A Pixel-based Complexity Model to Estimate Energy Consumption in Video Decoders

Victor H. Costa, Pedro A. Assuncao
Instituto de Telecomunicacoes
Politecnico de Leiria
Morro do Lena - Alto Vieiro, Leiria, Portugal
Email: 2151151@my.ipleiria.pt, amado@co.it.pt

Paulo J. Cordeiro
Esc. Superior Tecnologia e Gestao
Politecnico de Leiria
Morro do Lena - Alto Vieiro, Leiria, Portugal
Email: paulo.cordeiro@ipleiria.pt

***Abstract*—The increasing use of HEVC video streams in diverse multimedia applications is driving the need for higher user control and management of energy consumption in battery-powered devices. This paper presents a contribution for the lack of adequate solutions by proposing a pixel-based complexity model that is capable of estimating the energy consumption of an arbitrary software-based HEVC decoder, running on different hardware platforms and devices. In the proposed model, the computational complexity is defined as a linear function of the number of pixels processed by the main decoding functions, using weighting coefficients which represent the average computational effort that each decoding function requires per pixel. The results shows that the cross-correlation of frame-based complexity estimation with energy consumption is greater than 0.86. The energy consumption of video decoding is estimated with the proposed model within an average deviation range of about 6.9%, for different test sequences.**

I. Introduction

The complexity of High Efficiency Video Encoding (HEVC) has been thoroughly investigated in recent years in order to devise fast methods capable of reducing the computational time required by encoders, while not compromising the compression efficiency [1] [2] [3]. Given the computational complexity of HEVC standard encoders and the ever increasing use of multimedia applications and services over mobile networks, reducing the coding complexity has been particularly relevant for resource-constrained devices and to also increase the battery-life of portable equipment. However, despite the fact that HEVC encoders are much more computationally demanding than decoders, the most used operation in mobile devices is decoding, integrated in different types of video players. Therefore, in the vast majority of user devices, the amount of energy consumed for video decoding is in general much higher than for video encoding. This trend is expected to increase in the near future, as the mobile video traffic share foreseen for 2021 is about 78% [4].

In the past, the problem of HEVC decoding complexity has been addressed by several researchers for different purposes. For instance, the performance of H.264/AVC decoding was investigated in [5] to identify the main bottlenecks and to improve decoder architectures. Closer to the scope of this paper, a power consumption model for H.264/AVC video decoders with hardware accelerator, was presented in [6]. The authors show that a small number of parameters can be used to estimate the power consumption of hardware accelerators with a maximum prediction error of 10%. In [7], the authors analyse the HEVC decoder complexity to devise a tile partitioning method capable of achieving load balancing in multicore platforms with speedup gains and energy savings. A linear relation between processing time and energy consumption was found in [8], bringing relevant insight for developing software-based decoders. Based on the high correlation between processing times and energy consumption, the energy savings between different implementations can be directly estimated from speedup gains in processing time. More recently the same authors extended their previous work and developed a feature-based model, using multiple bit stream parameters (20) and including the impact of memory access in energy consumption models [9]. An interesting finding is that decoding time based models yields slightly better estimation accuracy than parametric ones (6% in comparison to 8%) while in the case of hardware based decoders the maximum estimation error is about 20%.

In this paper, a pixel-based complexity model is devised to estimate the amount of energy required for HEVC software-based decoding of video streams. The underlying principle of the proposed model is based upon the principle that each decoded pixel contributes to the computational complexity with an amount that depends on the cumulative processing time of the various decoding functions used to compute its value. The spatial and temporal resolutions of the decoded video are implicitly embedded in the model. The proposed method can be used to estimate the amount of energy (e.g. given as a percentage of the battery capacity) required by a software-based decoder using just few frames (e.g. 10) to compute the model parameters.

II. Decoder Complexity Analysis and Modelling

In the case of HEVC encoding tools, the computational complexity has been evaluated in the recent past. For instance, in [1] it was observed that among those that consume more computational power one can distinguish between the All-intra (AI) and Random Access (RA) encoding configurations. In the former, the most computationally intensive functions are

the transform and quantisation, intra prediction and entropy coding, while in the latter, Motion Estimation (ME) requires the most significant portion of encoding computational power for SAD calculations and fractional pixel search refinement.

However in the case of video decoding, such complexity measures are not useful because many of the encoding functions are not required at the decoding side. For instance, the rate-distortion optimisation (RDO) process include many encoding functions in several optimisation loops, which significantly contribute for the encoding complexity, but not for decoding. For these reasons the computational complexity of encoding functions cannot be straightforwardly translated to the decoding side. In [10], the decoding computational complexity in the AI configuration was found to be dominated by the inverse transform and deblocking filters while Motion Compensation and filters are the most complex functions in the RA configuration. Based on these results, the functions that most contribute for decoding complexity were defined as Inter (includes Motion Compensation), Intra, Inverse Transform, Deblocking Filter and SAO filter.

A. Pixel-based decoding complexity model

Based on the evidence found in previous works cited above, a linear complexity model for video decoding is proposed. The following decoding functions are defined as those which mostly contribute for the computational complexity of HEVC video decoders: Motion Compensation (Inter), Intra Prediction (Intra), Inverse Transform & Quantization (IT), Deblocking filter (Deblock), Sample Adaptive Offset filter (SAO).

In the proposed model, the computational complexity is defined as a weighted sum of the number of pixels processed by each decoding function defined above. The weighting factors are defined by the computational complexity per pixel of each decoding function. Then, for each CU, the corresponding complexity is given by the following expression:

$$C_{CU_i} = \sum_{m \in M} k(m).P_i(m) \tag{1}$$

with

$$M = \{Inter, Intra, IT, Deblock, SAO\} \tag{2}$$

$P_i(m)$ is the number of pixels in CU_i that are processed by function m and $k(m)$ is the corresponding complexity per pixel, as described above. Then the decoding complexity of a whole frame (C_f) is obtained by summing the complexities of all CUs, i.e.,

$$C_f = \sum_{i=1}^{N_{CU}} C_{CU_i} \tag{3}$$

The weighting coefficients $k(m)$ are model parameters, which depend on the decoder implementation. They can be determined online after decoding few frames of a video stream, counting the number of pixels processed by each function defined in M and measuring the computational complexity (i.e., processing timed) that is required for decoding a time-limited window of w consecutive frames. The ratio between the processing time and the number of pixels gives the average computational complexity per processed pixel for a given decoding function $m \in M$, i.e.,

$$k(m) = \frac{T_w(m)}{P_w(m)}, \qquad m \in M \tag{4}$$

Note that the parameters $k(m)$ are computed for different decoder implementations and hardware platforms, which means that the decoding complexity estimated by this model is implicitly adapted to different devices. This is an advantage of this model, which enables its implicit adaptation to different hardware platforms and/or software implementations.

B. Model parameters

To evaluate the accuracy of the complexity estimation model, its output was compared with the processing time of each decoding function measured by a code profiler, such as the Intel Vtune Amplifier XE. An experimental setup, comprising the reference decoder HM, running on Intel i7-2400 2.4GHz CPU with 24 GByte memory with Microsoft Windows, was implemented to obtain the computational complexity of decoding functions, measured as the accumulated processing time required by each method. All tests were conducted following common HEVC test conditions and software reference configurations defined by [1]. The tests were performed for All Intra, Random Access and Low Delay. Only the processing time of decoding functions was measured, thus excluding the I/O functions from the measurements in the profiler. Test sequences with two resolutions were used: HD (1920X1080 or 1080p), which is representative for most devices today, including mobile (e.g. smartphones and tablets) and 4K/UHD (3840X2160 or 2160p), which is representative for the next generation of high quality video. For 1080p, the five class B sequences from the JCT-VC test set have been used at 24, 50 and 60 frames per second (fps).

The model parameters $k(m)$ were first computed for several sequences. Since the results obtained from the various sequences exhibit a similar behaviour, only those regarding sequence Kimono are shown in this subsection. Table I shows the intermediate results that lead to the values of $k(m)$. The first column identifies the main decoding functions of HEVC decoder. Column #Pels represents the number of pixels processed by the functions from each decoding function. Note that, in this counting the same pixel can go through different functions, thus the #Pels in different functions may include counting the same pixel. The third column shows the computational time spent on each function. The fourth and last columns show the pixel-based complexity for each decoding function as given by equation 4. The last column $k(m)$ shows the normalized pixel-based complexity, using the case of Intra Prediction as the reference.

Using the model parameters obtained from each sequence, the generic values of $k(m)$ that are valid for the decoder implementation used in this work are shown in Table II. These were computed as the average $k(m)$ for all sequences and quantisation parameters.

Dec. Mod	#Pels	P.Time(s)	Time/Pel (s)	$k(m)$
Inter	458551808	11.37	2.48E−08	$k(Ir)$=1.45
Intra	39112192	0.67	1.71E−08	$k(Ia)$=1.00
Transform	120137328	1.12	9.28E−09	$k(IT)$=0.54
Deblock	75534608	3.12	4,13E−08	$k(dblk)$=2.42
SAO	67434068	0.89	1,32E−08	$k(SAO)$=0.77

TABLE I
MODEL PARAMETERS FOR KIMONO_1920X1080_24_QP27, RA.

Inter $k(Ir)$	Intra $k(Ia)$	Transform $k(Tr)$	Dblk $k(dblk)$	SAO $k(SAO)$
0.81	1	0.47	1.79	0.36

TABLE II
AVERAGE $k(m)$ VALUE FOR DECODING FUNCTIONS

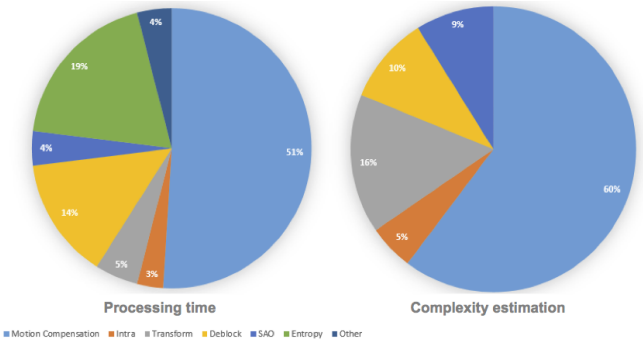

Fig. 1. Relative decoder complexity (Kimono, $QP = 27$, RA)

C. Complexity estimation vs processing time

Figure 1 presents the decoder's complexity per decoding function for sequence Kimono, measured by the code profiler as processing time (left) and estimated by the proposed model (right) while running the decoder. Besides the decoding functions included in set M, the pie-chart on the left also shows the processing time measured in entropy decoding and other functions (e.g., file I/O, etc). In the case of the complexity estimation given by the proposed model (i.e., pie chart on the right), entropy decoding is diluted in the main five functions. These graphs show that the relative decoding complexity burden estimated by the proposed model follows a distribution that is similar to the processing time measured by the code profiler. The main differences observed in corresponding functions of Figure 1 are due to the implicit inclusion of entropy decoding (19% of processing time) in the complexity estimation by the proposed model (i.e., pie chart on the right). For instance, this can be observed in the relative decoding complexity of the pixels accounted for the *Transform* and *Motion Compensation* functions, which is significantly higher than the corresponding processing time. In the case of the *Transform* this is justified by the complexity of entropy decoding functions being added to that of inverse quantisation and inverse transform, whereas for processing time, this is separately measured, i.e., not added to inverse quantisation and inverse transform. In the case of the *Motion Compensation*, the complexity of entropy decoding functions is added to complexity of CUs without coded coefficients (e.g. skip), whose pixels are not considered in the *Transform* function.

III. ESTIMATION OF ENERGY CONSUMPTION

The accuracy of the decoder complexity model presented in the previous section was evaluated for estimation of the energy consumption in video decoding. For this purpose, a lightweight performance tool suite, known as *likwid-perfctr* from LIKWID TOOLS was used to measure the decoding energy consumption [11],[12]. Using *likwid-perfctr*s marker API it is possible to measure the energy consumption of selected functions of an application by turning on/off hardware performance counters, which allows to obtain the CPU energy consumption per frame. Even though these tests were performed with minimal interference of other running processes, to minimize the impact of kernel processes on the intended measurements, the results were obtained running the same test several times (e.g., 6-8) and then normalising the average results. Also to get more accurate measurements, these started at each frame's first call of the decode method (class TDecTop) and finished after the execution of loop filters, when frames are fully reconstructed.

The normalised decoding complexity per frame, computed by the proposed model, was compared with the normalised energy consumption for each frame, measured by the *likwid-perfctr* tool. Figure 2 and Figure 3 show these results for sequences Ready Steady Go with QP=22 and Bosphorus with QP=37. These Figures clearly show that there is a high correlation between the computational complexity estimated by the proposed model and the energy consumption per frame. For other sequences and QPs, the profile behaviour is the same.

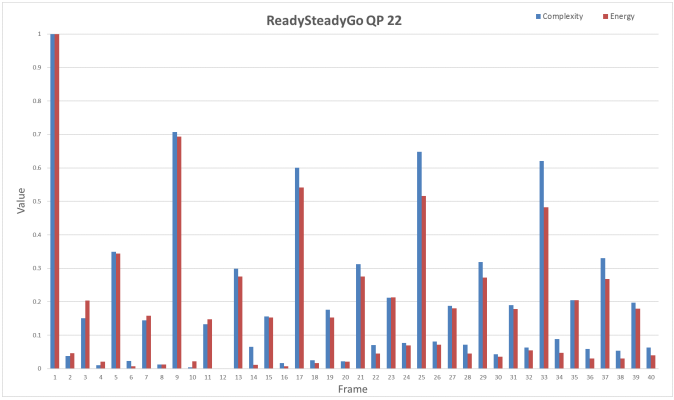

Fig. 2. Decoding complexity vs Energy consumption (ReadySteadyGo, QP22, RA)

To further evaluate the degree of similarity between the decoding complexity and energy consumption the Pearson Correlation Coefficient (PCC) was computed for 3 sequences, as shown in Table III. These results show that the proposed pixel-based decoding complexity model is capable of producing a complexity measure highly correlated with the energy consumption, which allows to estimate the amount

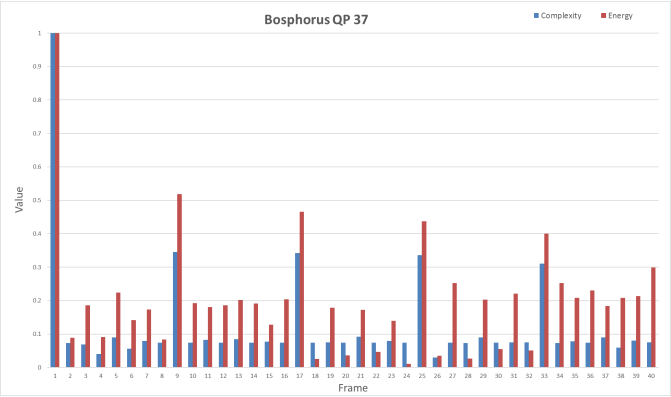

Fig. 3. Decoding complexity vs Energy consumption (Bosphorus, QP37, RA)

of energy necessary to decode time-limited video sequence and present such information to mobile users as percentage of the battery life. The next subsection describes the details of such application of the proposed model and presents accuracy results.

	QP22	QP27	QP32	QP37
Beauty	0.910	0.895	0.986	0.932
Bosphorus	0.984	0.971	0.914	0.857
ReadySteadyGo	0.951	0.960	0.959	0.938

TABLE III
PCC BETWEEN DECODING COMPLEXITY AND ENERGY CONSUMPTION PER FRAME

The mean squared error (MSE) between both normalised curves is shown in Table IV, where it can be seen that these two measures are quite similar. Since the decoding computational complexity and the energy consumed per frame are obtained through independent methods, the results shown in Table III and Table IV clearly demonstrate that the proposed model is quite accurate in estimating the energy consumption of video decoders. Note that the model parameters $k(m)$ are computed based on the processing time of each device and decoder implementation, thus the model is implicitly adapted to different decoders.

	QP22	QP27	QP32	QP37
Beauty	0.061	0.057	0.030	0.040
Bosphorus	0.010	0.006	0.008	0.007
ReadySteadyGo	0.008	0.009	0.005	0.003

TABLE IV
MSE BETWEEN DECODING COMPLEXITY AND ENERGY CONSUMPTION PER FRAME

A. Application in practical video decoders

A practical application of the proposed model is to estimate the percentage of battery-life that is required to decode a video stream, as described next. For this purpose, a compressed stream capable of enabling energy consumption estimation by different decoders should be generated. This is done by associating the normalised decoding computational complexity per frame, as computed by the proposed model, to compressed streams. Such new functionality can be implemented as side information that can be either fetched from a video server as a user option or sent along with the video stream as supplemental enhancement information (SEI). Then, after decoding an initial short time video segment (e.g. one GOP), during which the decoder measures the actual energy consumed to decode such frames, the total amount of energy required to decode the entire video stream can be computed from the quasi-constant relationship between the complexity and consumed energy per frame, i.e., $C_f/E_f \approx Const$. It was found that the Relative Standard Deviation (RSD) of such ratio is 3,7%, which indicates low dispersion of values around the mean.

Given the first few frames, such constant can be computed by using the corresponding decoding complexity and consumed energy. Then the remaining energy can be computed by simply using the normalised complexity associated to the whole stream and the constant ratio between complexity and consumed energy per frame. Note that such constant can be different for different sequences and QPs.

Table V shows a comparison between the energy per frame estimated by using the constant complexity-energy ratio and the actual energy measured by the *likwid-perfctr* tool. The complexity-energy ratio was computed after decoding and measuring the energy consumption of the first 10 frames of each sequence. Then the constant ratio is used for estimation of the energy / frame for all frames ahead in the sequence (i.e., 590). The comparison with the actual energy per frame shows that the estimation error ΔE lies in the range 1.5% - 14.8% for different sequences. The greater deviation happens for sequence ReadySteadyGo, which has the most active content in both the spatial and temporal dimensions. This suggests that either the constant complexity-energy ratio should be frequently updated while decoding the video stream to allow better estimation of dynamic scenes ahead or more frames should be initially used to measure the decoding and to calculate its value. By converting the energy estimation into a percentage of the battery capacity and admitting a maximum estimation error of about 15% (average $\Delta E = 6.9\%$), the proposed method can be integrated in a battery-life management tool for user control.

	Energy/frame (est.)	Energy/frame (real)	ΔE (%)	avg(ΔE)
Beauty				**5.5**
QP 27	8.96	8.36	7.2	
QP 32	6.21	6.30	1.5	
QP 37	6.00	5.56	7.9	
Bosphorus				**2.8**
QP 27	7.41	7.12	4.1	
QP 32	6.55	6.38	2.7	
QP 37	3.17	5.94	1.5	
ReadySteadyGo				**12.5**
QP 27	11.03	9.61	14.8	
QP 32	9.00	7.85	14.6	
QP 37	6.91	6.39	8.2	
		Global Average		**6.9%**

TABLE V
PREDICTED VS MEASURED ENERGY/FRAME USING ONLY THE FIRST 10 FRAMES FOR PARAMETER ESTIMATION (GLOBAL AVERAGE $\Delta E = 6.9\%$)

IV. Conclusion

This paper described a pixel-based complexity model for HEVC decoders that is highly correlated with the energy consumption. Such model is defined as a weighted sum of the number of pixels that are processed through each decoding function, which allows to account for the different decoding complexities required by each coding unit. The usefulness of the high correlation between complexity and energy consumption is shown to allow the application of the proposed model in helping mobile users to manage the battery-life of their devices, by providing estimations of the energy required to decode a video sequence prior its fully decoding.

References

[1] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "Hevc complexity and implementation analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1685–1696, Dec 2012.

[2] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz, "Complexity control of high efficiency video encoders for power-constrained devices," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, pp. 1866–1874, November 2011.

[3] G. Correa, P. A. Assuncao, L. V. Agostini, and L. A. da Silva Cruz, "Pareto-based method for high efficiency video coding with limited encoding time," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 9, pp. 1734–1745, Sept 2016.

[4] Cisco, "Global mobile data traffic forecast update 20162021," White paper, Tech. Rep., March 2017.

[5] M. Alvarez, E. Salami, A. Ramirez, and M. Valero, "A performance characterization of high definition digital video decoding using h.264/avc," in *IEEE International. 2005 Proceedings of the IEEE Workload Characterization Symposium, 2005.*, Oct 2005, pp. 24–33.

[6] X. Li, Z. Ma, and F. C. A. Fernandes, "Modeling power consumption for video decoding on mobile platform and its application to power-rate constrained streaming," in *2012 Visual Communications and Image Processing*, Nov 2012, pp. 1–6.

[7] H. Baik and H. Song, "A complexity-based adaptive tile partitioning algorithm for hevc decoder parallelization," in *2015 IEEE International Conference on Image Processing (ICIP)*, Sept 2015, pp. 4298–4302.

[8] C. Herglotz, E. Walencik, and A. Kaup, "Estimating the hevc decoding energy using the decoder processing time," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 513–516.

[9] C. Herglotz, D. Springer, M. Reichenbach, A. Kaup, and B. Stabernack, "Modeling the energy consumption of the hevc decoding process," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, no. 99, pp. 1–1, 2016.

[10] M. Wien, M. Budagavi, K. U. K. Mishra, and X. Xiu, "JCT-VC AHG report: Single-loop scalability (AHG16)," 2013.

[11] J. Treibig, G. Hager, and G. Wellein, "Likwid: A lightweight performance-oriented tool suite for x86 multicore environments," in *2010 39th International Conference on Parallel Processing Workshops*, Sept 2010, pp. 207–216.

[12] T. Rehl, J. Treibig, G. Hager, and G. Wellein, "Overhead analysis of performance counter measurements," in *2014 43rd International Conference on Parallel Processing Workshops*, Sept 2014, pp. 176–185.