Efficient Variable Block Size Selection for AVC Low Bitrate Applications

Ihab Amer and Graham Jullien
Advanced Technology Information Processing Systems (ATIPS)
2500 University Drive, NW
Calgary, AB, Canada, T2N 1N4
e-mails: {amer, jullien}@atips.ca

Wael Badawy
IntelliView Technologies Inc.
808-55 Avenue NE
Calgary, AB, Canada, T2E 6Y4
e-mail: badawy@intelliview.ca

Adrian Chirila-Rus and Robert Turney
DSP Systems Engineering, DSP Division, Xilinx Inc
115 South 4th street
Watertown, WI, USA, 53094
e-mails: {adrian.chirila-rus, robert.turney}@xilinx.com

Rana Hamed
German University in Cairo (GUC)
Main Entrance, Fifth District
New Cairo City, Egypt
email: rana.magdy-hamed@guc.edu.eg

Abstract— **The Advanced Video Coding (AVC) standard proposes the usage of Variable Block Size (VBS) motion-compensated prediction and mode decision aiming for an optimized Rate-Distortion (R-D) performance. Unlike Fixed Block Size (FBS) motion-compensated prediction, where all regions of the pictures are treated similarly in terms of temporal prediction, VBS increases the efficiency of encoding by allowing more active regions to be represented with more bits than less active ones. The main concern regarding the usage of VBS motion-compensated prediction is the dramatic increase it adds to the encoder computational requirements, which not only prevents the encoder from satisfying real-time constraints, but also makes it impractical for hardware implementation. This paper presents an efficient VBS selection scheme, which can be applied to any VBS Motion Estimation (ME) module, leading to significant reduction in its computational requirements with minor loss in the quality of the reconstructed picture. The computational requirements reduction is achieved by minimizing the number of required ME searches and simplifying the Mode Decision (MD) operation. In order to meet different applications' demands, the proposed algorithm can be adjusted to function at any of three operating points, trading off computational requirements with R-D performance. In the paper, the algorithm is described in detail, focusing on the theoretical computational requirements savings. This theoretical analysis is then supported with simulation results performed on three benchmark video sequences with various types of motion.**

Keywords- H.264/AVC, motion estimation, variable block size.

I. INTRODUCTION

To achieve a high coding efficiency, AVC deploys a set of new features in addition to enhancing a set of previously used features. In the inter-frame motion prediction, AVC allows for the usage of variable block sizes motion estimation/compensation that can support block sizes of 16×16, 16×8, 8×16 and 8×8 resulting in significant performance improvement. Even more, in the case when an 8×8 mode is chosen, further smaller blocks of sizes 8×4, 4×8 and 4×4 can be used. This method improves the motion tracking capabilities of the encoder by allowing inactive regions and regular movements to be represented with an optimal amount of motion information (e.g., 16×16 mode is represented by one motion vector), while for fast and highly irregular movements, the finer blocks can be used at the cost of increased motion information, but optimal error representation. In terms of computational and memory requirements, motion estimation is by far the most complex module in the entire AVC encoder. As will be shown later, recent studies show that it represents between 70% and 90% of the entire encoder computational requirements. The increase in computational and operational requirements brought by the usage of such new features requires algorithmic and implementation enhancements so that the compression algorithm can become useful in real applications. As a step towards the design and implementation of an entire computationally-efficient AVC encoder, this paper proposes a solution to the increased computational requirements of the AVC variable block size motion estimation/compensation (and mode decision) module. A novel, computationally-efficient algorithm for variable block size motion estimation and mode decision (acting at three operating points that meet various applications' requirements) is developed and described. The

obtained simulation results and analysis show that, for the variety of tested benchmark sequences, the encoder computational requirements can be reduced to less than half, at the expense of "minor" degradation in the quality. The work in this paper further emphasises on the work that has been introduced in [1].

The remainder of this paper is organized as follows: Section II describes the key concept behind the adoption of VBS ME/MD, showing its necessity, and the computationally-expensive way it is implemented in the direct approach of the AVC software reference model. A survey of recent efforts to solve this computational requirements issue is also given. In Section III, the proposed VBS selection scheme is described, accompanied with some theoretical computational requirements analysis to show its effectiveness. Section IV shows and discusses the experimental analysis and results obtained by encoding various benchmark video sequences. Then finally, Section V concludes this paper.

II. VBS Motion Estimation and Mode Decision

AVC defines seven block sizes for inter-prediction. The seven modes are shown in Figure 1. During the encoding process, an ideal encoder has to examine all these modes to achieve the best inter-prediction, which requires performing all the seven modes of searches, and then examining all the 259 possible combinations of macroblock (MB) partitioning schemes to choose the best one out of them. This VBS motion estimation and mode selection process result in significant performance improvement (in terms of rate-distortion). This is because, on the contrary to Fixed Block Size Motion Estimation (FBSME), where all sub-blocks in a MB have the same size, Variable Block Size (VBS) ME improves the motion tracking capabilities of the encoder by allowing it to give more "attention" to highly active sub-blocks (representing an active region with more sub-blocks would better describe it). However, this leads to dramatic increase in the computational requirements of the encoder compared to traditional FBSME-based encoders.

This section overviews the main concept of VBS ME/MD. It starts with Section A, where the necessity of VBS ME/MD is investigated. Then in Section B, a description of the AVC reference software implementation approach is given. Finally, Section C summarizes some recent efforts to reduce the computational requirements of VBS ME/MD.

A. Necessity of VBS ME/MD

In this section the necessity of VBS ME is evaluated by first demonstrating its usage when encoding different sequences with different types of motion, then by comparing the reconstructed sequences R-D performance when VBS ME is fully or partially disabled.

Figure 2 shows the occurrence percentage of different block sizes as chosen by the AVC software reference model "Joint Model JM 10.2", running with all the seven block mode types enabled. Full search is used for motion estimation. The test has been performed for a wide range of sequences, starting from QCIF (176×144) up to HD 1080p (1920×1080) and for different motion types.

Table 1 summarizes the characteristics of the sequences that were used in this experiment.

The impact of choosing the variable block size can be clearly seen. For example, the traditional 16×16 has been chosen as the optimal mode for almost 50% of the cases in the "Shields" sequence, a sequence with regular pan movement, while it was as small as 5% for high irregular movement in "Football" sequence, for which the small block sizes (4×4) has been the best option in 40% of the cases.

In addition, a more informative test was performed by running the JM several times for each sequence, disabling one or more inter search/decision mode in each run, then comparing the R-D curves generated from all runs. Figure 3 shows the impact of disabling some of the searching modes on the overall encoder efficiency. The sequence used for this test was Foreman CIF (30 fps). For simplicity, the testing conditions for the encoder runs included the following restrictions: the usage of only one reference frame and the usage of "Low Complexity Mode" for Rate-Distortion optimization. Each of the curves was generated with a different inter-search/decision-mode configuration than the other. Table 2 provides a description of the different cases used to create Figure 3. Any single curve in the graph represents a typical relation between rate and distortion in digital video coding. As expected, the two parameters are inversely proportional. This is reasonable since the better quality needed to be preserved in an encoded video sequence, the more bits required to represent the generated bitstream.

Rate is typically represented by the minimum number of bits per seconds required to transmit the generated bitstream without affecting the continuity of the reconstructed sequence. It depends on two parameters: the size of the generated bitstream, and the resolution/frame rate of the specific sequence. The quality of the reconstructed sequences is measured by the most widely-used *Peak Signal to Noise Ratio* (PSNR) parameter. Due to its easy calculation, PSNR is the most famous objective quality measurement in video coding. It is measured on a logarithmic scale and depends on the "Mean Squared Error" (MSE) between an original and an impaired image or video frame as shown in (1) and (2). *F(i,j)* and *f(i,j)* are the values of pixel *(i,j)* in the original and reconstructed images or video frame of resolution $N \times M$, while (2^n-1) is the highest-possible signal value in the video frame, where n is the number of bits per frame sample.

$$MSE = \frac{\sum_{\forall i,j} [f(i,j) - F(i,j)]^2}{N \times M} \quad (1)$$

$$PSNR = 10 \log_{10} \frac{(2^n - 1)^2}{MSE} \quad (2)$$

Figure 3 shows that the blue curve (Case 0, which uses all the variable block sizes) is the optimum one in terms of R-D performance. The brown curve (Case 5, where only a fixed 4×4 search/decision mode is used) has a relatively poor performance, especially at low bitrates (around 1 Mb/s), as most of the residual data is cut-off by the coarse quantization process (high QP value), giving more influence to the size of the encoded motion info on the output bitrate. Besides, the light-blue curve (Case 3, where only a fixed 16×16 search/decision mode is used) also shows a poor performance, especially at high bitrates (around 6 Mb/s), when the savings achieved by representing each macroblock by only one motion vector becomes negligible when compared to the overhead of encoding the residuals, which becomes higher for macroblocks with more "activity" inside.

Figure 4 shows that the generated R-D curves for all the tested sequences tend to demonstrate a relatively similar behaviour to what has been discussed above at different bitrates (the higher the sequence resolution, the higher the required bitrate).

In conclusion, VBS ME/MD is an important tool that has been used in AVC to optimize the R-D performance of the encoder. It allows inactive regions and regular movements to be represented with an optimal amount of motion information (e.g., 16×16 blocks), while for fast and highly irregular movements, the finer blocks are used (e.g., 4×4 blocks), achieving optimal error representation, at the cost of increased motion information.

B. Direct Implementation of VBS ME/MD

A feature of the way AVC reference software (JM) adopts *Rate-Distortion Optimization* (RDO) is by performing an exhaustive search for the "best" partitioning scheme (the one that gives the lowest R-D Cost) among all the 259 (4^4 (8×8 or smaller) sub-modes + 1 (8×16) + 1 (16×8) + 1 (16×16) modes) possible combinations. This can be shown in Figure 5 [2]. Hence, the JM searches exhaustively for the minimum possible distortion that can be achieved subject to a specific rate constraint. This can be expressed as follows: Find min(D) subject to $R<R_c$, which can be elegantly solved using *Lagrangian optimization* where a distortion term is weighted against a rate term as follows [3]:

$$\text{Find min(J), where } J = D + \lambda R \quad (3)$$

where J is the R-D Cost that needs to be minimized. D is the *Distortion*. JM uses the *Sum of Absolute Differences* (SAD) as the metric of distortion because of its less computational requirements compared to other metrics. SAD is calculated using (4), where F(i,j) and f(i,j) are the values of pixel (i,j) in the reference and the candidate block respectively. R represents the *Rate*. The way to estimate the rate differs based on the type of rate-distortion optimization scheme being used. For *high-complexity mode*, JM uses the exact number of bits for header, motion info, and transform coefficients (the way they look after the last encoding stage) as the metric of rate as shown in (5). λ (the Lagrange parameter that determines the importance of rate with respect to distortion) is exponentially related to the Quantization Parameter (QP) after multiplying it with a double-precision weight as shown in (6).

$$SAD = \sum_{\forall i,j} \left| f(i,j) - F(i,j) \right| \quad (4)$$

$$R = R_{\text{header (exact)}} + R_{\text{motion (exact)}} + R_{\text{coefficients (exact)}} \quad (5)$$

$$\lambda = \text{weight} \times 2^{(QP-12)/3} \quad (6)$$

For *low-complexity mode*, motion cost calculation is performed using a less complex scheme. A lookup table (LUT) is used to roughly estimate the number of bits needed to encode the difference between the motion vectors and the predictors as shown in (7). λ is also estimated roughly from QP by a lookup table as shown in (8).

$$R = LUT_{MV\_COST}[MV_{cand} - MV_{pred}] \quad (7)$$

$$\lambda = LUT_{QP2QUANT}[\max(0, QP-12)] \quad (8)$$

For simplicity reasons, and practical hardware implementation, any reference to rate-distortion optimization in the remaining part of this paper is a reference to the low-complexity mode, unless clearly mentioned otherwise. Recent complexity analysis was performed in [4] to estimate the distribution of complexity among different modules of the AVC encoder. A typical profile of the encoder complexity (by files) based on an Intel® Pentium™ III 1.0 *GHz* general purpose processor with 512 MB of memory is shown in the pie chart of Figure 6(a). Also, another run-time complexity analysis was performed in [6], where the reference software (Baseline profile) was executed on a Sun® Blade™ 1000 with UltraSPARC™ III 1 *GHz* processor. The run time percentage (approximated to the nearest integer value) of each module is shown in the pie chart of Figure 6(b). The figure shows that the computational requirements of motion estimation/compensation typically represent from 70% to 90% of the overall encoder computational requirements for a typical AVC encoder, which makes it a primary candidate for hardware acceleration. Besides, this module is the main target of encoder computational requirements reduction for most researchers. Most of the efforts aim at introducing "reasonable" quality degradation as an expense of computational requirements savings compared to the optimal (yet very complex) exhaustive-search solution. The next section surveys some of those efforts.

C. Recent Efforts to Reduce Complexity of VBS ME/MD

Many algorithms for fast ME and MD have been proposed in the literature. Most of them rely on the fact that human's visual system is typically insensitive to the degradation in PSNR that is less than 0.2 *dB*. Hence, they tend to reduce the computational requirements, at the expense of minor "unrecognizable" quality degradation. In this section, an overview of the recent efforts to reduce the VBS ME/MD in AVC is given. Some of the algorithms that have been introduced recently rely on exploiting video

features such as texture and edge information to predict the best possible mode. As an example, Wu et al [7] proposed to adjust the block sizes based on the homogeneity of the region in the MB. They observed that homogenous regions, which are determined by using the magnitude of the edge vector, tend to move together, and hence should not be split into smaller blocks. The results they provided indicate that the technique is not as effective for dynamic sequences as it is for inactive ones. They achieved a maximum computational requirements reduction of 45% for inactive sequences, but it did not exceed 10% reduction for active sequences. The tests they performed were on low resolution sequences only (QCIF and CIF), which leaves the effectiveness of their algorithm with high resolution sequences questionable. Lin et al [2] developed a combined algorithm for fast motion estimation and mode decision by exploiting the motion and cost information available from blocks that have been processed prior to the current block. The authors proposed to predict the mode of a current MB based on previously coded MBs, then perform a fast ME for this predicted mode. If the resulting cost is less than an adaptively calculated threshold, the algorithm skips any further calculations for this block. Although the authors did not investigate the potential of their algorithm for HW implementation, the way they described it makes it less likely to be implemented in hardware. A possible candidate HW design should be able to handle the worst case scenario, which is to calculate all the complex R-D costs for all the seven modes. This task is too complex to be executed within a reasonable number of clock cycles that fits typical encoder systems requirements unless a huge (non realistic) design is implemented. The authors did not provide any computational requirements analysis to justify the time savings they are claiming. They also claim that they observed a maximum PSNR degradation of 0.4 *dB* in their experiments on CIF sequences, which is big enough to generate visually noticeable defects. Yu et al [8] proposed a strategy that incorporates temporal similarity detection as well as the detection of different moving features within a macroblock. Their experiments were performed on QCIF resolution only, and they showed up to 9% bitrate overhead with highly dynamic sequences. The authors did not provide any visual justification of the quality they are claiming. Also they did not show any potential for hardware implementation. In [9], the author proposed an algorithm that relies mainly on two predictive factors: intrinsic complexity of the MB and mode knowledge from the previous frame. The algorithm added more than 5% bitrate overhead for dynamic sequences, and the maximum achieved time saving for all the tested QCIF and CIF sequences was 30%.

Tourapis et al [10] proposed to extend and adapt the concept of the Enhanced Predictive Zonal Search (EPZS) motion estimation algorithms within the AVC standard. They proposed additional modifications to the predictor consideration, and introduced a new refinement pattern and a new iterative refinement process to improve the efficiency of the algorithms. The results of their experiments on selected CIF sequences showed a speedup in the motion estimation process (without providing enough computational requirements analysis to justify this speedup). However, in that paper, the authors did not introduce any enhancements in the mode decision process. In [11], the authors had more focus on mode decision. They proposed a fast mode decision mechanism by considering that the mode decision error surface is monotonic with partition block size. Their approach depends on searching specific block sizes first, then based on the values of the resulting cost parameters, a decision was made whether to perform an early termination or not. If not, all possible search modes are performed. This means that for highly dynamic sequences, the speedup of the mode decision process is expected to be negligible. The authors did not provide any computational requirements analysis or comparisons, and they reported a speedup in the encoding process of the examined CIF sequences, without describing the configurations of the reference (JM) they are comparing with. In [12], the authors proposed fast mode decision and motion estimation processes with main focus on MPEG–2/AVC transcoding. The algorithm utilizes the motion information from MPEG–2 for an AVC encoding using EPZS. The target application for this work had limited processing power; hence the authors ignored the small sub-partitions that AVC standard offers. This had its effect on the obtained results for the tested CIF sequences. Highly dynamic sequences showed more than 0.3 *dB* degradation in PSNR, with a speedup of an order of magnitude in the encoding time.

Lee et al [13] proposed an early skip mode decision as well as a selective intra mode decision. They reported a maximum improvement of 30% in the required encoding time. Ahmad et al [14] proposed a scheme that is based on a 3D recursive search algorithm and takes into consideration the motion vector cost and previous frame information. They also reported a maximum decrease of 30% in the required encoding time. However, they did not provide any quality assessments; neither did they provide any computational requirements analysis of their algorithm. In [15], Kim et al proposed an algorithm that uses the property of All Zero Coefficients Block (AZCB), obtained by quantization and coefficient thresholding, to skip unnecessary modes. The authors reported an average speedup of two times for the tested QCIF and CIF sequences. However, based on the algorithm flow that they provided, the algorithm is expected to lose its computational efficiency when applied to highly dynamic sequences, as no early skipping will be performed in this case, and all the inter-search modes will be required. Jiang et al [16] proposed a low complexity VBS ME algorithm for video telephony communication. Their technique included classifying macroblocks in a frame into types, and applying different searching strategies for each type. They based their cost calculations on SAD only, ignoring the rate component. They did not propose any modification to the MD module. Hence, the algorithm efficiency is limited to the usage of the proposed ME algorithm only. In spite of the achieved time savings compared to the full search approach, the reported results showed noticeable degradation in encoding time when compared to other fast ME algorithms. Also, the authors' experiments on QCIF and CIF sequences showed that the

algorithm is confined in applications with sequences of low motion and high temporal correlation.

Tanizawa et al [17] proposed a fast rate-distortion optimization method targeting low complex mode decision. The proposed method was based on a 2-step hierarchical mode decision. In the first step, a simple R-D cost without tentative coding is calculated. One or more coding candidates are selected using the obtained R-D cost. The number of candidates varies with the value of the quantization parameter (QP). In the second step, the conventional RDO method is applied to the candidates that have been chosen in the first step. The authors timing comparisons did not include the motion estimation operation. A disadvantage of this method is its unsuitability for hardware implementation due to the extremely complex RDO calculation that is applied for the successful candidates of the first step. Dai et al proposed an algorithm in [18] that limits the candidate modes to a small subset by pre-encoding a down-sampled small version of the image. They reported a 50% reduction in encoding time for the three tested CIF sequences. Kuo et al [19] also proposed a multi-resolution scheme and an adaptive rate-distortion model with early termination rules to accelerate the search process. In addition, the authors derived a rule to estimate the bits resulting from residual coding. The reported results showed significant improvement in the encoding time for the tested CIF sequences when compared with the exhaustive full search technique. However, the algorithm showed poor R-D performance when applied to highly dynamic sequences. Chen et al [20] proposed a fast bits estimation method for entropy coding to be used in RDO instead of calculating the actual bitrate, which is an extremely complex operation if performed for each candidate location. The algorithm is based on simplifying the CAVLC only, which makes it inapplicable to the Main- and High-profile encoders that are intended to be configured to use CABAC. The authors did not propose any simplification to the motion search and mode decision strategy, which makes a hardware implementation of their method unpractical.

Rhee et al [21] introduced one of the earliest proposals to use VBS ME. In their work, they proposed two algorithms. One of them was extremely computationally extensive and they proposed to use it as a reference by the following research work in the field, while the other was a simpler version that is based on heuristics. Their work relied on local motion information. A set of candidate motion vectors of each fixed size small block is first obtained by full search whose matching error is less than a prescribed threshold. Neighbouring blocks are then merged in a quad-tree manner if they have at least one motion vector in common. The only modes the algorithm supported were the square modes (4×4, 8×8, and 16×6). Tu et al extended this work in [22] by taking the quantization parameter and the Lagrange cost function into consideration to determine the threshold for comparing the distance between the motion vectors of the two blocks. The experimentation was performed on QCIF and CIF sequences. No timing or computational requirements analysis was introduced. The authors proposed to start with 8×8 searches, and then merge all the way up in the tree to obtain larger block sizes. Hence the supported modes were (8×8, 16×8, 8×16, and 16×16) only. The authors then proposed in [23] a merge and split scheme, which gave the algorithm the ability to represent all search modes. They proposed to perform a "simple" refinement search after every merge or split operation. This extra search would introduce relatively large complexity overhead to the motion estimation module if the complexity of the main motion estimation search is comparable. The obtained results for the tested QCIF and CIF sequences were obtained at QP = 30 only and no rate distortion curves were given. In [24], the authors used the same merging scheme, with the possibility of excluding some low-probability modes in the mode decision process. Again, this extra search that is required after each merging or splitting would introduce relatively large complexity overhead to the motion estimation module if the complexity of the main motion estimation search is comparable.

Ates et al [25] proposed a hierarchical block-matching-based motion estimation algorithm that uses a common set of SAD computations for motion estimation of different block sizes. Based on the hierarchal prediction and the median motion vector predictor of AVC, the algorithm defines a limited set of candidate vectors; and the optimal motion vectors are chosen from this common set. The authors showed their complexity analysis; however, they did not support it with experimental timing measurements. This algorithm showed acceptable PSNR degradation and increase in required bitrates for the tested QCIF, CIF, and SIF sequences. The next section describes the proposed VBS selection scheme, accompanied with some theoretical computational requirements analysis to show its effectiveness.

III. THE PROPOSED VBS MODE DECISION ALGORITHM

The proposed variable block size motion estimation and mode decision algorithm is based on a merging scheme. It is a bottom-up approach approximation method that exploits the correlation of the smaller blocks motion vectors in a uniform or close motion vector field to build up the larger blocks. The algorithm uses the available typical motion estimation information such as predictor, refinement displacement, and cost. The cost here maybe directly considered as the R-D Cost described in (3) assuming that it is going to be pre-calculated in the motion estimation engine and passed as an input to the algorithm, or it might be considered as the SAD requiring the calculation of the R-D Cost within the algorithm module. A description of the main data structure of the algorithm is given in Section A. In Section B, a theoretical computational requirements analysis is introduced to emphasise the computational efficiency of the proposed algorithm. Finally Section C describes the flow of the proposed algorithm.

A. The Main Data Structure of the Proposed Algorithm

The main data structure of the algorithm is shown in Figure 7. It is mainly a decision tree that is used to decide about the "suitable" neighbouring nodes to approximately merge them upwards forming a parent node. The algorithm is based on the observation that, if the cost of a parent block is higher than the sum of costs of the children blocks, then the even larger block-size modes can be excluded. Each node in the tree represents a "legal" block partition. A node is represented by its best-representing motion vector that have been calculated by block-based motion estimation (or interpolated by merging), the predictor used as the anchor to start searching around, and the R-D cost accompanied with this motion vector and motion vector predictor.

B. Computational Complexity Analysis

The tree consists of four decision-levels and five nodes-levels as shown in Figure 7. Roughly, it can be assumed that all the *even* nodes-levels of the tree (level 0, level 2, and level 4) require approximately the same computational requirements (*1x* each level) to find the best motion vectors for all the nodes of a level and to calculate their accompanied costs, while the two *odd* nodes-levels of the tree (level 1 and level 3) require approximately a computational requirements of (*2x* each level) to find the best motion vectors and costs.

The overall ME/MD module computational requirements can be viewed as the sum of the VBS best motion vector search computational requirements and the VBS mode decision computational requirements. This is shown in (9).

$$Comp\_Req_{\_Total} = Comp\_Req_{\_MV\_Searches} + Comp\_Req_{\_VBS\_MD} \quad (9)$$

Hence, the overall inter-related computational requirements of the ME/MD module in the JM can be estimated using Figure 5 and Equation (9). Due to the exhaustive nature of the direct JM implementation, it is required to perform all the types of searches for all possible modes, which is equivalent to *7x*, in order to find the best match for each and every block type. Also, it is required to find the best partitioning scheme among all the 259 possible combinations, a relatively complex operation, whose computational requirements will not be described in details due to its dependence on the way it is implemented. Due to the dominance of the search computational requirements over the mode decision one, any reference to (JM 7x) in the remaining part of this paper is a reference to the exhaustive-search JM implementation of the ME/MD module.

By nature, the proposed merging tree makes the ME/MD module simpler than the JM implementation due to its hierarchical mode decision approximation scheme (simpler mode decision operation). However, as mentioned above, the dominant part that determines the overall module computational requirements is the influence it has on the computational requirements of the motion vector search part (first parameter of (9)). Therefore, the most effective cost reduction technique for the overall module would come by reducing the required number of searches. This led to the idea of allowing the algorithm to meet a wider base of market demands by making it adjustable to work at three different operating points, trading between computational requirements and R-D performance. The key concept is to reduce the required searches as much as possible, while still keeping the R-D degradation within the different allowance windows offered by different customers. Figure 8 describes the difference between the three operating points. The encircled levels refer to levels where the motion vectors of all their nodes are found by performing block-based searches. The arrows refer to the interpolated levels and where they are interpolated from. For *operating point 1*, the algorithm requires performing the 4×4, 8×8, and 16×16 types of searches. This corresponds to a search computational requirements of *3x* (based on the metric mentioned above), which is less than half the computational requirements of the JM exhaustive implementation (JM 7x). The motion vectors and the predictors of the other types of modes (4×8, 8×4, 8×16, and 16×8) are "interpolated" using the actually-calculated motion data of the child nodes in the levels that directly precede those levels in the merging tree structure. The algorithm working at this operating point would fit the requirements of high-resolution applications, such as SD and HD broadcasting. *Operating point 2* is the second mode of the algorithm. It requires performing 4×4 and 8×8 types of searches only, reducing the search computational requirements to *2x*, while the motion vectors and the predictors of the other types of modes (4×8, 8×4, 8×16, 16×8, and 16×16) are "interpolated" using the tree structure. *Operating point 3* is also introduced to meet the requirements of applications such as simple handheld devices where reduced computational requirements (and power consumption) is the main priority. This mode requires performing 4×4 searches only (level 0 of the tree), reducing the search computational requirements to *1x*, while the motion vectors and the predictors of all the other types of modes (4×8, 8×4, 8×8, 8×16, 16×8, and 16×16) are "interpolated" in a bottom-up fashion all the way through the tree structure.

Table 3 shows the schemes that are going to be referred to in the next section, with a brief description of each. Note that JM 1x, JM 2x, and JM 3x, are generated by running the JM, disabling some of inter search/decision modes. They are mainly included to evaluate the performance of VBS 1x, VBS 2x, and VBS 3x (operating point 3, 2, 1) respectively compared to their corresponding almost-same-computational-requirements JM implementations (in addition to the main comparison with the optimum R-D performance of JM 7x). A '√' in the above table refers to an actual MV search and mode decision. A '–' refers to an absence of this block-type search and mode decision, while an 'M' refers to a block-type that is interpolated by merging of child nodes. The number of comparisons required for VBS 1x, 2x, and 3x mode decision are derived assuming that each merging rule requires a single comparison. The MD computational requirements of VBS 2x and 3x also include

the additions and comparisons required for early mode skipping, while the computational requirements of the control logic are not stated. The MD computational requirements for JM 2x and 3x are not given in details because of being implementation-dependent.

C. Description of the Proposed Algorithm

In this section, a description of the algorithm that has been implemented is provided. The coverage of the algorithm to the three operating points mentioned above is also discussed. Figure 9 shows the flowchart of the algorithm. As can be seen, at operating point 3, the tree structure is parsed (checked for merging) starting from its level 0 (L0) all the way up to its level 4 (L4). At operating point 2, the sum of costs (using (3)) of each four neighbouring 4×4 nodes (each L0 quad) is compared with the corresponding 8×8 cost to decide whether to start with 4×4 quads followed by merge-checking for one level up, or to start with the 8×8 level (L2) followed by merge-checking for two levels up. Operating point 1 differs from operating point 2 by having an additional check for possible early decision to choose 16×16 mode, which makes the last-level merge-checking stage unnecessary. In Figure 9, it is assumed that the nodes costs are calculated on the spot; however, they can be passed as inputs to the VBS module as they have already been calculated during the search for best motion vectors. A key element that determines the efficiency of the above merging scheme is how accurate the merging rules are. On the other hand, its computational requirements are very influential in determining the second parameter in (9). Note that based on Figure 7, different rules maybe used for merge-checking of different nodes. However, for simplicity, the same merging rule is initially presumed to be applicable for all pairs of blocks. Other varieties of the algorithm may be applicable, such as changing the rule according to the level it is located in, or giving some nodes more priority than others by subjecting them to more accurate merging rules. It is clear that a suitable rule would be an accurate, yet simple one. The steps of the merge-checking rule are provided in Figure 10.

It is worth mentioning that for generality, the shown pseudo-code is based on the assumption that each searched node has been searched using N_MV different motion vectors predictors (N_MV is assumed to be 4 as an example), or that N_MV successful candidates are elected by the ME search engine rather than only the best one per search. This means that each search node will be initially marked by 4 best motion vectors and 4 motion vector predictors, which would help avoiding falling into local minima. For simplicity, all the testing results mentioned in the next section have been generated after assigning N_MV to 1. Also it was found that using Th_x = Th_y = 0 is a reasonable choice. The next section shows the experimental analysis and results obtained by encoding various benchmark video sequences.

IV. Experimental Analysis and Results

In this section, results of the performed experiments to evaluate the performance of the proposed algorithm are introduced. The main goal is to compare the algorithm at its three operating points to the different JM references mentioned in the previous section. Section A describes the method and experimentation environment, while Section B shows the detailed comparisons with the brutal-force method that is adopted by the reference software in terms of time complexity and R-D performance.

A. Method and Experimentation Environment

The evaluation process is done by performing a comprehensive test where various video sequences are encoded by seven versions of the encoder (see Table 3) throughout a specific range of Quantization Parameter (QP). The goal is to plot the seven generated R-D curves for each sequence to sense the closeness of the R-D performance of the encoder working with the proposed VBS ME/MD algorithm, to the optimum exhaustive solution. Besides, the different encoding times of various sequences with the tested versions of the reference software will be given, and compared to the "theoretical" computational requirements analysis that has already been introduced in Section B of the previous section. The sequences were selected to represent a variety of motion types. This helps in creating a wide range of input stimuli to test the algorithm behaviour. In order to cover a wider range of rate and quality requirements during the testing process, QP was set to vary throughout a broad range of values.

B. Obtained Results

This section starts by showing execution-time measurements and results in subsection 1, followed by the rate-distortion results in subsection 2.

1) Execution Time Measurments and Results

All the results that are discussed in this section have been obtained by running the seven versions of the AVC encoder (defined in Table 2) on a unified platform. Table 4 shows the time spent by each of the seven versions of the software on encoding ten frames of each of the tested sequences (with QP = 30). The values between parentheses represent the savings in computational time with respect to the JM7x version. The table shows that using VBS 1x, VBS 2x, or VBS 3x, reduces the encoding time of all the tested sequences at least by more than half when compared to the required encoding time for the pure JM 7x version. This conforms to the theoretical computational requirements analysis that has been discussed in Section B of the last section. The total encoding time for VBS 1x, VBS 2x, and VBS 3x is almost the same as for JM 1x, JM 2x, and JM 3x respectively (with minor increase due to the extra comparisons and additions). However, the next section shows that the improvement of the VBS *n*x algorithms over

the corresponding JM *n*x ones in terms of R-D performance is clear enough to neglect this minor complexity overhead, especially when targeting low bitrate applications. Also, the table shows that most of the encoding time is being spent on the motion estimation calculation, which conforms to the encoder complexity profile that was shown in Figure 6. Figure 11 shows a graphical representation of the obtained results. The results show that the more motion a sequence contains, the more time it spends in motion estimation, which in turns translates to an overall increase in encoding time.

2) Rate-Distortion Measurments and Results

Figure 12 to Figure 14 show the R-D behaviour of the tested versions of the encoder, with an emphasis on the low-bitrate region. The figures show that the generated graphs are consistent with the expected behaviour of the algorithm. It is clear that for all the sequences, the lower the bitrate, the more effective the algorithm appears to act (at all operating points). This is because at lower bitrates, motion data have a comparable effect on bitrate to the residual data; hence any savings are highly sensible.

For all sequences, VBS 1x may be used as an optimized version of JM 1x. Though, its performance is relatively poor when compared to VBS 2x, VBS 3x, or JM 7x, it can be used as a reasonable compromise when the target application requires low complexity and low power system with reasonable R-D behavior. For all the examined sequences, JM 7x does not outperform VBS 3x by more than 0.2 *dB* at any bitrate (around the target bitrate that suits the sequence resolution).

VBS 1x, VBS 2x, and VBS 3x introduce enhancements over JM 1x, JM 2x, and JM 3x respectively. The enhancements become more sensible at low bitrates. For example, VBS 1x introduces huge enhancement (around 8 *dB*) over JM 1x when encoding the sequence "Mobile and Calendar QCIF 30 fps" (Figure 14) targeting as low bitrate as 29.5 Kbps. Also, the merging operation that was performed resulting in VBS 2x boosted the curve of JM 2x for the sequence "Miss America QCIF 30 fps" (Figure 13) by (6 *dB*) at 26.5 Kbps bitrate.

In summary, the experimental analysis and results demonstrate the main contribution of the proposed algorithm. It is mainly the ability to exhibit acceptable R-D behavior for different sequences with various types of motion. Nevertheless, the ME/MD computational requirements are less than half the computational requirements of ME/MD of JM 7x. This leads to faster encoding time on software platforms, as well as smaller (hence less expensive) implementations on hardware platforms.

V. CONCLUSION

Having the VBS ME tool in the AVC standard improves its coding efficiency significantly. However, it also introduces extreme computational requirements to the encoder. The JM (AVC software reference model) has an exhaustive approach to implement VBS ME/MD. All seven types of motion estimation searches are performed, and then in the mode decision step, an exhaustive search follows to choose the best partitioning scheme among all possible combinations. Knowing that VBS ME and MD typically represent from 70% to 90% of the entire encoder computational requirements, many research efforts have been introduced in the literature to reduce their computational requirements. However, most of the solutions were local to specific types of simplified motion estimation searches.

In this paper a computationally-efficient VBS selection scheme was introduced. The scheme is applicable to any VBS ME module, leading to significant reduction in its computational requirements with minor loss in the quality of the reconstructed picture. Three versions of the proposed algorithm have been introduced in order to meet different applications' demands. Evaluation experiments were performed on three benchmark video sequences with various spatial and temporal characteristics ranging from smooth slow motion, up to random fast motion. Timing analysis of the performed experiments showed that the proposed algorithm (with its three versions) reduces the encoding time of all the tested sequences at least by half when compared to the required encoding time for the pure brutal-force solution. Objective quality measurement is represented by R-D performance. It has shown that, for all the performed tests, VBS 1x, VBS 2x, and VBS 3x introduce enhancements over JM 1x, JM 2x, and JM 3x respectively, with minor computational overhead. In general, the proposed algorithm is mostly effective with low-power decoder devices, with reduced computational resources, especially when targeting low-bitrate video applications.

ACKNOWLEDGMENT

The authors would like to thank the Advanced Technology Information Processing Systems (ATIPS) Laboratory, Alberta Informatics Circle of Research Excellence (iCORE), the Alberta Ingenuity Fund (AIF), the Natural Sciences and Engineering Research Council of Canada (NSERC), CMC Microsystems, Micronet R&D, Canada Foundation for Innovation (CFI), the Department of Electrical and Computer Engineering at the University of Calgary, Xilinx Inc., and the German University in Cairo (GUC) for supporting this research.

REFERENCES

[1] I. Amer, R. Hamed, W. Badawy, G. Jullien, and J. W. Haslett, "An Enhanced Lenient Merging Scheme for H.264 Variable Block Size Selection", proceedings of *International Conference on Advances in Multimedia (MMEDIA)*, Colmar, France, pp. 136-139, July 2009.

[2] S. Lin, C. Chang, C. Su, Y. Lin, C. Pan, and H. Chen, "Fast Multi-Frame Motion Estimation and Mode Decision for H.264 Encoders," Proceedings of *IEEE International Conference on Wireless Networks, Communications and Mobile Computing*, Vol. 2, pp. 1237-1242, June 2005.

[3] G.J. Sullivan and T. Wiegand, "Rate-Distortion Optimization for Video Compression," *IEEE Signal Processing Magazine*, Vol. 15, pp. 74 - 90, November 1998.

[4] W. Chung, "Implementing the H.264/AVC Video Coding Standard on FPGAs," *A white paper*. [Online]. Available:

[5] www.xilinx.com/publications/solguides/be_01/xc_pdf/p18-21_be1-dsp4.pdf

[6] Y.-W Huang, B.-Y Hsieh, S.-Y. Chien, S.-Y. Ma, and L.-G Chen, "Analysis and Complexity Reduction of Multiple Reference Frames Motion Estimation in H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 16, No. 4, pp. 507-522, April 2006.

[7] D. Wu, S. Wu, K. Lim, F. Pan, Z. Li, X. Lin, "Block INTER Mode Decision for Fast Encoding of H.264," Proceedings of *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 3, pp. 181-184, May 2004.

[8] A. Yu and G. Martin, "Advanced Block Size Selection Algorithm For Inter Frame Coding in H.264/MPEG–4 AVC," Proceedings of *IEEE International Conference on Image Processing*, Vol. 1, pp. 95-98, October 2004.

[9] A. Yu, "Efficient Block-Size Selection Algorithm for Inter-Frame Coding in H.264/MPEG–4 AVC," Proceedings of *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 3, pp. 169-172, May 2004.

[10] H. Tourapis and A. Tourapis "Fast Motion Estimation within the H.264 CODEC," Proceedings of *IEEE International Conference on Multimedia and Expo*, Vol. 3, pp. 517-520, July 2003.

[11] P. Yin, H. Tourapis, A. Tourapis, and J. Boyce, "Fast Mode Decision and Motion Estimation for JVT/H.264," Proceedings of *IEEE International Conference on Image Processing*, Vol. 3, pp. 853-856, September 2003.

[12] X. Lu, A. Tourapis, P. Yin, and J. Boyce, "Fast Mode Decision and Motion Estimation for H.264 with a Focus on MPEG–2/H.264 Transcoding," Proceedings of *IEEE International Symposium on Circuits and Systems*, Vol. 2, pp. 1246-1249, May 2005.

[13] J. Lee and B. Jeon, "Fast Mode Decision for H.264," Proceedings of *IEEE International Conference on Multimedia and Expo*, Vol. 2, pp. 1131-1134, June 2004.

[14] A. Ahmad, N. Khan, S. Masud, and M.A. Maud, "Efficient Block Size Slection in H.264 Video Coding Standard," *IEE Electronics Letters*, Vol. 40, No. 1, pp. 19-21, January 2004.

[15] Y.-H. Kim, J.-W. Yoo, S.-W. Lee, J. Shin, J. Paik, and H.-K. Jung, "Adaptive Mode Decision for H.264 Encoder," *IEE Electronics Letters*, Vol. 40, No. 19, pp. 1172-1173, September 2004.

[16] Y. Jiang, S. Li, and S. Goto, "A Low Complexity Variable Block Size Motion Estimation Algorithm for Video Telephony Communication," Proceedings of *IEEE International Midwest Symposium on Circuits and Systems*, Vol. 2, pp. 465-468, July 2004.

[17] A. Tanizawa, S. Koto, T. Chujoh, and Y. Kikuchi, "A Study on Fast Rate-Distortion Optimized Coding Mode Decision for H.264," Proceedings o*f IEEE International Conference on Image Processing*, Vol. 2, pp. 793-796, October 2004.

[18] Q. Dai, D. Zhu, and R. Ding, "Fast Mode Decision for Inter Prediction in H.264," Proceedings of *IEEE International Conference on Image Processing*, Vol. 1, pp. 119-122, October 2004.

[19] C.-H. Kuo, M. Shen, and C.-C. Kuo. "Fast Inter-Prediction Mode Decision and Motion Search for H.264," Proceedings of *IEEE International Conference on Multimedia and Expo*, Vol. 1, pp. 663-666, June 2004.

[20] Q. Chen and Y. He, "A Fast Bits Estimation Method for Rate-Distortion Optimization in H.264/AVC," Proceedings of *Picture Coding Symposium*, December 2004.

[21] I. Rhee, G.R., Martin; S. Muthukrishnan, and R.A. Packwood, "Quadtree-Structured Variable-Size Block-Matching Motion Estimation with Minimal Error," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 1, pp. 42-50, February 2000.

[22] Y.-K. Tu, J.-F. Yang, Y.-N. Shen, and M.-T. Sun, "Fast Variable-Size Block Motion Estimation Using Merging Procedure with an Adaptive Threshold," Proceedings of *IEEE International Conference on Multimedia and Exp*o, Vol. 2, pp. 789-792, July 2003.

[23] Z. Zhou, M.-T. Sun, and Y.-F. Hsu, "Fast Variable Block-Size Motion Estimation Algorithms Based on Merge and Split Procedures for H.264/MPEG–4 AVC," Proceedings of *IEEE International Symposium on Circuits and Systems*, Vol. 3, pp. 725-728, May 2004.

[24] Z. Zhou and M.-T. Sun, "Fast Macroblock Inter Mode Decision and Motion Estimation for H.264/MPEG–4 AVC," Proceedings of *International Conference on Image Processing*, Vol. 2, pp. 789-792, October 2004.

[25] H.F. Ates and Y. Altunbasak, "SAD Reuse in Hierarchical Motion Estimation for the H.264 Encoder," Proceedings of *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 2, pp. 905-908, March 2005.

[26] Y.-W. Huang, T.-C. Wang, B.-Y. Hsieh, and L.-G. Chen, "Hardware Architecture Design for Variable Block Size Motion Estimation in MPEG–4 AVC/JVT/ITU-T H.264," Proceedings of *IEEE International Symposium on Circuits and Systems*, Vol. 2, pp. 796-799, May 2003.

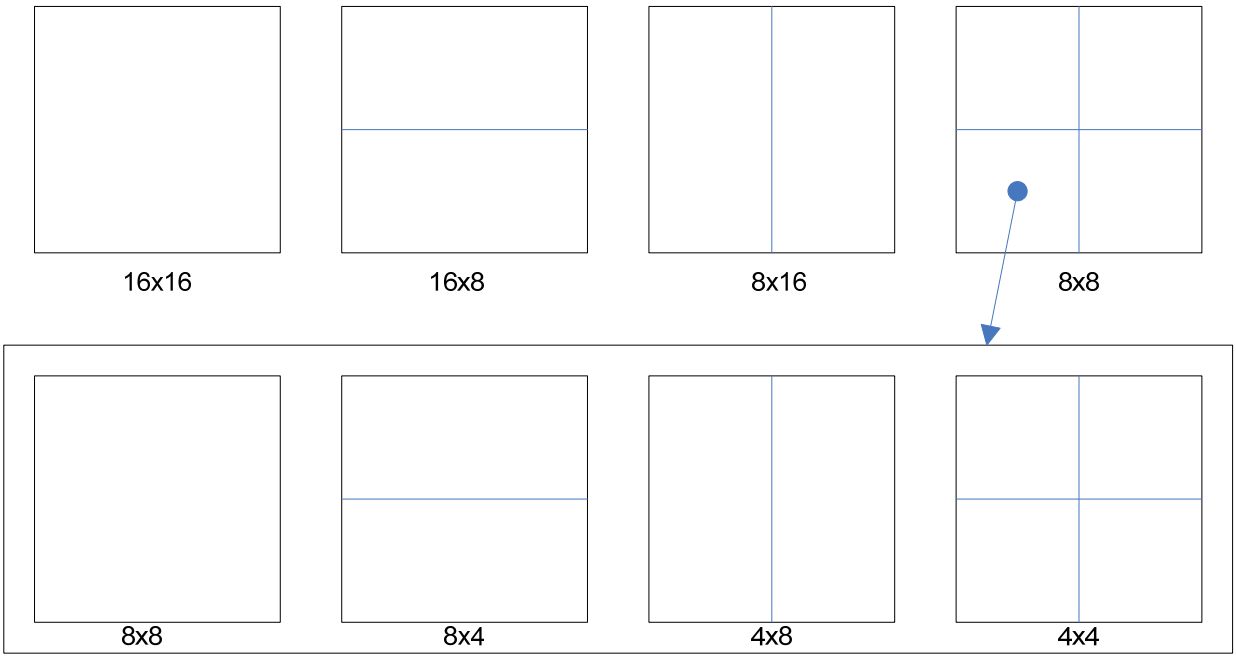


Figure 1. Variable block sizes defined in AVC

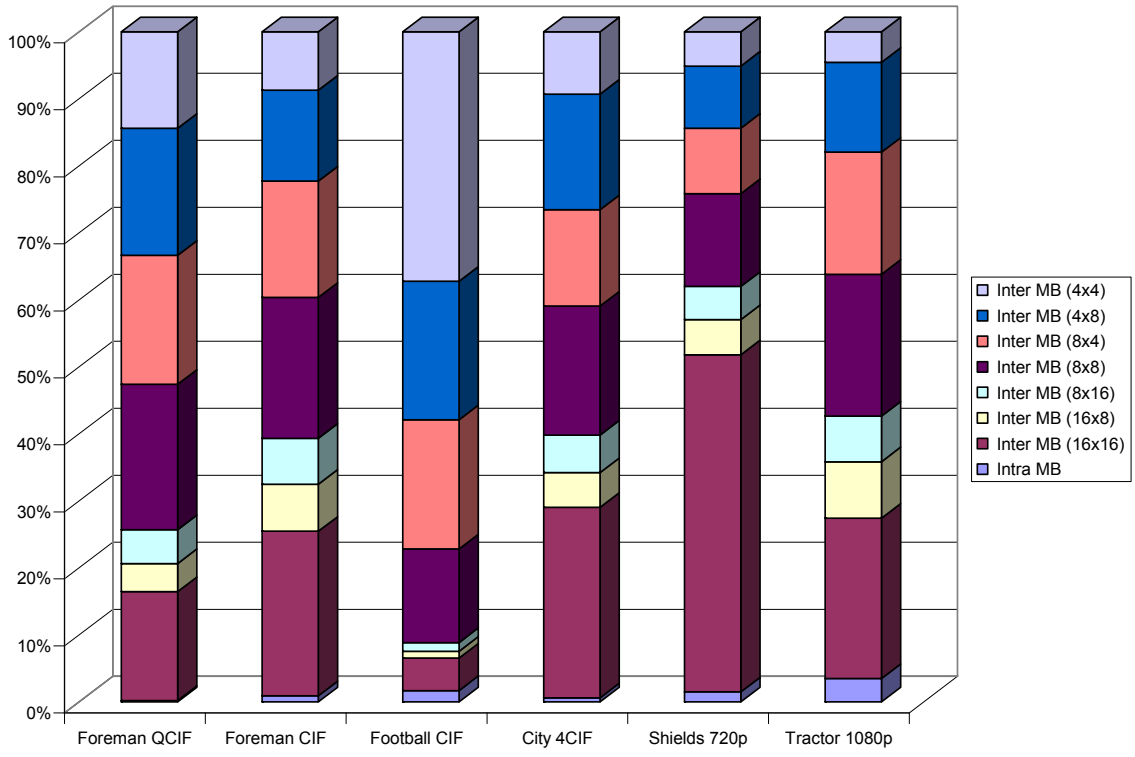


Figure 2. Optimal distribution of the various block sizes in the inter-predicted frames

Table 1. Summary of the characteristics of the tested sequences

Sequence	Type	Resolution	Frame Rate	Description of the first ten frames	Type of motion
Foreman	QCIF	176×144	30 fps	A man talking to a still camera	Slow limited motion
Foreman	CIF	352×288	30 fps	A man talking to a still camera	Slow limited motion
Football	CIF	352×288	30 fps	A part of a football game	Extensive motion
City	SD (4CIF)	704×576	30 fps	A scene of a city taken with a panning camera	Regular motion
Shields	HD (720p)	1280×720	60 fps	A person pointing at a group of shields while the camera is panning	Camera shooting of highly textured scenes
Tractor	HD (1080p)	1920×1080	60 fps	A Tractor working at field	Camera shooting of very high resolution

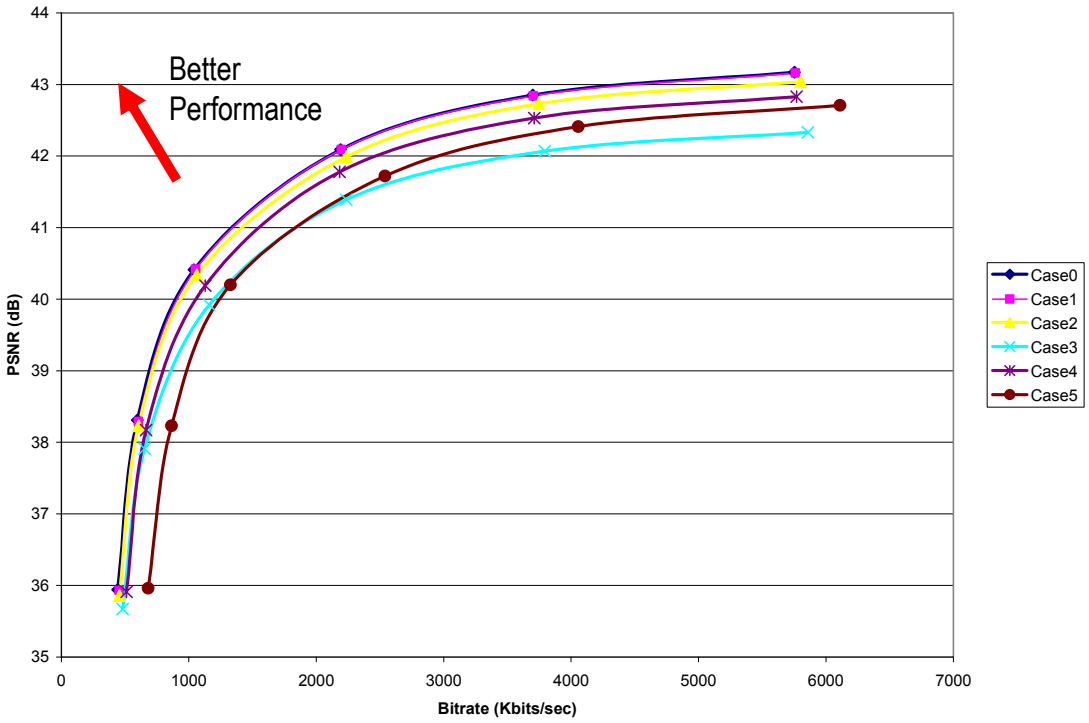


Figure 3. R-D behaviour of Foreman CIF (30 fps) with various inter searches/decision-modes enabled

Table 2. Description of the different curves in Figure 3

Curve	Description
Case 0	16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4 search/decision modes are enabled
Case 1	16×16, 8×8, 8×4, 4×8, and 4×4 search/decision modes are enabled
Case 2	16×16, 8×8, and 4×4 search/decision modes are enabled
Case 3	Only 16×16 search/decision mode is enabled
Case 4	Only 8×8 search/decision mode is enabled
Case 5	Only 4×4 search/decision mode is enabled

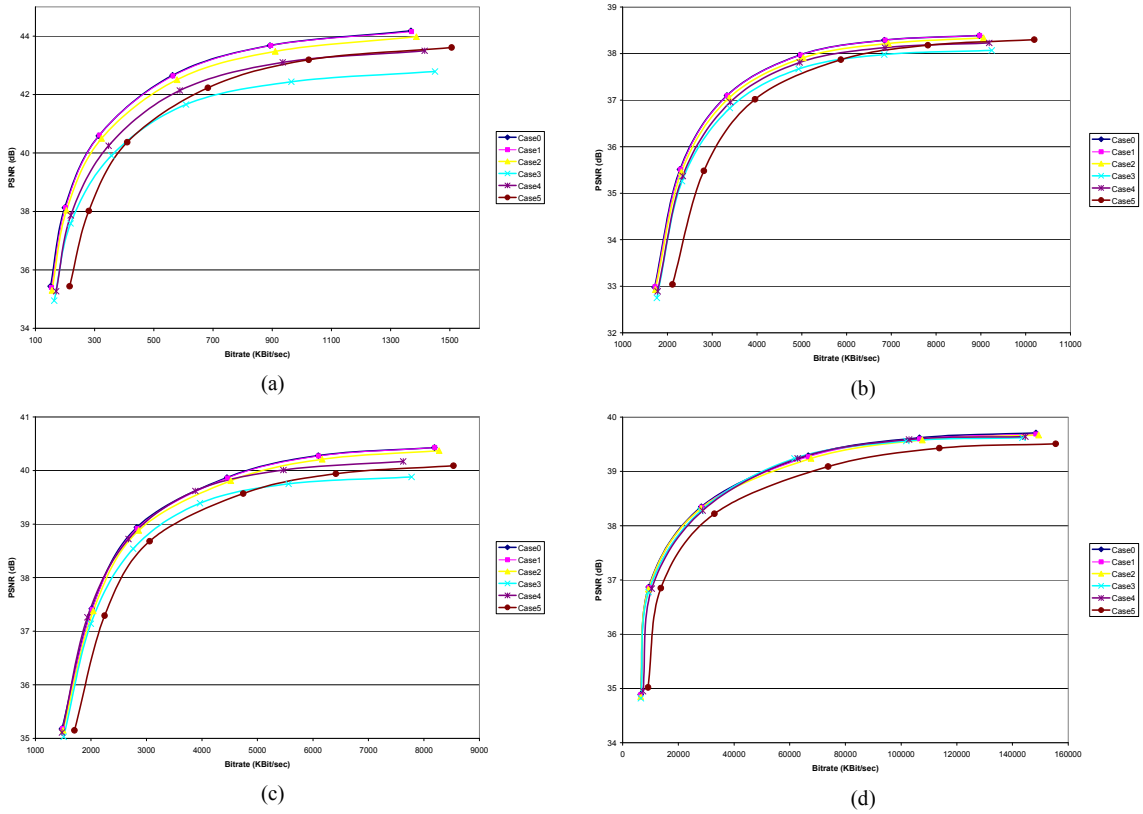


Figure 4. R-D behaviour of different sequences with different inter searches/decision-modes enabled
(a) Foreman QCIF (30 fps)
(b) Mobile CIF (30 fps)
(c) Football CIF (30 fps)
(d) Shields HD 720p (60 fps)

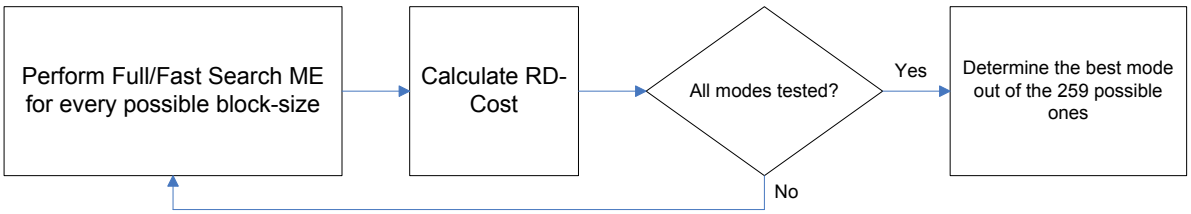
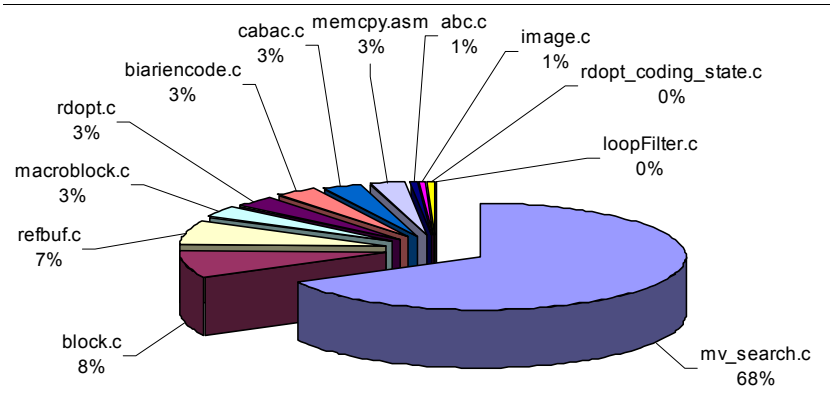
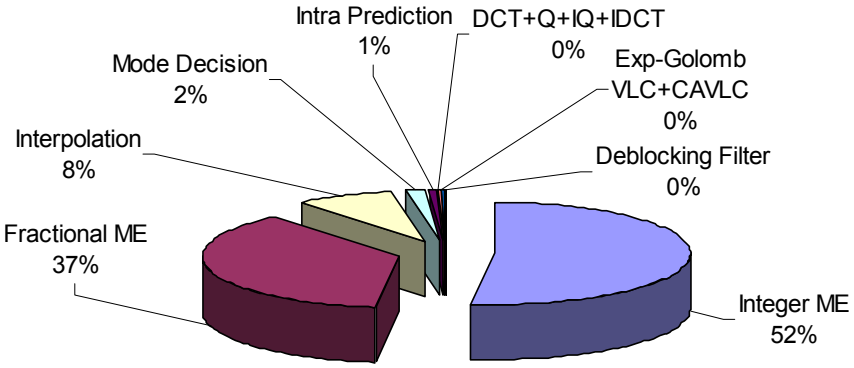


Figure 5. Exhaustive search for best partition scheme as adopted in AVC reference software



(a)



(b)

Figure 6. AVC encoder computational complexity
(a) profiled by files (from [4])
(b) profiled by functional modules (from [6])

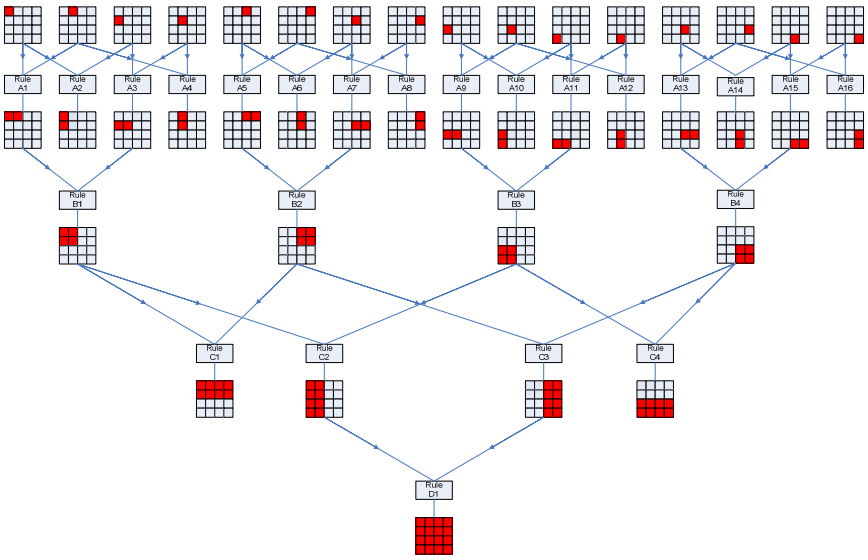


Figure 7. The main data structure of the algorithm: The merging tree

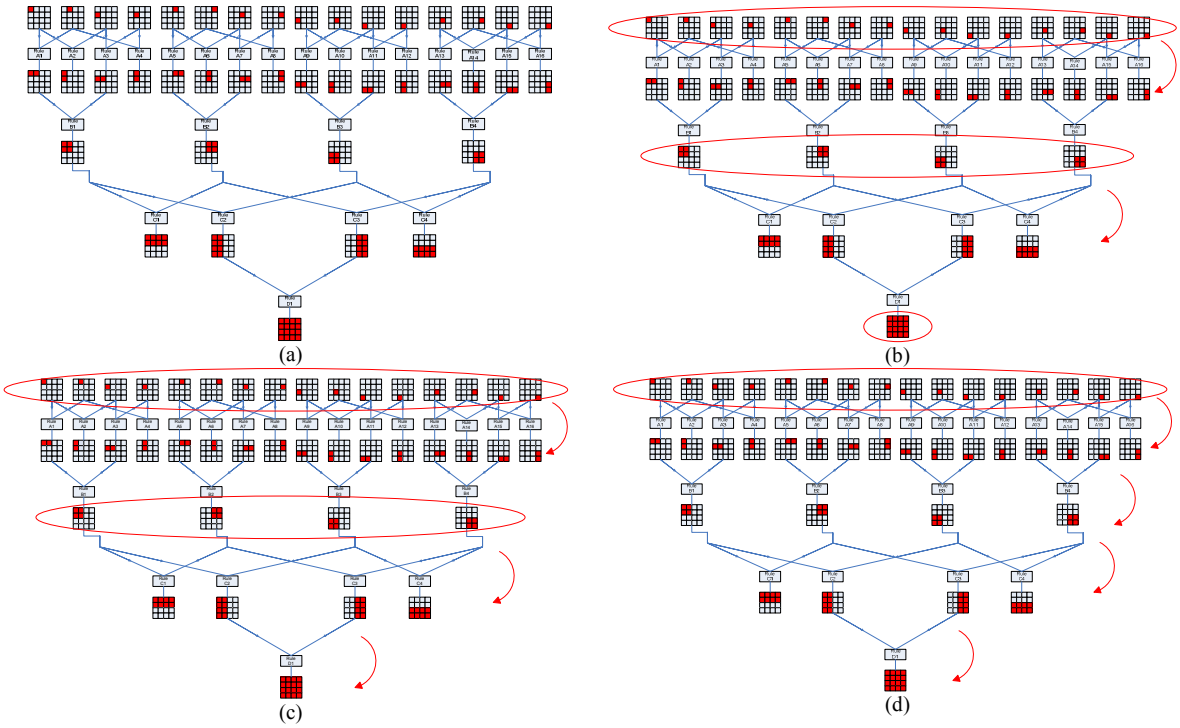


Figure 8. Required searches and interpolations for the algorithm's three operating points

(a) The merging tree (b) Operating point 1 (VBS 3x)
(c) Operating point 2 (VBS 2x) (d) Operating point 3 (VBS 1x)

Table 3. Required searches and computational complexities for the tested references

Ref.	Search/Decision Modes							Computational Requirements
	4×4	8×4	4×8	8×8	16×8	8×16	16×16	
JM 7x	√	√	√	√	√	√	√	7x MV search + searching 259 combs for MD
JM 3x	√	–	–	√	–	–	√	3x MV search + searching 17 combs for MD
JM 2x	√	–	–	√	–	–	–	2x MV search + searching 16 combs for MD
JM 1x	√	–	–	–	–	–	–	1x MV search + No mode decision
VBS 3x	√	M	M	√	M	M	√	3x MV search + 26 comps + 18 adds (for MD)
VBS 2x	√	M	M	√	M	M	M	2x MV search + 25 comps + 12 adds (for MD)
VBS 1x	√	M	M	M	M	M	M	1x MV search + 25 comps (for MD)

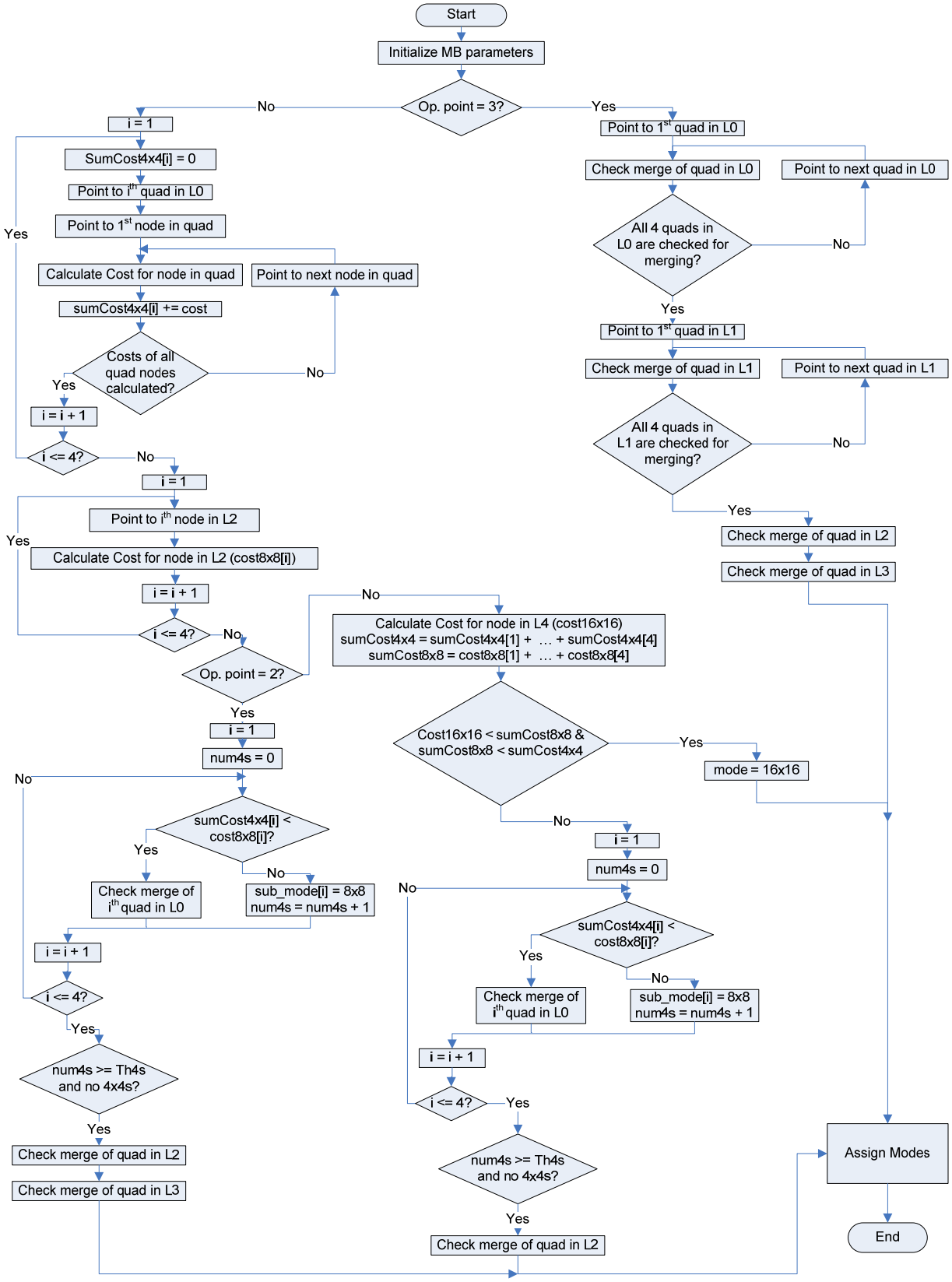


Figure 9. Flowchart of the proposed algorithm

1. If one of the nodes is "unavailable", then the parent node is also "unavailable", else:

2. Calculate the number of identical (or *semi-identical*) MVs in the pool of candidate MVs of each of the two nodes (4×4 = 16 possible pairs of MVs combinations). A simple rule to identify semi-identical MVs would be:

```
D_MV_x = abs(MV_x1 - MV_x2);
D_MV_y = abs(MV_y1 - MV_y2);

n_semi_iden_MV = 0;
If ((D_MV_x <= Th_x) && (D_MV_y  <= Th_y)){
    semi_identical = true;
    n_semi_iden_MV++;
    }
  Else
semi_identical = false;
```

This operation should be repeated to count the number of semi-identical MVs between the two nodes under test (n_semi_iden_MV).

3. Decide whether to merge the two nodes under test or not based on the following rule:

```
If (0<QP<12)                  merge = ((n_semi_iden_MV >12)?true:false);
  Else if (13<QP<25)merge = ((n_semi_iden_MV>9)?true:false);
  Else if (26<QP<38)merge = ((n_semi_iden_MV >6)?true:false);
Else                          merge = ((n_semi_iden_MV _MV>3)?true:false);
```

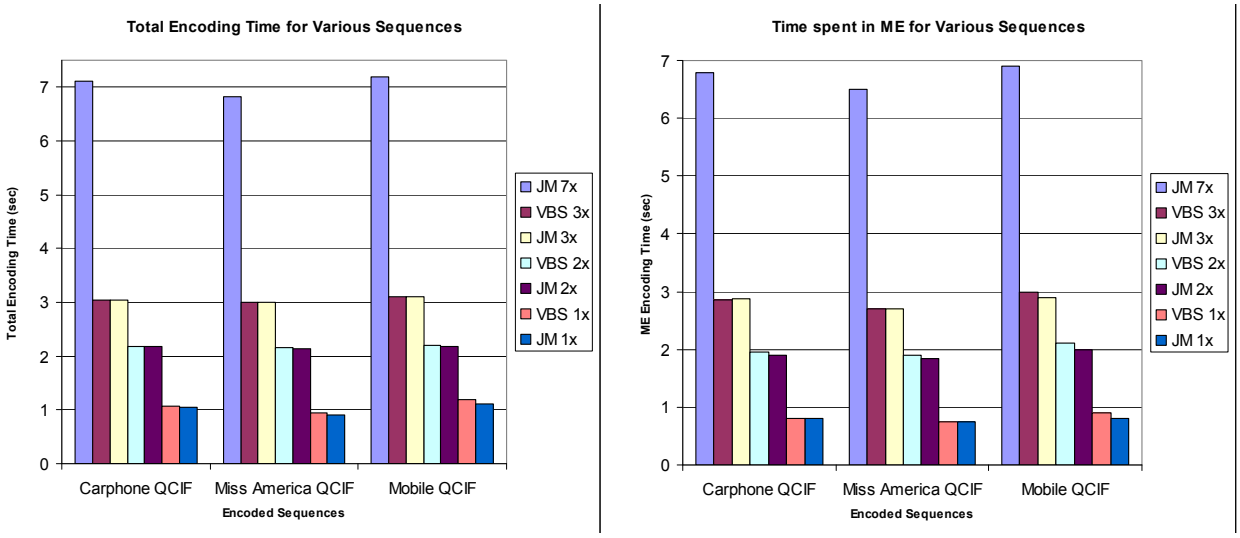
4. If the two nodes are chosen to be merged, then the parent node is marked as "available". The average MVs of the best 4 pairs of semi-identical MVs will be assigned to the parent node, This will be used to decide if this parent node is to be merged with its neighbour next-level node or not.

5. If the two blocks are not to be merged, then each of them will be marked with its best MV out of its 4 candidate ones, and the parent node will be marked as "unavailable".

Figure 10. Steps of the merge-checking rule

Table 4. Encoding time of the tested sequences via the seven versions of the reference software

		Carphone QCIF (30 fps)	Miss America QCIF (30 fps)	Mobile QCIF (30 fps)
JM 7x	Total Enc. time (sec)	7.11	6.83	7.2
	Time spent on ME (sec)	6.794	6.5	6.9
VBS 3x	Total Enc. time (sec)	3.302 (53.56%)	3.01 (59.93%)	3.11 (56.81%)
	Time spent on ME (sec)	2.861 (59.76%)	2.7 (58.46%)	3.0 (56.52%)
JM 3x	Total Enc. time (sec)	3.049 (57.12%)	2.99 (56.22%)	3.1 (56.94%)
	Time spent on ME (sec)	2.877 (57.65%)	2.71 (58.31%)	2.9 (57.97%)
VBS 2x	Total Enc. time (sec)	2.172 (69.45%)	2.153 (68.48%)	2.2 (69.44%)
	Time spent on ME (sec)	1.952 (72.55%)	1.9 (70.77%)	2.1 (69.57%)
JM 2x	Total Enc. time (sec)	2.172 (69.45%)	2.14 (68.67%)	2.18 (69.72%)
	Time spent on ME (sec)	1.892 (72.15%)	1.85 (71.54%)	2.0 (71.01%)
VBS 1x	Total Enc. time (sec)	1.061 (85.08%)	0.95 (86.09%)	1.2 (83.33%)
	Time spent on ME (sec)	0.813 (88.33%)	0.75 (88.46%)	0.9 (86.96%)
JM 1x	Total Enc. time (sec)	1.046 (85.29%)	0.9 (86.83%)	1.1 (84.72%)
	Time spent on ME (sec)	0.797 (88.27%)	0.74 (88.62%)	0.8 (84.06%)



(a) (b)

Figure 11. Time spent by the seven versions of the software on encoding the tested sequence
(a) Total encoding time (b) ME encoding time

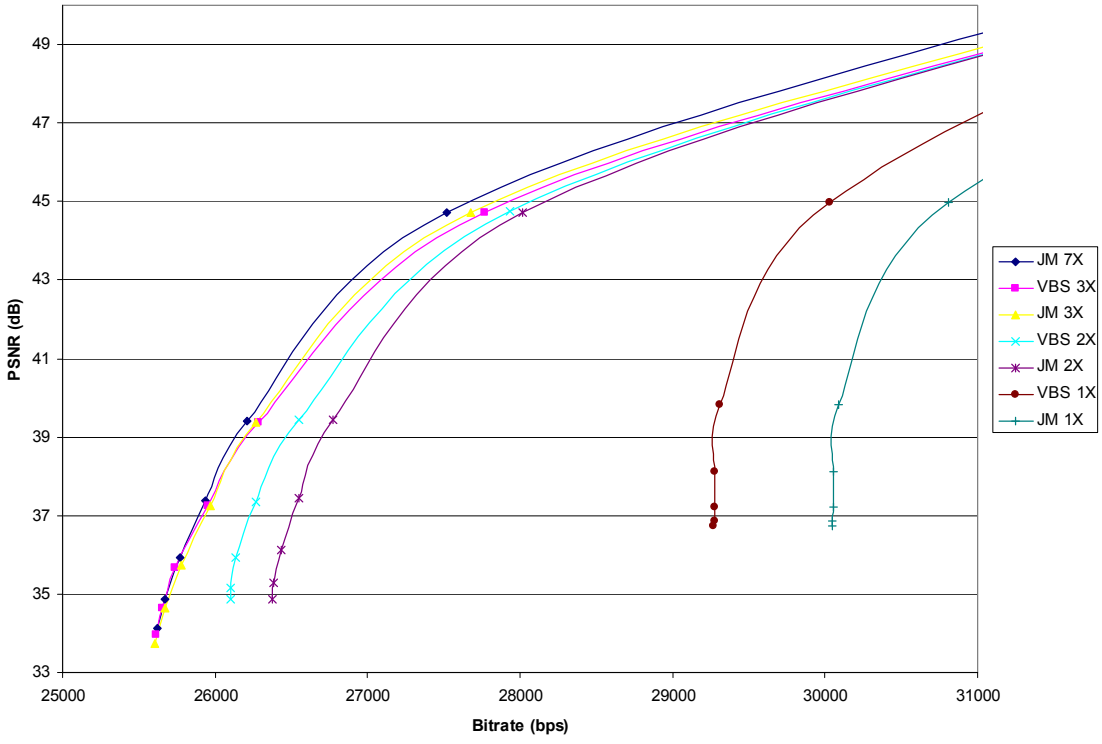


Figure 12. R-D behaviour for Carphone QCIF (30 fps) at low bitrate

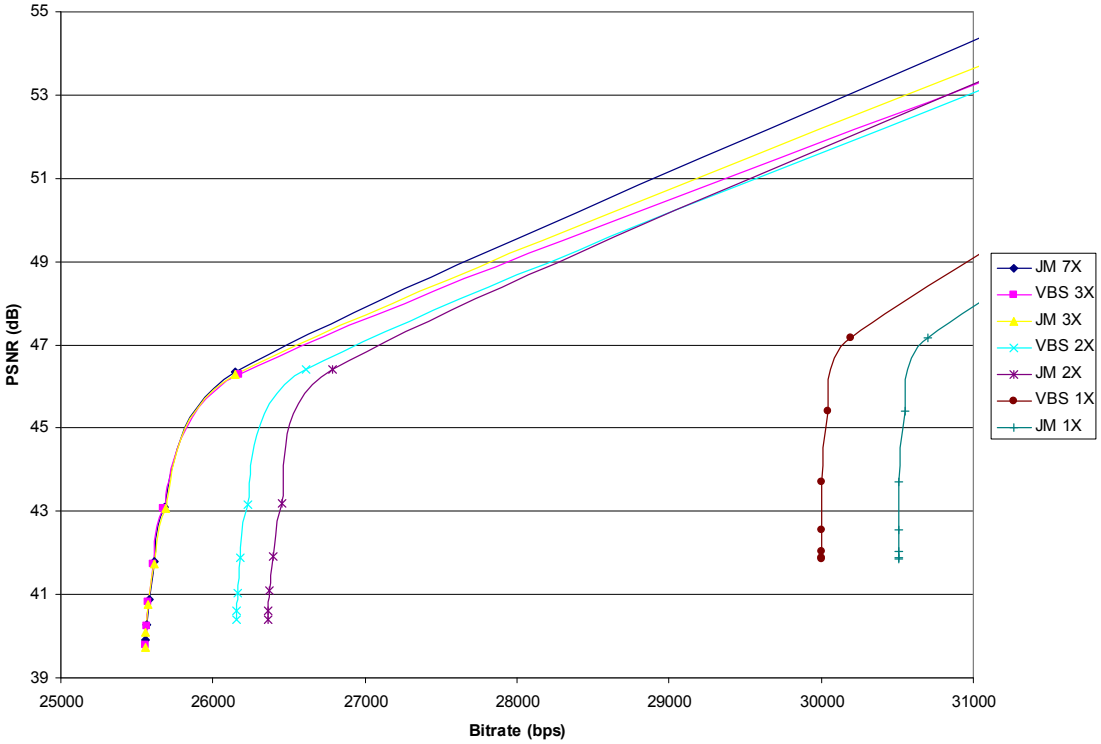


Figure 13. R-D behaviour for Miss America QCIF (30 fps) at low bitrate

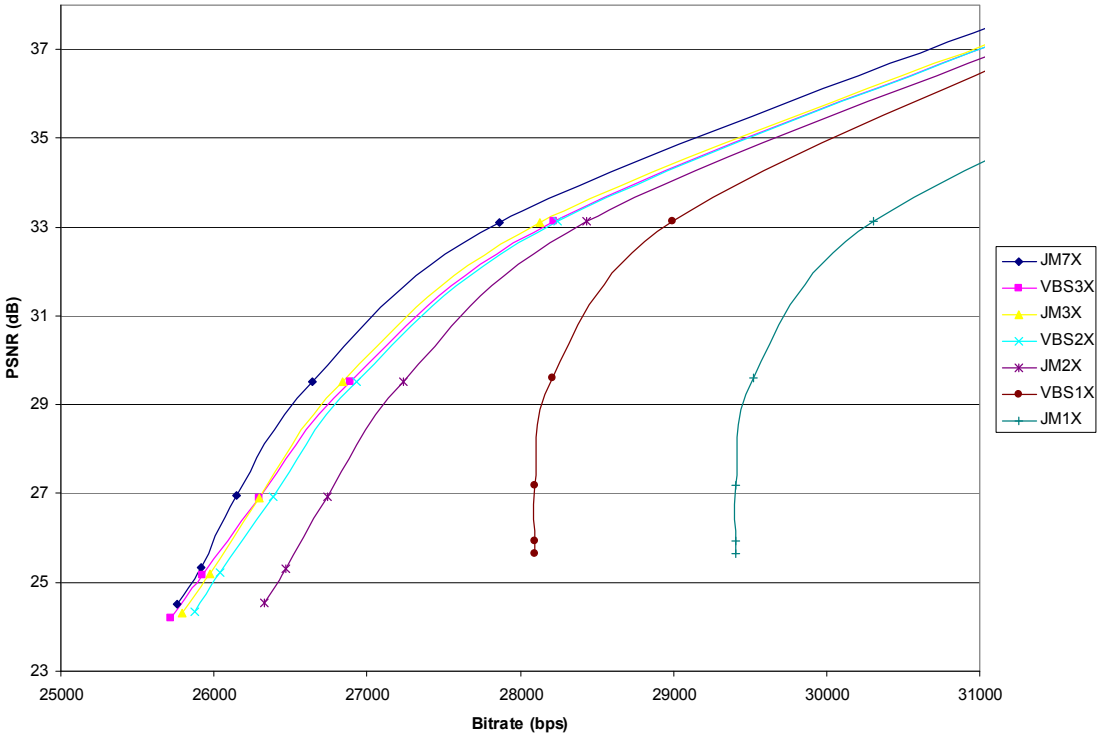


Figure 14. R-D behaviour for Mobile and Calendar QCIF (30 fps) at low bitrate