Dynamic Complexity Scaling for Real-Time H.264/AVC Video Encoding

Yuri V. Ivanov
School of Computer Science and Informatics
University College Dublin
Belfield, Dublin 4, Ireland
+(353) 1 716 29 15
yury.ivanov@ucd.ie

C. J. Bleakley
School of Computer Science and Informatics
University College Dublin
Belfield, Dublin 4, Ireland
+(353) 1 716 29 15
chris.bleakley@ucd.ie

ABSTRACT
The H.264 video encoding standard can achieve high coding efficiency at the expense of high computational complexity. Typically, real-time software implementation requires omission of most optional encoding tools leading to significantly reduced coding efficiency. This paper proposes a novel method for real-time H.264 encoding based on dynamic control of the encoding parameters to meet real-time constraints while minimizing coding efficiency loss. Experimental results show that the method provides up to 19% lower bit rate than conventional real-time encoding using fixed parameters with the same visual quality. The method allows real-time 30fps QCIF encoding on a Pentium IV with similar coding efficiency to full search baseline profile encoding.

Categories and Subject Descriptors
C.3 [**Special-Purpose and Application-Based Systems**]: *real-time and embedded systems.*

General Terms
Algorithms, Design.

Keywords
H.264, fast mode decision, complexity scaling, real-time video encoding.

1. INTRODUCTION

The H.264 standard [9] developed by the Joint Video Team (JVT) provides better coding efficiency than MPEG-4 and H.263 at low bit rates [17] but at the cost of significantly increased computational complexity. This makes it difficult to use software implementations of the encoder in practical real-time multimedia applications. Videophone conferencing, for example, requires every frame be encoded within 1/30 of a second to maintain a high quality frame rate and to provide low end-to-end delay.

MM'07, September 23–28, 2007, Augsburg, Bavaria, Germany.

A large number of algorithms have been proposed by researchers to reduce H.264 computational complexity. These algorithms are focused on new methods to reduce the complexity of the most computationally complex components of the video encoder, i.e., Motion Estimation (ME), Mode Decision (MD) and Discrete Cosine Transform (DCT) coding. In almost all cases, these algorithms are aimed at reducing total encoding time, rather than in meeting real-time constraints.

As in other video coding standards, H.264/AVC exploits the spatial, temporal and statistical redundancies of the source video. Since the amount of redundancy varies between frames, the computational complexity of, for example, mode decision can vary significantly between consequent frames. This effect is particularly important when using 'fast' encoding schemes in which MD and ME search are terminated early. Utilization of encoding tools such as Variable Block Sizes in MD, B-frames and multiple reference frames in ME can further increase variation in the processor workload and therefore create much greater variation in frame encoding complexity.

To the authors' knowledge, few papers consider the problem of dynamic complexity scaling in real-time H.264 video encoding, including utilization of frame buffers and maintaining constant processor workload. In contrast, most software solutions simply reduce the MD and ME search size globally, so as to meet real-time constraints in the worst case. This leads to significant reduction in coding efficiency.

The problem of real-time video encoding with dynamically varying computational complexity has a lot of similarities with the real-time rate-control problem [8]. That is, maintaining constant frame complexity in real-time encoding is similar to maintaining constant bit rate. Thus, this paper investigates the use of rate-control (RC) techniques for complexity control in real-time video encoding. The mode decision algorithm with dynamically variable complexity, proposed in this paper, is based on a MD class fast search algorithm and operates in real-time on an average Pentium IV PC. The technique for dynamic complexity scaling, proposed in the paper, is general and can be applied across a range of similar algorithms, such as [3].

The paper is organized as follows. Section 2 reviews related work in the field. Section 3 provides an analysis of frame complexity prediction methods and proposes a complexity prediction model. The proposed MD algorithm with dynamic complexity scaling is described in Section 4. Experimental results are presented in

Section 5 and discussed in Section 6. Finally, Section 7 concludes the paper.

2. RELATED WORK

Since the H.264 standard was adopted and its complexity was studied and analyzed [5][17], various methods had been proposed to reduce its computational complexity. In H.264, the most computationally complex components are Mode Decision and Motion Estimation [7]. Generally speaking, work done by researchers in the field can be divided into two categories.

Algorithms in the first category, such as [3][6][11-14][16] allow for complexity reduction of the total execution time of H.264. Some of these methods allow for several fixed complexity settings.

The conventional mode decision technique can be significantly improved by so-called 'early termination' (ET) and 'forward SKIP prediction' techniques [6][11][13][14]. These techniques assume that some block modes can be eliminated from the mode search without loss and that correct SKIP decisions may be made at the start of the MD process. The key to the success of these techniques is utilization of fast and efficient decision metrics.

Further Mode Decision complexity reduction involves more sophisticated approaches, such as [3][12], whereby all MBs in the frame are classified according to certain features of the video signal. Different MD search parameters are used for each class. For example, in [3], the authors propose Mode Group (MGs) classification that utilizes overlapped MGs based on a measure of the residual error with predefined empirical thresholds. The method provides 52% total complexity reduction, but only for P frames with Rate-Distortion Optimization (RDO) on. A feature-based approach with a risk minimizing Mode Decision was proposed in [12], but it is less effective – only reducing total encoding time by 20-30%.

In the second category, there are algorithms that are practical implementations of H.264 on particular hardware platforms, such as [4][15][18]. Unlike algorithms in the previous category, they are designed to operate under real-time conditions. The authors claim that they can provide a real-time video encoding on the selected hardware platform. For example, [18] discusses a real-time H.264 implementation on a TMS320C6416 DSP.

The authors do not concentrate only on the Mode Decision technique (or any other technique) by itself. Instead, they optimize different parts of the H.264 encoder to achieve the real-time goal. In [15], for instance, several techniques such as Intra prediction optimization, SAD optimization for Motion Estimation and DCT optimization are proposed. The resultant encoder was tested for its performance on the Pocket PCs and Smart phones.

These papers mainly discuss hardware-related issues of particular H.264 algorithm implementations (such as code optimizing techniques) and do not provide a deep discussion about how the proposed techniques can be utilized on other hardware platforms, if they can be utilized at all.

Finally, none of these papers investigate the dynamic complexity scaling problem, i.e. maintaining defined complexity when the processor's workload varies, or design of a real-time encoder working on a PC. We believe that the solution can be found among existing Rate Control techniques since the RC problem has many similarities with dynamic complexity scaling.

3. ANALYSIS

Rate control algorithms allocate a bit rate quota for encoding each video frame. The quota must be met with minimum loss in encoding efficiency. In the computational complexity control case, the problem can be formulated as:

$$\min(\Delta R), \min(\Delta D) \quad \text{such that} \quad T_{actual} < T_{quota} \qquad (1)$$

where ΔR is bit rate gain, ΔD is distortion, T_{actual} is the actual encoding time and T_{quota} is the time quota, which depends on frame rate.

As for RC algorithms [8], complexity scaling can be applied to the frame layer or the macroblock layer. At the frame layer, complexity scaling is applied to all MBs in a given frame in the same way. Thus, a single complexity setting is used for the whole frame. At the macroblock layer, complexity scaling is applied to each MB in the frame individually and there is a complexity setting for each MB. The choice of layer depends on the variation in complexity between different MBs when encoded using the same parameters.

3.1 Methods of Complexity Prediction

Current studies of RC algorithms in [8] show that solutions can be divided into two categories: those that operate without a buffer, and those that use a scene-content complexity estimation approach and require a buffer. Based on this, we propose the following schemes for dynamic complexity control.

In the first category, we consider fixed worst-case frame scheduling and truncated time scheduling. In fixed worst-case scheduling the time quota $T^{WC}{}_{quota}$ is the maximum frame encoding time as measured for the most computationally complex frame (e.g. in a high motion video sequence):

$$T^{WC}{}_{quota} = T^{WC}{}_{\text{frame\_limit}} \qquad (2)$$

The encoding parameters in worst-case scheduling are fixed such that the actual encoding time T_{actual} is less than $T^{WC}{}_{quota}$ in all cases. The unused processing time can be calculated as:

$$T^{WC}{}_{\text{unused}} = \sum_{i=0}^{N} (T^{WC}_{quota} - T_i) \qquad (3)$$

where N is the total number of frames in the video sequence and T_i is the actual processing time for the particular frame i.

This approach has a large disadvantage: since the actual encoding time T_{actual} depends on the temporal-spatial complexity of the source video material, it is impossible to take advantage of the unused processing time for video frames requiring less than maximum computational complexity.

Truncated time scheduling assumes that the encoding parameters can be adjusted dynamically to ensure that the encoding time T_{actual} is less than T_{quota}. All MBs in the frame are processed with fixed encoding complexity settings as in worst case scheduling,

but if the quota is exceeded then the rest of the MBs are skipped. The advantage of this approach is that it allows a reduction in the unused processing time compared to worst case: $T_{unused}^{TRUNC} < T_{unused}^{WC}$. Thus, higher encoding settings than in worst-case scheduling may be used, leading to greater average coding efficiency. The disadvantage is that the number of skipped MBs in some frames leads to high visual quality degradation for fast changing frames.

In the second category, we propose a scheduling scheme based on scene complexity estimation. In this case the MB encoding parameters are adjusted such that the predicted encoding time is on average equal to the time quota. The encoding time is predicted based on an estimate of scene complexity:

$$T_{predicted} = f(C_{frame}) \quad (4)$$

where $T_{predicted}$ is the predicted frame encoding time (or predicted T_{actual}) and C_{frame} is the estimated scene frame complexity. A study of previous RC algorithms [8] reveals that scene complexity can be estimated in a number of ways, e.g., by frame energy, number of allocated bits and by utilization of visual metrics like PSNR, MAD or SAD. The function f can be calculated adaptively, based on the C_{frame} , $T_{predicted}$ and T_{actual} obtained for the previous frames.

This scene complexity estimation approach is combined with a complexity control scheme that adjusts complexity dynamically. This ensures that $T_{actual} \approx T_{quota}$ and minimizes unused processing time. The disadvantage here is that large variations in C_{frame} produce large variations in $T_{predicted}$ and may eventually result in the predicted time exceeding the time quota.

A frame buffer allows for errors in prediction – excess coding time in one frame is simply subtracted from the quota in the next. The buffer size must be small for low delay applications, where a higher frame rate is desirable (i.e. 30 fps) and can be larger for applications that allow greater delay [8]. Small buffer sizes may result in buffer overflow for large errors in $T_{predicted}$. In order to avoid this, an efficient MB complexity scaling scheme is required in addition to accurate scene complexity prediction.

It can be concluded that of all of these schemes, scene complexity estimation is the most advantageous as it provides maximum flexibility in minimizing coding efficiency loss. Hence, scene complexity estimation is applied in this work and compared with the worst case scheduling.

3.2 Proposed Model

Experiments with various video sequences indicate that when a fast Mode Decision algorithm is used, the encoding time for each MB varies greatly. In fact, frame encoding time can be determined from the distribution of Mode Decision classes (or MD classes) across all MBs in the frame. Thus, the predicted encoding time for the current frame $T_{FastMD\_predicted}$ can be estimated as:

$$T_{FastMD\_predicted} = \sum_{i=1}^{i=5} n_i t_i + T_0 , \qquad t_i = const \quad (5)$$

where n_i is the number of MBs that belong to MD class i and t_i is the average encoding time for that MD class. For this model t_i can be determined experimentally across several high and low motion QCIF video sequences.

For fast Mode Decision algorithms that employ early termination schemes such as [6], t_i calculated per MB for the same MD class can vary greatly, depending on the stage of the MD process at which early termination occurred. To extend the complexity prediction model to these cases, a scheme for adaptive calculation of t_i is proposed.

The adaptive scheme involves storing the encoding time statistics for previously encoded frames and re-calculating t_i for all MD classes every Nth frame. The following methods of updating t_i were assessed experimentally for the proposed model as in Eq. (5) with $N = 5$:

1. Use the mean time for each MD class, calculated across all MBs in the previous frames:

$$t_i = mean\{T_0, T_1, ..., T_{N-1}\} \quad (6)$$

   This method is the simplest and most straight-forward solution.

2. Use mean time for MBs that did not have their class changed as a part of a scaling algorithm (i.e. were not demoted, see below):

$$t_i = mean\{T_{not\_demoted\_0}, ..., T_{not\_demoted\_N-1}\} \quad (7)$$

   As demotion happens for MBs with the lowest previous J (see Eq. (9)) within a class and since lower J values indicate higher scene complexity, then demoted MBs are harder to encode.

3. Use the maximum time measured for each MD class calculated across all N frames for MBs that were not early terminated:

$$t_i = \max\{T_{notET\_0}, ..., T_{notET\_N-1}\} \quad (8)$$

   Since early termination significantly reduces MD time, calculation of $T_{FastMD\_predicted}$ excludes early terminated MBs in order to provide a consistent estimate.

The experimental results are provided in Table 1. From the experimental results, it can be clearly seen that 'Maximum t_i' is very inaccurate. The other two methods provide accurate prediction with an average difference between predicted and actual times of about 7%. Since the 'mean not demoted' method has a higher Pearson correlation coefficient, it is deemed to provide the most accurate prediction for fast MD with early termination, so the final frame complexity prediction model chosen adopts Eq. (5) with adaptive t_i calculation as in Eq. (7).

Once frame complexity is predicted, macroblock layer complexity scaling can be performed. The goal is the adjustment of computational complexity for each MB in the frame in order that the time quota is met and coding efficiency degradation is minimized.

Table 1. Estimation of adaptive frame prediction models

Video sequence	Method	Prediction error, %			Pearson correlation between predicted and actual MD time
		min	*max*	*avg.*	
Carphone, QCIF	1	0.02	33.04	6.95	0.751
	2	0.13	33.75	7.73	0.771
	3	8.21	85.61	23.35	0.496
Hall, QCIF	1	0.02	46.70	3.95	0.949
	2	0.21	43.90	5.46	0.967
	3	4.18	88.62	22.94	0.625
Akiyo, QCIF	1	0.01	28.8	2.31	0.969
	2	0.02	29.13	3.42	0.983
	3	2.67	66.93	14.41	0.768

Since reducing complexity effects both bit rate and distortion, the need arises to unify both quantities into a single metric which is representative of the overall coding efficiency. At present, the rate-distortion model is adopted in H.264 Mode Decision [9] for making optimal decisions where both bit rate and distortion are important:

$$\min\{J\}, \quad \text{where} \quad J = D + \lambda \cdot R \tag{9}$$

where D is a distortion measure (usually Sum of Absolute Differences) and R represents bit rate. During video encoding, the Lagrange rate-distortion function J is minimized for a particular value of the Lagrange multiplier, λ.

Based on this, we introduce a coding efficiency metric, W, which is dependent on visual quality loss, ΔD, and bit rate increase, ΔR, relative to that achieved by the full complexity encoder. We define W as:

$$W = \Delta R + \mu \Delta D \tag{10}$$

where ΔR is a percentage, ΔD is PSNR in dB and μ is a constant relating bit rate loss and distortion increase. Thus, for any given computation complexity point C_i, the most efficient encoder can be identified as the one providing minimum W.

The constant μ can be interpreted as the percentage increase in bit rate equivalent to a 1 dB loss in PSNR. Previous work [1] determined that, for the frame sizes under investigation, a 10% decrease in bit rate is roughly equivalent to a loss of 0.5 dB in PSNR. In our work, μ was determined experimentally [7] and was set to 13.

Assuming, that the encoder parameter configurations form a discrete set, the problem of selecting the optimal encoder configuration for any given complexity point can be solved by Pareto analysis [2]. In this work, the efficiency of the encoder was assessed across a range of parameter configurations. These results were a projected on to a graph relating coding efficiency to computational complexity. The optimum encoder parameters were then identified as those points (C_i, W_i) which form the Convex Hull of the Individual Minima (CHIM) on the Pareto curve, shown as the blue line in Figure 1. Parameter configurations corresponding to points inside the CHIM are sub-optimal. In this work, the MD classes only contain parameter configurations which are on the CHIM.

The low complexity encoding algorithm considered in this work allocates an MD class to each MB based on an analysis of the MB's statistical properties. MBs likely to have low motion are allocated to an MD class with a narrow search range in ME. This reduces encoding time without loss of coding efficiency. MD class allocation based on MB statistics is used for initial frame encoding time prediction. If the prediction exceeds the quota, then some macroblocks must have their MD classes demoted to reduce total encoding time. In contrast, if the prediction time is less than the quota, then some MBs may have their MD classes promoted to improve coding efficiency.

Scaling complexity down for a particular MB is referred to as MD class demotion. For example, a macroblock initially allocated to class B can be demoted to class C. This reduces computational complexity by ΔC and reduces coding efficiency by ΔW, as defined in Eq. (10). For a given reduction in computational complexity of ΔC, it is clear that the best demotion class is on the convex hull of the Pareto curve.

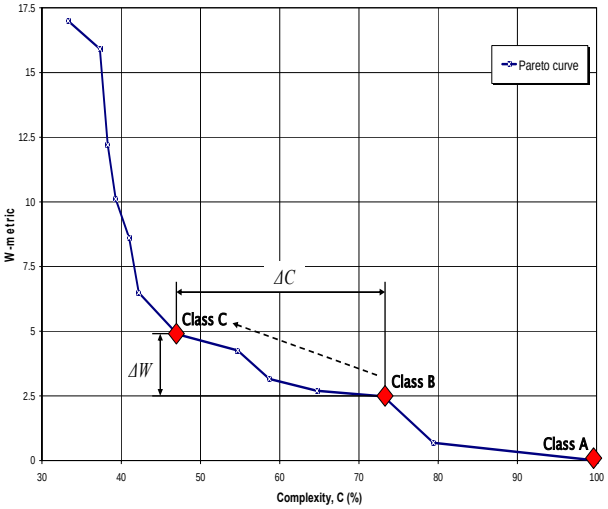

Figure 1. MB demotion example

It can be observed in Figure 1 that the gradient $\frac{\Delta W}{\Delta C}$ of the convex hull of the Pareto curve is lower between more computational complex MD classes (i.e. $\frac{\Delta W_{AB}}{\Delta C_{AB}} < \frac{\Delta W_{BC}}{\Delta C_{BC}}$).

That is, there is less coding efficiency loss for a given reduction in computational complexity. Hence, demotion should start with macroblocks that are in the more computationally complex MD classes.

In some cases, it may not be necessary to demote all MBs within a given class. Since demotion generally leads to a bit rate and distortion increase, demotion should start with MBs that have lowest previous J within the class. In this way, the most effectively coded MBs are demoted first, minimizing coding efficiency loss.

Scaling complexity up for a particular MB is referred to as MD class promotion. For example, a macroblock initially allocated to class C can be promoted to class B. This may improve MB coding efficiency, but also increases computational complexity. Promotion should start with the macroblocks that are allocated to less computationally complex MD classes, i.e. class E or D, and continues until the highest complexity class is reached (class A). Since promotion generally reduces bit rate and distortion, it should start with the MBs that have highest previous J within the given class, thus the less effectively coded MBs are promoted first.

The proposed frame complexity estimation model was tested and optimized for IPPP GOP structure. For B-frames the average encoding time t_i for each MD class is generally higher than for P-frames due to increased computational complexity. To overcome this, it is proposed to utilize different sets of t_i values for P- and B-frames, i.e. $\{t_0^P, t_1^P ... t_i^P\}$ and $\{t_0^B, t_1^B ... t_i^B\}$. Alternatively, a scaling coefficient γ can be applied for t_i^P values when dealing with B-frames. The particular value for γ is determined experimentally.

4. ALGORITHM

The fast Mode Decision algorithm used in these experiments is shown in Figure 2. It consists of two parts: MD class selection and fast Mode Decision with Early Termination.

The MD class selection algorithm chooses the macroblock class based on three metrics: *FD* [16], J_{prev} and SAD_{8x8} [6]. The Mode Decision algorithm with Early Termination utilizes *J* values from the previous frame (i.e. J_{prev}) in order to omit unnecessarily MD computations [6].

As is typical of most fast MD algorithms, algorithm's computational complexity is dependent on frame content. In order to achieve good coding efficiency, frames with high motion typically require more processing than frames with low motion.

The algorithm for real-time dynamic complexity scaling in the H.264 encoder is shown in Figure 3. The algorithm uses a one-frame buffer. The time quota T_{quota} is re-calculated after each frame is processed as:

$$T_{quota} = T_{\text{frame\_limit}} + \min(N \cdot T_{\text{frame\_limit}} - T_{total},\ T_{\text{frame\_limit}}) \quad (11)$$

In order to synchronize the buffer, the last frame encoding time T_{last_frame} cannot exceed the frame encoding time limit T_{frame_limit}. The value of T_{frame_limit} is selected in order to allow real-time encoding at the desired frame rate.

5. EXPERIMENTAL RESULTS

The MD algorithm for real-time complexity scaling was implemented in the JM [10] reference encoder and experimentally tested. No assembly language or other manual optimizations were applied. Several QCIF video sequences of 300 frames each were encoded with an "IPPP" GOP structure. Reference encoding used all seven VBS modes, CABAC entropy coder and RDO off.

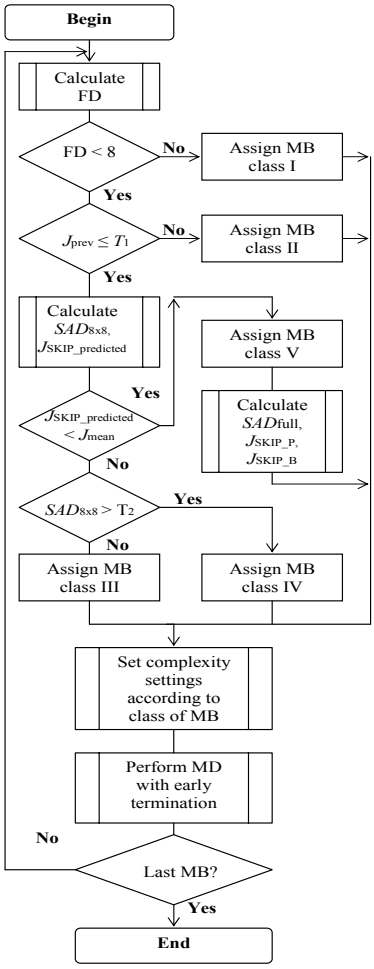

Figure 2. Fast MD algorithm with MD classes

In the experiments, *QP* was set to 28. The algorithm was tested under conditions, where the value of T_{frame_limit} was set to allow real-time encoding at 15, 20 and 30fps on a reference 3GHz Pentium IV PC with 1GB RAM. The obtained results for bit rate gain, quality degradation and complexity reduction versus non-real time JM running at full search baseline profile on the same PC are shown in Tables 2–4. The minus (–) sign denotes improvement for the new method.

Using Eqs. (12)–(14) we calculated the percentage of unused time T_{unused}, prediction error T_{error} and trim time T_{trim}:

$$T_{unused},\% = \sum_{n=1}^{n=300} \left| 1 - \frac{T_{actual\_n}}{T_{quota\_n}} \right| \quad (12)$$

$$T_{error},\% = \frac{\sum_{n=1}^{n=300} \left| T_{FastMD\_predicted\_n} - T_{actual\_n} \right|}{T_{total}} \quad (13)$$

$$T_{trim},\% = \frac{\sum_{n=1}^{n=300} \left| T_{FastMD\_predicted\_without\_demotion\_n} - T_{actual\_n} \right|}{T_{total}} \quad (14)$$

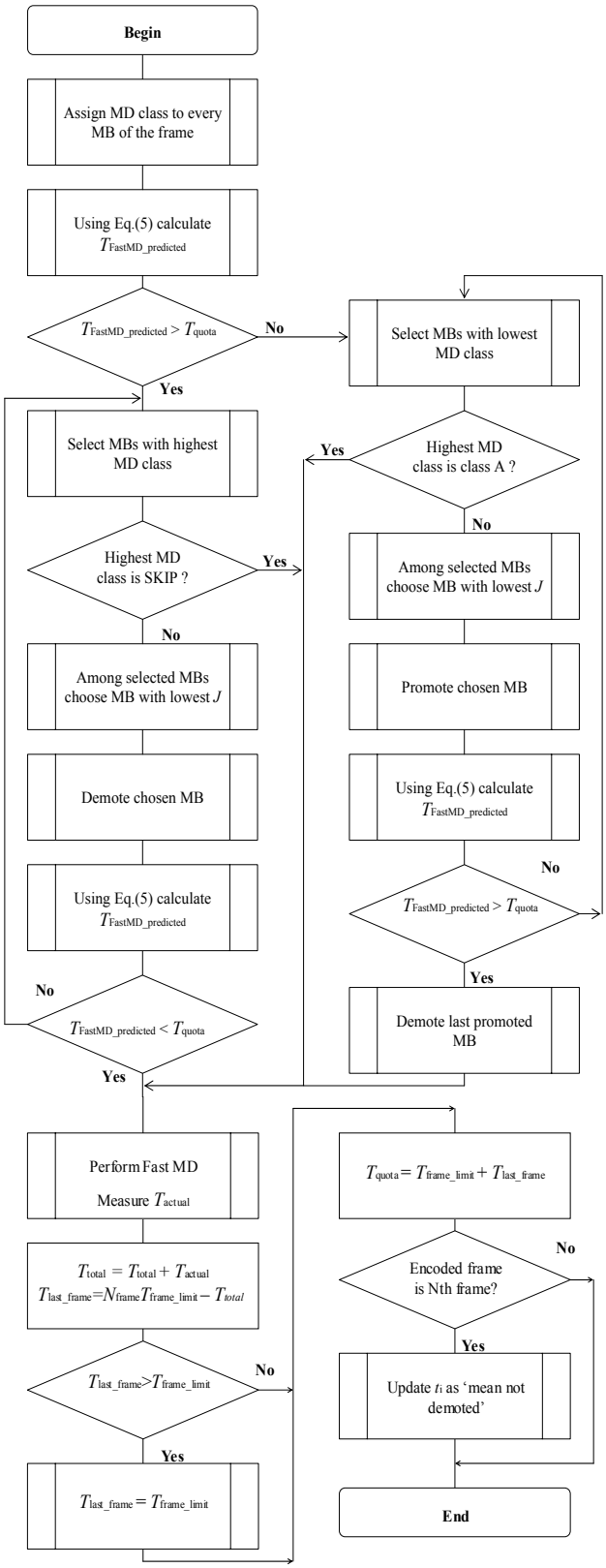

Figure 3. Real-time complexity scalable fast MD algorithm

For the proposed algorithm running on the reference PC at 30fps, the results obtained include the Pearson correlation between predicted and actual time.

All results are given in Tables 2–5.

Table 2. ΔBit rate, % for the proposed method vs. ref. JM

Video sequence	Frame rate		
	15fps	20fps	30fps
Carphone	0.05	1.04	15.25
Table tennis	2.48	3.85	11.16
Coastguard	0.26	0.87	6.48
News	0.84	0.76	5.90
Salesman	-0.5	-1.50	5.60
Grandmother	-1.23	-3.08	-1.82
Mother	-0.7	-1.00	1.68
Hall	0.03	-1.73	-4.41
Akiyo	-1.15	-1.80	-1.85
mean	***0.01***	***-0.28***	***4.22***

Table 3. ΔPSNR, dB for the proposed method vs. ref. JM

Video sequence	Frame rate		
	15fps	20fps	30fps
Carphone	0.11	0.28	0.49
Table tennis	0.24	0.24	0.46
Coastguard	0.06	0.08	0.13
News	0.09	0.12	0.38
Salesman	0.03	0.07	0.24
Grandmother	0.02	0.19	0.18
Mother	0.06	0.17	0.32
Hall	0.01	0.03	0.20
Akiyo	0.01	0.02	0.16
mean	***0.07***	***0.13***	***0.28***

Table 4. Total encoding time reduction, % for the proposed method vs. JM

Video sequence	Frame rate		
	15fps	20fps	30fps
Carphone	–43.30	–52.42	–73.27
Table tennis	–47.20	–52.10	–72.95
Coastguard	–46.73	–54.64	–73.10
News	–40.75	–48.03	–71.27
Salesman	–35.42	–48.70	–71.25
Grandmother	–32.97	–48.42	–70.54
Mother	–32.77	–47.91	–70.18
Hall	–40.92	–47.33	–70.97
Akiyo	–38.77	–47.41	–70.64
mean	***–39.87***	***–49.66***	***–71.57***

Table 5. T_{unused}, T_{error} T_{trim}, % and Pearson correlation, r for the proposed algorithm at 30fps

Video sequence	T_{unused}	T_{error}	T_{trim}	r between $\{T_{FastMD\_predicted}\}$ and $\{T_{actual}\}$
Carphone	3.81	3.39	74.95	0.581
Table tennis	5.75	5.55	50.59	0.546
Coastguard	4.12	3.88	98.27	0.509
News	6.17	5.88	21.11	0.557
Salesman	6.11	6.28	18.04	0.422
Grandmother	2.72	2.56	13.74	0.667
Mother	3.40	3.13	17.00	0.643
Hall	5.68	6.40	5.75	0.630
Akiyo	2.85	3.64	-2.92	0.738

Worst case scheduling was tested under the same simulation conditions. The H.264 complexity settings were selected as shown in Table 6, so the encoding time for the worst sequence allow encoding at 15, 20 and 30fps. The standard non-real time JM with full search baseline profile achieves only 8fps on the reference PC.

Table 6. H.264 settings for the worst case scheduling

H.264 settings	Worst scheduling at..			
	30fps	20fps	15fps	8fps
VBS modes	P16x16, all Intra modes	P16x16, P8x8, all Intra modes	All VBS modes	All VBS modes
Search range	1	4	6	8
Hadamard transform	off	on		

In order to compare the efficiency of both algorithms on the 30fps point, the difference in the bit rate and PSNR for the proposed algorithm were calculated relative to worst-case scheduling. The results are given in Table 7. In the Table 7, minus sign (–) indicates improvement for the method.

Table 7. Comparison of the proposed algorithm vs. worst-case scheduling

Video sequence	ΔBit rate, %	ΔPSNR, dB
Carphone	2.26	–0.04
Table tennis	–8.10	0.09
Coastguard	–6.84	–0.07
News	–11.10	–0.06
Salesman	–12.58	–0.06
Grandmother	–11.15	–0.08
Mother	–6.89	–0.18
Hall	–18.80	–0.1
Akiyo	–15.58	–0.19
mean	***–9.89***	***0.07***

6. DISCUSSION

From the experimental results in Tables 2–5, it can be concluded that the proposed algorithm provides dynamic complexity scaling for the selected sequences, making optimal real-time H.264 video encoding possible on a range of processors.

It can be observed from Table 5 that unused time T_{unused} and prediction error T_{error} for the algorithm are low, around 2.72–6.17% on average. Trim time T_{trim} shows the percentage of complexity that was scaled down by the algorithm from the original complexity. The minus sign for trim time (i.e. Akiyo) means that complexity was scaled up, not down, thus many MBs in that sequence were promoted.

From Table 5, it can be observed that, depending on the motion in the source video, the results can be roughly divided into three groups low ($T_{trim} < 10\%$), medium ($10\% \le T_{trim} < 50\%$) and high motion sequences ($T_{trim} \ge 50\%$).

For low motion sequences (i.e. Akiyo, Hall), the algorithm provides the best complexity scaling results. Prediction is quite accurate, and the Pearson correlation is quite high (0.63–0.73). These results show insignificant PSNR drop in visual quality and even bit rate reduction compared to the full search non-real time JM.

For the medium motion sequences (i.e. News, Salesman, Mother etc), the MB complexity scaling scheme results in around 1.65–6% of bit rate gain, which is due to the increased number of Intra macroblocks. The demotion of MD class C to class D is the most likely reason. Frame complexity prediction is accurate, the Pearson correlation is in the range 0.55–0.65.

Sequences that are high motion (i.e. Carphone, Tennis etc) are the most difficult for the algorithm to handle. Complexity reduction T_{trim} of 50–98% from non-scalable fast MD algorithm comes at the cost of a 0.5dB quality degradation and a bit rate increase of 10–15%. Due to high spatial and motion complexity there are few MBs which that not demoted, resulting in inaccuracy in adaptive t_i calculation. This results in poor frame complexity prediction (Pearson correlation is only 0.5). In fact, demotion of the original MD class A into class D seems to be necessary for complexity reduction, which results in a bit rate increase.

Notably, results for Coastguard are much better than for Carphone and Tennis, despite having the highest trim time (98%). This can be explained by the fact that Coastguard has the highest number of Intra blocks in the 'original' JM encoding, which results in comparatively low bit rate increase (only 8%) and insignificant quality degradation (0.13dB).

Based on these results, it can be concluded that the complexity scaling model needs to be improved to deal with high motion cases, where the demotion rate is high. A possible solution is utilization of multiple metrics for t_i for different demotions.

When comparing the proposed method with the worst scheduling approach, it can be concluded from Table 7 that the average bit rate reduction achieved for the method is almost 10% compared to worst case scheduling with almost 19% maximum (for Hall video sequence). PSNR is also better for the method for all sequences except Tennis. However, the difference in PSNR achieved is negligible. The results for previously published low complexity

H.264 algorithms clearly indicate that none of the previously proposed methods achieve similar results.

The bit rate gain and PSNR degradation was plotted against target processor performance for both methods. The performance of the reference PC that provides 30fps is equal to 33msec/frame.

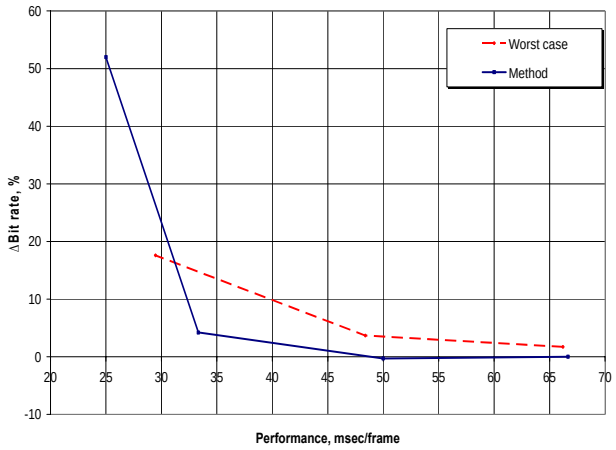

Figure 4. Bit rate gain for both methods

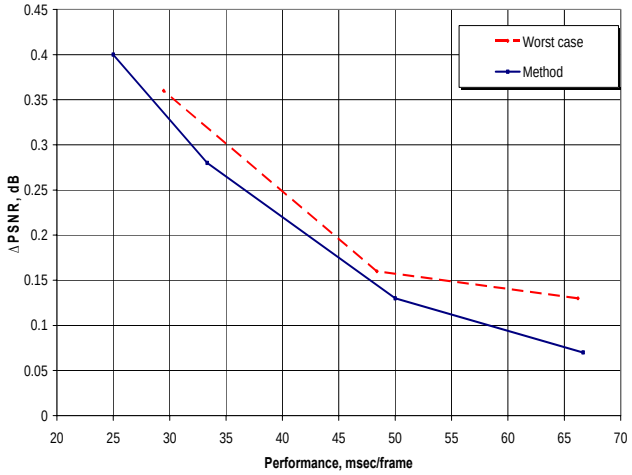

Figure 5. Quality degradation for both methods

In Figure 4, the bit rate curve for the proposed algorithm (solid line) lies much lower than for the standard worst case scheduling method (dashed line), which means that the proposed algorithm provides better encoding efficiency. It can be observed that the lower the frame rate – the closer the resultant bit rate curves, thus, the proposed algorithm becomes less effective relative to worst scheduling. Very high bit rate gain for the proposed algorithm at 35fps indicates that the processor's capabilities are the limiting factor and, in order to save processing time, the algorithm uses only Intra prediction and SKIPs. Therefore, the most efficient operating point for the algorithm is located on the 'knee' of the curve, which is around 30fps.

7. CONCLUSIONS

In this paper, various issues relating to dynamic real-time complexity scaling in H.264, such as complexity prediction methods, MB complexity scaling and time scheduling algorithms were investigated.

The proposed real-time complexity scalable MD algorithm significantly outperforms the standard worst case scheduling approach, both in terms of bit rate reduction (10% in average) and frame rate. It offers roughly 30% of the computational complexity required for full search with the baseline profile at the same coding efficiency and allows an efficient real-time video encoding on the reference Pentium IV PC.

8. REFERENCES

[1] Bjontegaard, G. *Calculation of average PSNR differences between RD curves.* Document VCEG-M33, ITU-T VCEG Meeting, Austin, 2001.

[2] Das, I. On characterizing the 'knee' of the Pareto curve based on Normal-Boundary Intersection, *Structural and Multidisciplinary Optimization, 18*, 3 (1999), 107–115.

[3] Feng, B., Zhu, G., and Liu, W. Fast Adaptive Inter Mode Decision Method for P Slices in H.264. In *Proc. of IEEE 3rd Int. Conf. on Consumer Communications and Networking (CCNC'06), 2* (2006), 745–748.

[4] Hsu, K.W., Xiang Li, and Chopra, R. An IC design for real-time motion estimation for H.264 digital video. In *Proc. of 48th Symp. On Circuits and Syst., 2,* (August 7–10, 2005), 1489–1493.

[5] Implementation Studies Group of ISO/IEC. *Main Results of the AVC Complexity Analysis.* Document ISO/IEC JTC1/SC29/WG11, Klagenfurt, 2002.

[6] Ivanov, Y. V., and Bleakley, C. J. Skip Prediction and Early Termination for Fast Mode Decision in H.264/AVC. In *Proc. of Int. Conf. on Digital Communications (ICDT).* (August, 2006).

[7] Ivanov, Y.V., and Bleakley, C. J. Survey and Pareto Analysis Method for Coding Efficiency Assessment of Low Complexity H.264 Algorithms. In *Proc. of 10th Irish Machine Vision and Image Processing Conf. (IMVIP)* (2006), 172–179.

[8] Jiang, M. and Ling, N. Low-Delay Rate Control for Real-time H.264/AVC Video Coding. *IEEE Trans. Multimedia, 8,* 3 (June 2006), 467–477.

[9] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG. *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H264|ISO/IEC 14496-10 AVC).* Document JVT-G050d35.doc, 7th Meeting: Pattaya, Thailand, March, 2003.

[10] JVT reference software JM 9.5, on the Web: http://iphome.hhi.de/suehring/tml.

[11] Kannangara, C. S., Richardson, I. E.G., Bystrom, M., Solera, J. R., Zhao, Y., MacLennan, A., and Cooney, R. Low-Complexity Skip Prediction for H.264 Through Lagrangian Cost Estimation. *IEEE Trans. Circuits Syst. Video Technol., 16,* 2 (2006), 202–208.

[12] Kim, C., and Jay Kuo, C. C. A Feature-based Approach to Fast H.264 Intra/Inter Mode Decision. In *Proc. of IEEE Int. Symp. Circuits and Systems (ISCAS'05), 1,* (May 23–26, 2005), 308–311.

[13] Kim, Y., Choe,Y., and Choi, Y. Fast Mode Decision Algorithm using AZCB Prediction. In *Proc. of Int. Conf. on Consumer Electronics. (ICCE'06). Digest of technical papers.* (Jan. 7-11, 2006), 33–34.

[14] Li, G. L., Chen, M. J., Li H. J., and Hsu, C. T. Efficient Motion Search and Mode Prediction Algorithms for Motion Estimation in H.264/AVC. In *Proc. of IEEE Int. Symp. Circuits and Systems (ISCAS'05) 6,* (May 23-26, 2005), 5481 – 5484.

[15] Rao, G. N., Prasad, RSV, Chandra, D. J., and Narayanan, S. Real-Time Software Implementation of H.264 Baseline Profile Video Encoder for Mobile and Handheld Devices Acoustics. In *Proc. of IEEE Int. Conf. on Speech and Signal Processing (ICASSP'06), 5,* (May 14-19, 2006), 457–460.

[16] Wang, H., and Zhu, Z. Fast Mode Decision and Reduction of the Reference Frames for H.264 Encoder. In *Proc. of Int. Conf. On Control and Automation (ICCA'05), 2,* (June 26–29, 2005), 1040–1043.

[17] Wiegand, T., Sullivan, G. J., Bjontegaard, G., and Luthra, A. Overview of the H.264/AVC Video Coding Standard. In *IEEE Trans.on Circuits Syst. Video Technol. 13,* 7 (2003), 560–576.

[18] Zhe, W., and Canhui, C. Realization and optimization of DSP based H.264 encoder. In *Proc. of IEEE International Symposium on Circuits and Syst. (ISCAS'06)* (May 21-24, 2006).