COMPLEXITY CONTROL FOR REAL-TIME VIDEO CODING

Emrah Akyol[*], *Debargha Mukherjee*[+], *Yuxin Liu*[+]
Email: eakyol@ee.ucla.edu, {debargha, yuxinl}@hpl.hp.com
[*] University of California, Los Angeles, CA, USA
[+] HP Laboratories, Palo Alto, CA, USA

ABSTRACT

A methodology for complexity scalable video encoding and complexity control within the framework of the H.264/AVC video encoder is presented. To yield good rate-distortion performance under strict complexity/time constraints for instance in real-time communication, a framework for optimal complexity allocation at the macroblock level is necessary. We developed a macroblock level fast motion estimation based complexity scalable motion/mode search algorithm where the complexity is adapted jointly by parameters that determine the aggressiveness of an early stop criteria, the number of ordered modes searched, and the accuracy of motion estimation steps for the INTER modes. Next, these complexity parameters are adapted per macroblock based on a control loop to approximately satisfy an encoding frame rate target. The optimal manner of adapting the parameters is derived from prior training. Results using the developed scalable complexity H.264/AVC encoder demonstrate the benefit of adaptive complexity allocation over uniform complexity scaling.

Index Terms − H.264/AVC, complexity control, scalable complexity, real-time video coding, videoconferencing, fast motion estimation.

1. INTRODUCTION

In recent video coding standards such as H.264/AVC [1], superior rate-distortion efficiency is achieved by incorporation of algorithmic features requiring complex motion and mode searches during encoding. Since the decoding complexity is less impacted, the use of such complex encoders is legitimate for applications where offline encoding is possible, ex. DVD distribution of movies. However, in applications requiring real-time video encoding and delivery, such as videoconferencing, live video broadcast or even live recording, complex encoders pose huge problems in maintaining the desired throughput and quality. Furthermore, real-time encoding on devices that are not only power-constrained (mobile devices) but also time varying in available power (battery powered devices, PCs), necessitates adaptability to optimally utilize the available power.

While rate allocation to macroblocks (MBs) in a rate-distortion optimal way is well studied in the context of image-video compression [2] optimal complexity allocation has not been analyzed as well. Further, most real-time encoders use a form of fast Motion Estimation (ME) (ex. [9]) with a variety of inter-dependent parameters that are confusing to adapt in order to allocate complexity differently. In this work, we first present a methodology to answer the following complexity related question for video encoding: what parameters are most suitable for complexity adaptation based on a fast ME scheme, and what is their optimal tuning strategy that results in the highest compression efficiency given a total complexity constraint at the encoder and a target quality? Next we use the developed complexity adaptation mechanism in conjunction with a control loop to achieve coding delay control. Note that this work is not just another method to decrease the complexity of H.264/AVC encoder with negligible compression efficiency loss, but a framework to scale the complexity, building on already optimized fast motion/mode searches, in a quasi-optimal manner (in the sense of minimizing R-D performance loss).

A few recent works also address the complexity scalability problem. In some of them [3][4][5], all MBs in a frame are ordered and more computation is allocated to the MBs with the highest distortion or distortion (SAD)-complexity slope in the entire frame step by step. However, there are two fundamental problems associated with these approaches. First motion vector predictors from spatial neighborhoods cannot be used effectively. Second, since all the MBs are processed simultaneously, huge book-keeping overheads are incurred. In our work, MBs are still processed in sequential order. In [6], complexity scalability is achieved only by assigning different number of SAD computations depending on a binary decision of static/non-static MB. Our framework [10] is more generic in handling multiple parameters for fast ME jointly. Furthermore, we attempt to find the optimal manner of adapting the parameters by a training based methodology.

2. MOTION-MODE SEARCH ALGORITHM

In this section we present a motion-mode search algorithm where the complexity is controlled by a few relevant parameters. H.264/AVC supports multiple encoding modes, and optimal encoding requires searching through all of them to find the best in the rate-distortion sense. In order to reduce complexity and scale it, we devise a mechanism where the modes are searched in a certain order with checks at the end of each mode to terminate the search or continue. Further, within each inter mode, the accuracy of the motion estimation process can be varied. The order of the modes is important for R-D performance. The ordering may be based on the statistical frequency of the optimal modes for a given type of video.

All SAD costs in the following refer to the $SAD+\lambda R_{\text{motion}}$ metric which is an approximation to the true $D+\lambda R$ cost. Further, the following notations are introduced in Table 1.

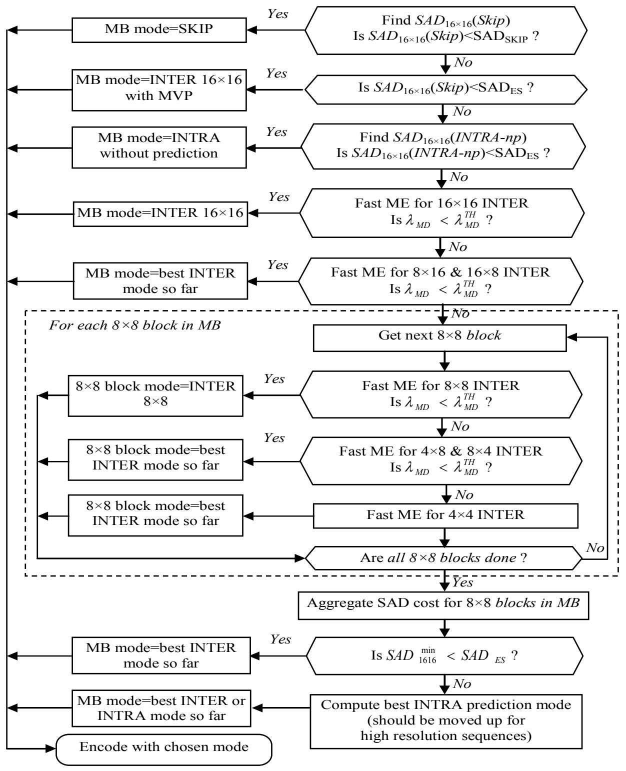
Table 1. Definition of parameters

$SAD_{16\times16}(mode)$ $SAD_{8\times8}(mode)$	Lowest SAD cost for a 16×16 MB and 8x8 block respectively, with given *mode*. Thus $SAD_{16\times16}(8\times16)$ is the total SAD cost for a 16x16 MB in the 8×16 mode obtained by summing the costs for two 8×16 blocks comprising it, while $SAD_{8\times8}(4\times4)$ is the total

	SAD cost for a 8×8 block in the 4×4 mode obtained by summing the costs for the four 4×4 constituent sub-blocks.
SAD_{SKIP}	Skip threshold which is only a function of the quantization parameter given as: $SAD_{SKIP}=N.2^{QP/6}/\sqrt{12}$, N being the number of pixels in 16×16 MB (N=256), similar to the formulation in [9].
$SAD_{pred}(m\times n)$	SAD cost prediction for a $m\times n$ block, obtained from SADs for neighboring blocks or collocated blocks from reference frame as in [9].
β	*Early stop threshold scaling factor*
SAD_{ES}	Early stop SAD cost threshold which is obtained by scaling $SAD_{pred}(16\times16)$ by β: $SAD_{ES}=\beta.SAD_{pred}(16\times16)$
ΔC	Computation required to search a mode or conduct a step of fast ME. It can be measured by real time taken or approximated by the number of SAD cost computations conducted.
λ_{MD}	*Mode search gradient* parameter that determines the number of INTER modes tested for each MB (n=16) or 8×8 block (n=8): $\lambda_{MD}=(SAD^{\min}_{n\times n}-SAD_{n\times n}(\text{mode}))/\Delta C$ where $SAD^{\min}_{n\times n}$ is the minimum $n\times n$ SAD cost in INTER mode searches conducted before the current mode search.
λ^{TH}_{MD}	*Mode search gradient threshold* parameter
λ_{ME}	*Motion estimation gradient* parameter computed at the end of each recursive step of fast ME: $\lambda_{ME}=\Delta SAD^{\min}_{n\times n}/\Delta C$ where $\Delta SAD^{\min}_{n\times n}$ is the change in minimum SAD after the search step.
λ^{TH}_{ME}	*Motion estimation gradient threshold* parameter

The general flow of the proposed complexity scalable mode search for QCIF/CIF sequences is given in Figure 1, and is explained below. Note that for higher resolution sequences, INTRA prediction modes are more useful, and hence they should be searched before the INTER modes are searched.

First, the SKIP mode SAD cost $SAD_{16\times16}(Skip)$ is computed based on the predicted motion vector (MVP). If $SAD_{16\times16}(Skip) < SAD_{SKIP}$ threshold mode search is finished and the MB encoded in SKIP mode. Otherwise, if $SAD_{16\times16}(Skip) < SAD_{ES}$, the early stop threshold, mode search is finished and the MB is encoded in the 16×16 INTER mode with motion vector equal MVP. Note that when this mode is chosen, the final encoded mode can still be SKIP, if all the quantized transform coefficients are zero. In the other case, $SAD_{16\times16}$(INTRA-np), the SAD cost for INTRA with no prediction is calculated, and if $SAD_{16\times16}$(INTRA-np) < SAD_{ES}, mode search is finished and the MB encoded in INTRA mode without prediction. Else, the search proceeds to INTER 16×16 fast ME. After ME, the mode search gradient λ_{MD} as described in Table 1 is computed, and compared with *mode search gradient threshold* λ^{TH}_{MD}; if $\lambda_{MD} < \lambda^{TH}_{MD}$, the mode search is finished and the MB encoded in INTER 16×16mode. Otherwise, fast ME for 8×16 and 16×8 modes are conducted, and the better of the two is computed. If $\lambda_{MD} < \lambda^{TH}_{MD}$, mode search is finished and the MB encoded with the best mode so far. In the other case, we proceed to search 8×8 block modes. For each 8×8 block, fast 8×8 ME is first conducted followed by λ_{MD} computation; if $\lambda_{MD} < \lambda^{TH}_{MD}$ mode search for that block is finished and the block is assigned the 8×8 mode. Otherwise, we continue to search 8×4 and 4×8 modes, find the better of the two, compute λ_{MD} and check if $\lambda_{MD} < \lambda^{TH}_{MD}$. If so, mode search for the block is finished by



Find $SAD_{16\times16}(Skip)$ Is $SAD_{16\times16}(Skip)$<SAD_{SKIP} ?
Yes
MB mode=SKIP
No
Is $SAD_{16\times16}(Skip)$<SAD_{ES} ?
MB mode=INTER 16×16 with MVP
Find $SAD_{16\times16}(INTRA\text{-}np)$ Is $SAD_{16\times16}(INTRA\text{-}np)$<$SAD_{ES}$?
MB mode=INTRA without prediction
Fast ME for 16×16 INTER Is $\lambda_{MD} < \lambda^{TH}_{MD}$?
MB mode=INTER 16×16
Fast ME for 8×16 & 16×8 INTER Is $\lambda_{MD} < \lambda^{TH}_{MD}$?
MB mode=best INTER mode so far
For each 8×8 block in MB
Get next 8×8 block
Fast ME for 8×8 INTER Is $\lambda_{MD} < \lambda^{TH}_{MD}$?
8×8 block mode=INTER 8×8
Fast ME for 4×8 & 8×4 INTER Is $\lambda_{MD} < \lambda^{TH}_{MD}$?
8×8 block mode=best INTER mode so far
Fast ME for 4×4 INTER
8×8 block mode=best INTER mode so far
Are all 8×8 blocks done ?
Aggregate SAD cost for 8×8 blocks in MB
Is $SAD^{\min}_{1616} < SAD_{ES}$?
MB mode=best INTER mode so far
Compute best INTRA prediction mode (should be moved up for high resolution sequences)
MB mode=best INTER or INTRA mode so far
Encode with chosen mode

Figure 1. General flowchart of complexity scalable mode search

assigning the best mode so far to it; otherwise 4×4 modes are searched and thereafter the block is assigned the best mode so far. When the modes for all of the 8×8 constituent blocks in a MB have been searched, we aggregate the overall SAD cost for the MB, and update the minimum SAD cost and the best mode so far. The minimum SAD cost so far is then compared with SAD_{ES} again to continue mode search with INTRA predictions or not. Note that there are a variety of INTRA predictors, and potentially complexity scalability may be achieved by ordering the search within them as well using simple features. While this paper does not consider such possibilities explicitly, the concepts in the paper can be readily extended to handle this case as well. Once the INTRA prediction modes have been searched, the overall best mode is chosen as the final mode.

The fast ME steps in the above algorithm at various block sizes is based on the one in [9], but we add the flexibility to scale the accuracy of the motion vectors searched. Because all fast ME schemes are essentially similar in that they compute a set of good predictors and conduct systematic non-recursive and recursive searches around the best predictor, our method can be readily extended to other fast ME schemes reported in the literature. In particular, we compute a *motion estimation gradient* parameter as described in Table 1 at the end of each recursive step of fast ME and terminate the search if $\lambda_{ME} < \lambda^{TH}_{ME}$, where λ^{TH}_{ME} is the *motion estimation gradient threshold* determining the accuracy of the motion estimation process. Note that the non-recursive steps that precede the recursive steps can be skipped based on comparing the SAD cost of the best predictor with the SAD predictors for the block.

Note that the ME gradient has been also used by other researchers [4] to scale complexity of fast ME methods. We

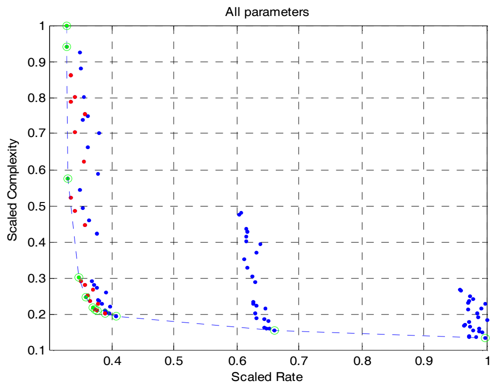


Figure 2. Rate-Complexity points and Convex Hull for a fixed quantization parameter (QP=32). Points on convex Hull are circled.

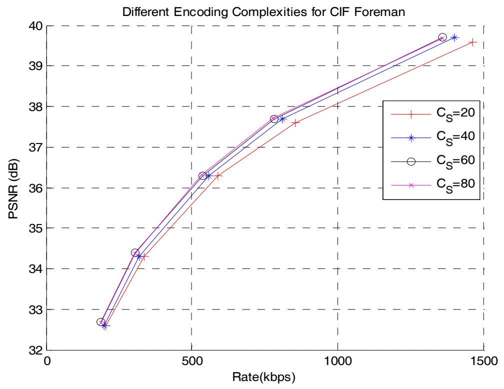


Figure 3. Rate-Distortion curves for different C_S values for the *Foreman* sequence at CIF 30fps resolution

also considered utilizing complexity parameters other than ME gradient, such as the number of predictors or search window size in [10], but they did not produce optimal R-D-C points.

We next consider parameters that are used to scale the complexity of encoding, given the algorithm above. We found using a training-based approach described in the next section, that the three parameters $\{\beta, \lambda_{MD}^{TH}, \lambda_{ME}^{TH}\}$ are the only viable parameters to scale complexity. The parameter β primarily determines the aggressiveness of early termination by choosing the INTER 16x16 mode with the predicted MV or INTRA mode without any prediction. The mode gradient threshold parameter λ_{MD}^{TH} determines the number of modes searched in large to small block size order. The motion estimation gradient threshold parameter λ_{ME}^{TH} determines the accuracy of the fast motion estimation steps for each block size.

3. TRAINING -BASED COMPLEXITY MAPPING

Generally, there are multiple inter-dependent parameters that can potentially be used to scale the complexity of a motion-mode search algorithm. However, allocation of complexity must be determined by a single controllable scalar complexity parameter C_S that can be adjusted with fine granularity at a MB level. Thus a mapping from C_S to the algorithmic parameters $\{\beta, \lambda_{MD}^{TH}, \lambda_{ME}^{TH}\}$ is necessary. Since the function that maps complexity parameters to the real complexity is implementation and platform dependent, we propose a training based generic methodology where the mapping is obtained by training on a given encoder implementation and platform.

First, we implemented the above algorithm on the fast ME [9] implementation included in the reference software [8]. Next, a training pool of typical videos is created, and run through the encoder with different combinations of finely sampled algorithmic parameters to generate rate-complexity (R-C) points, at a given constant quality (QP). From this collection of points, the convex hull is generated. Since only points on the convex hull are optimal for encoding for the given quality, parameter sets that do not generate points on the convex hull of the complexity-rate curve are pruned out. Figure 2 shows the points and the convex hull. For points on the convex hull, the optimal parameter combination for a given complexity level C_S can be readily read out. This information is next used to create a mapping from a complexity C_S to the corresponding algorithmic parameters. Since there may not be a set of parameters available at each C_S value, linear interpolation of the parameters is conducted to obtain the parameter values at each C_S point over a useful range. Fortunately, we also found this mapping to be largely independent of the quantization parameter QP, and therefore a convenient common mapping could be used irrespective of the target quality.

Note that the above training was in fact conducted not only for the three parameters mentioned above but also for several other potential parameters. However, it turned out that only changing $\{\beta, \lambda_{MD}^{TH}, \lambda_{ME}^{TH}\}$ generated points on the convex hull. Therefore we concluded that only these three parameters are useful for complexity scaling. The remaining ones are left at fixed optimal values. The training approach not only indicates the parameters that are useful for optimal scaling and provides the optimal mapping from C_S to them, but also provides the optimal fixed values for the other parameters.

The mapping for parameters $\{\beta, \lambda_{MD}^{TH}, \lambda_{ME}^{TH}\}$ obtained by the training procedure above leads to the following insight about the optimal scaling strategy. The highest complexity point for the algorithm is one where all INTRA and INTER modes are searched at the highest accuracy. To scale down complexity from this point, increasing the mode gradient threshold λ_{MD}^{TH} appears to be the best option, while keeping β and λ_{ME}^{TH} small. That is, the optimal strategy for complexity reduction at higher complexities is to reduce the number of INTER modes searched but not their accuracy. Complexity is decreased in this manner up to the point where only the 16x16 INTER mode can be searched. Thereafter, further complexity reduction is optimally achieved by increasing the motion estimation gradient threshold λ_{ME}^{TH} *i.e.* by reducing the accuracy of motion estimation process. If even further complexity reduction is desired, the optimal way would be to increase the early stop SAD threshold multiplier β. The minimum complexity point is achieved by setting the SAD threshold to a high number such that, all MBs are indeed encoded with MVP which reduces to zero, i.e., only the difference between consecutive frames is encoded. Thus the combination of the three parameters covers a wide enough range of complexities.

It is convenient to express the C_S parameter not in terms of the real complexity (encoding time), but in a scale between 0 and 100, where 100 corresponds to the maximum complexity point (where all INTER and INTRA modes are searched), and 0

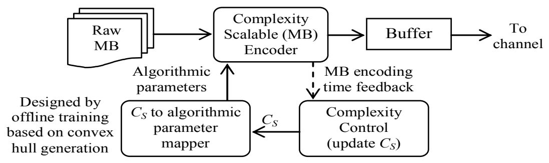

Raw MB
Complexity Scalable (MB) Encoder
Buffer
To channel
Algorithmic parameters
MB encoding time feedback
Designed by offline training based on convex hull generation
C_S to algorithmic parameter mapper
C_S
Complexity Control (update C_S)

Figure 4. Coding Delay Control Architecture

corresponds to the lowest complexity point where only the difference is encoded. Using this scale, Figure 3 shows the rate-distortion curves for different complexity (C_S) values.

4. COMPLEXITY CONTROL RESULTS

The scalar complexity parameter C_S per macroblock can now be used to control the coding delay of a software encoder. Ideally, the complexity allocation per MB should be such that the complexity-rate slope is identical for all MBs, and corresponds to a total coding delay target. This is an extension to the well known rate-distortion optimal bit allocation problem. However, the slope that corresponds to the complexity target is unknown, and further cannot be determined by methods such as bisection search as done in rate-distortion optimization, because complexity once expended cannot be reduced. Hence, we propose to control the complexity parameter using a control loop – specifically, a PID control loop.

Figure 4 shows our proposed architecture for coding delay control. The C_S parameter is updated on a MB by MB basis in a control loop to control the coding delay. The real encoding time per MB is measured by an accurate timer and fed back for the control. Specifically, we update the complexity parameter C_S using the complexity (coding time) error (deviation from target) based on a sliding window of previous M MBs:

$$C_S[i] = C_S[i-1] + K_P e[i-1] + K_D(e[i-1] - e[i-2]),\ e[i] = \sum_{k=0}^{M-1}(c[i-k] - C_T)$$

c denotes the real encoding time (complexity) for each MB measured with an accurate timer and C_T denotes the target complexity per MB. K_P and K_D denote proportional and derivative constants. Then, the complexity parameter C_S for each MB is mapped to encoding parameters before encoding. With this scheme, complexity can be controlled with fine granularity to achieve a target delay for the encoding time per any unit such as frame or group of frames.

Figure 5 shows the effectiveness of the coding delay control

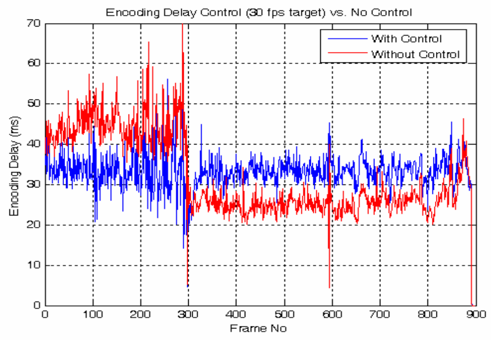


Figure 5. Coding delay per frame for controlled and uncontrolled encoders for a concatenation of three QCIF sequences (*Mobile + Akiyo + Foreman*). The controlled version maintains an average coding delay around the target set-point (33 ms for 30fps) even though the source statistics changes.

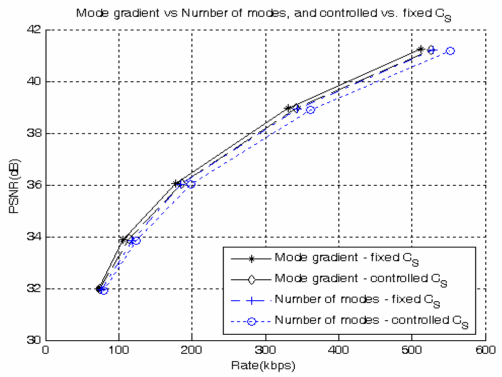


Figure 6. Rate-distortion curves for controlled vs. fixed C_S encoder.

mechanism for a concatenation of three different sequences using our software encoder based on implementation [9] in [8]. However, the control process, while necessary does have a penalty. Figure 6 compares the rate-distortion curves for a controlled C_S encoder vs. an encoder that uses a fixed C_S parameter. Each point in the controlled C_S version is run with the target coding delay equal to the overall encoding delay obtained with the fixed C_S version. Figure 6 also shows the superiority of the proposed complexity allocation over an alternative strategy where the tunable λ_{MD}^{TH} parameter is replaced by a parameter N_{MD} which is the number of modes to search in a pre-defined order, starting from 16×16 all the way to 4×4. This approach performs worse than the proposed gradient-based one by about 0.2dB PSNR.

5. CONCLUSION

A complexity scalable H.264/AVC encoder is developed for use in a real-time delay-constrained video communication. In the future, we will explore joint rate and complexity control to drive real-time buffer controlled streaming.

6. REFERENCES

[1] Draft ITU-T Rec. and Final Draft International Standard of Joint Video Specification, Joint Video Team, May 2003.

[2] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Magazine*, vol. 15 no. 6, pp. 23–50, November 1998.

[3] P. L. Tai, S. Y. Huang, C. T. Liu, J. S. Wang, "Computation aware scheme for software-based block motion estimation," *IEEE Trans. Circ. and Syst. Video Tech.*, vol. 13, no. 9, Sep. 2003.

[4] Z. Yang, H. Cai, and J. Li, "A framework for fine-granular computational-complexity scalable motion estimation," *IEEE Int. Symp. on Circuits and Systems*, Japan, 2005.

[5] C. Kim, J. Xin, A. Vetro, C. C. J. Kuo, "Complexity scalable motion estimation for H.264/AVC," *Proc. SPIE, Visual Comm. Image Proc.*, vol. 6077, pp. 109-20, Jan 2006.

[6] Z. He, Y. Liang, L. Chen, I. Ahmad, D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraint," *IEEE Trans. Circ. and Syst. Video Tech.*, vol. 15, May 2005.

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

[8] JVT Reference Software Version 10 [Online]. Available: http://iphome.hhi.de/suehring/tml/index.htm

[9] Z. Chen and Y. He, "Fast Integer and fractional pel motion estimation," Joint Video Team (JVT), JVT-F017, Dec. 2002.

[10] E. Akyol, D. Mukherjee, "Complexity control for video compression and transmission", *HP Labs Tech Report*, Nov 2006.