

# Efficient HEVC to H.264/AVC Transcoding with Fast Intra Mode Decision

Jun Zhang<sup>1,2</sup>, Feng Dai<sup>1</sup>, Yongdong Zhang<sup>1</sup>, and Chenggang Yan<sup>1,2</sup>

<sup>1</sup> Advanced Computing Research Laboratory, Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology,  
Chinese Academy of Sciences, Beijing, China

<sup>2</sup> University of Chinese Academy of Sciences, Beijing, China  
{zhangjun01, fdai, zhyd, yanchenggang}@ict.ac.cn

**Abstract.** High Efficiency Video Coding (HEVC) standard will soon reach its final draft. To provide the widely deployed H.264/AVC devices with HEVC video contents, transcoding pre-encoded HEVC video into H.264/AVC format is highly necessary. Computational complexity of H.264 hinders real-time transcoding. In this paper, we propose an efficient HEVC to H.264 intra frame transcoder to accelerate the time-consuming H.264 intra mode decision while ensure rate distortion (RD) performance. The proposed transcoder incorporates a support vector machine (SVM) based macroblock (MB) partition mode decision and a fast prediction mode decision. Compared with the reference transcoder which employs exhaustive search mode decision, our proposed transcoder can save 68.83% of transcoding time with negligible 2.32% bit-rate increase on average.

**Keywords:** Video Transcoding, HEVC, H.264/AVC, Fast Mode Decision.

## 1 Introduction

In order to satisfy the increasing demand for better visual quality, VCEG and ISO/IEC have formed a group named Joint Collaborative Team on Video Coding (JCT-VC) to develop a new video coding standard HEVC [1] with the target of outperforming all existing standards. HEVC will soon reach its Final Draft as an International Standard, so more and more videos are to be encoded in this outstanding next generation standard. However, as H.264/AVC [2] is now being widely adopted, there will be inevitably a long term co-existence of them and the coming HEVC videos need to be converted into H.264 format to play on legacy H.264 compatible devices. For example, a high definition HEVC digital TV program cannot be displayed on a mobile phone that supports only H.264 unless transcoded. Thus, transcoding pre-encoded HEVC video to H.264 format is definitely needed.

Due to complex coding tools, H.264/AVC encoding is very time-consuming and many researches on accelerating it can be seen in [3-8]. By the same token, the encoder side of an HEVC to H.264 transcoder also needs to be accelerated for real-time applications.

In H.264 encoding, intra mode decision plays an important role and it's computationally complex because of so many coding modes to choose from. In order to speed up this process, many efficient algorithms have been proposed in the past several years which can be classified into two categories. The first one generally employs a simplified computation of RD cost. In [5], the author estimates rate using  $\rho$ -domain model based on the number of zero quantized transform coefficients. In [6] and [7] SATD (sum of absolute transformed differences) based methods are used in which the standard deviation of SATD coefficients are used to estimate rate and show better efficiency than [5]. The second category picks out unnecessary modes and eliminates them. In [8], the variance of a MB is used to decide the partition mode ( $16 \times 16$ ,  $8 \times 8$ ,  $4 \times 4$ ) and filter based approach is adopted to reduce the candidate prediction modes. A MPEG-2 to H.264/AVC transcoder proposed in [9] fully utilizes the DCT coefficients contained in the incoming MPEG-2 stream to do fast mode decision. Learning based approaches [10-12] can get further acceleration. To the best of our knowledge, no works about HEVC/H.264 transcoding have been published yet. Methods listed above can't be directly employed to HEVC/H.264 transcoder because differences between these two standards are relatively significant.

This paper proposes an efficient HEVC to H.264/AVC intra frame transcoder with fast MB partition mode decision and fast prediction mode decision. Experimental results show that our proposed transcoder can get good performances with negligible quality loss but considerable time saving, as compared with the method of exhaustive search.

The remainder of this paper is organized as follows. In section 2 we briefly review intra prediction and mode decision. Our proposed transcoder is fully expounded in section 3. Then in section 4 we show our experimental results. Finally this paper is briefly concluded.

## 2 Intra Prediction

### 2.1 Intra Prediction in HEVC

The concept of MB in previous video coding standards is not directly inherited by HEVC. Instead, a more flexible block splitting manner is adopted. A picture is firstly divided uniformly into unlapped square units named Largest Coding Unit (LCU) which is the top-most level unit in the splitting hierarchy. LCUs are split recursively into four equally sized units in a quad-tree manner and a leaf node of the resulting recursive quad-tree is called a Coding Unit (CU) which also has square size. An intra CU can be further partitioned into four Prediction Units (PUs) which are the basic units carrying prediction information in intra slices. Additionally, there is a concept of Transform Unit (TU) which is the basic unit for intra prediction and transform and it's limited to a CU. It should be noticed that the PU and TU shape may not be square in inter slices but they are always square in intra slices.

In order to represent complex textures or image contents with different directions more efficiently than H.264/AVC, HEVC adopts a more flexible direction representation method which can provide up to 33 directional intra modes and improved accuracy as shown in Figure 1. In addition, two non-directional modes named DC and planar are available. The number 35 mode which means predicting from luma component belongs only to chroma component.

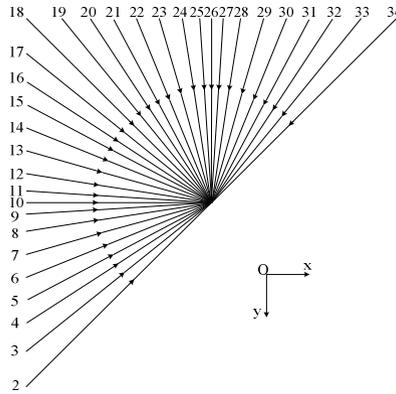


Fig. 1. The 33 directional intra prediction modes in HEVC for luma component

### 2.2 Intra Prediction in H.264/AVC

In H.264/AVC, a MB can be partitioned into blocks of size 16×16, 8×8 or 4×4 for intra prediction of its luma component, while both chroma components are predicted as a single block. For a 4×4 luma block, up to nine prediction modes can be used, including the DC mode and 8 directional modes. The prediction modes for 8×8 luma blocks are exactly the same as those for 4×4 blocks expect that the block to be predicted is 8×8. For a 16×16 luma block, there are four modes namely vertical, horizontal, DC and plane respectively can be use. The prediction modes for chroma component are very similar to those of 16×16 luma block except for different numbering. Because intra prediction of chroma component is much computationally easier than luma, we don't consider it in this paper.

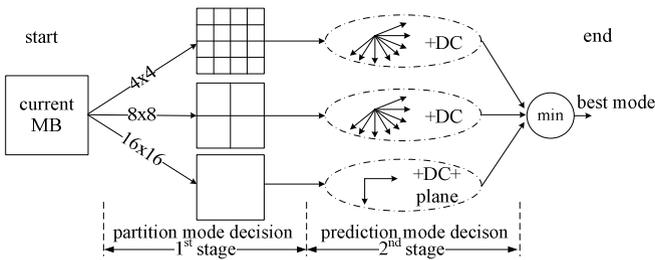


Fig. 2. H.264 intra mode decision for luma component

H.264 intra mode decision tries to find the best partition mode and prediction mode combination which minimize the RD cost for current MB. This process can be modeled as a two-stage decision process as shown in Figure 2 from which we can see that pursuing the best coding mode means searching for a two-stage path from the start to end with the minimum RD cost. The best partition mode is the choice in the first stage and the best prediction mode is the choice in the second stage.

### 3 Proposed Transcoder

#### 3.1 Architecture

The conceptually straightforward cascaded architecture is adopted in this paper as shown in Figure 3. We firstly decode the inputting pre-encoded HEVC stream using a HEVC decoder, generating reconstructed pictures and decoding information contained in the HEVC stream. The following H.264/AVC encoder takes the reconstructed pictures and encodes it, outputting H.264 stream.

During the encoding, the HEVC decoding information is utilized to accelerate intra mode decision. It has been mentioned that intra mode decision can be modeled as a two-stage decision process. As shown in Figure 3, partition mode and prediction mode are determined using fast algorithms in the first and second stage respectively.

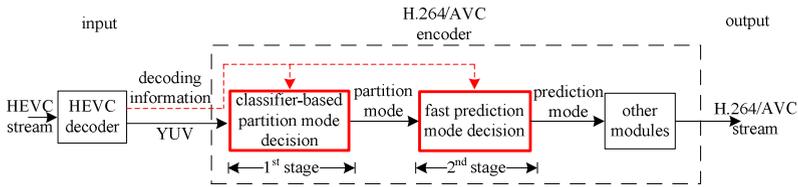


Fig. 3. Outline of our proposed transcoder

#### 3.2 Classifier-Based Partition Mode Decision

In H.264 full search mode decision, all three partition modes 16×16, 8×8 and 4×4 are computed to find the best, so it's very time-consuming. To ease the computational overhead, we model the partition mode decision as a classification problem. Specifically, we extract useful decoding information of current MB to compose a feature vector and send the vector to a pre-trained classifier (section 4.1). The output of the classifier is a class label corresponding to a particular partition mode which is probably the best one for current MB. This process is illustrated in Figure 4.



Fig. 4. The pipeline of our proposed classifier based partition mode decision

To make a compromise between coding quality and computational load, instead of the direct 3-class classifier, we adopt a 2-class classifier. As [8] does, 8×8 mode is always computed; the classifier outputs class label 0 or 1 indicating whether 16×16 or 4×4 mode should be additionally computed and then compared with 8×8 mode to find the best. Thus, totally two modes are computed rather than full search.

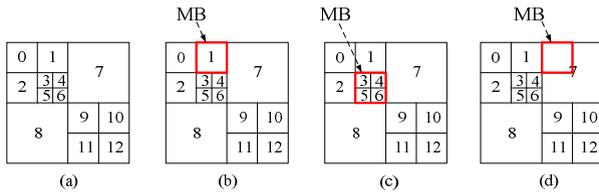
There is a lot of information contained in the inputting HEVC stream, so which information to extract to compose the feature vector may significantly influence the

classification accuracy and eventually the coding quality. An ideal feature vector should be able to reflect the complexity extent of a MB. After a lot of experiments, we decide to use the information listed below as features to compose the vector.

**Prediction Bits.** During the HEVC decoding process, we count for each MB area (a 16×16 area) the bits that are used for coding HEVC syntax elements related to intra prediction including CU splitting flag, PU partition flag and intra prediction mode signaling. We call this feature *PB* in this paper. *PB* is the first dimension of the feature vector.

As the concept of MB does not exist in HEVC but a flexible block splitting, during the counting, we convert CUs to MB areas as the example shown in Figure 5 in which the highlighted block represents current MB area. Figure 5 (a) shows the CU splitting. If current MB lies precisely on a CU of size 16×16 like Figure 5 (b), the *PB* of this CU is directly assigned to this MB; if current MB area covers more than one CUs as (c) shows, the *PB* of this MB area is set to the sum of *PBs* of the covered CUs; otherwise, if the size of a CU is larger than 16×16 as Figure5 (d) shows, then *PB* of each contained MB is set to the complete *PB* of this CU divided by the number of MBs in the CU.

*PB* is able to reflect the coding complexity of current MB area to some extent because a complicated MB area usually needs to be more accurately predicted in order to minimize the residual signal energy and get better RD performance, which leads to large *PB*.



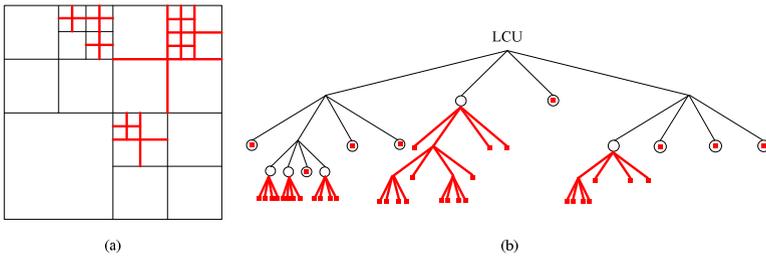
**Fig. 5.** The relationship between HEVC CUs and H.264/AVC MB areas

**Residual Bits.** It's not convincing enough to measure the complexity of a MB area only by *PB* because it is the residual signal bits that occupies most part in the generated bits stream. Like *PB*, for each MB area, we count the bits that are used for coding HEVC syntax elements related to residual signal, including TU splitting flag, coded block flag (Cbf) and transform coefficients. This feature is called *RB* in this paper. The conversion from CUs to MBs is performed exactly the same way as the *PB* computing does as shown in Figure 5.

The average quantization parameter (AQP) in each MB area is also computed from the HEVC stream using which the residual complexity (*RC*) can be formulated as (1). For simplicity but without loss of generality, in our implementation the QP for every frame and block of the inputting HEVC stream is constant so AQP is same for every MB area. Thus, only *RB* is needed to reflect the complexity of a MB area and we put it in the second dimension of the feature vector.

$$RC = AQP \cdot RB . \tag{1}$$

**Splitting Depth.** As mentioned, HEVC adopts a highly flexible block splitting method which recursively split LCU into four equally sized units. Similarly, a CU can further be split recursively. Both these two processes are performed in a quad-tree manner with the first tree called coding tree (*CTree*) whose leaf nodes are CUs and the second one transform tree (*TTree*) whose leaf nodes are TUs. A *TTree* is rooted by a CU. TUs are the basic units for intra prediction and transform. An example of unit partition and its quad-tree representation are shown in Figure 6 (a) and (b) respectively. Thin lines (black) represent CU splitting and coding tree while bold (red) lines represent TU splitting and transform tree. In Figure 6 (b), hollow circles mean CUs and solid squares (red) represent TUs. Leaf nodes attached with both shapes mean that these CUs are predicted and transformed as a whole without further splitting, i.e. they contain only one TU.



**Fig. 6.** Coding tree and transform tree

These two splitting processes in HEVC offer a content-adaptive coding method during which flat areas are most likely to be coded in shallow depths of *CTree* and *TTree* while complex areas are probably coded in deep depths. Thus, the depth information contained in the HEVC stream may play an important role in estimating spatial complexity. For a MB area, we calculate the average depth of *CTree* and average depth of *TTree* within it by simple mean computations utilizing the decoding information. These two features are named  $D_{CTree}$  and  $D_{TTree}$  respectively and act as the next two dimensions of the feature vector.

$$FV = (PB, RB, D_{CTree}, D_{TTree}) . \tag{2}$$

By now, the feature vector can be formulated as (2). The classifier is trained as follows. Pre-encoded HEVC files are decoded to generate reconstructed YUV files and feature vectors of each MB area. We then encode the reconstructed YUVs using a full search H.264 encoder. During the encoding, if the optimal partition mode for one MB decided by the encoder is 16×16 or 4×4, it's taken as *ground truth* and recorded along with the corresponding feature vector of this MB, which are jointly called a training instance. Extensive instances are inputted into a classifier to train it.

During transcoding, the feature vector of each MB area is generated and sent to the pre-trained classifier to directly decide the best partition mode.

### 3.3 Fast Prediction Mode Decision

A number of prediction modes can be used in H.264 intra coding, especially for 4×4 and 8×8 blocks. It's time-consuming to do full search. In this section we propose a fast intra prediction mode decision for HEVC to H.264 transcoding. We select a subset of the H.264 prediction modes to do mode decision, denoted as  $M$ . Only the modes in the subset  $M$  are computed and others are omitted. The modes in  $M$  are called candidate modes. In our proposed algorithm, DC and the most probable mode (MPM, 8×8 and 4×4 blocks only) are always computed due to their high probability of being the best [8,9], i.e.  $M$  is initialized to {DC} or {DC, MPM}. By referring corresponding HEVC modes, the other modes are selectively added into  $M$  as follows.

In HEVC, there are 35 intra prediction modes for luma component including two non-directional modes planar (0) and DC (1) and 33 directional ones (2 to 34) as shown in Figure 1. The minimal coding block in HEVC is 4×4, so the prediction modes of a certain area can be expressed, stored and accessed in 4×4 block unit as HM [14] does. We assume that the corresponding HEVC prediction modes for a H.264 block area are  $m_0, m_1, \dots, m_{n-1}$  where  $m_k$  ( $k = 0, 1, \dots, n-1$ ,  $n=1$  for 4×4 block,  $n=4$  for 8×8 block and  $n=16$  for 16×16 block) means the  $k^{th}$  HEVC prediction mode in Z-scan order within this block as shown in Figure 7.

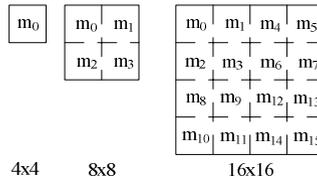


Fig. 7. HEVC prediction modes in a H.264 block. A square represents a 4×4 block area

**1) All Non-directional.** We observe by experiments that if all HEVC prediction modes within one block are planar or DC, i.e.  $m_k \leq 1$  ( $k = 0, 1, \dots, n-1$ ), then this block will be coded in H.264 vertical or horizontal mode with a high probability. So if current block meets this condition,  $M = M \cup \{0, 1\}$ .

**2) Non-directional and Directional.** However, if both non-directional and directional HEVC modes exist in a block, then  $M = \{0, 1, 2, 3\}$  for 16×16 block and  $M = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$  for 8×8 or 4×4 block. This means all modes will be computed because the dominant texture direction in this block is not obvious.

**3) All Directional** If HEVC prediction modes within current block are all directional ones which means  $m_k \geq 2$  ( $k = 0, 1, \dots, n-1$ ), the following procedures apply.

*Some Definitions.* As shown in Figure 1, we denote the angles of the 33 directional modes relative to x-axis as  $\alpha_i$  ( $i = 2, 3, \dots, 34, -\pi < \alpha_i \leq \pi$ ). Similarly, we denote the angles of eight H.264/AVC directional intra modes as  $\beta_i$  ( $0 \leq i \leq 8, i \neq 2, -\pi < \beta_i \leq \pi$ ).

We further define the angle distance  $D_{\gamma,\eta}$  between two angles  $\gamma$  and  $\eta$  ( $-\pi < \gamma, \eta \leq \pi$ ) as (3) where *abs* means absolute value and *min* returns the smaller of its two operands.

$$D_{\gamma,\eta} = \min\{\text{abs}(\gamma - \eta), \pi - \text{abs}(\gamma - \eta)\} . \tag{3}$$

Based on definitions above, we define the mean mode  $m^*$  of a series of HEVC directional modes  $m_0, m_1, \dots, m_{n-1}$  (between 2 and 34) as (4).

$$m^* = \arg \min_i \left( \sum_{k=0}^{n-1} D_{\alpha_i, \alpha_{m_k}} \right) . \tag{4}$$

*Candidates Selection.* We firstly compute the mean mode  $m^*$  of all the HEVC modes  $m_0, m_1, \dots, m_{n-1}$  contained in this block as formula (4). Then,

- a) For 8x8 or 4x4 block, we find that the probability of mode 0 or 1 being the best is relatively high, so they are always selected,  $M=MU\{0,1\}$ . We then compute the angle distances between  $m^*$  and the rest six H.264 directional modes, and denoted them as  $D_{\alpha_{m^*}, \beta_j}$  with  $3 \leq j \leq 8$ . We select the three smallest angle distances and the corresponding mode  $j$  is added into  $M$ ,  $M=MU\{j\}$ .
- b) For 16x16 block, if  $6 \leq m^* \leq 14$ ,  $M=MU\{1\}$ ; if  $22 \leq m^* \leq 30$ ,  $M=MU\{0\}$ ; otherwise,  $M=MU\{0,1,3\}$ .

Summary of the proposed algorithm are given in Table 1.

**Table 1.** Candidate modes selection in the proposed fast prediction mode decision

<i>block size</i> <i>HEVC mode</i>	<i>4x4</i>	<i>8x8</i>	<i>16x16</i>
<i>all m<sub>k</sub> ≤ 1</i>	$M=\{0, 1, 2, \text{MPM}\}$		$M=\{0, 1, 2\}$
<i>not all m<sub>k</sub> ≤ 1</i>	—	$M=\{0,1,\dots,8\}$	$M=\{0,1,2,3\}$
<i>all m<sub>k</sub> ≥ 2</i>	calculate $m^*$ as formula (4)		
	$M=\{\text{three modes between 3 and 8}\} \cup \{0,1,2, \text{MPM}\}$ .	if $6 \leq m^* \leq 14$ , $M=\{1,2\}$ ; if $22 \leq m^* \leq 30$ , $M=\{0,2\}$ ; else $M=\{0,1,2,3\}$ .	

## 4 Experimental Results

Using H.264/AVC reference software JM18.3 [13], HEVC reference software HM6.1 [14] and a well-known library libsvm3.12 [15] for support vector machine (SVM), all of which are the latest version at the start of our work, we evaluate the performance of our proposed heterogeneous HEVC to H.264/AVC transcoder. The experiment environments and configurations are as follows.

- a) Implemented on a PC with Intel 3.10GHz i5-2400 processor and 4GB RAM.
- b) Nine standard test sequences are used, including 1080p *Kimono*, *ParkSene*, *Cactus*, *BasketballDrive*, *BQTerrace*, *Tennis* and 720p *Vidio1*, *Vidio3* *Vidio4*. The first 200 frames of each sequence are used.

- c) Original raw YUV videos are firstly encoded by HM encoder to get pre-encoded HEVC stream files using the default high efficiency-all intra configuration file. Specifically, all frames are encoded into intra pictures, LCU size is set to 64x64, QP to 32.
- d) For the H.264/AVC encoder, High profile is adopted. All incoming frames are encoded as intra; QP is set to 24, 28, 32, and 36 respectively to get the experimental data; RDO is set on; PCM is disabled.

In the next sub-sections, we evaluate the performance of our proposed algorithms separately and jointly in the transcoding context. The experimental results of the corresponding transcoders are compared in terms of BD-rate [16] and time reduction to those of the reference cascaded transcoder in which the exhaustive search mode decision is employed. Average PSNR of all three components defined as  $(6 * YPSNR + UPSNR + VPSNR)/8$  is used to calculate average BD-rate.

**Table 2.** Performance comparison between the reference transcoder and our transcoder

Class	Sequence	1 <sup>st</sup> stage		2 <sup>nd</sup> stage		Two-Stage	
		BD-rate (%)	Time (%)	BD-rate (%)	Time (%)	BD-rate (%)	Time (%)
1080p	<i>Kimono</i>	+0.00	-51.84	+3.53	-52.55	+3.54	-75.16
	<i>ParkScene</i>	+0.02	-32.66	+2.75	-48.64	+2.75	-64.21
	<i>Cactus</i>	+0.05	-37.74	+2.24	-45.44	+2.29	-63.24
	<i>BasketballDrive</i>	+0.38	-47.96	+1.33	-41.75	+1.75	-67.16
	<i>BQTerrace</i>	+0.13	-32.04	+1.60	-40.21	+1.75	-57.77
	<i>Tennis</i>	+0.05	-54.47	+2.73	-42.93	+2.77	-72.62
720p	<i>Vid101</i>	+0.10	-51.38	+2.20	-45.57	+2.28	-71.00
	<i>Vid103</i>	+0.20	-46.37	+1.96	-45.27	+2.12	-68.27
	<i>Vid104</i>	+0.13	-51.43	+1.98	-47.75	+2.11	-72.38
	<b>average</b>	<b>+0.13</b>	<b>-45.10</b>	<b>+2.21</b>	<b>-45.57</b>	<b>+2.32</b>	<b>-68.63</b>

#### 4.1 Evaluation of the SVM-Based Partition Mode Decision

*Training the Classifier* We choose SVM due to its excellent classification accuracy. Four 1080p video materials including smooth and textured ones in spatial variation namely *BQTerrace*, *BasketballDrive*, *Cactus* and *Kimono* are firstly encoded by HM encoder to generate pre-encoded HEVC files. These files are then decoded by HM decoder to produce reconstructed YUV files and corresponding feature vectors of each MB area as specified in section 3.2. The reconstructed YUVs are then encoded by JM18.3. During the encoding process, if the best partition mode for one MB decided by JM is 16x16 or 4x4, it's taken as the ground truth. The ground truth and corresponding feature vector of a MB are jointly called a training instance. We randomly select 10,000 instances which are inputted into a SVM to train it. Cross-fold validation is performed. The resultant 2-class classifier achieves a classification accuracy of about 98% on the whole training set.

The training process is performed offline and the final output is a SVM mode file. The transcoder loads the model file into memory on its start-up and it classifies the incoming feature vector of a MB according to the model file.

*Transcoding* We replace the full search approach in JM18.3 encoder with SVM-based partition mode decision while the prediction mode decision remains unchanged, i.e. try all available prediction modes. The performance comparison to the reference transcoder is shown in  $1^{st}$  stage column of Table 2. We can see time reduction is relatively low for textured videos such as *BQTerrace* and *Cactus*, and relatively high for smooth ones such as *Kimono*. The reason is that  $4\times 4$  mode is much more computationally intensive than  $16\times 16$  but may be skipped without computing for less complex videos [8]. On the whole, the performance of the proposed algorithm is satisfactory with almost half time saving but tiny bit-rate increase.

### 4.2 Evaluation of the Fast Prediction Mode Decision

In order to verify the proposed prediction mode decision algorithm separately, we implement it based on JM18.3 encoder with all three partition modes ( $16\times 16$ ,  $8\times 8$  and  $4\times 4$ ) computed. The results are shown in  $2^{nd}$  stage column of Table 2 from which we can see that the average time reduction is nearly the same as  $1^{st}$  stage column with a little BD-rate increase.

### 4.3 Evaluation of the Two-Stage Fast Mode Decision

We incorporate both these two fast mode decision algorithms into JM18.3 to get the final proposed transcoder. The RD performance and transcoding speed relative to the reference transcoder are shown in *Two-Stage* column of Table 2.

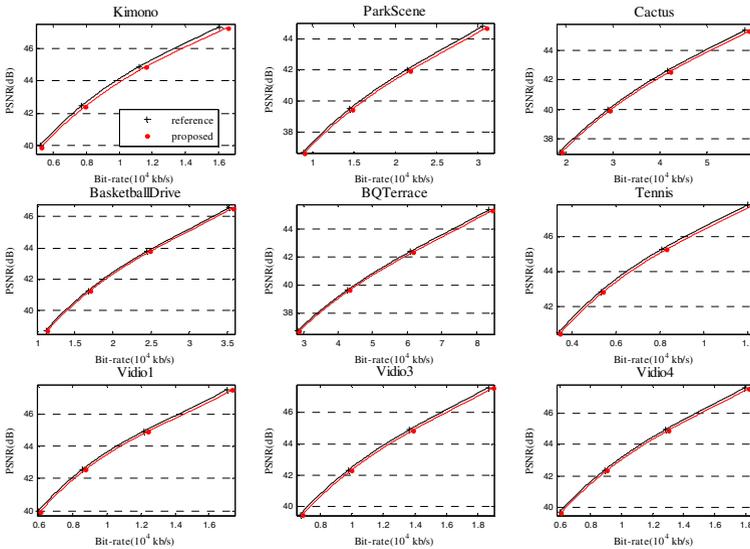


Fig. 8. RD curves of the reference and our proposed transcoder

As shown in Table 2, 68.63% transcoding time can be saved with 2.32% rate increase on average. Obviously the RD performance loss is negligible in view of the considerable time reduction. The RD plots of transcoding are depicted in Figure 8. The negligible differences between these two curves and the transcoding acceleration verify the efficiency of our proposed algorithms.

## 5 Conclusion

In this paper, an efficient HEVC to H.264 intra frame transcoder is proposed. The transcoder incorporates a SVM-based partition mode decision algorithm and a fast prediction mode decision algorithm. Experimental results obtained by implementing these two algorithms separately and jointly in transcoding context demonstrate their effectiveness. Using our proposed transcoder, considerable time saving can be obtained with negligible bit-rate increase.

**Acknowledgments.** This work is supported by National Nature Science Foundation of China (61102101, 61272323), National Key Technology Research and Development Program of China (2012BAH06B01), Co-building Program of Beijing Municipal Education Commission.

## References

1. Bross, B., Han, W.-J., Ohm, J.-R., Sullivan, G.J., Wiegand, T.: High Efficiency Video Coding (HEVC) Text Specification Draft 6, JCTVC-H1003, San José, CA, USA, pp. 1–10 (February 2012)
2. Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, document JVT-G050.doc, ITU-I Rec. H.264 and ISO/IEC 14496-10 AVC (2003)
3. Zhang, Y., Yan, C., Dai, F., Ma, Y.: Efficient parallel framework for H.264/AVC deblocking filter on many-core platform. *IEEE TMM* 14(1), 510–524 (2012)
4. Yan, C., Dai, F., Zhang, Y., Ma, Y., Chen, L., Fan, L., Zheng, Y.: Parallel deblocking filter for H.264/AVC implemented on Tile64 platform. In: ICME 2011, Barcelona, Spain, pp. 1–6 (2011)
5. Kim, H., Altunbasak, Y.: Low-complexity macroblock mode selection for H.264/AVC encoder. In: ICIP 2004, October 24–27, pp. 765–768 (2004)
6. Lee, Y.-M., Sun, Y.-T., et al.: SATD-Based Intra Mode Decision for H.264/AVC Video Coding. *IEEE TCSVT* 20(3), 463–469 (2010)
7. Lin, Y., Lee, Y.-M., et al.: Efficient Algorithm for H.264/AVC Intra frame Video Coding. *IEEE TCSVT* 20(10), 1367–1372 (2010)
8. Yi-Hsin, H., Ou, T.-Sh., et al.: Fast Decision of Block Size, Prediction Mode, and Intra Block for H.264 Intra Prediction. *IEEE TCSVT* 20(8), 1122–1132 (2010)
9. Xingang, L., Yoo, K.-Y., et al.: Low Complexity Intra Prediction Algorithm for MPEG-2 to H.264/AVC Transcoder. *IEEE Trans. Cons. Electro.* 56(2), 987–994 (2010)
10. Chiang, C.-K., Pan, W.-H., et al.: Fast H.264 Encoding Based on Statistical Learning. *IEEE TCSVT* 21(9), 1304–1315 (2011)

11. Lu, Z.-Y., Jia, K.-B., Siu, W.-C.: Low-complexity Intra Prediction Algorithm for Video Down-Sizing Transcoder. In: VCIP 2011, vol. 6-9, pp. 1–4 (2011)
12. Fern, G., Kalva, H., et al.: An MPEG-2 to H.264 Video Transcoder in the Baseline Profile. *IEEE TCSVT* 20(5), 763–768 (2010)
13. JM Reference Software Version 18.3, <http://iphome.hhi.de/suehring/tml/>
14. HM Reference Software Version 6.1, [https://hevc.hhi.fraunhofer.de/svn-/svn\\_HEVCSoftware/](https://hevc.hhi.fraunhofer.de/svn-/svn_HEVCSoftware/)
15. Libsvm Version 3.12, <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>
16. Bjontegaard, G.: Calculation of average PSNR differences between RD-curves. In: Document VCEG-M33, 13th Meeting VCEG, Austin, Texas, USA, pp. 2–4 (April 2001)