Statistical Framework for Video Decoding Complexity Modeling and Prediction

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

***Abstract*— Video decoding complexity modeling and prediction is an increasingly important issue for efficient resource utilization in a variety of applications, including task scheduling, receiver-driven complexity shaping, and adaptive dynamic voltage scaling. In this paper we present a novel view of this problem based on a statistical framework perspective. We explore the statistical structure (clustering) of the execution time required by each video decoder module (entropy decoding, motion compensation, etc.) in conjunction with complexity features that are easily extractable at encoding time (representing the properties of each module's input source data). For this purpose, we employ Gaussian mixture models (GMMs) and an expectation-maximization algorithm to estimate the joint execution-time—feature probability density function (PDF). A training set of typical video sequences is used for this purpose in an offline estimation process. The obtained GMM representation is used in conjunction with the complexity features of new video sequences to predict the execution time required for the decoding of these sequences. Several prediction approaches are discussed and compared. The potential mismatch between the training set and new video content is addressed by adaptive online joint-PDF re-estimation. An experimental comparison is performed to evaluate the different approaches and compare the proposed prediction scheme with related resource prediction schemes from the literature. The usefulness of the proposed complexity-prediction approaches is demonstrated in an application of rate-distortion-complexity optimized decoding.**

***Index Terms*— Clustering methods, complexity modeling, complexity prediction, parametric density estimation, prediction theory, statistical analysis, video coding.**

I. Introduction

The PROLIFERATION of media-enabled portable devices based on low-cost processors makes resource-constrained video decoding an important area of research [12]–[16]. In addition, modern state-of-the-art video coders tend to be highly content-adaptive [11], [17] in order to achieve the best compression performance. Therefore, they exhibit a highly variable complexity and rate profile [18]. This makes efficient resource prediction techniques for video decoding a particularly important and challenging research topic. As a result, several fine-grain multimedia adaptation frameworks that combine with power-aware hardware modules to optimize battery life under quality constraints have been proposed [16], [19]. In order for the adaptation to be efficient, much work has recently focused on the video encoding and decoding complexity prediction [5]–[12], [14], [18]–[20].

In this paper, following a statistical learning perspective, we investigate the intrinsic relationship of the execution time requirements of each decoding module with simple feature variables representing the source and algorithm characteristics. We term the per-module execution time as real complexity metric (RCM) and try to establish whether it can be predicted from feature variables easily obtainable during encoding time. We demonstrate the usefulness of predicting module-specific execution time in scalable video decoders with an application that performs selective decoding of compressed content based on complexity-distortion optimization.

Statistical and machine learning techniques have been widely used in the image and video processing literature [1], [2]. However, this paper presents the first systematic attempt to bring these statistical tools in the emerging domain of video decoding complexity analysis and complexity prediction (a system overview is given in Section II). Our statistical point of view in these problems is inspired by the intuition that similar video sequences (in terms of content characteristics), when decoded under similar conditions, should produce a similar decoding complexity profile. The statistical analysis of our experimental results (Sections III and IV) confirms our expectation and furthermore provides useful insights into properties of video decoding complexity. Using these insights as well as domain-specific knowledge, we modify and properly extend standard statistical methods and models for the problem of video decoding resource prediction (Sections V–VII). Applications of decoder-driven bitstream shaping based on the derived results are discussed (Section VIII).

II. System Overview and Model Description

The proposed system consists of two main blocks as seen in Fig. 1: the joint probability density function (PDF) estimation module for RCM complexity features, which operates during the offline training; and the online complexity prediction module. The adaptive PDF re-estimation is an optional

Manuscript received October 16, 2006; revised January 23, 2008. First version published April 7, 2009; current version published July 22, 2009. This work was supported by the NSF (Grant NSF 0509522). N. Kontorinis acknowledges the generous support of the Alexander S. Onassis Public Benefit Foundation. This paper was recommended by Associate Editor Z. He.

N. Kontorinis and M. van der Schaar are with the Department of Electrical Engineering, University of California, Los Angeles, CA, 90095-1594, USA (e-mail: nkontor@ee.ucla.edu; mihaela@ee.ucla.edu).

Y. Andreopoulos was with the University of California, Los Angeles. He is currently with the Department of Electronic and Electrical Engineering, University College London, London, WC1E 7JE, U.K. (e-mail: iandreop@ee.ucl.ac.uk).

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.2009.2020256

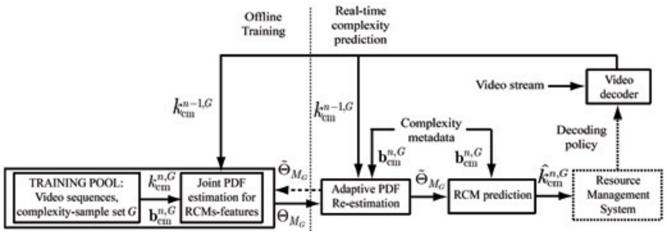

Offline Training
Real-time complexity prediction
$k_{\text{cm}}^{n-1,G}$
$k_{\text{cm}}^{n-1,G}$
Video stream
Video decoder
Complexity metadata
Decoding policy
$\mathbf{b}_{\text{cm}}^{n,G}$
$\mathbf{b}_{\text{cm}}^{n,G}$
$\tilde{\Theta}_{M_G}$
TRAINING POOL: Video sequences, complexity-sample set G
$k_{\text{cm}}^{n,G}$
$\mathbf{b}_{\text{cm}}^{n,G}$
Joint PDF estimation for RCMs-features
Θ_{M_G}
Adaptive PDF Re-estimation
$\tilde{\Theta}_{M_G}$
RCM prediction
$\hat{k}_{\text{cm}}^{n,G}$
Resource Management System

Fig. 1. Proposed complexity-driven system architecture.

enhancement module that increases the system prediction accuracy but also the overhead for online complexity prediction.

We assume that there is a joint PDF characterizing the statistical behavior of RCMs $k_{\text{cm}}^{n,G}$ (where cm is the RCM type) and the complexity features $\mathbf{b}_{\text{cm}}^{n,G}$. For each video frame, these features can be easily extracted based on the intra frames, the motion-compensated difference frames, and the motion vectors produced during encoding time, and can be stored or transmitted along with the compressed bitstream with minimal overhead [7]. Unlike the RCMs that depend on the particular algorithm used to decode and reconstruct the video data as well as on the implementation platform, these features depend only on the source properties (frames and motion vectors). The PDF estimation (Section III) provides the prediction module with the model parameters (denoted by Θ_{M_G}). When a more advanced model is used based on Markov chains (Section V), the prediction module also utilizes the model's state transition probabilities. In this case, there is also feedback from the decoder with the true values of RCMs of previously decoded frames ($k_{\text{cm}}^{n-1,G}$ in Fig. 1) in order to assess the prediction performance and recalibrate the model. The prediction module is provided with the complexity feature vector $\mathbf{b}_{\text{cm}}^{n,G}$ for each unit to be decoded (e.g., a video frame or a GOP) and, in conjunction with the PDF estimation that was performed offline, attempts to predict the RCMs of these units. When the adaptive PDF re-estimation module is present, it adaptively predicts the RCMs using either the offline-trained GMM model or the current sequence statistics (Section VI). A resource management system can utilize the complexity prediction for quality vs. energy tradeoffs [13], [16], or for scheduling the execution in multiple threads/processors [8].

A. Module-Specific Execution Times as Real Complexity Metrics

The decoding part of most modern video coders consists of an entropy decoding module, an inverse-transform module, a motion compensation module, and a pixel interpolation module (for fractional-pixel motion compensation) [3], [5], [11], [21]. In our recent work [6], [7], we identified an associated "generic" cost metric for each module. In this paper we are considering the execution time as the real complexity metric of each module. Even though execution time measurements are tied to a particular decoder platform, they capture the "real" complexity of processing (decoding) a particular compressed bitstream. Hence, they are relevant for a variety of applications such as operating system task scheduling [13], adaptive shaping of a compressed bitstream [7], dynamic voltage scaling methods [19], scheduling in multiple processors [8], etc. In addition, even though we assume that training can be performed based on a particular decoder of interest and by using representative sequences, we remark that, during online encoding, the required decoding execution time cannot be measured in most application scenarios, since decoders run on remote platforms.[1]

The module-specific breakdown of execution time is performed by partitioning the decoding software into the following modules: entropy decoding functions (ED_tics), inverse transform function (IT_tics), motion compensation (MC_tics), and fractional-pixel interpolation module (FI_tics).

For single-threaded single-processor execution, the summation of the execution time for all modules participating in the reconstruction of a video frame represents the decoding complexity for the particular frame [9], [11], [12], [15]. All remaining parts of the decoder cause negligible execution time overhead in comparison to these modules. A resource monitor can be implemented at the receiver, which measures the processor cycles ("tics") [22] required for the completion of the operations performed by each module, in order for them to be used during the training process. In this paper, we follow the generic approach of assuming that the required time for the processing of every frame by each module can be measured in real time. The vast majority of general-purpose processors or programmable digital signal processors have built-in registers for this purpose [22]. This process is of very low complexity and provides accurate measurements, which can be straightforwardly converted to time measurements based on the frequency of the underlying processor [22]. Moreover, minimal software instrumentation is required and this can be done in a similar fashion for a variety of underlying architectures and software implementations.

For each video frame n, with $1 \leq n \leq N$ and N the total number of frames in our sample space, we define the following complexity features:

1) the percentage of decoded nonzero transform coefficients $p_T(n)$;
2) the percentage of decoded nonzero motion vectors $p_M(n)$ (out of the maximum possible motion vectors per video frame);
3) the percentage of nonzero interpolated fractional-pixel positions $p_I(n)$;
4) the sum of magnitudes of the non-zero coefficients $\mathcal{T}_{\text{nonzero}}(n)$;
5) the sum of the run lengths of zero coefficients $\mathcal{T}_{\text{runlen}}(n)$.

These features depend only on the input source data (motion-compensated difference frames and motion vectors). Hence they can be computed at encoding time and transmitted to the decoder with the compressed bitstream [6], [7].

B. Model Variables and Notation

The elementary complexity that we model and predict in this paper is the sum of the module-specific RCMs over the

[1] In video streaming applications, encoding is typically performed in a powerful server workstation while decoding may occur in personal computers, laptops, portable video players, cell-phones, etc.

same-type frames in a GOP. Following terminology used in open-loop video coding [17], [21], our adaptation unit (AU) will be each temporal level in every GOP. Notice that the temporal levels can be seen as categories of frame types within each GOP [21]. In our experiments, we have grouped together temporal level 5 and 4 (corresponding to intra frames (I) and uni-directionally predicted (P) frames, respectively) since they consist of one frame per GOP. Temporal levels 1–3 correspond to hierarchies of bi-directionally predicted (B) frames using variable block-size motion estimation and compensation (similar to the hierarchical B pictures of H.264/AVC [11], [21]). Finally, temporal level 0 corresponds to the output (reconstructed) video frames. Hence, the theory and methods of this paper can be applied for complexity modeling in conventional I-, P-, and hierarchical B-frames of hybrid coders [3], [21] (e.g., in H.264/AVC [11]).

In order to segment all the available frames of each temporal level (frame type) into uniquely identifiable subsets, we define the parameter index set $G = (R, \text{tlev}; \text{SEQ}, \text{GOP})$. The parameters R, tlev, SEQ, and GOP refer respectively to the streaming bit rate, the temporal level, and the specific video sequence and GOP-number within the video sequence that the sample adaptation unit belongs to. Moreover, in our notation we use the semicolon (;) to separate the leftmost parameters that remain constant from the rightmost parameters that are the indices to our sample subset.

For the adaptation unit n that is characterized by the parameter index[2] $G = (R, \text{tlev}, \text{SEQ}, \text{GOP})$, we denote the numerical value of RCM cm (where $\text{cm} \in \{\text{ED\_tics}, \text{IT\_tics}, \text{MC\_tics}, \text{FI\_tics}\}$) by $k_{\text{cm}}^{n,G}$. Additionally, using the above notation, let $\mathbf{b}_{\text{cm}}^{n,G}$ denote the $(s-1)$th dimensional[3] feature vector associated with the RCM cm of the AU n indexed by the parameter index set G. Notice that the value for $(s-1)$ as well as the identity of the relevant features to each RCM varies and will be determined later with a pruning process. In the most general case, if we consider all the (not evidently correlated) features, we have

$$\mathbf{b}_{\text{cm}}^{n,G} = \left[p_T(n)\ \mathcal{T}_{\text{nonzero}}(n)\ \mathcal{T}_{\text{runlen}}(n)\ p_M(n)\ p_I(n)\right]. \quad (1)$$

By joining $k_{\text{cm}}^{n,G}$ and $\mathbf{b}_{\text{cm}}^{n,G}$, we define the s-dimensional sample vector $\mathbf{x}_{\text{cm}}^{n,G} = [k_{\text{cm}}^{n,G}\ \ \mathbf{b}_{\text{cm}}^{n,G}]$. Using the corresponding Greek letters for random variables, we define the random variable κ_{cm}^{G} for the RCM cm in the parameter set indexed by G, the feature vector random variable β_{cm}^{G}, and the vector random variable $\chi_{\text{cm}}^{G} = [\kappa_{\text{cm}}^{G}\ \ \beta_{\text{cm}}^{G}]$. We will refer to χ_{cm}^{G} as a "target complexity multivariable" and denote its s-dimensional joint probability density function as $P_{\text{cm}}(\chi_{\text{cm}}^{G})$. We will estimate this joint density function using the training dataset $X_{\text{cm}}^{G} = \{\mathbf{x}_{\text{cm}}^{1}, \mathbf{x}_{\text{cm}}^{2}, \ldots, \mathbf{x}_{\text{cm}}^{N}\}$, where N is the total number of AUs in the parameter set G. Then, we will use this density estimate, in conjunction with other statistical information, to predict the RCMs from instantiations of the complexity features.

C. Problem Description

In typical resource prediction problems we are concerned with the mean absolute error. In this paper, however, we use the least mean-square error (LMSE) criterion to quantify the accuracy of the prediction. Optimal complexity predictors in the LSME sense can be expressed analytically and also require little computational effort during the online prediction. Moreover, LSME-based predictors tend to produce individual errors of smaller magnitude [23] by penalizing large errors more heavily than e.g., the L_1-norm criterion.

Let $\hat{\kappa}_{\text{cm}}^{n_{\text{online}},G}$ denote the estimator of the RCM value $k_{\text{cm}}^{n_{\text{online}},G}$ of the sample set G that was used to train the estimator. Here n_{online} denotes the current AU index inside the video sequence that is being predicted. The minimization problem we solve, according to the least mean square error criterion, is

$$\min_{\hat{\kappa}_{\text{cm}}^{n_{\text{online}},G}} \mathrm{E}\left(\left(\hat{\kappa}_{\text{cm}}^{n_{\text{online}},G} - k_{\text{cm}}^{n_{\text{online}},G}\right)^2\right) \quad (2)$$

where the expectation is over all the adaptation parameters (e.g., temporal levels), all the AU indices (e.g., all the GOP), and all the RCMs cm. The above problem assumes knowledge of the joint RCM feature probability density functions $P_{\text{cm}}(\chi_{\text{cm}}^{G})$ defined in the previous section. To obtain $P_{\text{cm}}(\chi_{\text{cm}}^{G})$ from our training data, we need to address the probability density estimation problem. As explained and justified next, we employ a GMM for the PDF and use the EM algorithm to perform parametric density estimation.

III. Gaussian Mixture Modeling of RCMs—Complexity Features of Joint PDFs

In this section we introduce the Gaussian mixture model for the RCM complexity features statistical modeling. In order to tie our statistical analysis with prediction results, we next present our baseline prediction algorithm that is an LMSE-optimal predictor naturally customized for the GMM formulation. Indicative results for both the statistical properties and the prediction performance are presented, and certain interesting dimensions of decoding complexity are highlighted.

A. Gaussian Mixture Model for PDF Estimation of RCM Complexity Features

For illustrational purposes, we display our approach for the complexity of entropy decoding (i.e., RCM ED_tics) and using our domain knowledge, we consider the three complexity features that are intuitively more related to ED_tics. Thus, our complexity target variable $\chi_{\text{cm}}^{G} = \left[\kappa_{\text{cm}}^{G}\ \ \mathbf{b}_{\text{cm}}^{G}\right]$ will be

$$\chi_{\text{ED\_tics}}^{G} = \left[\kappa_{\text{ED\_tics}}^{G}\quad \mathcal{T}_{\text{nonzero}}^{G}\quad \mathcal{T}_{\text{runlen}}^{G}\quad p_{T}^{G}\right]$$

for some sample set indexed by G. Of course, with the appropriate selection of features and modifications in the dimensions of the variables and the model, the approach generalizes to any subset of complexity features.

Let $\hat{P}_{\text{ED\_tics}}(\chi_{\text{ED\_tics}}^{G})$ denote the density estimate of the four-dimensional entropy decoding complexity target variable

$$\chi_{\text{ED\_tics}}^{G} = \left[\kappa_{\text{ED\_tics}}^{G}\quad \mathcal{T}_{\text{nonzero}}^{G}\quad \mathcal{T}_{\text{runlen}}^{G}\quad p_{T}^{G}\right].$$

[2] When no semicolon is used, all the parameters are constant and the index set reduces to a single index.

[3] Note that we use $s-1$ dimensions to simplify notation, so that our target variable has s dimensions.

The Gaussian mixture model assumes that the density estimate can be written as a mixture of Gaussian components

$$\hat{P}_{\text{ED\_tics}}\left(\chi^G_{\text{ED\_tics}}\right) = \sum_{m=1}^{M_G} w^G_{\text{ED\_tics},m}\phi^G_{\theta_m}\left(\chi^G_{\text{ED\_tics}}\right) \tag{3}$$

where M_G (number of components) is determined through information-theoretic criteria as described in Section IV-B. Here, $\phi^G_{\theta_m}(\chi^G_{\text{ED\_tics}})$ denotes a 4-D Gaussian distribution $\mathcal{N}(\mathbf{m}^G_{\text{ED\_tics},m}, \mathbf{C}^G_{\text{ED\_tics},m})$ with parameters $\theta_m = (\mathbf{m}^G_{\text{ED\_tics},m}, \mathbf{C}^G_{\text{ED\_tics},m})$, where $\mathbf{m}^G_{\text{ED\_tics},m}$ is the mean vector

$$\mathbf{m}^G_{\text{ED\_tics},m} = \left[\mu_{\kappa^G_{\text{ED\_tics},m}} \quad \mu_{\mathcal{T}^G_{\text{nonzero},m}} \quad \mu_{\mathcal{T}^G_{\text{runlen},m}} \quad \mu_{p^G_T,m}\right] \tag{4}$$

and $\mathbf{C}^G_{\text{ED\_tics},m}$ is the 4 × 4 positive definite (symmetric) covariance matrix, compactly written as

$$\mathbf{C}^G_{\text{ED\_tics},m} = \begin{bmatrix} \mathbf{C}^G_{\kappa^G_{\text{ED\_tics}},m} & \mathbf{C}^G_{\kappa^G_{\text{ED\_tics}}\mathbf{b}^G_{\text{ED\_tics}},m} \\ \mathbf{C}^G_{\mathbf{b}^G_{\text{ED\_tics}}\kappa^G_{\text{ED\_tics}},m} & \mathbf{C}^G_{\mathbf{b}^G_{\text{ED\_tics}},m} \end{bmatrix} \tag{5}$$

where $\mathbf{C}^G_{\kappa^G_{\text{ED\_tics}},m}$, $\mathbf{C}^G_{\kappa^G_{\text{ED\_tics}}\mathbf{b}^G_{\text{cm}},m}$, and $\mathbf{C}^G_{\mathbf{b}^G_{\text{cm}},m}$ are the RCM, RCM feature, and the feature covariance matrices, respectively.

The parameters $w^G_{\text{ED\_tics},m}$ are the mixing (weight) coefficients, which denote the relative importance of each Gaussian in the total distribution. They satisfy the normalization condition

$$\sum_{m=1}^{M_G} w^G_{\text{ED\_tics},m} = 1. \tag{6}$$

For brevity, we will refer to each of the components as Gaussian component θ_m and to a specific Gaussian mixture with parameters $\{\theta_m | 1 \leq m \leq M_G\}$ as the mixture Θ_{M_G}.

Domain knowledge suggests that these features should be highly correlated. To determine the extent of their interdependence, we computed the cross-correlation of $\mathcal{T}_{\text{nonzero}}$ and each of $\mathcal{T}_{\text{runlen}}$ and p_T, for all the temporal levels of nine representative video sequences and several decoding bitrates. With the exception of the "Sailormen" sequence that had weaker correlation coefficients $\rho_{\mathcal{T}_{\text{nonzero}}\mathcal{T}_{\text{runlen}}} \approx -0.9$ and $\rho_{\mathcal{T}_{\text{nonzero}}p_T} \approx 0.85$, all the rest of the sequences had coefficients below -0.98 and above 0.97, respectively, which confirms our intuition. The strong dependence of the three features in our coder implementation indicates that using all of them is redundant. Hence, we performed dimensionality reduction of our sample space and kept only one of the three features ($\mathcal{T}_{\text{nonzero}}$) for our statistical analysis. Experiments with various subsets of the features in a scalable coder [17] yielded no tangible improvement in prediction results as compared to the cases when only $\mathcal{T}_{\text{nonzero}}$ was used (we omit these experiments here for brevity of description). Still, the more generic model description is useful for other coding frameworks, where usage of several features may lead to better prediction results.

B. Online Prediction Based on Complexity Features and Offline Training

Let $\hat{\theta}_m, 1 \leq m \leq M_G$ denote the mth Gaussian component estimated through expectation maximization (EM) in Section IV. The probability of a certain (s-dimensional) RCM cm and its feature pair $(\kappa^G_{\text{cm}}, \beta^G_{\text{cm}})$ given $\hat{\theta}_m$ is

$$\begin{aligned} P\left(\kappa^G_{\text{cm}}, \beta^G_{\text{cm}} \mid \hat{\theta}_m\right) &= \frac{1}{(2\pi)^{s/2}\sqrt{|\mathbf{C}^G_m|}} \\ &\times \exp\left[-\frac{1}{2}\left(\begin{bmatrix} \kappa^G_{\text{cm}} \\ (\beta^G_{\text{cm}})^T \end{bmatrix} - \begin{bmatrix} \mu_{\kappa^G_{\text{cm}},m} \\ (\mu_{\beta^G_{\text{cm}},m})^T \end{bmatrix}\right)^T\right. \\ &\times \left.(\mathbf{C}^G_m)^{-1}\left(\begin{bmatrix} \kappa^G_{\text{cm}} \\ (\beta^G_{\text{cm}})^T \end{bmatrix} - \begin{bmatrix} \mu_{\kappa^G_{\text{cm}},m} \\ (\mu_{\beta^G_{\text{cm}},m})^T \end{bmatrix}\right)\right]. \end{aligned} \tag{7}$$

The Gaussian form of the above joint conditional probability yields analytic solutions for the optimal LMSE estimator [23], which, given $\mathbf{b}^{n,g}_{\text{cm}}$ and $\hat{\theta}_m$, is

$$\begin{aligned} \hat{\kappa}^G_{\text{cm},m} &= \mathrm{E}\left[\kappa^G_{\text{cm}} \mid \mathbf{b}^{n,g}_{\text{cm}}, \hat{\theta}_m\right] \\ &= \mu_{\kappa^G_{\text{cm}},m} + \mathbf{C}^G_{\kappa^G_{\text{cm}}\mathbf{b}^G_{\text{cm}},m}\left(\mathbf{C}^G_{\mathbf{b}^G_{\text{cm}},m}\right)^{-1}\left(\mathbf{b}^{n,g}_{\text{cm}} - \mu_{\beta^G_{\text{cm},m}}\right). \end{aligned} \tag{8}$$

The last equation gives an explicit estimator for the instantiation of each RCM cm using the available *online* complexity feature vector $\mathbf{b}^{n,G}_{\text{cm}}$ and the parameters of the Gaussian component $\hat{\theta}_m$. Since the parameter training is performed *offline*, as described in Section III, the matrix inversion and multiplication in (8) are pre-computed; thus only one matrix-vector multiplication and vector subtraction are performed online. Notice that the derived expectations of (8) are mixed according to the mixing coefficients w^G_m in (3), i.e., the probability that $\hat{\theta}_m$ is responsible for χ^G_{cm}, prior to observing $\mathbf{b}^{n,G}_{\text{cm}}$. The conditional expectation $\hat{\kappa}^G_{\text{cm}}$ for the whole GMM Θ_{M_G} given the observed complexity feature $\mathbf{b}^{n,G}_{\text{cm}}$ is thus

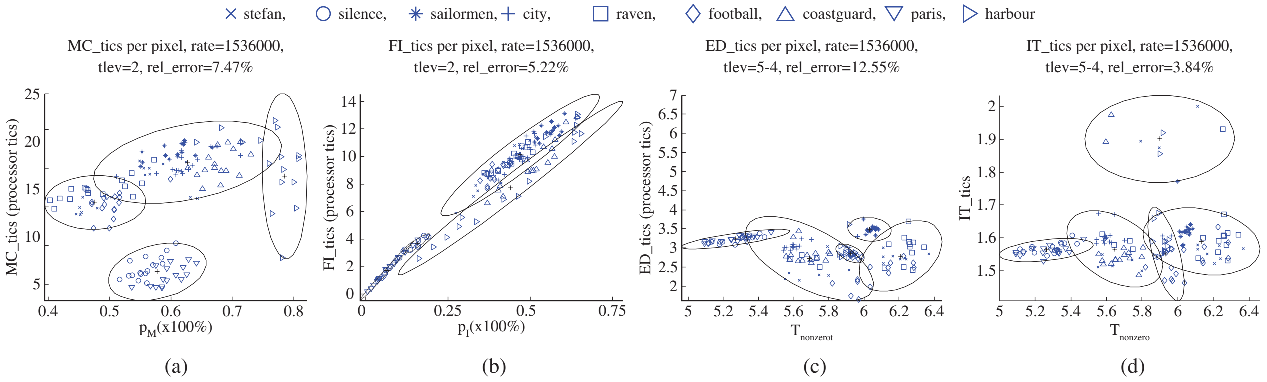
$$\hat{\kappa}^G_{\text{cm}} = \sum_{m=1}^{M_G} w^G_m \cdot \hat{\kappa}^G_{\text{cm},m} \tag{9}$$

which, given the derived estimators of (8) and the mixing coefficients w^G_m, is computationally inexpensive in comparison to the complexity of video decoding. We remark that the online model execution complexity was negligible in comparison to the video decoding complexity (less that 2% in all cases).

C. Statistical Analysis of RCMs Using the Proposed Scheme and Observations

Our proposal to predict complexity through offline PDF training on the RCM feature space and LMSE prediction using the current feature value is based on the assumption that similar (in the complexity sense) sequences mostly lie on the same regions of the RCM feature space. Thus, when these regions do not overlap excessively across the feature direction, features can efficiently discriminate and predict the RCM values.

In the following figures and tables we show indicative and motivating results from a representative set of standard CIF test video sequences. The spatial domain version of the video coder [17] used in our experiments encoded the first 256 frames of nine sequences: "Stefan," "Silence," "Sailormen," "City," "Raven," "Football," "Coastguard," "Paris,"



(a) (b) (c) (d)

Fig. 2. Examples of GMM fittings and LMSE prediction results for (a) motion compensation, (b) fractional interpolation RCMs, (c) ED_tics, and (d) IT_tics. Each experimental point corresponds to one GOP and the results are presented in average number of processor tics per pixel of each GOP.

and "Harbour" using multiframe variable blocksize motion-compensated prediction and update steps within multiple decomposition (temporal) levels. For each GOP, the total number of temporal levels[4] was set to four (i.e., three sets of hierarchical B-frames and one P-frame). Each GOP had 16 frames. In all figures and tables, we show results from prediction with GOP granularity (i.e., the sum of the RCMs and features in each temporal level for each GOP were calculated leading to $N_{\rm seq} = 16$ points per sequence) and prediction with temporal level granularity. The results are normalized to indicate the average number of processor cycles ("tics") per pixel (computed across temporal levels and GOP).

We start by presenting an indicative example for the complexity measurements of the motion compensation (using p_M as complexity feature) and the fractional-pixel interpolation module (using p_I as feature). For each case we present results corresponding to an indicative temporal level (prediction hierarchy). Apart from the data points, we also depict the fitted Gaussian mixture model. Fig. 2(a)–(b) presents the related results. The corresponding prediction results shown in the titles are presented in more detail in Table I and will be discussed separately.

The first thing to notice in Fig. 2(a)–(b) is the self-clustering of most sequences' GOP in the RCM complexity feature space. We observed similar results across temporal levels and bitrates. This "self-clustering" property will be further discussed and exploited for increased complexity prediction accuracy in Section V. There are, however, some notable exceptions, such as the sports sequence "Football" and the sequence "Raven" that have irregular motion of multiple objects and also camera motion. Moreover, in the case of FI_tics, on top of the self-clustering behavior we note a strong linearity in the RCM complexity feature space, which could potentially lead to a simpler model. Overall there also appears to be some "cross-clustering" among different sequences that indicates the similarity of sequences in a decoding complexity sense. For instance, notice that GOP from "Paris" and "Silence" cluster together and this behavior was consistent across most rates and temporal levels. Although the cross-clustering is quite weak in general compared to the self-clustering, it can play a complementary role to the latter and enhance prediction during changes in video content. We also remark that the cases without clustering that are present in the results of Fig. 2(a)–(b) will be handled based on techniques that improve the prediction based on feedback from the decoder RCM measurements (introduced in a later section).

TABLE I

Mean Relative Prediction Error [Percentage—as Defined in (10)] per Sequence for the Motion Compensation and Fractional-Pixel Interpolation RCMs. The Mean Prediction Error Is the Average of the Prediction Error for Each Temporal Level of Each GOP of Each Sequence

	MC_tics	FI_tics		MC_tics	FI_tics
Stefan	10.69	4.17	**Football**	6.09	3.21
Silence	37.84	16.43	**Coastguard**	11.21	7.74
Sailormen	22.46	3.23	**Paris**	38.62	25.17
City	13.95	2.94	**Harbour**	28.25	20.05
Raven	11.12	3.58			

In Table I we present the corresponding prediction errors by predicting the measurements of the training set via the optimal estimator of (8). The errors are calculated as the mean relative error[5] (in percentages) per temporal level of each GOP and each sequence

$$\text{mean\_rel\_err}(\text{tlev}, \text{GOP}, \text{SEQ}) = \frac{\sum_{n=1}^{N_{\max}} \left| k_{\rm cm}^{n,G} - \hat{k}_{\rm cm}^{n,G} \right|}{\sum_{n=1}^{N_{\max}} k_{\rm cm}^{n,G}} \cdot 100\% \tag{10}$$

where the index n is over all the GOP in that particular temporal level (frame type) for one sequence and $N_{\max} = N_{\rm seq} \cdot 2^{5-\rm tlev}$ based on the properties of temporal levels in the coder used in our experiments [17]. The cases that exhibit large prediction error will be handled with approaches based on online measurement feedback, as explained later.

For the entropy decoding and the inverse transform RCMs, i.e., ED_tics and IT_tics, we present an indicative example at

[4] In a four-temporal-level video coder, entropy decoding is performed in levels 4 through 1 (nothing is transmitted for level 0—which corresponds to the output frames of the reconstructed video sequence) and for the intra-frames of the additional fifth level. Reconstructed frames via motion compensation are created in levels 3 through 0 (no frames are created via motion compensation in temporal level 4 and 5).

[5] Even though the natural error criterion to report for LMSE estimation and prediction would be SNR, in order to adhere to the relevant resource prediction literature [12]–[16], we report the error based on (10).

high bit rate, where the complexity measurements are more prominent and exhibit higher variability across the different video sequences. We are using $\mathcal{T}^G_{\text{nonzero}}$ as feature, based on the dimensionality reduction proposed previously (Section III-A). Fig. 2(c)–(d) and Tables II and III contain indicative results. The scaling of the measurements is normalized relative to the average value of all RCMs present in our experimental pool, in order to show that these metrics have low importance in the overall decoding execution time. Notice that this is true even for the presented case of tlev = 5, which corresponds to intra frames.

Overall, similar remarks to the motion compensation and interpolation cases regarding the self- and cross-clustering apply here as well. The provided prediction examples for IT_tics (Table II) demonstrate that the vast majority of the mean relative prediction errors are below 10% and actually in quite a few cases they are below 5%. On the other hand, worse prediction performance is obtained for the ED_tics (Table III). However, this large error is not expected to affect the overall complexity prediction significantly, since ED_tics and IT_tics do not contribute much to the overall complexity. Since the prediction is performed per GOP, the errors in each temporal level can partially cancel out. Thus, the relative errors per temporal level serve as indicators of how well the prediction is done per temporal level, but their average is generally more than the actual mean total error per GOP for a particular sequence. The later is given under the "Total$_1$" column in Tables II and III.

Although we made observations about clustering that are behind the proposed algorithms and the prediction results of Tables I and III, the target of this paper is *complexity prediction* and not the clustering itself. Weaker clustering may still lead to good prediction and vice versa. Moreover, the results in this section do not contain all the improvements we propose in our framework (such as adaptive prediction based on online measurement feedback) and they are provided in this section as an initial motivation.

IV. Offline Training for the GMM Parameter Estimation

To learn the parameters of the GMM we employ an EM algorithm that is biased toward our goal for reduced complexity prediction error. The following section introduces the basic EM scheme for our problem while Section IV-B discusses the appropriate selection of the GMM components based on an information-theoretic criterion.

A. Parameter Estimation With the Standard Expectation-Maximization Algorithm

To specify a d-dimensional Gaussian function one needs d coordinates for the mean and

$$\binom{d+1}{2} = \frac{1}{2}d(d+1)$$

elements for the covariance matrix (the rest are given from the symmetry). For a GMM Θ_{M_G} with M_G Gaussian components, $M_G - 1$ mixing coefficients (due to the normalization condition) are required. Thus, in the general case, the number $a(M_G)$ of parameters one needs to specify in (3) is

$$a(M_G) = M_G \cdot \left(\frac{1}{2}d(d+1) + d + 1\right) - 1. \tag{11}$$

For our specific example with the four-dimensional entropy decoding complexity target variable [see (4), (5)] we would need $a(M_G) = 15 \cdot M_G - 1$ parameters. However, under the dimensionality reduction imposed on the problem, this number decreases significantly to $a(M_G) = 6 \cdot M_G - 1$.

The estimation of the GMM parameters is performed through the EM algorithm [24], [25]. EM is an iterative optimization procedure that attempts to maximize the likelihood function of the data

$$L\left(\Theta_{M_G}; X^G_{\text{cm}}\right) = \prod_{n=1}^{N}\left(\hat{P}_{\text{cm},\Theta_{M_G}}\left(\mathbf{x}^{n,G}_{\text{cm}}\right)\right) \tag{12}$$

or equivalently the more convenient log-likelihood function of the data

$$l\left(\Theta_{M_G}; X^G_{\text{cm}}\right) = \sum_{n=1}^{N}\log\left(\hat{P}_{\text{cm},\Theta_{M_G}}\left(\mathbf{x}^{n,G}_{\text{cm}}\right)\right) \tag{13}$$

over all the models Θ_{M_G}. Here, $\hat{P}_{\text{cm},\Theta_{M_G}}(\mathbf{x}^{n,G}_{\text{cm}})$ denotes the probability of the sample $\mathbf{x}^{n,G}_{\text{cm}}$ under the model Θ_{M_G}.

In the case of Gaussian mixture models, EM alternatively computes the expected values of the mixing coefficients per sample, also called *responsibilities* of each Gaussian component for each sample, given the current model parameters and data and then updates the model parameters given the new responsibilities and the data.

In the expectation step, the responsibility of all components $\hat{\theta}_m$, $1 \leq m \leq M_G$, for all data samples $\mathbf{x}^{n,G}_{\text{cm}}$, $1 \leq n \leq N$, is computed using

$$\hat{\gamma}_{n,m} = w^G_{\text{cm},m} \cdot \phi^G_{\theta_m}\left(\mathbf{x}^{n,G}_{\text{cm}}\right) \Bigg/ \left[\sum_{m=1}^{M_G} w^G_{\text{cm},m} \cdot \phi^G_{\theta_m}\left(\mathbf{x}^{n,G}_{\text{cm}}\right)\right]. \tag{14}$$

In the maximization step, EM computes the weighted means and covariances

$$\mu^G_{\text{cm},m}(i) = \sum_{n=1}^{N} \hat{\gamma}_{n,m} \cdot \mathbf{x}^{n,G}_{\text{cm}}(i) \Bigg/ \sum_{n=1}^{N} \hat{\gamma}_{n,m} \tag{15}$$

$$C^G_{\text{cm},m}(i,j) = \frac{\begin{array}{c}\sum_{n=1}^{N}\Big[\hat{\gamma}_{n,m} \cdot \left(\mathbf{x}^{n,G}_{\text{cm}}(i) - \mu^G_{\text{cm},m}(i)\right) \\ \cdot \left(\mathbf{x}^{n,G}_{\text{cm}}(j) - \mu^G_{\text{cm},m}(j)\right)\Big]\end{array}}{\sum_{n=1}^{N} \hat{\gamma}_{n,m}} \tag{16}$$

as well as the updated mixing coefficients

$$w^G_{\text{cm},m} = \sum_{n=1}^{N} \hat{\gamma}_{n,m} \Bigg/ N. \tag{17}$$

At the start of the algorithm the parameters of the GMM are picked randomly. In addition, a good way to construct initial guesses for the means is to pick any M_G samples at random. To initialize the covariance matrices, one can use the total

TABLE II

MEAN RELATIVE PREDICTION ERROR [PERCENTAGE—AS DEFINED IN (10)] PER TEMPORAL LEVEL AND SEQUENCE FOR IT_tics, GIVEN $\mathcal{T}^{G}_{\text{nonzero}}$ AND THE FITTED GAUSSIAN MIXTURE FROM THE OFFLINE TRAINING

IT_tics	Rate = 384 kbps					Rate = 1536 kbps				
	tlev = 5, 4	tlev = 3	tlev = 2	tlev = 1	Total_1	tlev = 5, 4	tlev = 3	tlev = 2	tlev = 1	Total_1
Stefan	3.01	9.81	5.47	4.42	**3.33**	9.53	6.31	21.35	2.05	**5.02**
Silence	2.93	7.61	7.55	6.29	**4.58**	1.20	6.34	7.33	2.20	**3.29**
Sailor-men	3.70	4.69	5.00	2.98	**2.71**	1.28	6.59	9.50	3.63	**4.06**
City	1.39	4.21	5.36	6.91	**4.48**	1.37	4.06	6.25	2.74	**3.08**
Raven	2.04	8.46	5.39	2.46	**2.40**	3.78	4.02	12.19	4.05	**4.62**
Football	4.47	12.23	7.50	6.45	**4.81**	6.24	5.96	43.04	3.14	**12.65**
Coast-guard	2.16	5.38	9.22	8.83	**6.67**	5.59	6.80	15.49	5.11	**5.41**
Paris	1.21	3.23	7.51	5.83	**4.62**	0.93	6.26	8.41	2.56	**3.89**
Harbour	1.26	7.90	6.35	9.13	**5.69**	4.59	6.52	20.97	10.76	**9.54**
Total_2	**2.46**	**7.06**	**6.60**	**5.92**	**4.37**	**3.84**	**5.87**	**16.06**	**4.03**	**5.73**

covariance of the data, while the initial mixing coefficients are usually considered all equal to $1/M_G$.

The algorithm is assumed to have converged when the log-likelihood function $l(\Theta^{(j)}_{M_G}; X^G_{\text{cm}})$ in the jth iteration differs from $l(\Theta^{(j-1)}_{M_G}; X^G_{\text{cm}})$ by less than the machine precision threshold ε.

Although the EM algorithm is well suited for GMM estimation, there are two potential problems with its practical application. Firstly, there are potential convergence problems. Like all the stochastic optimization procedures, EM can be trapped in local maxima of the log-likelihood and thus yield suboptimal results, or suffer from the singularity matrix problem that will prohibit the algorithm from converging to a solution. These problems are ameliorated when a large enough number of random starting points (model parameters) are used during the training and then the best solutions are chosen.

The second problem involves the selection of the appropriate complexity of the model in order to capture the true underlying probability distribution without overfitting the training data. Our approach to handle this is discussed in the next section.

B. Selection of Appropriate GMM Complexity—Number of GMM Clusters to Use

There are several criteria that penalize higher model complexity. One of the most often used in conjunction with GMMs is the Bayesian information criterion (BIC)

$$\text{BIC}(M_G) = -\,2 \cdot \text{loglik}(M_G) + a(M_G) \cdot \log N \qquad (18)$$

where $\text{loglik}(M_G)$ is the maximized log-likelihood function (the log-likelihood function for the best GMM $\Theta^*_{M_G}$ with M_G Gaussian components)

$$\text{loglik}(M_G) = \max\Big(l\left(\Theta_{M_G}; X^G_{\text{cm}}\right)\Big) = \sum_{n=1}^{N_{\max}} \log P\left(\mathbf{x}^{n,g}_{\text{cm}} | \Theta^*_{M_G}\right). \qquad (19)$$

In this paper, we use BIC to determine the appropriate GMM complexity. Following [26], we select the number M_G that gives rise to the first decisive local minimum for BIC, i.e.,

$$M^*_G = \min\big\{M_G : (\text{BIC}(M_G - 1) > \text{BIC}(M_G)) \\ \wedge (\text{BIC}(M_G) < \text{BIC}(M_G + 1))\big\}. \qquad (20)$$

In order to justify the usage of BIC in the domain of video decoding resource modeling and prediction (since BIC is a general model selection criterion), we compared the performance for values of M_G within a distance of 3 from the optimal M^*_G according to the BIC, and picked the one that produced the best prediction results. Thus, even though the GMM was trained with the log-likelihood as cost function, among those optimal models close to the best BIC value, we selected the one that minimized our average prediction error per temporal level in our training set. This alternative method of model complexity selection yielded on average only marginally better prediction results (in most cases less than 3% improvement). Since the BIC also yielded less complex models (all of them containing less than seven Gaussian components), its use in our application domain is justified.

V. ADAPTIVE ONLINE COMPLEXITY PREDICTION THROUGH A MARKOV CHAIN MODEL OF DOMINANT GAUSSIAN COMPONENTS

The baseline prediction presented earlier builds upon the cross-clustering of similar sequences and thus is useful during scene changes or when no decoding complexity measurement feedback is provided. However, video source characteristics exhibit, in general, strong short-term autocorrelation that is exploited in state-of-the-art video coders. This autocorrelation manifests itself also in decoding complexity characteristics, as was shown by various authors through autoregressive and adaptive linear models of complexity [7], [16] and was further confirmed through the strong self-clustering behavior observed in Section III-C. Here, we seek to take advantage of the strong short-term autocorrelation in our statistical modeling and prediction approach using Markov-chain models.

A. Model Description and Intuition

We model the sequence of Gaussian components θ_m that are expected to be responsible[6] for the sequence of observations of the complexity target random variable χ^G_{cm} as a Markov chain. We call these Gaussian components *dominant*.

[6] Since our model is probabilistic, any Gaussian component can be responsible for a particular complexity variable instantiation, albeit with different—often very small—probability.

TABLE III

Mean Relative Prediction Error [Percentage—as Defined in (10)] per Temporal Level and Sequence for ED_tics, Given $\mathcal{T}^{G}_{\text{nonzero}}$ and the Fitted Gaussian Mixture from the Offline Training

ED_tics	**Rate = 384 kbps**					**Rate = 1536 kbps**				
	tlev = 5, 4	**tlev = 3**	**tlev = 2**	**tlev = 1**	**Total_1**	**tlev = 5, 4**	**tlev = 3**	**tlev = 2**	**tlev = 1**	**Total_1**
Stefan	31.19	47.95	26.73	50.26	**40.33**	19.00	13.68	14.02	20.08	**14.00**
Silence	24.73	14.26	14.42	12.41	**8.16**	10.25	13.02	17.87	13.98	**13.88**
Sailor-men	16.35	14.42	11.00	10.84	**4.19**	11.93	9.28	7.25	14.17	**4.51**
City	18.07	9.99	6.65	12.56	**9.62**	2.93	4.40	7.38	7.42	**5.30**
Raven	9.70	12.90	11.54	26.83	**14.45**	9.46	6.57	9.40	9.22	**6.56**
Football	39.20	67.42	36.73	104.81	**70.33**	30.08	9.76	14.02	9.50	**10.15**
Coast-guard	5.16	18.45	19.95	11.85	**9.12**	9.39	5.70	5.29	7.32	**3.82**
Paris	27.25	11.08	15.07	17.73	**17.97**	10.34	13.72	21.36	19.61	**17.44**
Harbour	4.16	23.78	16.40	16.76	**12.87**	9.56	7.97	9.86	12.89	**7.28**
Total_2	**19.54**	**24.47**	**17.61**	**29.34**	**20.78**	**12.55**	**9.34**	**11.83**	**12.69**	**9.21**

Although we performed experiments with a variety of Markov-chain orders for our model, it appears that the majority of gain in the complexity-prediction results appears already from an order-1 model ("Markov-1"). Hence, we focus our analysis on this case for the remainder of the paper. A detailed study on whether higher order models are truly beneficial for some complexity-prediction applications under the proposed framework is left as a future topic.

Let θ_m^n denote the Gaussian components that produce the complexity target variable instantiations $\mathbf{x}_{\text{cm}}^{n,G}, n \in \{1, \ldots, N_{\text{online}}\}$ and θ_i^{n-1} the Gaussian component that produces the complexity target variable instantiation $\mathbf{x}_{\text{cm}}^{n-1,G}$, for a particular video sequence and in a particular measurement set G, where N_{online} is the total number of AUs in the currently decoded video sequence. In most of the cases, and provided that the motion compensation part of the coder is successful enough in capturing the intrinsic scene motion, we expect the complexity in the entropy decoding $\mathbf{x}_{\text{cm}}^{n,G}$ (and similarly in the inverse filtering) to lie close to the complexity $\mathbf{x}_{\text{cm}}^{n-1,G}$ of the previous AU. Intuitively, this complexity "inertia" is expected much more in the low temporal levels (i.e., prediction among frames that are closer to each other in time [27]) than in the high ones that are noisier and less correlated. In the case of shot changes, we expect jumps to different regions of the complexity sample space. Similar behavior is expected for the complexity of motion compensation and interpolation modules.

Knowing θ_i^{n-1} therefore provides valuable information for the probability to have the next dominant Gaussian component θ_m^n. This information is captured in our model through the state transition probabilities $P_\theta(i, j) = P(\theta_i^n \mid \theta_j^{n-1})$ that are trained offline from our training dataset as follows.

We calculate for all training sequences and all the temporal levels the state transition probabilities $P_\theta(i, j) = P(\theta_i^n \mid \theta_j^{n-1})$ of the dominant Gaussian components using the conditional occurrence frequencies from our training set. The identification of the dominant Gaussian components is done with comparison of the Gaussian component likelihoods for the observed complexity data $\mathbf{x}_{\text{cm}}^{n,g}$. The transition probability matrices are at most 6×6 since all our Gaussian mixtures contain at most six Gaussian components (see Section III). Typical transition probability matrices are illustrated in Table IV. We used the ED_tics results for temporal levels 5 and 4 at 512 and 1024 kbps with the optimum number of Gaussian components $M_G = 4$ and $M_G = 3$, respectively. The probability concentration in the diagonal agrees with the self-clustering property.

To further motivate the importance of the correct identification of the dominant Gaussian component for the prediction accuracy, we present next an oracle prediction scheme that assumes this information is a-priori available.

B. Selection of Appropriate GMM Complexity—Number of GMM Clusters to Use

This section presents the results for an oracle algorithm that predicts the RCM given the complexity feature and under the assumption we know the dominant Gaussian component the current sample belongs to. This component can be trivially determined *a posteriori* by finding the Gaussian component that yields the largest probability for the observed complexity metric-feature pair after decoding. However, this component cannot be determined *a priori* in a completely reliable manner, since this essentially assumes that the RCM that we want to predict is known, which is true only after the decoding is completed. In this sense, the oracle algorithm is useful as an upper bound on the prediction accuracy that practical algorithms can achieve within our statistical framework.

The results of Table V for ED_tics are significantly better than our baseline algorithm and remain quite good even for low bit rates, where prediction tends to be more unstable. Similar improvements were observed for IT_tics, MC_tics, and FI_tics with the average prediction error reducing to 3, 7.5, and 2.5% respectively. Due to space limitations, these results are omitted. Thus, it becomes clear that the correct identification of the dominant Gaussian component can play a major role in the prediction accuracy results. In conjunction with the strong self-clustering property (see Section III) and through the transition probabilities of Table IV, we expect that the introduction of memory in our prediction scheme should provide significant improvements in the proposed framework's prediction performance. In other words, finding *a posteriori* the dominant Gaussian component for the previous measurement will help us improve our prediction for the current

TABLE IV
EXAMPLES OF TRANSITION PROBABILITY MATRICES. THE jTH ELEMENT OF EACH ROW IN THE TABLES INDICATES THE PROBABILITY OF TRANSITION FROM THE DOMINANT GAUSSIAN COMPONENT i TO DOMINANT GAUSSIAN COMPONENT j

Gaussian component t number	$j = 1$	$j = 2$	$j = 3$	$j = 4$
$i = 1$	0.93	0.00	0.07	0.00
$i = 2$	0.00	0.76	0.22	0.02
$i = 3$	0.05	0.28	0.67	0.00
$i = 4$	0.00	0.08	0.00	0.92

Gaussian component t number	$j = 1$	$j = 2$	$j = 3$
$i = 1$	0.94	0.00	0.06
$i = 2$	0.00	1.00	0.00
$i = 3$	0.10	0.00	0.90

measurement. An algorithm to accomplish this is described next.

C. Online Prediction with Additional Online Measurement Feedback

Assuming that after the offline training we have reliable estimates for the transition probabilities $P_\theta(i, j) = P(\theta_i^n \mid \theta_j^{n-1})$ as well as the observation probabilities $P(\mathbf{b}_{\text{cm}}^{n,g} \mid \theta_i^n)$ with $i, j \in \{1, \ldots, M_G\}$, we propose the following extension to the prediction approach.

Instead of using the (stationary) mixing coefficients w_m^G obtained from the EM training of the offline data, we update the mixing coefficients at each prediction step to obtain a sequence $\tilde{w}_m^n$. The update is done in two phases. We denote with $\tilde{w}_m^n = \tilde{P}(\theta_m^n)$ the current weight estimate that was used in the prediction of $\hat{k}_{\text{cm}}^n$. Once the decoding is done, we get the actual value k_{cm}^n as feedback from the decoder. Now, for all $m \in \{1, \ldots, M_G\}$ we calculate (using Bayes' law) the posterior probability of having the mth Gaussian component produce the current complexity observation

$$P(\theta_m^n \mid k_{\text{cm}}^n, \mathbf{b}_{\text{cm}}^{n,G}) = \frac{P\left(k_{\text{cm}}^n, \mathbf{b}_{\text{cm}}^{n,G} \mid \theta_m^n\right) \cdot \tilde{P}\left(\theta_m^n\right)}{\sum_{i=1}^{M_G} P\left(k_{\text{cm}}^n, \mathbf{b}_{\text{cm}}^{n,G} | \theta_i^n\right) \cdot \tilde{P}\left(\theta_i^n\right)}. \quad (21)$$

We update the current probabilities for the Gaussian components by setting $\tilde{P}(\theta_m^n) \leftarrow P(\theta_m^n \mid k_{\text{cm}}^n, \mathbf{b}_{\text{cm}}^{n,G})$. Then, we obtain the probability to get each Gaussian component in the next AU using the state transition probabilities

$$\tilde{P}\left(\theta_m^{n+1}\right) = \sum_{i=1}^{M_G} P\left(\theta_m^{n+1} | \theta_i^n\right) \cdot \tilde{P}\left(\theta_i^n\right). \quad (22)$$

The updated probabilities constitute the mixing coefficients $\tilde{w}_m^{n+1}$ for the next AU complexity prediction. The rest of the prediction proceeds as in the static LMSE prediction case, with the only difference that we use $\tilde{w}_m^{n+1}$ in (9).

VI. VARIANTS OF RCM PREDICTION

In the next sections we will consider alternative prediction schemes that can reduce in some cases either the prediction error or the prediction overhead. We will also show a way to adaptively enhance the trained PDF when "atypical" sequences (in complexity terms) are encountered.

A. Alternative Prediction Scheme: Maximum-Likelihood (ML) Prediction

In this section we use again the LMSE estimator given $\mathbf{b}_{\text{cm}}^{n,G}$ and $\hat{\theta}_m$, as described in (8), but instead of mixing the estimators for all the Gaussian components as in (9) we only consider the most likely Gaussian component $\hat{\theta}_{m^*}$ for the observed feature value $\mathbf{b}_{\text{cm}}^{n,G}$

$$m^* = \arg\max_m P\left(\mathbf{b}_{\text{cm}}^{n,g} \mid \hat{\theta}_m\right). \quad (23)$$

Once we obtain the solution of the optimization in (23) through M_G probability evaluations and $M_G - 1$ comparisons, we use the modified LMSE estimator with one Gaussian component

$$\begin{aligned}\hat{\kappa}_{\text{cm},m^*}^G &= \mathrm{E}\left[\kappa_{\text{cm}}^G \mid \mathbf{b}_{\text{cm}}^{n,G}, \hat{\theta}_{m^*}\right] \\ &= \mu_{\kappa_{\text{cm}}^G,m^*} + \mathbf{C}_{\kappa_{\text{cm}}^G \mathbf{b}_{\text{cm}}^G,m^*}^G \left(\mathbf{C}_{\mathbf{b}_{\text{cm}}^G,m^*}^G\right)^{-1} \\ &\quad \times \left(\mathbf{b}_{\text{cm}}^{n,G} - \mu_{\beta_{\text{cm}}^G,m^*}\right). \quad (24)\end{aligned}$$

The intuition is that, provided the components are not substantially overlapping in the feature subspace, the most likely Gaussian component will be (most probably) the correct one and the related RCM value $\hat{\kappa}_{\text{cm},m^*}^G$ will be closer to the true one, thus reducing the noise from other components. As discussed in Section III-C, the condition of less overlapping is better satisfied in higher rates and temporal levels, where the ML estimator performs better than the LMSE estimator. Moreover, for the motion-estimation-related RCMs, the ML estimator should also work better. However, whenever there is significant Gaussian component overlapping and the sequence being currently decoded is similar to the Gaussian that is less densely populated in the training set (i.e., has smaller *a priori* probability), then all the predictions may differ considerably from the true RCMs. In this case, the averaging of the LMSE estimator will yield better results. At this point, it is interesting to note the similarity of the ML prediction to the local linear regression model approach used in [6] and [7]. ML prediction disregards less significant components (which are more distant in the Mahalanobis sense), thus essentially performing a prediction using a more localized region of the RCM feature space. In this sense, it is quite similar to the local regression scheme of [6] and [7], albeit the different sampling conditions prevent us from expecting similar prediction results.

B. Adaptive Online Density Re-Estimation

Since video statistics can be extremely diverse, we examine the possibility of enhancing the prediction accuracy of the proposed complexity prediction system by adaptively switching between the offline GMM and the re-estimation of the joint RCM complexity feature PDF. This would be very useful for

TABLE V

RESULTS FROM ORACLE PREDICTION: MEAN RELATIVE PREDICTION ERROR [PERCENTAGE—AS DEFINED IN (10)] PER TEMPORAL LEVEL AND SEQUENCE FOR ED_tics

ED_tics	Rate = 384 kbps					Rate = 1536 kbps				
	tlev = 5, 4	**tlev = 3**	**tlev = 2**	**tlev = 1**	**Total$_1$**	**tlev = 5, 4**	**tlev = 3**	**tlev = 2**	**tlev = 1**	**Total$_1$**
Stefan	6.81	11.36	9.06	7.55	**6.40**	12.36	14.77	16.07	19.66	**13.37**
Silence	2.50	5.92	5.90	7.28	**4.35**	1.03	9.39	4.13	12.09	**6.20**
Sailor-men	4.27	7.91	8.00	5.89	**2.97**	4.15	11.85	7.25	16.20	**6.85**
City	3.75	4.62	7.51	4.91	**2.93**	4.57	5.07	5.47	5.01	**3.07**
Raven	9.94	10.07	7.06	7.33	**3.78**	8.81	5.71	8.24	9.08	**6.60**
Football	11.98	10.20	8.71	7.77	**6.17**	19.15	9.66	8.78	8.56	**8.11**
Coast-guard	4.48	7.53	6.70	6.23	**3.95**	5.07	6.59	5.94	9.42	**4.21**
Paris	2.63	3.80	11.37	7.20	**5.74**	0.91	8.44	2.34	17.28	**8.77**
Harbour	3.91	4.61	5.71	5.31	**2.18**	6.23	5.83	8.51	13.20	**6.81**
Total$_2$	**5.59**	**7.34**	**7.78**	**6.61**	**4.27**	**6.92**	**8.59**	**7.41**	**12.28**	**7.11**

sequences that predominantly fail to fit in the existing training set, i.e., are "atypical."

For each GOP n, with $n > 1$, we perform this process as follows. For every new RCM measurement that comes from the decoder, we update the RCM feature mean and covariance matrices estimated from the current video sequence in all temporal levels (online Gaussian component estimation). Let $\hat{\theta}^{n-1}_{\text{online}}$ denote the Gaussian component corresponding to the online mean and covariance matrices calculated from GOP $1, \ldots, n-1$. In parallel with the prediction procedures described above, we perform LMSE prediction of the current GOP RCMs using only $\hat{\theta}^{n-1}_{\text{online}}$ and the current GOP complexity feature $\mathbf{b}^{n,G}_{\text{cm}}$. After we receive the decoder's feedback, we compare the prediction error between the "online mode" and the "Offline-trained GMM mode." Based on the comparison, we select the way to determine which mode—online or GMM—we will use for the next prediction

$$\begin{aligned}&\{\text{active\_mode}\}^{n+1} = \arg\min_{\text{mode}} \left(\text{err}^n_{\text{mode}}\right)\\ &\text{mode} \in \{\text{online, offline-GMM}\}\,.\end{aligned} \tag{25}$$

In order to discourage frequent mode switching due to small noisy variations of the prediction error, we include a "bias" scaling factor to the error of the active prediction mode, i.e.,

$$\text{err}^n_{\text{mode}} = x_{\text{mode}} \cdot \left(k^{n,G}_{\text{cm}} - \left\{\hat{k}^{n,G}_{\text{cm}}\right\}_{\text{mode}}\right)^2 \tag{26}$$

with $x_{\text{mode}} = 1.0$ if mode $\neq \{\text{active\_mode}\}^n$ and $0.0 < x_{\{\text{active\_mode}\}^n} < 1.0$. After defining the "bias" scaling factor $x_{\{\text{active\_mode}\}^n}$ empirically based on preliminary simulation results, it was kept constant during our experimentation with the proposed approach.

VII. SUMMARY OF PROPOSED PREDICTION ALGORITHMS

In Fig. 3 we show the pseudo-code for the combined Markov-1–re-estimation algorithm, which is the most extended version of the algorithms we propose. The rest of the algorithms can be derived as follows. For Markov-1: Skip step 6 and 8b. Also skip the error calculation and the complexity prediction using the Re-estimation part. For LMSE: As above, plus skip the GMM model online measurement and update altogether (steps 4 and 8). Without online model updates Markov-1 becomes LMSE and can be used when no online feedback is available. For Max-likelihood variants: In any of the previous variants, substituting (24) for (8) in the complexity prediction (step 7) yields the corresponding ML variants. Experiments are detailed in the following section, where we examine the prediction accuracy based on the adaptive re-estimation of the PDF model versus the offline-GMM based prediction.

VIII. EXPERIMENTAL VALIDATION

We utilized a scalable video coder [17] that incorporates a variety of advanced motion prediction tools found in state-of-the-art standardized video coders. The decoding algorithm was implemented in platform-independent optimized C code and executed in an Intel Core Duo processor within the Windows XP operating system. Profiling of the execution time for each module was done using the built-in processor counters [22].

The utilized sequences for our experiments were all Common Interchange Format (CIF) resolution video clips with 30 frames/s replay rate. With all the advanced coding options enabled (long temporal filters, multihypothesis motion-compensated prediction and update, etc.), real-time decoding was not possible without platform-dependent software optimization. Hence, similar to prior work [9], [11], [12], [18], we resorted to simulation-based results.

A. Impact of Module-Specific RCMs in the Overall Decoding Complexity

Table VI shows the percentile of the execution time of each module for reconstructing four typical video sequences used in our experiments. Results for three indicative bit rates are provided. The results indicate that the importance of each module in the resulting execution time varies depending on the decoding bit rate and the sequence used. Clearly, motion compensation and fractional-pixel interpolation appear to be the dominant components in the decoder's complexity.

B. Comparisons of Proposed Prediction Schemes

In this section we are comparing the proposed methods among each other, and also with respect to the results obtained

TABLE VI

PERCENTILE OF THE EXECUTION TIME ATTRIBUTED TO EACH MODULE FOR DECODING FOUR INDICATIVE VIDEO SEQUENCES. THE MOTION COMPENSATION AND FRACTIONAL PIXEL INTERPOLATION WHERE AVERAGED OVER ALL BIT RATES SINCE THEY DO NOT VARY WITH THE DECODING BIT RATE ($\sim$ 1% VARIATION)

Bit rate (kbps)	Tempete		Foreman		Mobile		News	
	Entropy decoding	Inverse transform	Entropy decoding	Inverse transform	Entropy decoding	Inverse transform	Entropy decoding	Inverse transform
384	2.17	3.58	2.14	3.42	1.67	2.99	7.26	11.40
1024	3.08	3.54	3.71	3.52	2.62	3.01	10.68	10.51
1536	3.88	3.56	3.72	3.52	3.15	3.09	12.05	10.52
Average:	**3.04**	**3.56**	**3.19**	**3.49**	**2.48**	**3.03**	**10.00**	**10.81**
Motion parameters	Motion compensation	Fract.-pixel interpolation	Motion compensation	Fract.-pixel interpolation	Motion compensation	Fract.-pixel interpolation	Motion compensation	Fract.-pixel interpolation
	60.99	**32.41**	**60.63**	**32.69**	**61.81**	**32.68**	**52.26**	**26.93**

```
1.   While (∃ more frames to decode)–Iteration $m$
2.     ∀ module cm ∈ {ED_tics, IT_tics, MC_tics, FI_tics}
3.       ∀ tlev
4.         Measurement: Get the decoding measurement $k_{\mathrm{cm}}^{m-1}$ from the previous prediction unit (frame $m - 1$).
5.         Error calculation: Calculate with (26) the weighted errors between the previous frame measurement and the
           predictions from Markov-1 and Re-estimation algorithms
6.         Algorithm selection: Determine which algorithm/model to use for current frame prediction using (25).
7.         Complexity prediction: Calculate the new complexity estimate $\hat{k}_{\mathrm{cm}}^{m}$ using the LMSE predictor (24) for both
           models (GMM updated with "Markov-1" and single Gaussian updated with "Re-estimation"). Output the value
           for the selected model to the resource management system.
8.         Models updates: Using measurement $k_{\mathrm{cm}}^{m-1}$:
8a.        a) Markov-1 update: update the Gaussian component probabilities (mixing coefficients of the GMM) for the
           next prediction cycle using (21) and (22),
8b.        b) Adaptive PDF re-estimation: update the current sequence mean and covariance.
```

Fig. 3. Pseudo-code for the complete set of proposed prediction algorithms.

from a state-of-the-art approach for video decoding complexity prediction based on linear regression, that was proposed in the relevant literature [16], [19]. The comparison includes our main proposals, i.e., the LMSE-based prediction with and without feedback and the adaptive PDF re-estimation.[7]

We present experimental prediction results for the various RCMs using the schemes proposed in this paper. We denote as "LMSE" the baseline LMSE estimation algorithm [i.e., using (8), (9) and only offline training] and as "Markov-1" the Markov-enhanced LMSE estimation that adapts the weights of the mixture model used in the predictor based on the feedback received from the decoder (Section V-C). Finally, we denote as "Re-est" the method that performs PDF re-estimation based on online feedback from the decoder (Section VI-B). The results are given in Tables VII–IX. The set of video sequences mentioned in Section V-C was used as the training set for our offline GMM and state transition probabilities, while the sequences "Tempete," "Foreman," "Mobile," and "News" comprised the test set.

Starting with the most-important part, i.e., MC_tics, notice that the offline "LMSE" method performs surprisingly well in comparison to the other approaches that require online feedback from the decoder, with the exception of the "News" sequence. The complexity prediction for the fractional-pixel interpolation FI_tics is also quite accurate in the "LMSE" method, but the "Re-est" method performs better in general. Regarding the entropy decoding and inverse transform complexities, i.e., ED_tics and IT_tics, the "Re-est" method is again a superior predictor. This indicates that a combination of the different approaches could be beneficial, and it would also not require online feedback for all the RCMs of the previously decoded GOP.

We also compare against the regression-based approach proposed in previous work [16], [19], which was shown to

[7] The proposed ML approach of Section VI-A produced similar results to the module-specific linear regression approach (adapted from [16]) and hence its results are omitted for brevity of description. However, Section VI-A is relevant to the overall scope of our work and shows how different variants of the proposed approach can derive similar schemes to other methods proposed in the literature.

TABLE VII
AVERAGE RELATIVE PREDICTION ERROR OF THE "LMSE" ALGORITHM (PERCENTAGE)

Bit rate (kbps)	Tempete		Foreman		Mobile		News	
	ED_tics	IT_tics	ED_tics	IT_tics	ED_tics	IT_tics	ED_tics	IT_tics
384	24.89	6.29	11.32	3.93	10.12	2.14	25.44	7.58
512	24.27	4.53	7.13	4.92	4.75	3.20	18.49	2.66
896	5.42	2.98	11.60	2.44	13.56	2.45	19.30	1.86
1024	5.95	2.76	22.93	4.07	4.47	3.61	13.41	1.99
1280	5.34	3.85	7.46	2.48	4.43	3.37	17.97	3.22
1536	10.52	4.95	7.57	3.56	6.09	3.56	18.33	5.00
Average:	**12.73**	**4.23**	**11.34**	**3.57**	**7.24**	**3.06**	**18.82**	**3.72**
Motion parameters	MC_tics	FI_tics	MC_tics	FI_tics	MC_tics	FI_tics	MC_tics	FI_tics
	3.38	**10.90**	**6.67**	**4.36**	**12.52**	**1.63**	**28.57**	**19.40**

TABLE VIII
AVERAGE RELATIVE PREDICTION ERROR OF THE MARKOV-1 PREDICTION ALGORITHM (PERCENTAGE)

Bit rate (kbps)	Tempete		Foreman		Mobile		News	
	ED_tics	IT_tics	ED_tics	IT_tics	ED_tics	IT_tics	ED_tics	IT_tics
384	13.46	5.12	10.96	2.90	4.07	2.31	15.46	7.20
512	10.70	3.80	12.36	4.37	3.58	1.42	12.52	3.59
896	7.85	2.32	11.63	3.58	8.72	2.60	5.75	1.66
1024	6.37	3.55	23.02	4.74	7.20	3.62	5.88	1.28
1280	8.20	4.00	13.29	2.68	6.61	3.46	11.35	2.71
1536	10.77	4.23	10.87	2.53	6.62	3.64	10.11	3.22
Average:	**9.56**	**3.84**	**13.69**	**3.47**	**6.13**	**2.84**	**10.18**	**3.28**
Motion parameters	MC_tics	FI_tics	MC_tics	FI_tics	MC_tics	FI_tics	MC_tics	FI_tics
	11.21	**2.61**	**10.37**	**4.76**	**12.65**	**1.43**	**15.96**	**6.90**

outperform other related methods. The results can be found in Table X. This approach adapts the coefficients of a linear predictor based on linear regression with previously obtained decoding RCMs (per module). We experimented with various predictor lengths for each module of the decoder and selected (per module) the one that provided the best results on average. It can be seen that the combination of the best choices from the proposed methods systematically outperforms the regression-based approach proposed in [16] and [19], with only a few exceptions. Finally, we remark that, on condition of the GMM accuracy on RCM feature modeling, the proposed methods consist of the optimal predictor in the MSE sense. This is in contrast to previously proposed schemes [7], [13], [19] that perform complexity prediction based on heuristic predictors without any guarantee of optimality.

C. Selective Bit Stream Decoding Based On Complexity Estimates

We conclude our experimental investigations by demonstrating an application of the proposed complexity modeling and estimation. One missing aspect of conventional rate-distortion modeling for video coders is a model estimating the complexity associated with processing of a certain component of a compressed bit stream. Our approach consists of such a framework since it provides reliable estimates of the associated cost of decoding a certain component of the compressed bit stream. As such, the streaming server of a video session may selectively decide to omit certain parts of the compressed stream from the transmission in order to satisfy real-time constraints of a certain decoder, or simply to reduce the transmission overhead, since the complexity model predicts that the decoder will exceed the real-time decoding constraints if these parts are used. This was initially termed "complexity-constrained bitstream shaping" in our previous work [7]. Notice that our approach that provides module-by-module real complexity metrics is very suitable for this task. In addition, the use of a scalable bitstream ensures that decoding occurs even if certain texture or motion (motion vectors or interpolation information) is omitted.

We set two upper bounds on the processing cycles (expressed in tics/pixel) permissible for the video decoding of each GOP (which corresponds to approximately 0.53 s under the utilized settings) and two corresponding upper bounds on the average bit rate per GOP. Then, based on the proposed LMSE prediction algorithm we estimate the decoding time per temporal level and per module. We couple these measurements with a distortion model for the impact of each component of each temporal level [6]. Then, based on the rate-distortion complexity optimization algorithm of our previous work (Fig. 3 of [6]) we selectively drop certain substreams for each temporal level in order to satisfy the total constraints set for each GOP (rate and complexity). For each texture or motion vector bitstream corresponding to each

TABLE IX

AVERAGE RELATIVE PREDICTION ERROR OF THE "RE-EST" ALGORITHM USING ADAPTIVE ONLINE PDF RE-ESTIMATION (PERCENTAGE)

Bit rate (kbps)	Tempete		Foreman		Mobile		News	
	ED_tics	IT_tics	ED_tics	IT_tics	ED_tics	IT_tics	ED_tics	IT_tics
384	8.51	5.22	8.30	5.86	3.92	3.56	11.56	8.53
512	7.56	5.33	6.88	5.97	3.45	4.25	9.81	4.11
896	5.19	2.69	5.24	4.03	5.06	3.70	6.65	3.39
1024	4.31	3.98	12.60	4.87	4.77	4.31	7.96	2.57
1280	4.41	4.21	3.82	3.07	4.29	2.76	8.33	3.58
1536	5.27	5.14	2.97	2.46	4.19	4.49	7.69	4.70
Average:	**4.24**	**2.81**	**6.04**	**3.27**	**3.24**	**3.47**	**5.00**	**3.09**
Motion parameters	MC_tics	FI_tics	MC_tics	FI_tics	MC_tics	FI_tics	MC_tics	FI_tics
	6.16	**2.36**	**7.20**	**3.98**	**11.27**	**2.42**	**10.51**	**6.61**

TABLE X

AVERAGE PREDICTION ERROR OF THE LINEAR REGRESSION METHOD USED IN PREVIOUS WORK ON DECODING COMPLEXITY PREDICTION [16], [19]

Bit rate (kbps)	Tempete		Foreman		Mobile		News	
	ED_tics	IT_tics	ED_tics	IT_tics	ED_tics	IT_tics	ED_tics	IT_tics
384	6.44	5.60	7.48	6.24	4.99	7.07	16.87	24.91
512	5.15	3.47	7.80	7.16	5.68	8.73	8.12	5.90
896	5.01	1.05	4.36	3.49	6.34	4.41	8.97	4.05
1024	5.06	4.28	22.93	5.10	6.68	6.60	10.25	4.56
1280	3.72	2.47	3.51	3.37	7.05	5.69	9.63	3.81
1536	4.06	5.17	2.58	2.23	6.27	5.46	5.59	5.27
Average:	**4.91**	**3.67**	**8.11**	**4.60**	**6.17**	**6.33**	**9.91**	**8.08**
Motion parameters	MC_tics	FI_tics	MC_tics	FI_tics	MC_tics	FI_tics	MC_tics	FI_tics
	12.41	**2.88**	**10.03**	**3.57**	**15.23**	**3.13**	**12.16**	**6.12**

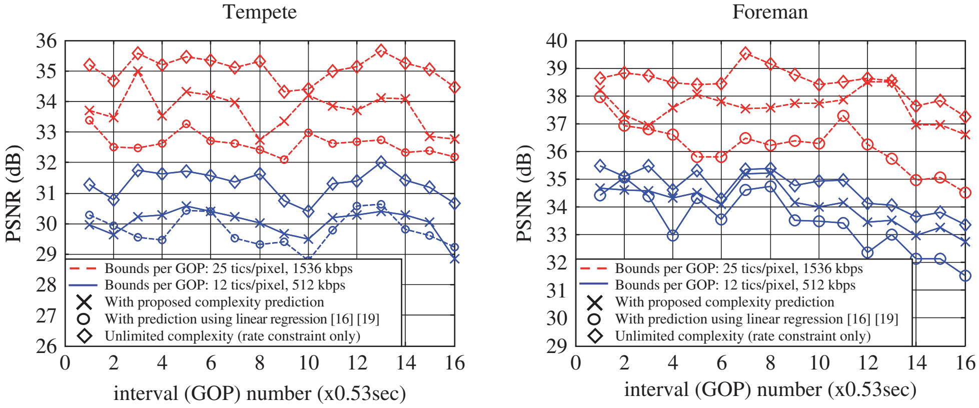


Fig. 4. Rate-distortion complexity optimized decoding. Two complexity and rate upper bounds were set for each GOP (interval of 0.53 s). In all cases we use the model-derived estimates for the expected number of cycles of each operational setting.

video frame, the scalable bitstream parser can select from the truncation points corresponding to the bitrates reported in Tables VII–X (if under the constraint set per GOP). From the obtained complexity-distortion set of points, we select the one closest to the complexity constraint that has the highest PSNR estimate (based on the distortion model).

Representative average peak-signal-to-noise (PSNR) results are presented in Fig. 4 for the different bounds set for the processing cycles. As a reference, we also include the corresponding results when using the linear regression method for complexity estimation [16], [19] (corresponding prediction results can be found in Table X). In addition, the results when complexity is unconstrained are included in the figure to provide an upper bound. Overall, there is a progressive decrease in quality when selective components are removed based on the distortion-complexity estimates. The results of Fig. 4 demonstrate that the proposed complexity prediction approach enables higher PSNR under the complexity bounds than the case where linear regression is used, since, on average, operating points that are closer to the bound are selected. This brings the results of the proposed approach closer to the upper bound that performs only rate-distortion optimization without complexity constraints. We observed similar improvements offered by the proposed approach when the Re_est complexity prediction algorithm was used.

IX. Conclusion

In this paper, we have proposed and discussed several statistical modeling and prediction schemes for video decoding complexity based on module-specific execution times (which we term as real complexity metrics—RCMs) and easily extractable complexity features. We have shown that there is significant self- and cross-clustering in the temporal-level partitioned RCM complexity feature domain for many video sequences (Fig. 2). The clustering is highly correlated to the content of the video sequences and can be successfully used to engineer optimal and effective prediction algorithms.

References

[1] D. Zhong, H. J. Zhang, and S.-F. Chang, "Clustering methods for video browsing and annotation," in *Proc. SPIE Storage and Retrieval for Image and Video Databases*, vol. 2670. 1996, pp. 239–246.

[2] Y. Xu, E. Saber, and A. M. Tekalp, "Object segmentation and labeling by learning from examples," *IEEE Trans. Image Process.*, vol. 12, no. 6, pp. 627–638, Jun. 2003.

[3] F. Wu, S. Li, R. Yaw, X. Sun, and Y.-Q. Zhang, "Efficient and universal scalable video coding," in *Proc. IEEE Int. Conf. Image Process., (ICIP)*, vol. 2. 2002, pp. 37–40.

[4] Q. Zhang, *et al.*, "Power-minimized bit allocation for video communication over wireless channels," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 6, pp. 398–410, Jun. 2002.

[5] 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.

[6] 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.

[7] 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.

[8] I. Ahmad, S. M. Akramullah, M. L. Liu, and M. Kafil, "A scalable off-line MPEG-2 video encoding scheme using a multiprocessor system," *Elsevier Parallel Computing*, vol. 27, no. 6, pp. 823–846, May 2001.

[9] J. Valentim, *et al.*, "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.

[10] 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.

[11] J. Ostermann, *et al.*, "Video coding with H.264/AVC: Tools, performance, complexity," *IEEE Circ. Syst. Mag.*, vol. 4, no. 1, pp. 7–28, Jan. 2004.

[12] S. Regunathan, *et al.*, "A generalized video complexity verifier for flexible decoding," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, vol. 3. Sep. 2003, pp. 289–292.

[13] J. Liang, *et al.*, "Adaptive multi-resource prediction in distributed resource sharing environment," in *Proc. IEEE Int. Symp. Cluster Comput. Grid, (CCGrid-04)*, Apr. 2004, pp. 293–300.

[14] H. J. Stolberg, *et al.*, "A platform-independent methodology for performance estimation of multimedia signal processing applications," *J. VLSI Signal Process.*, vol. 41, no. 2, pp. 139–151, 2005.

[15] 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.

[16] W. Yuan, K. Nahrstedt, S. V. Adve, D. L. Jones, and R. H. Kravets, "GRACE-1: Cross-layer adaptation for multimedia quality and battery life," *IEEE Trans. Mobile Comput.*, vol. 5, no. 7, pp. 799–815, Jul. 2006.

[17] 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.*, vol. 19, no. 7, pp. 653–673, Aug. 2004.

[18] M. Horowitz, *et al.*, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 704–716, Jul. 2003.

[19] W. Yuan and K. Nahrstedt, "Energy-efficient CPU scheduling for multimedia applications," *ACM Trans. Comput. Syst.*, vol. 24, no. 3, pp. 292–331, Aug. 2006.

[20] D. Turaga, *et al.*, "Complexity scalable motion compensated wavelet video encoding," *IEEE Trans. Circ. Syst. Video Technol.*, vol. 15, no. 8, pp. 982–993, Aug. 2005.

[21] J. R. Ohm, *Multimedia communication technology, in Series: Signals and Commun. Technol.*, 1st ed. New York: Springer-Verlag, 2004.

[22] Intel Corp. Intel VTune® data collector enabling kit for I/O Processors. *Application Note 273892-001* [Online]. Available: http://developer.intel.com/design/iio/applnots/27389201.pdf accessed June 2009.

[23] A. H. Sayed, *Fundamentals of Adaptive Filtering*. New York: Wiley, 2003.

[24] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.

[25] T. Hastie, "The elements of statistical learning," in *Series: Statistics*, New York: Springer-Verlag, 2001.

[26] C. Fraley and A. E. Raftery, "How many clusters? Which clustering method? Answers via model-based cluster analysis," *Comput. J.*, vol. 41, no. 8, pp. 578–588, 1998.

Nikolaos Kontorinis (S'03) received the Diploma in electrical engineering from the University of Patras, Greece, in 2002, and the M.Sc. degree in signal processing from the University of California, Los Angeles (UCLA), in 2004. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering, UCLA.

His research interests include statistical modeling for power-aware video coding systems, machine learning techniques for video applications, and financial models in networked video transceivers.

Mr. Kontorinis was a recipient of the Alexander S. Onassis Public Benefit Foundation Scholarship for Greeks studying abroad in 2004–2006.

Yiannis Andreopoulos (S'01–M'04) received the Ph.D. degree in applied sciences from the Vrije Universiteit Brussel, Brussels, Belgium, in 2005, as well as the M.Sc. degree in signal and image processing systems and the Electrical Engineering Diploma, both from the University of Patras, Patras, Greece.

He is presently Lecturer at the Department of Electronic and Electrical Engineering, University College London, London, U.K. His research interests are in the domain of multimedia signal processing.

Mihaela van der Schaar (SM'04) received the M.Sc. and Ph.D. degrees in electrical engineering from Eindhoven University of Technology, Eindhoven, the Netherlands.

He is currently an Associate Professor in the Department of Electrical Engineering, University of California, Los Angeles (UCLA), where she leads the Multimedia Communications and Systems Laboratory.