Rate-Distortion and Complexity Optimized Motion Estimation for H.264 Video Coding

Hasan F. Ates, *Member, IEEE,* and Yucel Altunbasak, *Senior Member, IEEE*

***Abstract*—H.264 video coding standard supports several inter-prediction coding modes that use macroblock partitions with variable block sizes. Rate-distortion optimal selection of both the motion vectors and the coding mode of each macroblock is essential for an H.264 encoder to achieve superior coding efficiency. Unfortunately, searching for optimal motion vectors of each possible subblock incurs a heavy computational cost. In this paper, in order to reduce the computational burden of integer-pel motion estimation without sacrificing from the coding performance, we propose a rate-distortion and complexity joint optimization framework. Within this framework, we develop a simple method that determines for each macroblock which partitions are likely to be optimal. Motion vector search is carried out for only the selected partitions, thus reducing the complexity of the motion estimation step. The mode selection criteria is based on a measure of spatio-temporal activity within the macroblock. The procedure minimizes the coding loss at a given level of computational complexity either for the full video sequence or for each single frame. For the latter case, the algorithm provides a tight upper bound on the worst case complexity/execution time of the motion estimation module. Simulation results show that the algorithm speeds up integer-pel motion estimation by a factor of up to 40 with less than 0.2 dB loss in coding efficiency.**

***Index Terms*—video coding, H.264, motion estimation, complexity, rate-distortion, optimization.**

I. INTRODUCTION

It has been shown that H.264 video coding standard achieves considerably higher coding efficiency over such previous standards as H.263 and MPEG-4 Visual [1]. One of the novelties contributing to H.264's superior performance is a rich set of coding modes to choose from for each macroblock (MB) [2]. These modes allow the encoder to try different MB partitions, multiple reference frames and inter/intra prediction methods in order to find a rate-distortion (R-D) optimal coding strategy. Unfortunately, evaluating the coding performance of all these modes incurs a substantial increase in the computational complexity of the encoder.

Motion estimation (ME) and mode decision (MD) are the most computationally demanding modules of an H.264 encoder. When a MB is partitioned into several smaller subblocks, rate-distortion optimal mode decision requires estimating the optimal motion vectors (MVs) of each such subblock. Hence, each separate MB partition brings additional complexity to the ME module. The H.264 coding standard [3] allows MBs (16×16 pixel) to be divided into two 16×8 or two 8×16 or four 8×8 subblocks. Each 8×8 block can be further partitioned into 8×4, 4×8 and 4×4 subblocks. Therefore, in H.264, a naive full-search method (FSM) will require significantly higher number of computations than the single block type case.

In recent years, there have been many attempts at developing fast ME algorithms for the H.264 encoder. Some of these work adapt the ideas previously developed for other standards and for a single block size (i.e. 16×16) to the multiple block sizes of H.264 [4], [5], [6], [7]. Early termination techniques have also been investigated to speed up the ME process for each partition [8], [9], [10]. Different search patterns, such as multi-directional search pattern (MDRPS) [11], adaptive hexagon-based search (AHBS) [5], modified diamond search pattern (DSP) [6], have been designed to increase the ME accuracy. In most cases, ME algorithms designed for H.264 tend to focus on each block size separately, and perform fast ME for each partition [4], [5], [12], [13]. For instance, EPZS [4] uses special search patterns around block specific MV predictions for estimating the optimal MVs of each MB partition. This approach could provide satisfactory speed-up factors in each partition. However, since separate MV searches are employed for different block sizes, the overall computational cost could still be significant.

Since the encoder eventually needs the optimal MVs of only one partition, i.e. the optimal partition, having to search for the MVs of other partitions seems like a waste of encoder time. However, it is difficult to determine which partition(s) is(are) likely to be optimal without performing ME first. There exist many methods that try to eliminate certain modes before or during the ME phase [14], [15], [16], [17], [18], [19], [20]. The eliminated modes are excluded from the ME and/or MD steps. The mode selection is based on various measures, such as the MB variance [21], absolute frame difference [22], a measure of local motion content and MV statistics [23], [24], [25], edge information [14], partially completed ME results [26], [27], [28], [29], and so forth. In these methods, MBs are basically classified as high or low-detailed, and low-detailed MBs are assumed to have no gain from the use of small-sized partitions. Despite some success, these ideas do not provide a justification of whether the choices made during the mode selection process are meaningful both from a R-D efficiency and also a computational complexity point of view. In other words, it is generally not clear how different parameters of these algorithms affect both the coding performance and the execution time of the encoder.

For real-time video coding applications, it is important to be

This research was supported by the TÜBİTAK Grant 104E125.

H.F. Ates is with the Department of Electronics Engineering, Işık University, Şile, 34980 İstanbul, TURKEY (e-mail: hfates@isikun.edu.tr).

Y. Altunbasak is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0250 USA (e-mail: yucel@ece.gatech.edu).

able to control the complexity of the encoder without affecting the R-D performance too much. For instance, in case the ME step is taking too much time, the encoder should have a robust mechanism to speed up ME without a significant degradation in the video quality. Such a mechanism requires a thorough understanding of how different parameters of the ME module affect the R-D performance and the total execution time. In [30], the authors introduce an operational method for optimizing ME with respect to the trade-off between R-D and complexity. However, the method is not very flexible, because, to adjust ME complexity, the encoder has to choose from a limited number of best average settings.

In this paper, we extend our work in [31] and develop a theoretical framework for joint optimization of R-D performance and ME complexity. This framework is used to derive an optimal method to speed up the ME module with minimum loss in coding efficiency. The method decides for each MB which partitions are likely to be optimal and performs integer-pel MV search for only the selected block sizes. The mode decisions are based on a spatio-temporal gradient measure that gives the amount of spatial and motion activity within the MB. In simple terms, smaller partitions are preferred for MBs with significant spatial and temporal information, and larger partitions are preferred for smoother MBs with little motion. When selecting partitions to search during ME, the algorithm aims to minimize the overall R-D cost of mode estimation error for the video sequence. Therefore, a partition is skipped only if the reduction in computations justifies the possible reduction in coding efficiency.

Compared to existing mode selection methods in literature, we propose a novel way to estimate and evaluate the coding cost of ME speed-up. Except for [30], [27], none of the existing work mentioned above analyze the R-D cost of changing the heuristic parameter settings of the algorithms. Since the value of each parameter is determined through various simulations in many video sequences, all these tests would probably have to be repeated for finding the "right" set of parameters at a different level of complexity. In this paper, we address this problem and develop a general framework in which R-D costs could be minimized at any level of ME complexity.

Consequently, our R-D and complexity joint optimization framework provides the encoder with a flexible way to adjust the ME complexity while achieving a desired level of video quality. In case a time budget is set for the ME step in a single frame, the mode selection criteria is easily adapted such that ME execution finishes within the maximum allowed time budget. Most of the fast ME algorithms described above ignore the worst-case complexity of the ME module and try to minimize average complexity. However, to guarantee uninterrupted real-time operation, the encoder hardware should be designed to code frames at a specified rate even when each frame consumes the maximum amount of time during the ME process. In other words, worst-case complexity is a more critical design parameter for smooth real-time encoding than the average complexity. In this paper, we present an effective method to limit the worst-case ME execution time with as little loss of coding efficiency as possible.

The main contribution of our work is this "complexity adaptation feature" of the proposed algorithm. The method targets optimal control of frame-level ME complexity by adapting the parameters of the mode decision process appropriately. The joint optimization framework guarantees a robust and controlled adaptation procedure without having to worry about sudden and significant losses in R-D efficiency. As to our knowledge, existing methods in literature are not suitable for such complexity control, for a couple of reasons. One reason is because these algorithms do not provide a robust mechanism to change the parameter settings "on the fly" and any misguided attempt to do so may cause significant coding losses. More crucially, in most methods, such as [17], [18], [19], [20], [25], [28], the mode decisions are based on some information about the MBs that is not available *a priori*, such as the coding modes of neighboring MBs or partial ME results. This makes it hard, if not impossible, to predict and control the overall frame-level ME complexity. We overcome these two issues by adjusting our parameters within an optimized framework, and by using a simple and effective gradient measure that can be computed prior to coding a frame. In the simulations section, we compare our algorithm to a simpler heuristic approach to emphasize the importance of our optimized decision framework.

Section II describes the joint optimization framework. In Section III, we derive the optimal mode selection criteria and explain the details of the algorithm. Section IV mentions some practical issues and the estimation procedure for the algorithm parameters. Section V describes the extension of the method for adaptive control of the frame-level complexity/execution time of the ME module. In Section VI, we present the simulation results, analyze and discuss the optimality of the approach, and compare the performance of the algorithm with UMHexagonS [12] and FSM. The results show that there is negligible loss in video quality, despite a major reduction in complexity. We conclude the paper in Section VII.

II. RATE-DISTORTION AND COMPLEXITY JOINT OPTIMIZATION

Motion estimation (ME) constitutes a significant portion of total complexity in H.264 encoding. Despite the existence of several fast ME algorithms in literature, having to find the optimal motion vectors (MVs) for all possible MB partitions places a strict restriction on how much the ME complexity can be reduced. In this paper, we propose to estimate which partitions/modes are most likely to be optimal before performing any ME, and reduce the number of different partitions that have to be searched for optimal MVs.

Figure 1 provides a high-level summary of our approach. The control module collects some data, D, from the current frame, and uses this data to control the complexity of the ME module. Based on D and the desired level of ME complexity, the control module forms the optimized decision function, $\mathcal{Q}(\mathrm{D})$, and passes this information to the ME module. ME module processes each MB based on $\mathcal{Q}$; that is, $\mathcal{Q}$ tells the ME module which partitions to search for each MB. The control module optimizes its decisions based on probabilistic models

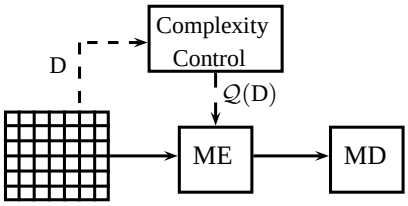


Fig. 1: Algorithm block diagram.

of computational complexity and R-D costs of sub-optimal mode decisions. In the following paragraphs, we define D, $\mathcal{Q}$, and these probabilistic models.

The data D is essential for accurate estimation of the optimal modes. As for D, we propose to use the magnitude of the spatio-temporal gradient of each MB as defined below:

$$\mathrm{D} = 2\mathrm{D}_T + 0.5\mathrm{D}_x + 0.5\mathrm{D}_y, \tag{1}$$

where

$$\mathrm{D}_T = \sum_{x=1,y=1}^{16,16} |c(x,y) - r(x,y)|, \tag{2}$$

$$\mathrm{D}_x = \sum_{x=1,y=1}^{15,16} |c(x,y) - c(x+1,y)|, \tag{3}$$

$$\mathrm{D}_y = \sum_{x=1,y=1}^{16,15} |c(x,y) - c(x,y+1)|. \tag{4}$$

Here c and r are current and previous reference frames, respectively. In the formula, the temporal gradient, D_T, is given a bigger weight than the spatial components, because simulations show that D_T is more informative when deciding which modes could be optimal. For 8×8 and smaller partitions, we also compute the spatio-temporal gradients of the 8×8 subblocks, say D^k, $1 \le k \le 4$. We claim that the spatio-temporal gradient, D, provides a reliable measure of the high-frequency details and varying motions in a block. In other words, MBs with a high value of D are more likely to contain multiple objects with different motions, hence could benefit from the use of smaller block sizes. This gradient measure is chosen for its simplicity and effectiveness, although the following framework could just as well be applied to other criteria for selecting the MB coding mode.

Having computed D and D^k's for each MB, we would like to skip integer-pel ME for partitions with low likelihood of being optimal and reduce the overall execution time of the ME module in the H.264 encoder. For reducing computational complexity with as little loss of coding efficiency as possible, we propose an efficiency - complexity (E-C) optimization method. The method decides, for each MB, which partitions are worth the amount of computations required to perform integer-pel ME. An optimal decision is made by minimizing the following average Lagrangian cost:

$$\mathcal{J}_{MB} = E_{MB} + \lambda_E C_{MB}, \tag{5}$$

where E_{MB} and C_{MB} are measures for expected loss of coding efficiency and average amount of ME computations for a MB, respectively; λ_E is an appropriate Lagrangian multiplier. In this paper, we will look at two separate but related problems:

- **Unconstrained Optimization**: Minimize $\mathcal{J}_{MB}$ for a given and fixed λ_E. This problem tries to figure out the optimal trade-off between rate-distortion performance and complexity for a given video sequence.
- **Constrained Optimization**: Minimize $\mathcal{J}_{MB}$ subject to the constraint $\sum_{i=1}^{N} C_i \le CB$, where N is the number of MBs in a frame. In this problem, we use a complexity budget, CB, to place a worst-case limit on the total ME complexity of each frame. λ_E is chosen adaptively for each frame in order to satisfy this constraint.

E_{MB} and C_{MB} are composed of separate performance terms for each possible coding mode. Imagine that seven different partitions are represented by the modes, $m_1 = 16\times16$, $m_2^1 = 16\times8$, $m_2^2 = 8\times16$, $m_3 = 8\times8$, $m_4^1 = 8\times4$, $m_4^2 = 4\times8$, $m_5 = 4\times4$. For modes $m_2^1 = 16\times8$ and $m_2^2 = 8\times16$, the gradient D does not help us discriminate which of the two modes is more likely; in other words, given D, the probability of the two modes being optimal is almost equal. That is why we consider the two modes together as one, i.e. $m_2 = (m_2^1$ or $m_2^2)$. Likewise, $m_4 = (m_4^1$ or $m_4^2)$. Note that, the directional components of the gradient, D_x and D_y, could have been considered separately to differentiate these modes. However, it turns out that this would increase the complexity of the approach and provide only a marginal gain in coding efficiency.

As for the smaller sized partitions, i.e. $j = 3, 4, 5$, each 8×8 subblock can have a different optimal partition. However, the overall partition of the MB has to be of size 8×8 and below. To represent this, we define an extra symbol m_0 which indicates that each 8×8 subblock is m_j for some $j \ge 3$:

$$\hat{m} = m_0 \Rightarrow \hat{m}^k = m_{j_k},\ \exists j_k \ge 3,\ 1 \le k \le 4, \tag{6}$$

where $\hat{m}$ represents the optimal coding mode of the MB, and $\hat{m}^k$ represent the optimal coding mode of the 8×8 subblock k, $1 \le k \le 4$.

We use the the gradients D and D^k's to evaluate each mode m_j and decide whether to perform integer-pel ME or skip this step. We define appropriate decision sets for each block size; integer-pel ME is performed only if the gradient is within this set. These sets are as follows:

$$\mathcal{Q}_j = \{\mathrm{D} : T_j^- \le \mathrm{D} \le T_j^+\}, \quad j = 1, 2. \tag{7}$$

$$\mathcal{Q}_j^k = \{\mathrm{D}^k : T_j^- \le \mathrm{D}^k \le T_j^+\}, \quad j = 3, 4, 5. \tag{8}$$

Hence, if $\mathrm{D} \notin \mathcal{Q}_j$, then partition m_j is not included in integer-pel ME. If a block size is skipped, median MV prediction of H.264, MV_{med}, is assigned as the optimal MV for subblocks of that size and sub-pel ME and mode decision are performed as before. In general words, these sets define for each partition a connected interval of gradient values in which the partition has a high likelihood of being optimal.

For 8×8 and smaller partitions, the decision to perform ME will depend on the gradient of the corresponding 8×8 subblock. This is meaningful since, within a MB, some subblocks could be detailed and some could be smooth or could belong to a static background. As a result, different partitions could be

more likely to be optimal for different subblocks. However, with the same reasoning, sometimes the MB might not require small partitioning even though one of its 8x8 subblocks could be high-detailed. For instance, if three of the subblocks belong to the static background and if the fourth subblock belongs to a slow-moving object, m_1 or m_2 could just as well be optimal. In such cases, we should be able to skip ME for all 8×8 and smaller partitions. For that purpose, we define an extra set:

$$\mathcal{Q}_0 = \{\mathrm{D} : T_0^- \leq \mathrm{D} \leq T_0^+\}. \tag{9}$$

This set will be used first to decide for the whole MB whether it is worth performing ME for any 8×8 partitioning at all (i.e. mode m_0). If the answer is yes, each 8×8 subblock will be separately handled to decide which of the three modes, $j = 3, 4, 5$, to look at.

Having defined the decision sets, we would like to compute the expected R-D loss and ME complexity based on this formulation. For the average computational cost, we define the expected complexity of each partition as follows. For $j = 1, 2$:

$$\mathcal{E}[C_j] = \int_{T_j^-}^{T_j^+} c_j(\mathrm{D}) p(\mathrm{D}) d\mathrm{D}, \tag{10}$$

where $\mathcal{E}[\cdot]$ is the expectation operator, $p(\mathrm{D})$ is the probability density function (pdf) of the MB gradient D, and $c_j(\mathrm{D})$ is the integer-pel ME complexity of mode m_j as a function of D. For $j = 3, 4, 5$, a partition is searched if both $\mathrm{D} \in \mathcal{Q}_0$ and $\mathrm{D}^k \in \mathcal{Q}_j^k$. Therefore,

$$\mathcal{E}[C_j] = \int_{T_0^-}^{T_0^+} \int_{T_j^-}^{T_j^+} c_j(\mathrm{D}^k) p(\mathrm{D}^k, \mathrm{D}) d\mathrm{D}^k d\mathrm{D}, \tag{11}$$

where $p(\mathrm{D}^k, \mathrm{D})$ is the joint pdf of D and subblock gradient D^k. Without loss of generality, we can assume that all 8×8 subblocks have the same pdf and same cost function (i.e. $p(\mathrm{D}^k, \mathrm{D})$ and $c_j(\mathrm{D}^k)$ are the same for all k, $1 \leq k \leq 4$). Therefore, the integral will be equal for all subblocks. To simplify notation, we take $c_j(\mathrm{D}^k)$ as 4 times the cost of a single subblock. As a result, for a given MB, the overall average ME complexity becomes:

$$C_{MB} = \sum_{j=1}^{5} \mathcal{E}[C_j]. \tag{12}$$

As for the loss of coding efficiency, we evaluate the cost of mode estimation error for each mode. A mode incurs a R-D penalty only if it is excluded from ME and it turns out to be the optimal mode. That is, for $j = 0, 1, 2$,

$$\begin{aligned}\mathcal{E}[E_j] &= \int_0^{T_j^-} d_j(\mathrm{D}) p(\mathrm{D}, m_j) d\mathrm{D} + \int_{T_j^+}^{\infty} d_j(\mathrm{D}) p(\mathrm{D}, m_j) d\mathrm{D} \\ &= \int_{\mathrm{D}\notin\mathcal{Q}_j} d_j(\mathrm{D}) p(\mathrm{D}, m_j) d\mathrm{D},\end{aligned} \tag{13}$$

where $p(\mathrm{D}, \hat{m})$ is the joint pdf of MB gradient D and the optimal mode $\hat{m}$ (i.e. $\hat{m} = m_j$ means optimal mode is m_j), and $d_j(\mathrm{D})$ is the R-D cost of mis-estimation for each mode as a function of D. For $j = 3, 4, 5$, we define:

$$\mathcal{E}[E_j] = \int_{T_0^-}^{T_0^+} \int_{\mathrm{D}\notin\mathcal{Q}_j} d_j(\mathrm{D}^k) p(\mathrm{D}^k, \mathrm{D}, m_j) d\mathrm{D}^k d\mathrm{D}, \tag{14}$$

For mode m_0, we have separated the two sources of R-D loss: coding loss when none of the smaller partitions are included in the MV search, $\mathcal{E}[E_0]$; and coding loss when $\mathrm{D} \in \mathcal{Q}_0$ but ME is skipped for the optimal partition of some 8×8 subblocks, $\mathcal{E}[E_j]$, $j = 3, 4, 5$.

Assuming same pdfs and same cost functions again for all subblocks, we take $d_j(\mathrm{D}^k)$, $j = 3, 4, 5$, as 4 times the R-D cost of mis-estimation in each subblock. As a result, the expected loss of R-D efficiency is given by:

$$E_{MB} = \sum_{j=0}^{5} \mathcal{E}[E_j]. \tag{15}$$

Hence the overall expression becomes:

$$\mathcal{J}_{MB} = \sum_{j=0}^{5} \mathcal{E}[E_j] + \lambda_E \sum_{j=1}^{5} \mathcal{E}[C_j]. \tag{16}$$

In this formulation, the expected complexity and the expected R-D cost of a MB/subblock is calculated independent of the (spatial and temporal) neighboring MBs/subblocks and their optimal coding modes. It is possible to extend this formulation and include both the neighboring MB/subblock information and the gradient as multiple measures that will be jointly used to better estimate the optimal mode of a MB/subblock. This requires a higher order modeling of the pdfs defined above (e.g. as a Markov process), and it is outside the scope of this paper. In Section VI, we discuss the effects of these and other simplifications on the performance of the algorithm.

III. OPTIMAL THRESHOLD SELECTION

Given the pdfs defined above, and the performance functions d_j, c_j, we would like to find the optimal values of the thresholds $T_j^{\pm}$ that are used to define the decision sets. So, we take the partial derivatives of $\mathcal{J}_{MB}$ with respect to (w.r.t.) each threshold and set to zero.

First, we take the derivative of $\mathcal{E}[C_j]$ w.r.t $T_j^{\pm}$, for $j = 1, 2$:

$$\frac{\partial \mathcal{E}[C_j]}{\partial T_j^{\pm}} = \pm c_j(T_j^{\pm}) p(T_j^{\pm}), \tag{17}$$

Likewise, for $j = 0, 1, 2$, the derivative of $\mathcal{E}[E_j]$ w.r.t. $T_j^{\pm}$ is equal to:

$$\frac{\partial \mathcal{E}[E_j]}{\partial T_j^{\pm}} = \mp d_j(T_j^{\pm}) p(T_j^{\pm}, m_j), \tag{18}$$

For $j = 3, 4, 5$, the formulation is similar, except for the dependency on $\mathcal{Q}_0$:

$$\begin{aligned}\frac{\partial \mathcal{E}[C_j]}{\partial T_j^{\pm}} &= \pm c_j(T_j^{\pm}) \int_{T_0^-}^{T_0^+} p(T_j^{\pm}, \mathrm{D}) d\mathrm{D} \\ &= \pm c_j(T_j^{\pm}) p(T_j^{\pm} | \mathrm{D} \in \mathcal{Q}_0) P(\mathrm{D} \in \mathcal{Q}_0), \quad (19)\end{aligned}$$

and likewise,

$$\frac{\partial \mathcal{E}[E_j]}{\partial T_j^{\pm}} = \mp d_j(T_j^{\pm}) p(T_j^{\pm}, m_j | \mathrm{D} \in \mathcal{Q}_0) P(\mathrm{D} \in \mathcal{Q}_0), \quad (20)$$

where $p(\mathrm{D}^k | \mathrm{D} \in \mathcal{Q}_0)$ and $p(\mathrm{D}^k, \hat{m}^k | \mathrm{D} \in \mathcal{Q}_0)$ are the conditional pdf of subblock gradient D^k and the conditional joint pdf of D^k and optimal subpartition mode $\hat{m}^k$, given that $\mathrm{D} \in \mathcal{Q}_0$, respectively.

Finally, we also need the partial derivatives of $\mathcal{E}[C_j]$ and $\mathcal{E}[E_j]$ w.r.t $T_0^{\pm}$, for $j = 3, 4, 5$. Taking the partial derivatives of equations (11) and (14) w.r.t. $T_0^{\pm}$, we arrive at,

$$\begin{aligned}\frac{\partial \mathcal{E}[C_j]}{\partial T_0^{\pm}} &= \pm \bar{c}_j(T_0^{\pm}) P(\mathrm{D}^k \in \mathcal{Q}_j^k, T_0^{\pm}), \quad (21) \\ \frac{\partial \mathcal{E}[E_j]}{\partial T_0^{\pm}} &= \pm \bar{d}_j(T_0^{\pm}) P(\mathrm{D}^k \notin \mathcal{Q}_j^k, T_0^{\pm}, m_j) \quad (22)\end{aligned}$$

where, $P(\mathrm{D}^k \in \mathcal{Q}_j^k, T_0^{\pm})$ is the probability that $\mathrm{D}^k \in \mathcal{Q}_j^k$ and $\mathrm{D} = T_0^{\pm}$, and, for $j = 3, 4, 5$,

$$\begin{aligned}\bar{c}_j(T_0^{\pm}) &= \int_{T_j^-}^{T_j^+} c_j(\mathrm{D}^k) \frac{p(\mathrm{D}^k, T_0^{\pm})}{P(\mathrm{D}^k \in \mathcal{Q}_j^k, T_0^{\pm})} d\mathrm{D}^k, \quad (23) \\ &= \mathcal{E}[c_j(\mathrm{D}^k) \mid \mathrm{D}^k \in \mathcal{Q}_j^k, T_0^{\pm}] \quad (24)\end{aligned}$$

and similarly,

$$\bar{d}_j(T_0^{\pm}) = \mathcal{E}[d_j(\mathrm{D}^k) \mid \mathrm{D}^k \notin \mathcal{Q}_j^k, T_0^{\pm}, m_j] \quad (25)$$

Now, we use these results to derive the optimality conditions for the thresholds, $T_j^{\pm}$. For $j = 1, 2$, setting the derivative of the Lagrangian cost to zero, we get:

$$\frac{\partial \mathcal{J}_{MB}}{\partial T_j^{\pm}} = \mp d_j(T_j^{\pm}) p(T_j^{\pm}, m_j) \pm \lambda_E c_j(T_j^{\pm}) p(T_j^{\pm}) = 0, \quad (26)$$

Therefore,

$$\frac{p(T_j^{\pm}, m_j)}{p(T_j^{\pm})} = P(m_j | T_j^{\pm}) = \lambda_E \frac{c_j(T_j^{\pm})}{d_j(T_j^{\pm})}, \quad (27)$$

where $P(m_j | T_j^{\pm})$ is the conditional probability of optimal mode being m_j given that $\mathrm{D} = T_j^{\pm}$. The solutions of this equation give the extremum points of $\mathcal{J}_{MB}$.

For the optimal thresholds $T_j^{\pm *}$ to minimize $\mathcal{J}_{MB}$, we also need to check the second order derivative and see that:

$$\left. \frac{\partial^2 \mathcal{J}_{MB}}{\partial {T_j^{\pm}}^2} \right|_{T_j^{\pm} = T_j^{\pm *}} > 0. \quad (28)$$

Assuming $\frac{\partial (c_j / d_j)}{\partial T_j^{\pm}} \approx 0$ (see Section IV), we can show that this is true as long as:

$$\begin{aligned}\left. \frac{\partial P(m_j | T_j^-)}{\partial T_j^-} \right|_{T_j^- = T_j^{-*}} &> 0, \quad (29) \\ \left. \frac{\partial P(m_j | T_j^+)}{\partial T_j^+} \right|_{T_j^+ = T_j^{+*}} &< 0. \quad (30)\end{aligned}$$

From Figures 3 and 4, we realize that these two conditions are satisfied.

Likewise, the optimal solutions for $j = 3, 4, 5$, satisfy:

$$P(m_j | T_j^{\pm}, \mathrm{D} \in \mathcal{Q}_0) = \lambda_E \frac{c_j(T_j^{\pm})}{d_j(T_j^{\pm})}, \quad (31)$$

For $T_0^{\pm}$, we derive $\frac{\partial \mathcal{J}_{MB}}{\partial T_0^{\pm}}$ and set it equal to zero:

$$\begin{aligned}0 = \mp d_0(T_0^{\pm}) p(T_0^{\pm}, m_0) \pm \sum_{j=3}^{5} \bar{d}_j(T_0^{\pm}) P(\mathrm{D}^k \notin \mathcal{Q}_j^k, T_0^{\pm}, m_j) \\ \pm \lambda_E \sum_{j=3}^{5} \bar{c}_j(T_0^{\pm}) P(\mathrm{D}^k \in \mathcal{Q}_j^k, T_0^{\pm}). \quad (32)\end{aligned}$$

Notice that, $\hat{m}^k = m_j, j \geq 3 \Rightarrow \hat{m} = m_0$. Therefore, we can write,

$$\begin{aligned}p(\mathrm{D}^k, T_0^{\pm}) &= p(\mathrm{D}^k | T_0^{\pm}) p(T_0^{\pm}), \quad (33) \\ p(\mathrm{D}^k, T_0^{\pm}, m_j) &= p(\mathrm{D}^k, m_j | T_0^{\pm}, m_0) p(T_0^{\pm}, m_0) \quad (34)\end{aligned}$$

Here, m_j represents that $\hat{m}^k = m_j$, and likewise m_0 means $\hat{m} = m_0$. Consequently, Eqn. (32) becomes,

$$P(m_0 | T_0^{\pm}) = \lambda_E \frac{\bar{c}_0}{\bar{d}_0}, \quad (35)$$

where

$$\bar{c}_0 = \sum_{j=3}^{5} \bar{c}_j(T_0^{\pm}) P(\mathrm{D}^k \in \mathcal{Q}_j^k | T_0^{\pm}) \quad (36)$$

$$\bar{d}_0 = d_0(T_0^{\pm}) - \sum_{j=3}^{5} \bar{d}_j(T_0^{\pm}) P(\mathrm{D}^k \notin \mathcal{Q}_j^k, m_j | T_0^{\pm}, m_0). \quad (37)$$

Here $\bar{c}_0$ and $\bar{d}_0$ can be interpreted as the average complexity and average cost of estimation error for mode m_0, respectively. $\bar{d}_0$ represents the difference in coding loss between not searching any of the smaller partitions vs. making individual decisions for each 8×8 subblock. If we hadn't used the gradient measures of 8×8 subblocks; that is, if we had searched for all small partitions whenever $\mathrm{D} \in \mathcal{Q}_0$, then simply $\bar{c}_0 = \sum_{j=3}^{5} \bar{c}_j(T_0^{\pm})$ and $\bar{d}_0 = d_0(T_0^{\pm})$.

IV. Parameter Estimation

Solutions of the equations (27), (31), (35) yield the optimal thresholds that define the decision sets and minimize the Lagrangian cost $\mathcal{J}_{MB}$. In order to solve these equations, we need to know, for each partition, the complexities c_j, the R-D costs d_j, and the conditional distributions $P(m_j | T_j^{\pm})$, for $j = 0, 1, 2$, and $P(m_j | T_j^{\pm}, \mathrm{D} \in \mathcal{Q}_0)$, for $j = 3, 4, 5$. In the above formulation, these parameters are defined as functions

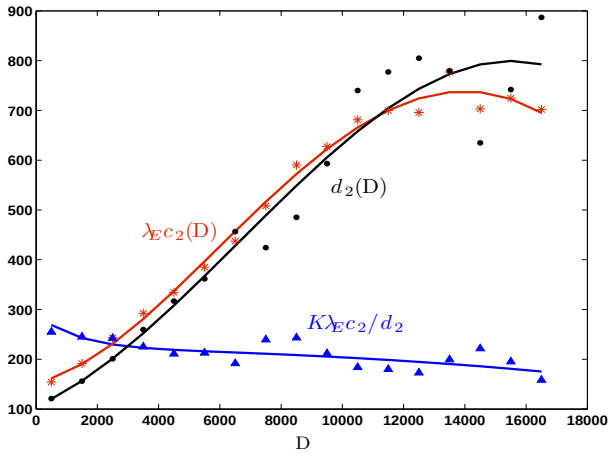


Fig. 2: $\lambda_E c_2$, d_2 and $\lambda_E c_2/d_2$ as a function of D, for *carphone* ($QP = 24$).

of the MB/subblock gradients D and D^k. More generally, c_j, d_j and the conditional probabilities can change according to the video content, the compression level and the quantization parameter QP. Therefore, accurate modeling of the algorithm parameters require extensive simulations on a training set of video sequences coded at various compression levels.

Note that, our early mode selection strategy could be used together with almost any fast ME algorithm in the literature (except for those algorithms in which there is a strong dependence between the MV searches of different partitions). In this paper, we have chosen to incorporate our method into the fast ME algorithm provided in JM reference software, called UMHexagonS [12]. That is why UMHexagonS is also used during the training phase when determining the necessary parameters. We discuss below the implications of using this algorithm on the modeling of the parameters.

During training, video sequences are coded using constant QP. This is required, because especially d_j's and the conditional distributions heavily depend on the value of QP. However, once the optimal thresholds are determined for each QP, the algorithm could be used in a coding scenario with rate control, i.e. changing QP.

Simulations show that c_j and d_j strongly depend on the MB content (e.g. level of motion and high-frequency details). The simple spatio-temporal gradient D is meant to represent this content information. However, the relationship between a MB and its actual c_j, d_j values is actually much more complicated than what the gradient D can single-handedly capture. In the following, we briefly mention these complications, and then provide our own reasoning and solution to these problems:

- Most fast ME algorithms, including UMHexagonS, employ early termination techniques and MB adaptive decisions that make the ME execution time heavily dependent on the MB content. While the actual ME time can change drastically between MBs of similar gradient, we observe that, on average, the computational cost increases with increasing D (see Figure 2).
- A similar argument applies to R-D costs d_j, although the actual relationship between the R-D costs and the MB content is even more complicated. In simple terms, as the MB gets more detailed, its coding cost increases, and so does the cost of mis-estimating its coding mode, that is d_j (see Figure 2).
- The R-D loss of skipping ME of a given partition cannot be evaluated independent of the other partitions. For instance, the average R-D cost of skipping both m_1 and m_2 is larger than the sum of average costs of skipping only m_1 and only m_2. This is expected, since, when m_1 is skipped, m_2 is usually the next best choice and vice versa. This observation has to be taken into account while computing the parameters d_j.
- The suboptimal decisions (i.e. suboptimal MVs and suboptimal mode decision) made for a MB affect the R-D performance while coding future MBs. This is also expected, since MVs of a given MB is used to predict the MVs of the next MB, causing propagation of the suboptimality. This dependency impedes an exact calculation of the R-D cost of any mode estimation error. However, on the average, we expect the R-D costs of these secondary effects to cancel out, justifying the assumption of independent MBs.

TABLE I: Complexity Parameters for Different Sequences

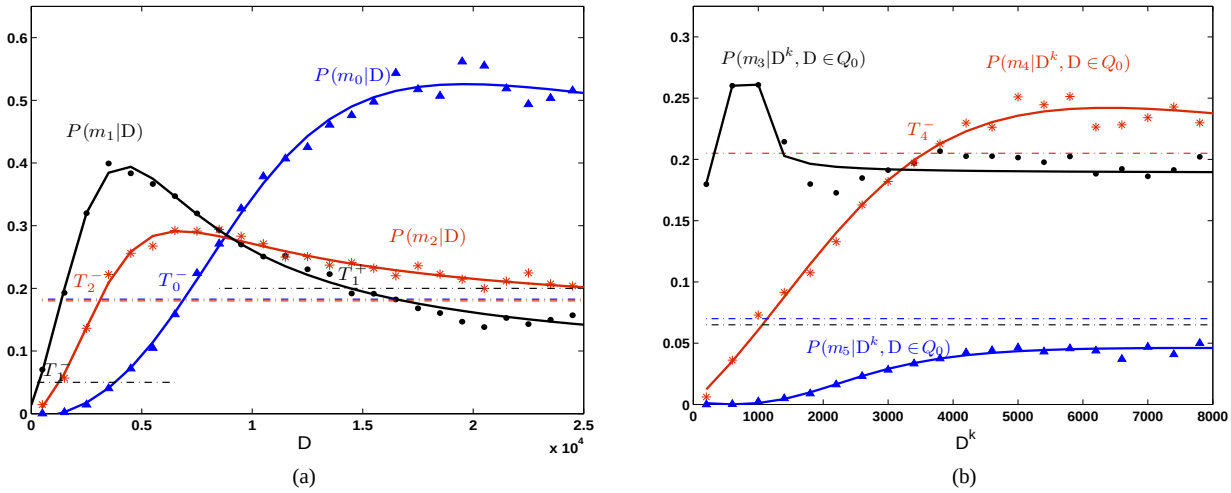
		c_1	c_2	c_3	c_4	c_5
carphone	QP=24	1.0	1.7	1.2	3.7	1.5
	QP=32	1.0	1.7	1.2	4.1	1.5
foreman	QP=24	1.0	1.7	1.2	4.0	1.5
	QP=32	1.0	1.8	1.4	4.5	1.5
mobile	QP=24	1.0	1.8	1.3	4.0	1.3
	QP=32	1.0	1.9	1.4	4.2	1.3
tennis	QP=24	1.0	1.8	1.2	3.7	1.3
	QP=32	1.0	1.8	1.4	4.3	1.4
stefan	QP=24	1.0	1.8	1.2	3.6	1.3
	QP=32	1.0	1.8	1.3	3.9	1.3

Despite these issues, the behaviors of average c_j and average d_j are well defined as functions of D. In other words, even though there exist many MB-dependent and algorithmic factors that affect the actual c_j and d_j, the expected values of these parameters depend mainly on the gradient D. Figure 2 shows average $\lambda_E c_2$, for a suitable λ_E, and average d_2 vs. D, for the MBs of *carphone* sequence. We observe similar behavior for other partitions m_j and for other tested video sequences.

Figure 2 also shows $\lambda_E c_2/d_2$ (magnified by $K = 200$ for better visualization). We see that this ratio is almost constant for all D (except for very small D in which case d_2 is relatively small). This behavior is also common to all partitions and tested sequences. Simulations also indicate that the relative ME complexities for different partitions, i.e. c_j/c_i, stay almost the same for all D. Table I lists c_j for different tested sequences, relative to the value of c_1.

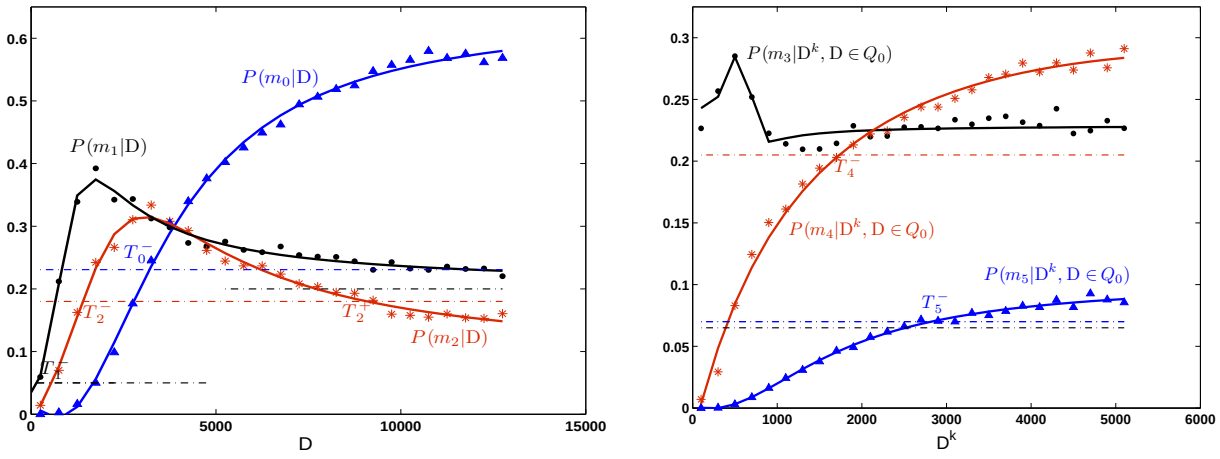
Based on these two observations, we propose to take the ratios c_j/d_j, c_j/c_i, and d_j/d_i to be constant, independent of D or D^k. The parameters c_j and d_j need only to be defined relative to one another, irrespective of the actual values.

Note that, the specific shape of these plots is a function of the fast ME algorithm being used, i.e. UMHexagonS.



(a) (b)

Fig. 3: Conditional probabilities (QP=32): (a) modes m_1, m_2, m_0; (b) modes m_3, m_4, m_5



(a) (b)

Fig. 4: Conditional probabilities (QP=24): (a) modes m_1, m_2, m_0; (b) modes m_3, m_4, m_5

For other ME algorithms, these plots could be somewhat different. In general, assuming constant c_j/d_j and c_j/c_i is a crude approximation. In the simulations section, we show that these approximations do not compromise the optimality of the approach in terms of the actual coding results.

By averaging the actual execution times for different MB gradients and through different test sequences, we arrive at the following set of relative ME complexities: $c_1 = 1, c_2 = 1.8, c_3 = 1.3, c_4 = 4.1, c_5 = 1.4$.

When computing the relative values of d_j, we consider the fact that, in all cases, the optimal thresholds satisfy the following inequalities:

$$T_1^- \leq T_2^- \leq T_0^- \leq T_1^+, \quad T_0^- \leq T_2^+. \tag{38}$$

Hence, whenever $\mathrm{D} < T_1^-$, ME should be skipped for all partitions of the MB. This decision has typically a high marginal cost, i.e. d_1 is relatively large. On the other hand, if $\mathrm{D} > T_1^+$, m_1 is skipped but m_2 and some of the smaller partitions are typically active. Therefore, the marginal cost of skipping ME for m_1 is much lower this time. If we represent these two different costs by d_1^- and d_1^+, then $d_1^- \gg d_1^+$. For other partitions m_j, the R-D costs d_j also change based on which modes are active. But these variations are comparably smaller or do not affect the optimal thresholds, and hence there is no need to differentiate between d_j^- and d_j^+.

For the subpartitions, the thresholds exhibit a similar pattern:

$$T_3^- \leq T_4^- \leq T_5^-. \tag{39}$$

Hence, whenever $\mathrm{D}^k \geq T_5^-$, m_3 and m_4 (and possibly m_1, m_2) are already active, implying a small d_5 on the average. This leads to $T_5^- = \infty$ and m_5 is skipped in all MBs. Nonetheless, simulations show that skipping m_5 could be costly for some video frames (e.g. for very detailed MBs that contain objects experiencing different motions), especially when QP is low. Therefore, we set d_5 to a suitable level that will allow MV search for m_5 whenever necessary.

By carefully considering these interactions between different partitions and evaluating the R-D costs for the training set of video sequences, we have determined the following set of

TABLE II: Optimal thresholds for different QP and λ_E

	λ_E	T_0^-	T_1^-	T_1^+	T_2^-	T_4^-	T_5^-
QP=24	0.02	2230	0	∞	725	610	1245
	0.04	2980	80	∞	1180	1195	2090
	0.06	3200	245	9120	1630	2725	4260
	0.08	3040	350	3090	2260	∞	∞
QP=32	0.02	5110	65	∞	1620	1680	3000
	0.04	6590	260	20310	2580	2670	∞
	0.06	6285	440	11190	3615	∞	∞
	0.08	7330	610	7435	5895	∞	∞

relative d_j parameters to yield the best efficiency-complexity trade-off: $d_0 = 1.0, d_1^- = 1.0, d_1^+ = 0.25, d_2 = 0.5, d_3 = 1.0, d_4 = 1.0, d_5 = 1.0$.

As for the computation of conditional probability distributions, we observe that the probabilities change with overall video content, and even with the content of each frame. Similar to c_j and d_j, the probability of a mode being optimal depends not only on the MB gradient but also on other factors, such as the local motion content. As mentioned before, this requires the use of a higher order statistical model, which is not considered in this paper. Instead, we limit our analysis to video sequences that yield similar distributions, i.e. sequences with moderate-to-high motion content and spatial details. Hence, we determine a fixed set of *a priori* distributions and use it in all of the tested sequences. In the simulations section, we discuss the impact of this simplification on the performance of the algorithm.

As mentioned before, the quantization parameter QP significantly affects the conditional probabilities of different partitions. With higher QP, fewer MBs prefer 8×8 and smaller partitions, and the probability of these modes being optimal decreases at all MB gradients (see Figure 3 and Figure 4). Therefore, the training sequences are coded using fixed values of QP, and the resulting optimal statistics are used to derive separate conditional distributions for corresponding QP values.

Figure 3 and Figure 4 show the derived distributions for QP=32 and QP=24. In these figures, rational functions (having second order numerators and denominators) are fitted through the empirical histograms. The fit is rather accurate most of the time, and it simplifies the computation of the optimal thresholds.

The optimal thresholds are computed through equations (27), (31), (35) for different rate-distortion and complexity trade-offs (i.e. for different values of the Lagrangian parameter λ_E). Note that, the equations (31) and (35) depend on both $T_0^{\pm}$ and $T_j^{\pm}$, for j=3,4,5. Therefore we start with initial estimates of $\bar{c}_0$ and $\bar{d}_0$, iteratively update the values $T_0^{\pm}$ and $T_j^{\pm}$, and converge to the optimal solutions after a couple of iterations.

Figure 3 and 4 illustrate the thresholds satisfying the optimality equations. For instance, when QP=32, with $\lambda_E = 0.05$ c_j, d_j as defined above, the optimal thresholds are computed as: $T_0^- = 6915$, $T_1^- = 350$, $T_1^+ = 14430$, $T_2^- = 3070$, $T_2^+ = 41850$, $T_4^- = 2000$ (The other thresholds are either 0 or ∞). Note that $T_3^- = 0$, $T_3^+ = \infty$; that is, we always search 8×8 partition as long as D $\in \mathcal{Q}_0$. Also, $T_5^- = T_5^+ = \infty$ implies that integer-pel ME is not performed for 4×4 partition.

For all QP and λ_E values, we always set $T_0^+ = T_3^+ = T_4^+ = T_5^+ = \infty$. This does not have any significant effect on the performance, since there are only few MBs with very high gradients. A sample set of thresholds are tabulated in Table II for different QP and λ_E.

V. Frame- and MB-Level Threshold Adaptation

As explained in the previous section, when λ_E and QP are fixed, the same set of global thresholds are used throughout the video sequence that is being coded. Provided that all the assumptions of the model are satisfied, this leads to an R-D and complexity optimal performance. On the other hand, different frames of a video sequence might have significantly different content, and this approach might lead to significant variations in total execution time among frames.

Large variations in the execution time impede an efficient real-time hardware implementation of the algorithm. For smooth and real-time encoding, the encoder needs to code frames at a rate that is equal to the frame display rate. In the worst case scenario when each frame consumes the maximum amount of time during ME process, the encoder still has to guarantee real-time operation. Therefore, if two ME algorithms that have similar average execution time and similar data flow and memory requirements are compared, the one with less variations in the execution time will have a more efficient hardware implementation. Consequently, it is important to achieve almost constant total execution time among all frames of a sequence so as to simplify the hardware design of the encoder and to guarantee smooth real-time operation.

In this section, we use the constrained optimization scenario to adapt the Lagrangian parameter λ_E and the corresponding thresholds so that a given complexity budget is satisfied for the whole frame, i.e. $\sum_{i=1}^{N} C_i \leq CB$.

Note that, the ME complexity of i^{th} MB is equal to C_i, where

$$C_i = I_1 c_1 + I_2 c_2 + \sum_{j=3}^{5} \sum_{k=1}^{4} I_j^k c_j^k, \tag{40}$$

where $c_j^k = c_j/4$, and

$$I_j = \begin{cases} 1, & \text{if D} \in \mathcal{Q}_j \\ 0, & \text{else} \end{cases} \tag{41}$$

$$I_j^k = \begin{cases} 1, & \text{if D} \in \mathcal{Q}_0 \ \& \ \text{D}^k \in \mathcal{Q}_j^k \\ 0, & \text{else} \end{cases} \tag{42}$$

Assume that, for a given QP, we define a set of Lagrangian parameters, $\mathcal{S} = \{\lambda_E^{(l)}\}_{1 \leq l \leq L}$ ($\lambda_E^{(l)} < \lambda_E^{(l+1)}$), and the corresponding optimal thresholds $\{T_j^{\pm(l)}\}_{1 \leq l \leq L}$. Then, before starting to code a frame, we compute the gradients of all MBs and find $\lambda_E^{(l)} \in \mathcal{S}$ that satisfies the time budget, i.e. $CB^{(l)} \leq CB$, where $CB^{(l)} = \sum_{i=1}^{N} C_i^{(l)}$:

- Find $\lambda_E^{(l)} \in \mathcal{S}$ such that $\lambda_E^{(l)} \approx \lambda_E^{\text{pr}}$, which is the Lagrangian parameter of the previous frame.
- If $CB^{(l)} < CB$, then $\delta l = 1$,
  - Increment l by 1 as long as $CB^{(l)} < CB$.
- Else if $CB^{(l)} > CB$, then $\delta l = -1$

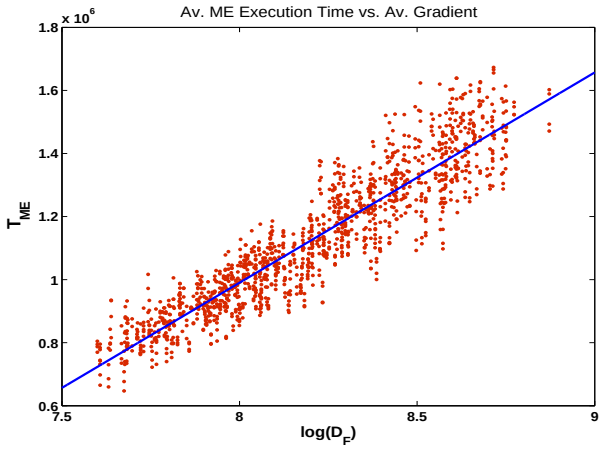


Fig. 5: Average ME Time vs Average Frame Gradient, for *carphone*.

– Decrement l by 1 as long as $CB^{(l)} > CB$.

- Set

$$\lambda_E = \alpha\lambda_E^{(l)} + (1-\alpha)\lambda_E^{(l+\delta l)}, \quad (43)$$

$$T_j^{\pm} = \alpha T_j^{\pm(l)} + (1-\alpha)T_j^{\pm(l+\delta l)}, \quad (44)$$

where

$$\alpha = \frac{CB^{(l+\delta l)} - CB}{CB^{(l+\delta l)} - CB^{(l)}}. \quad (45)$$

In this formulation, the frame-level ME complexity is measured in terms of the relative values of c_j. However, we know from the previous section that c_j's are actually functions of D. Figure 2 shows how c_j increases with increasing D, and converges to an approximate worst-case limit for high values of D. Therefore, for the full frame, ME execution time increases with increasing D_F, the average frame gradient, as well.

Figure 5 plots the relationship between T_{ME}, average ME execution time, and $\log(\mathrm{D}_F)$, for the frames of *carphone* sequence. As seen from this figure, up to a good approximation, the execution time increases linearly with $\log(\mathrm{D}_F)$. In view of this observation, we model c_j's as a function of average frame gradient, as follows. For $1 \le j \le 5$:

$$c_j(\mathrm{D}_F) = \beta(\mathrm{D}_F)c_j, \quad \beta(\mathrm{D}_F) = 1 + a(\log(\mathrm{D}_F) - b). \quad (46)$$

Here, the slope a can be seen as a modeling parameter that depends on the ME algorithm being used. For UMHexagonS, we set $a = 0.5$ and $b = 8.5$.

Now, we update the complexity budget for each frame to account for this model. We once again assume constant relative c_j, but compute the average frame gradient D_F and set the complexity budget such that

$$\sum_{i=1}^{N} C_i \le \frac{CB}{\beta(\mathrm{D}_F)} \quad (47)$$

is satisfied. Hence, depending on the average frame gradient, the budget is adapted so that overall execution time stays almost the same.

In addition to this frame-level update of the thresholds, we also perform MB-level adaptation to further decrease the fluctuations in the total ME execution time among frames. For that purpose, we compare the actual ME execution times of the MBs, X_i, $1 \le i \le N$, with the actual time budget TB. In order not to cause any extra computational burden, we apply a simple strategy:

- For MB $n = 10k, k \in \mathbb{Z}$, compare the actual ME time so far with the expected time:
  - If $\sum_{i=1}^{n} X_i > \frac{n}{N}TB + \epsilon$, then choose $\lambda_E^u = \lambda_E^{\mathrm{av}} + \delta\lambda$;
  - Else if $\sum_{i=1}^{n} X_i < \frac{n}{N}TB - \epsilon$, then choose $\lambda_E^u = \lambda_E^{\mathrm{av}} - \delta\lambda$;
  - Else $\lambda_E^u = \lambda_E^{\mathrm{av}}$;
- Find $\lambda_E \in \mathcal{S}$ such that $\lambda_E \approx \lambda_E^u$.
- Set $\lambda_E^{\mathrm{av}} = (n\lambda_E^{\mathrm{av}} + 10\lambda_E)/(n+10)$.

In this procedure, once in every 10 MBs, λ_E is chosen from the set $\mathcal{S}$ as slightly higher or lower than the average λ_E^{av} for the frame, depending on whether the actual ME execution time is higher or lower than the expected time so far, respectively.

Note that, most video frames contain both smooth static regions (e.g. background) and detailed moving regions (e.g. boundary of a moving object), which causes significant fluctuations of execution time even within a single frame. Updating λ_E for every such fluctuation will have an adverse effect on the coding results. Therefore, by performing the update procedure once in every 10 MBs, and choosing the parameters δ and ϵ conservatively, we avoid λ_E to change drastically from its average value.

In a coding scenario where encoded frames are first stored in a large buffer before transmission, the time budget could be set at the GOP (group of pictures) level, i.e. the total ME time of frames in the GOP should be less than this limit. In that case, the budget of each frame would be adjusted according to the remaining encoding time for the given GOP. That means, for frame $i, 1 \le i \le M$:

$$TB_i = \frac{TB_{\mathrm{GOP}} - \sum_{j=1}^{i-1} T_{\mathrm{ME}}^j}{M - i + 1}, \quad (48)$$

where T_{ME}^j is the ME execution time of frame j, and M is the number of frames in the GOP. Since the time lost in one frame could be made up in the other frames of the GOP, these budgets are not necessarily strict. Hence, MB-level threshold updates would be less frequently needed. Having a looser time limit for each frame should improve the coding performance of the algorithm.

VI. SIMULATIONS

A. Testing Conditions

The simulations are performed for video sequences *carphone* (QCIF), *foreman* (CIF), *mobile* (CIF), *tennis* (SIF), *stefan* (SIF) at 30 fps, (search range $[-16, 16]$). All frames except the first one are coded as P-frames. Two reference frames are allowed. The CAVLC entropy coder is used. For coding with constant QP, the quantization parameters are $QP = 24, 28, 32, 36$. The optimal thresholds are determined, at all QP values, for the set $\mathcal{S} = \{\lambda_E : \lambda_E = 0.005k, 1 \le$

TABLE III: Performance comparison with FSM

carphone	δPSNR (dB)	δbitrate(%)	Speed-up
Comp-Opt(1)	-0.18	+3.47	33.7
Comp-Opt(2)	-0.14	+2.68	26.9
Comp-Opt(3)	-0.17	+3.18	26.5
UMHexagonS	-0.07	+1.39	9.4
foreman	δPSNR (dB)	δbitrate(%)	Speed-up
Comp-Opt(1)	-0.14	+3.44	35.3
Comp-Opt(2)	-0.13	+3.06	32.3
Comp-Opt(3)	-0.13	+3.16	28.3
UMHexagonS	-0.05	+1.31	9.4
mobile	δPSNR (dB)	δbitrate(%)	Speed-up
Comp-Opt(1)	-0.06	+1.17	25.2
Comp-Opt(2)	-0.07	+1.46	29.6
Comp-Opt(3)	-0.09	+1.77	33.2
UMHexagonS	-0.01	+0.21	9.6
tennis	δPSNR (dB)	δbitrate(%)	Speed-up
Comp-Opt(1)	-0.10	+2.60	36.5
Comp-Opt(2)	-0.12	+3.12	37.4
Comp-Opt(3)	-0.14	+3.68	38.8
UMHexagonS	-0.03	+0.84	12.0
stefan	δPSNR (dB)	δbitrate(%)	Speed-up
Comp-Opt(1)	-0.17	+2.98	25.4
Comp-Opt(2)	-0.19	+3.21	24.7
Comp-Opt(3)	-0.23	+3.92	25.3
UMHexagonS	-0.07	+1.26	9.9

TABLE IV: Comparison with FSM (Rate Control enabled)

carphone	δPSNR (dB)	Speed-up
Comp-Opt(1)	-0.16	33.4
UMHexagonS	-0.07	9.2
foreman	δPSNR (dB)	Speed-up
Comp-Opt(1)	-0.14	33.6
UMHexagonS	-0.05	9.1
mobile	δPSNR (dB)	Speed-up
Comp-Opt(1)	-0.06	25.9
UMHexagonS	-0.02	9.4
tennis	δPSNR (dB)	Speed-up
Comp-Opt(1)	-0.10	34.5
UMHexagonS	-0.04	11.6
stefan	δPSNR (dB)	Speed-up
Comp-Opt(1)	-0.17	23.8
UMHexagonS	-0.06	9.8

$k \leq 30\}$. The algorithm is incorporated into JM software version 8.2 [32], and used together with UMHexagonS. For rate control, JM's own algorithm is used.

Three versions of the algorithm Complexity-Optimization (abbr. Comp-Opt) are tested during simulations:

- **Comp-Opt v.1**: unconstrained case, with fixed λ_E for the full sequence. The value of λ_E is adjusted for each QP, such that total ME execution time stays almost the same.
- **Comp-Opt v.2**: constrained optimization, with fixed complexity budget $CB = 3N$; $\beta(\mathrm{D}_F) = 1$ for all frames.
- **Comp-Opt v.3**: constrained optimization, with $CB = 3N$; $\beta(\mathrm{D}_F)$ is computed from equation (46). MB-level update is also used, with $TB \approx 133N\mu$sec.

B. Comparison with UMHexagonS

The results in Table III and IV compare the performance of our approach with UMHexagonS in terms of speed-up factor (i.e. the ratio between ME execution times), average PSNR loss in dB (at equal bitrates) and percentage change in bitrate (at equal PSNR) with respect to FSM as given in [32]. Average PSNR loss and bitrate change are computed from the R-D curves as described in [33].

We see from Table III that Comp-Opt yields a speed-up factor of 25-40 with less than 0.2 dB PSNR loss at equal bitrates (except for *stefan* using version 3). The speed-up factor over UmHexagonS is between 2.5-3.5. Note that, UmHexagonS does not provide a high speed-up factor over FSM, but the results in terms of PSNR and bitrate are almost as good as FSM. Therefore, our algorithm also gives very good PSNR and bitrate results, but achieves only moderate speed-up. If Comp-Opt is used together with faster ME algorithms, the speed up will be much higher but the PSNR and bitrate will get worse.

Table IV shows the simulation results for Comp-Opt v.1 when rate control is enabled. Each MB uses the thresholds optimized for its own QP value. The results are very similar to the case with constant QP. Hence, the optimality of the algorithm does not suffer due to changing QP.

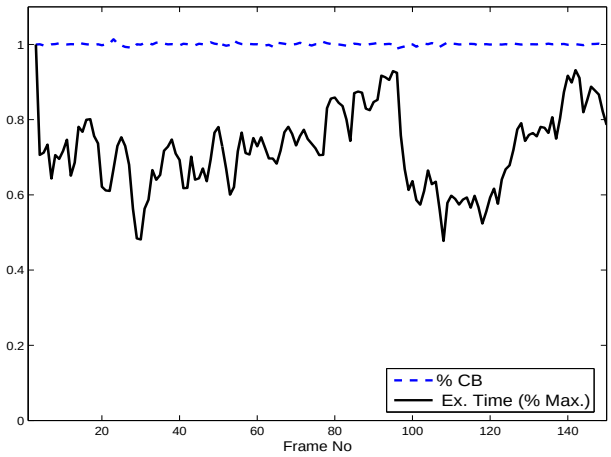


Fig. 6: Clock cycles and execution time per frame using Compt-Opt v.2 (*foreman* QP=32).

Figure 6 shows the percentage of complexity budget CB spent and the normalized ME execution time (w.r.t. the maximum execution time) for frames of *foreman*, coded with $QP = 24$ using Compt-Opt v.2. We see that even though CB is almost perfectly met at each frame, the execution times still show variations among frames. This is expected, since relative complexity parameters c_j do not reflect the actual execution time of video frames with different average gradients.

Compt-Opt v.3 reduces the time variations by adjusting the budget according the frame content and by updating λ_E based on the actual ME time. Figure 7 shows the ME execution time for the frames of *carphone*, coded with $QP = 24$, both for UMHexagonS and for Compt-Opt v.3. In order to better visualize the variations, the mean time is subtracted from both plots. Compt-Opt v.3 reduces the standard deviation of the execution time by a factor of about 12.5. For version 1 and version 2, the reduction factor is merely 1.3 and 5.2, respectively.

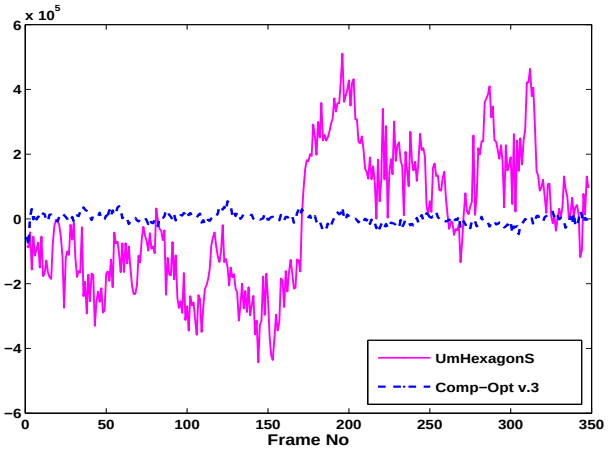


Fig. 7: Execution time per frame for UMHexagonS and Compt-Opt v.3 (*carphone* QP=32).

The reduction of time variations in Compt-Opt v.3 comes with some degradation in the R-D and complexity performance. That is, even though the results for Compt-Opt v.1 and Compt-Opt v.2 are very similar, Compt-Opt v.3 yields worse PSNR at equal complexity or lower speed-up factor at equal PSNR loss. This is natural, since for detailed frames with high ME time, λ_E has to be set at a high level, causing significant R-D losses. Also, the simple but suboptimal MB-level adaptation strategy is not good for the coding performance. As discussed before, if the time budget is set at the GOP level instead of frame-level, then the coding performance of Compt-Opt v.3 should be better.

Since ME complexity is adapted at the MB-level in Compt-Opt v.3, some MBs might have been coded at a very low level of ME complexity (i.e. high λ_E). This could potentially lead to significant variations in coding quality of individual MBs. Hence, we look at how much R-D costs change w.r.t. UMHexagons throughout the MBs of tested sequences. Let us define, for each MB, E_o and ΔE as the R-D cost of UMHexagonS and the R-D cost difference between Comp-Opt v.3 and UMHexagonS, respectively. For MBs with sufficiently large R-D costs, say when the cost is larger than the mean of E_o, we calculate the number of MBs for which $\Delta E > 0.5E_o$. This number gives an indication about the percentage of MBs that experience a relatively large coding quality degradation. It turns out that this value is less than 1% of total number of MBs for $QP = 36$ and goes as low as 0.1% when $QP = 24$.

When we compare the worst-case ME execution time instead of the average time, Compt-Opt v.3 gives a speed-up factor of 32.1 over FSM, as opposed to 26.5 given in Table III for *carphone*. Table V indicates similar speed-up improvements for other test sequences as well. Since worst-case complexity is a more important design parameter than average complexity, we believe that the performance of Compt-Opt v.3 is actually very promising.

C. Discussions about the optimal performance

Throughout the derivation of Comp-Opt algorithm, we have proposed several simplifications to the optimal solution and justified these choices by related observations on the tested sequences. The use of constant average values for c_j and d_j, and fixed probabilities for all video sequences are crucial approximations and have to be validated with further tests. For that purpose, we implement an "oracle" algorithm that uses the exact ME times and the best estimate of R-D costs (as a function of D) to choose, for each MB, the optimal set of partitions that minimizes the Lagrangian cost. This oracle yields PSNR values that are at most 0.03 dB better than Comp-Opt v.1, at equal bitrate and speed-up factors. Therefore, the use of average parameter settings leads to a minor deviation from the optimal performance.

The proposed optimization framework is important for robust and efficient adaptation of the thresholds to satisfy any level of ME complexity. For a fixed level of complexity, the thresholds of the decision sets might be selected by trial and error, without resorting to the complicated procedure presented in this paper. However, our optimized framework becomes crucial when the ME complexity is to be adaptively controlled. To illustrate this, we implement a simple threshold update strategy as follows:

- As long as $\sum_{i=1}^{N} C_i > CB$, update the thresholds:

$$T_j^{\pm} = T_j^{\pm}(1 \mp \alpha). \tag{49}$$

The algorithm starts with an initial set of thresholds that yield higher total complexity than the budget, and iteratively updates these values until the budget is satisfied. Table VI compares the performance of this heuristic algorithm with that of Comp-Opt v.2. In this table, for almost equal speed-up factors, Comp-Opt v.2 is 0.10-0.25 dB better. As the speed-up ratio increases, so does the PSNR difference between these two approaches. It turns out that the performance of this approach is very much dependent on the initial thresholds, and could actually lead

TABLE V: Speed-up factors for Compt-Opt v.3

Speed-up	worst-time	average-time
carphone	32.1	26.5
foreman	36.4	28.3
mobile	35.1	33.2
tennis	72.3	38.8
stefan	30.2	25.3

TABLE VI: Performance comparison with heuristic approach

carphone	δPSNR (dB)	δbitrate(%)	Speed-up
Comp-Opt(2)	-0.24	+4.71	49.2
Heuristic	-0.42	+8.13	50.4
foreman	δPSNR (dB)	δbitrate(%)	Speed-up
Comp-Opt(2)	-0.25	+5.92	52.9
Heuristic	-0.42	+9.62	53.8
mobile	δPSNR (dB)	δbitrate(%)	Speed-up
Comp-Opt(2)	-0.11	+2.23	49.8
Heuristic	-0.20	+4.05	50.3
tennis	δPSNR (dB)	δbitrate(%)	Speed-up
Comp-Opt(2)	-0.19	+4.70	64.4
Heuristic	-0.45	+10.82	65.8
stefan	δPSNR (dB)	δbitrate(%)	Speed-up
Comp-Opt(2)	-0.30	+5.07	35.9
Heuristic	-0.40	+6.76	37.2

to very poor results if these initial values are not selected appropriately.

Our use of a robust complexity adaptation strategy is a major improvement when compared to other mode decision methods in the literature. As discussed in the introduction, some of these methods use sophisticated MB-adaptive measures to decide which modes to search for. However, since they are MB-adaptive and cannot be computed before the coding begins, these measures are not suitable for frame-level control of the overall ME complexity. In that respect, spatio-temporal gradient is both suitable and very effective. We still continue to search for better measures and look into higher order prediction models as well.

VII. CONCLUSION AND FUTURE WORK

In this paper, we develop a general framework for joint optimization of R-D efficiency and computational complexity in H.264 encoder. This framework is applied successfully to integer-pel ME module, and moderate speed-up with little loss of coding efficiency is observed.

The optimization framework defined in this paper is equally applicable to various other parts of the encoding process, such as full ME (integer and sub-pel), or ME and MD together. In the latter case, if a mode is skipped, it is excluded from R-D cost calculations and optimal mode decisions as well.

The optimization framework allows the encoder to adjust its complexity "on the fly" to satisfy stringent timing requirements. This feature of our algorithm establishes a new direction in the encoder optimization research, which should be further explored. Therefore, we continue to seek more sophisticated models that are capable of improving the mode estimation accuracy and hence the overall coding efficiency.

REFERENCES

[1] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 13, no. 7, pp. 688–703, July 2003.

[2] I.G. Richardson, *H.264 and MPEG-4 Video Compression*, Wiley, West Sussex, England, 2003.

[3] ITU-T, *Prepublished Recommendation H.264*, 2003.

[4] H. Cheong and A.M. Tourapis, "Fast motion estimation within the H.264 codec," in *Proc. IEEE Int. Conf. Multimedia and Expo*, Baltimore, MD, July 2003, vol. 3, pp. 517–20.

[5] J. Zhang, Y. He, S. Yang, and Y. Zhong, "Performance and complexity joint optimization for H.264 video encoding," in *Proc. Int. Symposium Circuits Systems*, May 2003, pp. 25–28.

[6] W.I. Choi, B. Jeon, and J. Jeong, "Fast motion estimation with modified diamond search for variable motion block sizes," in *Proc. IEEE Int. Conf. Image Processing*, Sept. 2003, vol. 2, pp. 371–4.

[7] K-K. Ma and G. Qiu, "Unequal-arm adaptive rood pattern search for fast block-matching motion estimation in the JVT/H.26L," in *Proc. IEEE Int. Conf. Image Processing*, Barcelona, Spain, Sept. 2003.

[8] L. Yang, K. Yu, L. Li, and S. Li, "An effective variable block-size early termination algorithm for H.264 video coding," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 15, no. 6, pp. 784–788, June 2005.

[9] C.-W. Lam and L.-M. Po, "Fast block motion estimation with early acceptance technique in H.264/JVT," in *Proc. Int. Symposium Circuits and Systems*, May 2005, vol. 2, pp. 1513–16.

[10] J. Xu, Z. Chen, and Y. He, "Efficient fast ME predictions and early-termination strategy based on H.264 statistical characters," in *Proc. Int. Conf., Inform., Communication and Signal Processing*, Dec. 2003, vol. 1, pp. 218–22.

[11] G.L. Li, M.J. Chen, H.J. Li, and C.T. Hsu, "Efficient search and mode prediction algorithms for motion estimation in H.264/AVC," in *Proc. Int. Symposium Circuits Systems*, May 2005, vol. 6, pp. 5481–84.

[12] Z. Chen, P. Zhou, and Y. He, "Fast motion estimation for JVT," in *JVT-G016.doc, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG*, 7th Meeting: Pattaya II, Thailand, 7-14 March 2003.

[13] X. Li, E.Q. Li, and Y.K. Chen, "Fast multi-frame motion estimation algorithm with adaptive search strategies in H.264," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, May 2004, vol. 3, pp. 369–72.

[14] D. Wu and *et al*, "Fast intermode decision in H.264/AVC video coding," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 15, no. 6, pp. 953–8, June 2005.

[15] Z. Zhou, J. Xin, and M.-T. Sun, "Fast motion estimation and inter-mode decision for H.264/MPEG-4 AVC encoding," *J. Vis. Commun. Image*, vol. 17, no. 2, pp. 243–63, April 2006.

[16] H. Wang, S. Kwong, and C.-W. Kok, "An efficient mode decision algorithm for H.264/AVC encoding optimization," *IEEE Trans. Multimedia*, vol. 9, no. 4, pp. 882–8, June 2007.

[17] A. Chang, P.H.W. Wong, Y.M. Yeung, and O.C. Au, "Fast multi-block selection for H.264 video coding," in *Proc. Int. Symposium Circuits and Systems*, May 2004, vol. 3, pp. 817–20.

[18] A. Chang, O.C. Au, and Y.M. Yeung, "A novel approach to fast multi-block motion estimation for H.264 video coding," in *Proc. IEEE Int. Conf. Multimedia and Expo*, July 2003, vol. 1, pp. 105–8.

[19] S.-F. Lin, C.-Y. Chang, C.-C Su, Y.-L. Lin, C.-H. Pan, and H. Chen, "Fast multi-frame motion estimation and mode decision for H.264 encoders," in *Proc. IEEE Int. Conf. Wireless Networks, Communications and Mobile Computing*, June 2005, vol. 2, pp. 1237–42.

[20] D. Zhang, Y. Shen, S. Lin, and Y. Zhang, "Fast inter frame encoding based on modes pre-decision in H.264," in *Proc. IEEE Int. Conf. Multimedia and Expo*, July 2005, p. 4 pp.

[21] A.C. Yu, "Efficient block-size selection algorithm for inter-frame coding in H.264/MPEG-4 AVC," in *Proc. IEEE ICASSP*, May 2004, vol. 3, pp. 169–172.

[22] X. Jing and L.P. Chau, "An efficient inter mode decision approach for H.264 video coding," in *Proc. IEEE Int. Conf. Multimedia Expo*, June 2004, vol. 2, pp. 1111–14.

[23] Y. Shen, D. Zhang, C. Huang, and J. Li, "Fast mode selection based on texture analysis and local motion activity in H.264/JVT," in *Proc. IEEE Int. Conf. Comm. Circuits Systems*, June 2004, vol. 1, pp. 539–42.

[24] T.-Y. Kuo and C.-H. Chan, "Fast variable block size motion estimation for H.264 using likelihood and correlation of motion field," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 16, no. 10, pp. 1185–95, Oct. 2006.

[25] T.-Y. Kuo and C.-H. Chan, "Fast macroblock partition prediction for H.264/AVC," in *Proc. IEEE Int. Conf. Multimedia and Expo*, June 2004, vol. 1, pp. 675–78.

[26] C. Grecos and M. Yang, "Fast mode prediction for the baseline and main profiles in the H.264 video coding standard," *IEEE Trans. Multimedia*, vol. 8, no. 6, pp. 1125–34, Dec. 2006.

[27] S. Saponara, M. Casula, F. Rovati, D. Alfonso, and L. Fanucci, "Dynamic control of motion estimation search parameters for low complex H.264 video coding," *IEEE Trans. Consum. Electron.*, vol. 52, no. 1, pp. 232–9, Feb. 2006.

[28] P. Yin, H. Cheong, A.M. Tourapis, and J. Boyce, "Fast mode decision and motion estimation for JVT/H.264," in *Proc. IEEE Int. Conf. Image Processing*, Barcelona, Spain, Sept. 2003, vol. 3, pp. 14–17.

[29] T. Shimizu, A. Yoneyama, H. Yanagihara, and Y. Nakajima, "A two-stage variable block size motion search algorithm for H.264 encoder," in *Proc. IEEE Int. Conf. Image Processing*, Oct. 2004, vol. 3, pp. 1481–84.

[30] J. Stottrup-Andersen, S. Forchhammer, and S.M. Aghito, "Rate-distortion-complexity optimization of fast motion estimation in H.264/MPEG-4 AVC," in *Proc. IEEE Int. Conf. Image Processing*, Oct. 2004, vol. 1, pp. 111–14.

[31] H.F. Ates, B. Kanberoglu, and Y. Altunbasak, "Rate-distortion and complexity joint optimization for fast motion estimation in H.264 video coding," in *IEEE Int. Conf. Image Processing*, Oct. 2006.

[32] http://bs.hhi.de/˜suehring/tml/index.htm, "JM reference software version 8.2," .

[33] G. Bjontegaard, "Calculation of average PSNR differences between R-D curves," *Doc. VCEG-M33*, Apr. 2001.