overall complexity-prediction error for different predictor complexity bounds based on the optimization of (25) for four sequences ("Coastguard," "Paris," "News," and "Stefan") not belonging to the training set for the bootstrap estimation process. Similar to our previous experiments, the prediction error was measured for each decoded frame and a prediction interval of one GOP was selected. We begin the prediction with confidence interval corresponding to $\varepsilon[q] = 1.0\ \forall q \in \mathfrak{Q}$, and, for each consecutive GOP, we adapt the confidence parameter $\varepsilon[q]$ (parameter for each operation and each temporal level) to the one which produces the closest prediction to the observed errors. During our experimentation we observed that, depending on the sequence characteristics and the mismatch between the offline estimated error and the actual prediction error, frequent changes in the predictor configuration may occur. This is not desirable since additional error is caused until the new predictor converges to the (quasi-)steady state. Hence, to indicate the penalization cause by the new predictor convergence time, we included a bias for the error estimate of the current predictor configuration by scaling it down by a certain percentage. The best scaling value was determined experimentally.

A. Practical Complexity and Communication Overhead for GCM-Based Adaptive Complexity Prediction

Due to the fact that we are performing software-only decoding simulations and the utilized codec does not support real-time decoding under the chosen experimental setting, it is hard to explicitly quantify the real effect of the complexity predictor versus the entire decoder complexity. However, considering that one of the most complex modules of the decoder is the inverse spatial transform, which by itself can operate in real-time in our decoder, preliminary results indicate that the most complex instantiation of the predictor used in the results of Table III and tested as a software-only C++ module, required approximately 25% of the execution time of the inverse transform. This complexity is substantially reduced as the bound on the predictor MAC operations is decreased. In addition, further optimization of the decoder or the complexity predictor implementation may change the relative complexity overhead added into the overall decoding process. Hence, the practical quantification of this overhead in a real-time environment remains a future research topic.

Finally, concerning the transmission overhead for the communication of the GCM values to the decoder, in all the experiments reported in Table III, the overhead was ranging from 0.5% to no more than 8% of the overall transmission bit rate. Since there is a linear relationship between (18) and (20) (complexity and information requirements of GCM-based prediction), the higher the bound set for $\mathrm{O}_{\mathrm{bound}}(\mathbf{n}_1)$, the more the GCM transmission overhead.

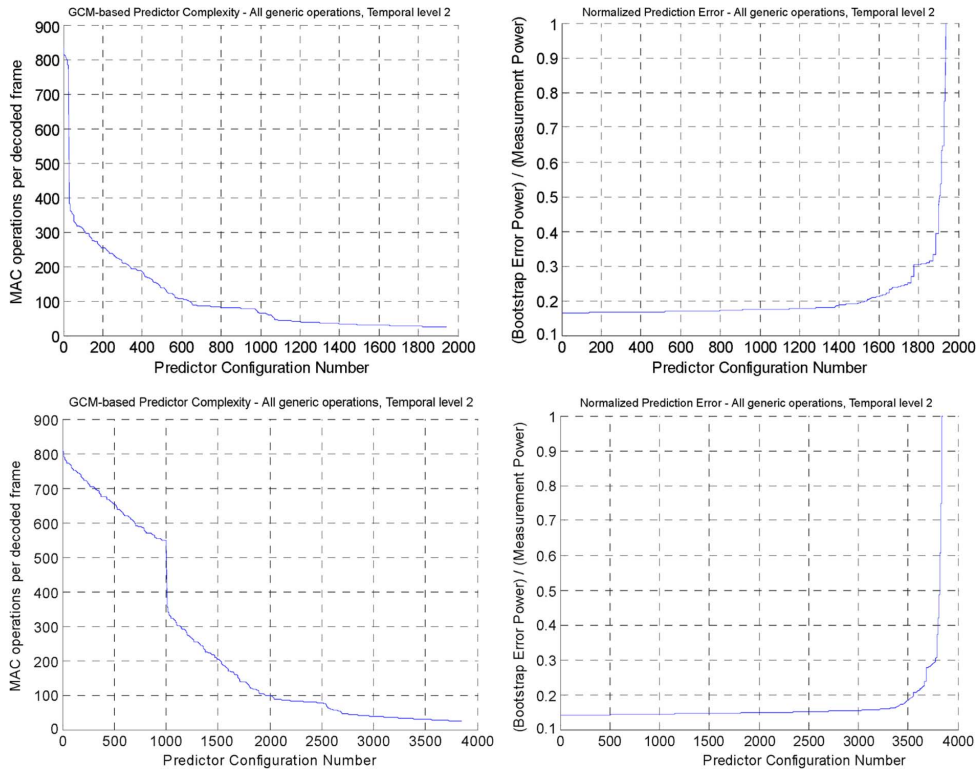

Fig. 6. Complexity and bootstrap error of the GCM-based prediction for the case of temporal level two. Top: confidence interval $\forall g : \varepsilon[g, 2] = 1.0$. Bottom: confidence interval $\forall g : \varepsilon[g, 2] = 0.5$. Each predictor configuration number corresponds to a particular setting for the predictor order, predictor adaptation period, and granularity of the application of the predictor from the parameter space of Fig. 3.

TABLE III
RESULTS OF RESOURCE-CONSTRAINED COMPLEXITY PREDICTION. PREDICTION WAS APPLIED FOR THE DECODING COMPLEXITY OF EACH SUBSEQUENT GOP OF THE SEQUENCE AND THE PREDICTION ERROR IS MEASURED PER VIDEO FRAME

$O_{\text{bound}}(\mathbf{n}_1)$ (in MAC operations per decoded frame of each GOP)	500	200	50
Av. $[e_{\text{out}}(\mathbf{n}_1)/c_{\text{out}}(\mathbf{n}_1)]^2$ (256 Kbps)	2.88%	5.74%	7.45%
Av. $[e_{\text{out}}(\mathbf{n}_1)/c_{\text{out}}(\mathbf{n}_1)]^2$ (512 Kbps)	3.77%	4.44%	6.91%
Av. $[e_{\text{out}}(\mathbf{n}_1)/c_{\text{out}}(\mathbf{n}_1)]^2$ (1024 Kbps)	3.11%	4.82%	7.21%

B. Discussion

The results of the statistical estimation of the expected error versus the predictor complexity shown in Fig. 6 serve as a validation that the proposed GCM-based complexity prediction approach provides lower resource-prediction error for increased computation bound for the predictor. This was additionally tested in practice with new sequences as demonstrated by the results of Table III. The monotonic relationship between predictor complexity and expected prediction error is a significant aspect of the proposed framework since, in this way, depending on the delay deadline for the decoding of each GOP, one can assign appropriate complexity to the resource prediction such that sufficient prediction accuracy is obtained for the particular application. The proposed framework appears to be able to provide good complexity estimates over a variety of sequences and bit rates, even though all the offline training steps and the online adaptation are essentially agnostic to the specific decoding operations and their implementation details. In fact, even though the estimation experiments for Table III were performed under a slightly different processor and under different processor load (which is anyway not constant in the Windows XP operating system) in comparison to our training stage, the online adaptation of the confidence interval for the bootstrap error estimates appears to cope well with these effects.

The proposed complexity-prediction framework can be used in several cases where delay-bounded video decoding is to be performed at a receiver. For example, under the flexible video decoding model [19], the proposed adaptive prediction can be used to accurately estimate the remaining time for each frame of each GOP. These estimates can be communicated to the operating system which can decide to increase or decrease the resources allocated to the particular task [29]–[31] as long as the delay deadline is not violated.

Another application of the proposed complexity-prediction framework is in admission control for a video bitstream to be decoded by a particular system. As shown in our previous work [17], the complexity mapping can generate bounds on each of the possible operations (MC, IO_H, IO_D, FC, and RS) and request that these bounds be satisfied for each GOP of the decoded video sequence. This effectively creates a dynamic complexity-specification (C-SPEC) negotiation mechanism between the video server and each client decoder that implements the complexity prediction. Although we have presented some preliminary results of such a negotiation in our previous work [17], [18], we plan to investigate this issue further in our future research.

VI. Conclusion

Accurate and generic methods for video-decoding resource prediction can provide significant improvements for scheduling and resource optimization algorithms in multimedia platforms. This paper analyzed in a systematic manner the various alternatives involved in autoregressive linear prediction or prediction based on generic complexity metrics. Our analysis led to the definition of an offline estimation process for resource-constrained complexity prediction followed by online adaptation of the prediction parameters. A large parameter space was taken into consideration. Our results indicated that the GCM-based approach provides better resource-prediction accuracy with lower computational requirements for the adaptive predictor. Moreover, under the proposed optimization approach, the resource-prediction accuracy is gracefully improved with increased predictor complexity, thereby providing an adaptable framework useful for a variety of applications involving resource optimization for multimedia decoding.

References

[1] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 704–716, Jul. 2003.

[2] V. Lappalainen *et al.*, "Complexity of optimized H.26L video decoder implementation," *IEEE Trans. Circuits, Syst. Video Technol.*, vol. 13, no. 7, pp. 717–725, Jul. 2003.

[3] J. Valentim, P. Nunes, and F. Pereira, "Evaluating MPEG-4 video decoding complexity for an alternative video complexity verifier model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 11, pp. 1034–1044, Nov. 2002.

[4] S. Saponara, C. Blanch, K. Denolf, and J. Bormans, "The JVT advanced video coding standard: Complexity and performance analysis on a tool-by-tool basis," in *Proc. Packet Video Workshop, (PVC)*, Apr. 2003, pp. 98–109.

[5] M. Ravasi and M. Mattavelli, "High-abstraction level complexity analysis and memory architecture simulations of multimedia algorithms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 5, pp. 673–684, May 2005.

[6] M. Ravasi and M. Mattavelli, "High-level algorithmic complexity evaluation for system design," *J. Syst. Architec.*, vol. 48, pp. 403–427, 2003.

[7] A. C. Bavier, A. B. Montz, and L. Peterson, "Predicting MPEG execution times," in *Proc. ACM SIGMETRICS*, 1998, pp. 131–140.

[8] P. A. Dinda and D. R. O'Hallaron, "An evaluation of linear models for host load prediction," in *Proc. IEEE Int. Symp. High Perf. Distrib. Comput.*, Aug. 1999, pp. 87–96.

[9] W. Yuan and K. Nahrstedt, "Energy-efficient CPU scheduling for multimedia applications," *ACM Trans. Computer Syst.*, vol. 24, no. 3, pp. 292–231, Aug. 2006.

[10] J. Liang, K. Nahrstedt, and Y. Zhou, "Adaptive multi-resource prediction in distributed resource sharing environment," in *Proc. IEEE Int. Symp. Cluster Comput. Grid*, Apr. 2004, pp. 293–300.

[11] D. G. Sachs *et al.*, "Cross-layer adaptive coding to reduce energy on general-purpose processors," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2003, vol. 3, pp. 109–112.

[12] H. J. Stolberg, M. Berekovic, and P. Pirsch, "A platform-independent methodology for performance estimation of streaming media applications," in *Proc. IEEE Int. Conf. Multimedia Expo, (ICME)*, Aug. 2002, vol. 2, pp. 105–108.

[13] H. J. Stolberg, M. Berekovic, and P. Pirsch, "A platform-independent methodology for performance estimation of multimedia signal processing applications," *J. VLSI Signal Process.*, vol. 41, pp. 139–151, 2005.

[14] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 5, pp. 645–658, May 2005.

[15] M. Mattavelli and S. Brunetton, "Implementing real-time video decoding on multimedia processors by complexity prediction techniques," *IEEE Trans. Consum. Electron.*, vol. 44, no. 3, pp. 760–767, Aug. 1998.

[16] G. Landge *et al.*, "Complexity metric driven energy optimization framework for implementing MPEG-21 scalable video decoders," in *Proc. IEEE Int. Conf. Accoust., Speech, Signal Process.*, Mar. 2005, vol. 2, pp. 1141–1144.

[17] Y. Andreopoulos and M. van der Schaar, "Complexity-constrained video bitstream shaping," *IEEE Trans. Signal Process.*, vol. 55, no. 5, pp. 1967–1974, May 2007.

[18] M. van der Schaar and Y. Andreopoulos, "Rate-distortion-complexity modeling for network and receiver aware adaptation," *IEEE Trans. Multimedia*, vol. 7, no. 3, pp. 471–479, Jun. 2005.

[19] S. Regunathan, P. A. Chou, and J. Ribas-Corbera, "A generalized video complexity verifier for flexible decoding," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2003, vol. 3, pp. 289–292.

[20] M. T. Sun and A. R. Reibman, Eds., *Compressed Video Over Networks*. New York: Marcel Dekker, 2001.

[21] T. Wiegand *et al.*, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[22] J. R. Ohm, M. van der Schaar, and J. W. Woods, "Interframe wavelet coding—Motion picture representation for universal scalability," *Signal Process.: Image Commun.*, vol. 19, no. 9, pp. 877–908, Oct. 2004.

[23] H. Schwarz *et al.*, Technical Description of the HHI Proposal for SVC CE1 ISO/IEC JTC1/SC29/WG11,m11244, Oct. 2004.

[24] Y. Andreopoulos, A. Munteanu, J. Barbarien, M. van der Schaar, J. Cornelis, and P. Schelkens, "In-band motion compensated temporal filtering," *Signal Process.: Image Commun. (Special Issue on "Subband/Wavelet Interframe Video Coding")*, vol. 19, no. 7, pp. 653–673, Aug. 2004.

[25] Intel Corp., "Intel VTune Data Collector Enabling Kit for I/O Processors," Application Note 273892-001.

[26] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[27] B. Hassibi, A. H. Sayed, and T. Kailath, "H^∞ optimality of the LMS algorithm," *IEEE Trans. Signal Process.*, vol. 44, no. 2, pp. 267–282, Feb. 1996.

[28] A. M. Zoubir and B. Boashash, "The bootstrap and its application in signal processing," *IEEE Signal Process. Mag.*, pp. 56–76, Jan. 1998.

[29] M. Srivastava, A. Chandrakasan, and R. Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programmable computation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 4, no. 1, pp. 42–55, Mar. 1996.

[30] S. Irani *et al.*, "An overview of the competitive and adversarial approaches to designing dynamic power management strategies," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 12, pp. 1349–1362, Dec. 2005.

[31] W. Yuan *et al.*, "GRACE: Cross-layer adaptation for multimedia quality and battery life," *IEEE Trans. Mobile Comput.*, to be published.

Yiannis Andreopoulos (M'00) is a Lecturer with the Electronic Engineering Department, Queen Mary University, London, U.K., since October 2006.

His research interests are in the fields of transforms, multimedia architectures and complexity modeling, video coding, and multimedia transmission through unreliable media.

Mihaela van der Schaar (SM'04) is an Assistant Professor with the Electrical Engineering Department, University of California, Los Angeles.

She has published extensively on multimedia compression, processing, communications, networking and architectures and holds 22 U.S. patents with several more pending.