FAST HEVC INTRA CODING ALGORITHM BASED ON MACHINE LEARNING AND LAPLACIAN TRANSPARENT COMPOSITE MODEL

Yi Shan and En-hui Yang, Fellow, IEEE

Dept. of ECE, University of Waterloo, Waterloo, ON, Canada (Emails: {y9shan; ehyang}@uwaterloo.ca)

ABSTRACT

Compared with H.264, High Efficient Video Coding (HEVC) improves the coding efficiency by 50% at the price of significant increase in encoding time, due to Rate Distortion Optimization (RDO) on large variations of block sizes and prediction modes. In this paper, a fast intra coding algorithm is proposed to alleviate the high computational complexity of HEVC intra-frame coding. The proposed algorithm is based on machine learning and Laplacian Transparent Composite Model (LPTCM). Features called Summation of Binarized Outlier Coefficient (SBOC) vectors are firstly extracted from original frames by using LPTCM and then fed into online trained Support Vector Machine (SVM). Two SVMs are combined to predict Coding Unit (CU) decisions so that the encoding process can be significantly sped up. Additionally, a performance controller is introduced to ensure the robustness of machine learning models. It is shown by experiments that compared with HM 16.3, the proposed algorithm reduces the encoding time, on average, by 48% with negligible increase in BD-rate.

Index Terms— High Efficiency Video Coding, Fast Intra Coding, Support Vector Machine, Transparent Composite Model.

1. INTRODUCTION

High Efficiency Video Coding (HEVC) is the latest video coding standard developed by the Joint Collaborative Team on Video Coding (JCT-VC), which is jointly established by ISO/IEC Moving Picture Experts Group (MPEG) and IUT-T Video Coding Experts Group (VCEG) [1]. In comparison with the previous video coding standard H.264, HEVC has improved the compression performance by nearly 50% bit-rate reduction under the same distortion [2]. However, the computational cost of HEVC has been increased tremendously [3].

Take HEVC intra coding as an example. The main reason for its high computation complexity lies in its novel quad-tree structure and adoption of a large number of Intra Prediction Mode (IPM), which is up to 35 [4]. To decide the optimal quad-tree structure and IPM, the encoder has to traverse all possible combinations through Rate Distortion Optimization (RDO), which in turn increases the computation complexity dramatically. Based on our experiments, if the Coding Unit (CU) decision can be made a priori, the encoding time can be saved by nearly 65% when compared with the normal HM encoder. In other words, with optimal CU decision available in advance, nearly 65% encoding time can be saved. As such, an effective way to reduce the encoding time of HEVC intra coding is to predict CU decision. On the other hand, because 35 IPMs need to be checked, another strategy is to narrow the range of IPM candidates for Rough Mode Decision (RMD) or RDO.

Therefore, in the literature, most works on fast HEVC intra coding can be categorized into three types. The first type concentrates on CU size decision, like [5] [6]. Another type is only based on IPM decision, like [7]. The third type chooses to combine these two principles, like [8] [9] [10]. Within the three categories, a problem on how to achieve better performance has resulted in variety of solutions. Traditional fast algorithms are mainly based on fixed designs and threshold-based methods [10] [11]. In [5], features, which include mean and variance of the image, are extracted to compare with a threshold. By merging the Prediction Unit (PU) blocks from bottom to top, two PU maps are generated to narrow down the range of RDO. However, the threshold in [5] is obtained from experiments and it can only change with Quantization Parameter (QP) rather than the content of videos. To make fast algorithms more adaptive, many machine learning methods are applied in this topic. In [12], a fast CU decision algorithm based on Neutral Network (NN) is proposed for compression of screen content while causes a relatively large loss in coding efficiency. For machine learning methods, a prime problem is how to extract and process features . Method in [8] provides a dimensionality reduction idea that utilizes Laplacian Transparent Composite Model (LPTCM). A feature called Outlier Block Flag (OBF) is used to train a Bayes model that predicts the CU decision. However, the feature used in [8] is only a single number and the information utilized may not be sufficient for accurate prediction. On the other hand, model selection and training are also crucial

THIS WORK WAS SUPPORTED IN PART BY THE NATURAL SCIENCES AND ENGINEERING RESEARCH COUNCIL OF CANADA UNDER GRANT RGPIN203035-11, AND BY THE CANADA RESEARCH CHAIRS PROGRAM.

factors in success of machine learning. Since Bayes model usually needs more training samples than other methods, Support Vector Machine (SVM) may be a better choice when handling high-dimensional features. In [9], a combined algorithm is proposed, where off-line trained SVMs are used to decide CU size and online trained thresholds are used to reduce IPM candidates. In [6], 3 SVM models based on different features are trained to predict the CU decision collaboratively, but the result is not very satisfactory. Therefore, there are many innovations need to be made in feature extraction, model training, and robustness control.

In this paper, we propose an online fast intra coding algorithm based on switching SVM together with a new feature set called Summation of Binarized Outlier Coefficients (SBOC) vector. Two SVMs of different parameter settings are trained online to predict the CU decision. Additionally, a performance controller is introduced to avoid the loss caused by ineffective prediction. By skipping and terminating CU checking on some depth of the quad-tree, the encoding time can be significantly saved with negligible loss in coding efficiency. Through the All Intra Main (AIM) full test, the performance of the proposed algorithm outperforms those CU decision based algorithms in the literature.

The rest of the paper is organized as follows. Some background knowledge of SVM and two types of CU decision are briefly introduced in Sec. 2. In Sec. 3, the proposed method will be introduced in detail. In Sec. 4, a full test is carried out to compare the proposed method with some high-quality works in the literature. Finally, conclusions are drawn in Sec. 5.

2. SUPPORT VECTOR MACHINE AND CU DECISION

SVM is a boundary-based classification method, which uses hyperplanes to separate data points of different classes in an N-dimensional space. In our application, the problem is modeled as binary classification, and the hyperplane can be represented as

$$\boldsymbol{w}^T\boldsymbol{x} + b = 0, \tag{1}$$

where $\boldsymbol{x} \in R^N$ is the feature vector, $\boldsymbol{w}$ is the coefficient vector of the hyperplane and b is the offset. The hyperplane is trained from training set

$$S_t = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2)..., (\boldsymbol{x}_M, y_M)\}. \tag{2}$$

Each sample in the S_t is a feature-label pair, where label $y \in \{-1, +1\}$. Under our application, $y = +1$ in the training set represents the case that splitting the current CU into sub-CUs can achieve better coding efficiency. While, $y = -1$ means splitting current CU will result in larger RD-cost and the CU partition should be terminated. Because, in general case, the training data may not be perfectly separable, the error and penalty need to be considered. To balance the performance with computational complexity, the SVM used in our algorithm is a linear weighted SVM with penalty parameter C [13]. The optimal hyperplane is chosen as the one that minimized the systemic error. It is equivalent with solving the following optimization problem:

$$\min_{\boldsymbol{w},b} \{\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C(W^+ \sum_{i=1}^{|S_t^+|} \xi_i + W^- \sum_{j=1}^{|S_t^-|} \xi_j)\} \tag{3}$$

subject to

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i, and\, \xi_i \geq 0 \quad \forall(\boldsymbol{x}_i, y_i) \in S_t, \tag{4}$$

where C is the penalty parameter and ξ_i is the slack variable [14]. W^+ is the weight for positive error while W^- is the weight for negative error. Sample vectors satisfying $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) = 1$ are called support vectors, which can uniquely decide the hyperplane. After the model is trained, the classification rule for new data sample $\boldsymbol{x}$ is

$$y_{new} = sign(\boldsymbol{w}^T\boldsymbol{x}_{new} + b), \tag{5}$$

and y_{new} is used as the prediction of CU decisions.

In our application, $y_{new} = +1$ means CU skip while $y_{new} = -1$ means CU termination. If the decision is CU skip, the encoder will skip all the process on the current depth, directly split the CU into four sub-CUs and jump to the next depth of the quad-tree. If the decision is CU termination, the encoder will terminate the CU splitting and keep the current CU size as the final choice.

3. PROPOSED METHOD

3.1. Fast CU Decision Algorithm: Structure and Logic

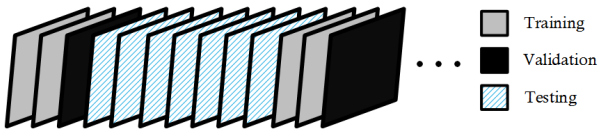

Fig. 1. Train-validate-test period

To make the algorithm adaptive with any input sequences, an periodical online training scheme is used as high-level structure of our proposed algorithm. Input video frames are periodically segmented into several train-validate-test groups, as shown in Figure 1. Suppose each group has P frames, the first T frames are used to collect training data set and train the SVM models. There are two SVM models: SVM1 and SVM2. SVM1 is designed to achieve high precision on CU skip. Therefore, the weight of negative error for SVM1 is larger than that of positive error ($W_1^- > W_1^+$). SVM2 is

expected to have high precision on CU termination and it has $W_2^+ > W_2^-$. These two SVMs will switch between each other and work together as a Decision Maker (DM) in the following frames.

To avoid the performance loss due to ineffective case and overfitting, another V frames called validation frames are used to verify the effectiveness of predictions. Based on the performance feedback from those V frames, a performance controller starts working to switch off the predictions for unsatisfactory decisions in specific quad-tree depths. A precision threshold Th_P is set to be 0.8 in the main version. The definition of precision is as follow:

$$Precision(i) = \frac{N(y_{pred} = i, y_{true} = i)}{N(y_{pred} = i)}, i = +1, -1 \quad (6)$$

where $N(y_{predict} = i)$ is the number of samples that are predicted to be class i, $N(y_{pred} = i, y_{true} = i)$ is the number of samples that are correctly predicted to be class i. If the precision of one prediction is less than Th_P, the prediction from the corresponding SVM will be switched off. If predictions from both SVMs are switched on, the DM will trust the SVM with higher precision.

After this, the DM is applied into the rest $P-T-V$ frames to accelerate the encoding. Feature vectors are extracted from each CU and further fed into the DM for prediction. In a CU of a prediction frame, if the DM outputs +1 and the decision is switched on through validation, the encoder will execute CU skip; if the DM outputs -1 and the decision is activated, the CU splitting will be terminated.

3.2. Feature Extraction

Here comes to how feature vector $\boldsymbol{x}$ is extracted. Firstly, the original frame is segmented into 4×4 blocks and each block is transformed by 4×4 Discrete Cosine Transform (DCT). A DCT coefficient map can be generated. After that, the LPTCM models can be online obtained by maximum likelihood estimation. This step is finished before the encoding so that the features are available at each depth of the quad-tree. Equation 7 shows the probability density function of LPTCM [15]

$$p\left(\mathrm{y}|\mathrm{y}_c, b, \lambda\right) = \begin{cases} \frac{b}{1-e^{-\mathrm{y}_c/\lambda}} \frac{1}{2\lambda} e^{-|\mathrm{y}|/\lambda}, \text{ if } |\mathrm{y}| < \mathrm{y}_c \\ \frac{1-b}{2(a-\mathrm{y}_c)}, \text{ if } \mathrm{y}_c < |\mathrm{y}| \leq a \\ \max\left\{ \frac{b}{1-e^{-\mathrm{y}_c/\lambda}} \frac{1}{2\lambda} e^{-|\mathrm{y}|/\lambda}, \frac{1-b}{2(a-\mathrm{y}_c)} \right\}, \text{ if } |\mathrm{y}| = \mathrm{y}_c \\ 0, \; otherwise, \end{cases} \quad (7)$$

where a is the largest value of the DCT coefficients and y_c, b, λ are model parameters estimated online. The main body of LPTCM ($|\mathrm{y}| < \mathrm{y}_c$) is Laplacian distribution, while the tail ($\mathrm{y}_c < |\mathrm{y}| \leq a$) is modeled by uniform distribution. For DCT coefficients that fall into the main body, they are categorized as inlier. While for large DCT coefficients that exceed the y_c, they are called outliers. Outliers account for very small percentage of AC coefficients (about 1.2%), but they contain very important information about the content of images. The process of feature extraction is shown in Figure 2. Inliers and the DC coefficients will be suppressed to 0 and outliers will be quantized into 1. Then the binary outliers within each 4×4 block will be summed up to form a single value, which is called SBOC. Since CU size is always larger than 4×4, a CU contains multiple SBOC values. These SBOC within one CU will be raster scanned into a multidimensional vector as the final feature. For example, a CU of size 16×16 will result in a feature vector of $(16/4)^2 = 16$ dimensions. It should be pointed out that, in comparison with [16], where a single value is used for each CU, the feature set here is much richer. It contains information about both the number and the position of outliers.

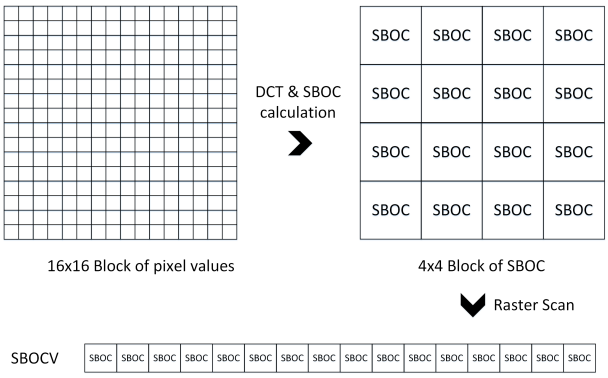

Fig. 2. Feature extraction process

4. EXPERIMENTAL RESULTS

To test the effectiveness of our proposed algorithm, a full test is carried out under the common test condition provided by the official document [17]. The full test includes 24 test sequences of 6 classes and all sequences were encoded under AIM configuration. The QPs are set to 22, 27, 32 and 37. To calculate the BD-rate [18] and Time Saving (TS), test results of fast algorithms are compared with normal HM that the algorithms based on (e.g. HM 16.3 for our proposed method, HM 14.0 for [9]). TS is calculated according to Equation (8):

$$TS = \frac{T_{HM} - T_{fast}}{T_{HM}} \times 100\% \quad (8)$$

where T_{HM} and T_{fast} are the encoding time of the original HM and the fast algorithm, respectively. For training period, parameters are set to be $P = 60, T = 2, V = 1$. The machine

Table 1. Full-test results of [9], [16] and proposed method on the same machine

	TCSVT2016 [9]		TCSVT2015 [16]		Proposed (Main)	
Class	BD-Rate	TS	BD-Rate	TS	BD-Rate	TS
ClassA (4 Seq.)	0.73%	51.70%	0.33%	41.01%	0.53%	54.76%
ClassB (5 Seq.)	0.63%	49.95%	0.43%	38.40%	0.78%	48.90%
ClassC (4 Seq.)	0.62%	41.20%	0.40%	34.23%	0.83%	42.27%
ClassD (4 Seq.)	0.54%	37.95%	0.34%	33.67%	0.71%	48.65%
ClassE (3 Seq.)	1.10%	56.76%	0.66%	41.28%	0.74%	51.78%
ClassF (4 Seq.)	2.03%	53.05%	0.62%	47.88%	0.98%	49.02%
Overall Average	0.92%	48.15%	0.46%	39.29%	0.78%	48.03%

used to run experiments is of Intel(R) Core (TM) i7-4790 3.60GHz CPU, and 8GB RAM. In the main version of the proposed algorithm, the parameters are set to be: $Th_p = 0.8$, $C = 0.000002$, $W_1^+ = 1$, $W_1^- = 20$, $W_2^+ = 50$, $W_2^- = 1$. Since the encoding time vary from machine to machine, we have regenerated the test of [16] and [9] together with our method on the same machine. Here, I would like to thank authors of [16] and [9] for their sharing the executable encoders. The result of different video classes are shown in the Table 1. On average, the proposed algorithm can achieve 48.03% time reduction with only 0.78% increase in BD-rate. Our algorithm works very well for 2560×1600 4K videos (Class A), which may have the strongest need for reducing encoding time. The average TS for 4K video is 54.76% with only 0.53% increase in BD-rate. For benchmarks, because the models in [9] is off-line trained and data of class F is not included in their training set, the result of [9] on class F is not very satisfactory, which causes 2.03% BD-rate increase. However, for our online method, the BD-rate increase of class F is only 0.98%. To some extent, this comparison shows the advantage of online training.

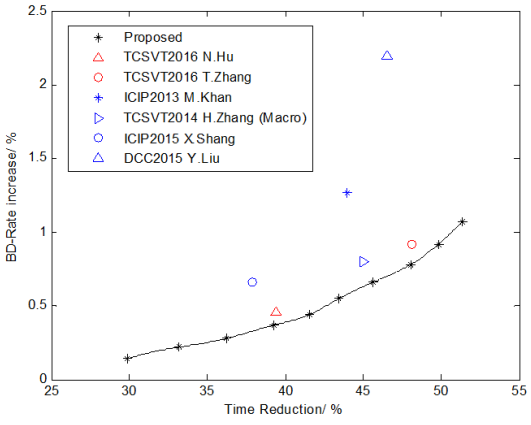

Fig. 3. Performance comparison

To provide more intuitive comparisons, the Th_p is changed from 0.76 to 0.94 to reach different trade-off points. Therefore, in Figure 3, a curve is generated and compared with benchmarks. Some recent results in the literature are also cited and plotted in the figure and the specific data in benchmarks are shown in Table 2. For benchmark points above the curve, one can always find a black point on the curve that outperforms the benchmark in both TS and BD-rate. For example, the result of [16] is 39.29% TS with 0.46% BD-rate. While, the black point on its right achieves 41.53% TS with 0.44% BD-rate, which shows that our proposed method outperforms the Bayes method in [16]. For [10], it is a synthetic method and the paper reports the contribution of each part individually. Since our method is based the CU decision, only the result of CU decision part (Macro) in [10] is cited.

Table 2. Results cited from other benchmarks

Encoder Version	TS	BD-rate	From
[5]ICIP2013	44.00%	1.27%	Paper
[10]TCSVT2014(Macro)	45%	0.8%	Paper
[11]ICIP2015	37.91%	0.66%	paper
[6]DCC2015	46.50%	2.20%	Paper

5. CONCLUSION

This paper has proposed a new fast CU decision algorithm for HEVC intra coding based on machine learning and LPTCM. Experimental results have shown that the main version of the proposed algorithm can achieve significant TS (overall 48.03%) with negligible increase in BD-rate. It compares favorably with other fast algorithms for HEVC intra coding proposed recently in the literature. Its new ingredients include an effective feature set called SBOC vector extracted from each original frame by using LPTCM, DM consists of two switching SVMs and its performance controller are determined during the validation period. Since the proposed algorithm only utilizes the feature extracted from original frames, it does not rely on internal data of encoding. This makes the proposed method more collaboration-friendly with methods based other principles (e.g. IPM decision).

6. REFERENCES

[1] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[2] Jens-Rainer Ohm, Gary J Sullivan, Heiko Schwarz, Thiow Keng Tan, and Thomas Wiegand, "Comparison of the coding efficiency of video coding standards-sincluding high efficiency video coding (hevc)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669–1684, 2012.

[3] Frank Bossen, Benjamin Bross, Karsten Suhring, and David Flynn, "Hevc complexity and implementation analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1685–1696, 2012.

[4] Jani Lainema, Frank Bossen, Woo-Jin Han, Junghye Min, and Kemal Ugur, "Intra coding of the hevc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1792–1801, 2012.

[5] Muhammad Usman Karim Khan, Muhammad Shafique, and Jorg Henkel, "An adaptive complexity reduction scheme with fast prediction unit decision for hevc intra encoding," in *Image Processing (ICIP), 2013 20th IEEE International Conference on*. IEEE, 2013, pp. 1578–1582.

[6] Yen-Chun Liu, Zong-Yi Chen, Jiunn-Tsair Fang, and Pao-Chi Chang, "Svm-based fast intra cu depth decision for hevc," in *Data Compression Conference (DCC), 2015*. IEEE, 2015, pp. 458–458.

[7] Liang Zhao, Li Zhang, Siwei Ma, and Debin Zhao, "Fast mode decision algorithm for intra prediction in hevc," in *Visual Communications and Image Processing (VCIP), 2011 IEEE*. IEEE, 2011, pp. 1–4.

[8] Nan Hu and En-Hui Yang, "Fast mode selection for hevc intra-frame coding with entropy coding refinement based on a transparent composite model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 9, pp. 1521–1532, 2015.

[9] Tao Zhang, Ming-Ting Sun, Debin Zhao, and Wen Gao, "Fast intra mode and cu size decision for hevc," 2016.

[10] Hao Zhang and Zhan Ma, "Fast intra mode decision for high efficiency video coding (hevc)," *IEEE Transactions on circuits and systems for video technology*, vol. 24, no. 4, pp. 660–668, 2014.

[11] Xiwu Shang, Guozhong Wang, Tao Fan, and Yan Li, "Fast cu size decision and pu mode decision algorithm in hevc intra coding," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1593–1597.

[12] Fanyi Duanmu, Zhan Ma, and Yao Wang, "Fast cu partition decision using machine learning for screen content compression," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4972–4976.

[13] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[14] Chih-Chung Chang and Chih-Jen Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, pp. 27, 2011.

[15] En-Hui Yang, Xiang Yu, Jin Meng, and Chang Sun, "Transparent composite model for dct coefficients: Design and analysis," *IEEE Trans. Image Process.*, vol. 23, no. 3, pp. 1303–1316, 2014.

[16] Nan Hu and En-Hui Yang, "Erratum to fast mode selection for hevc intra-frame coding with entropy coding refinement based on a transparent composite model," .

[17] F Bossen and HM Common, "test conditions and software reference configurations, jct-vc doc," *L1100, Jan*, 2013.

[18] Gisle Bjontegaard, "Calcuation of average psnr differences between rd-curves," *Doc. VCEG-M33 ITU-T Q6/16, Austin, TX, USA, 2-4 April 2001*, 2001.