EFFICIENT PU MODE DECISION AND MOTION ESTIMATION FOR H.264/AVC TO HEVC TRANSCODER

Zong-Yi Chen[1], Jiunn-Tsair Fang[2], Tsai-Ling Liao[1], and Pao-Chi Chang[1]

[1]Department of Communication Engineering, National Central University, Jhongli, Taiwan
[2]Department of Electronic Engineering, Ming Chuan University, Taoyuan, Taiwan

ABSTRACT

H.264/AVC has been widely applied to various applications. However, a new video compression standard, High Efficient Video Coding (HEVC), had been finalized in 2013. In this work, a fast transcoder from H.264/AVC to HEVC is proposed. The proposed algorithm includes the fast prediction unit (PU) decision and the fast motion estimation. With the strong relation between H.264/AVC and HEVC, the modes, residuals, and variance of motion vectors (MVs) extracted from H.264/AVC can be reused to predict the current encoding PU of HEVC. Furthermore, the MVs from H.264/AVC are used to decide the search range of PU during motion estimation. Simulation results show that the proposed algorithm can save up to 53% of the encoding time and maintains the rate-distortion (R-D) performance for HEVC.

KEYWORDS

H.264/AVC, HEVC, PU, Fast Algorithm, Transcoder

1. INTRODUCTION

H.264/AVC is currently one of the popular video compression standards [1]. However, with increasingly demands on the high quality of video services, previous video compression standards are no longer satisfied. The new video compression standard, High Efficient Video Coding (HEVC) [2], had been finalized in April 2013. During the transfer to HEVC, transcoders become practical tools continuing to serve users. An efficient transcoder from H.264/AVC to HEVC focusing on the prediction unit (PU) is proposed.

Video transcoding refers to converting the video content from one format into another. For the transcoder of H.264/AVC to HEVC, the video bit-stream of H.264/AVC is fully decoded, and some parameters are extracted. These parameters are mainly motion vectors (MVs), residuals, and modes from each coding block. With strong relation between these two standards, these extracted parameters can be reused for HEVC, and the HEVC encoder does not need to do full prediction when encoding. Figure 1 shows the typical fast transcoder architecture. The proposed method is to build up the relation from these extracted parameters of H.264/AVC to predict the PU of HEVC. Then, the original procedure to predict PU by HEVC encoder can be skipped, and therefore the encoding time of HEVC is reduced.

Most related works for the transcoding focus on the fast mode decision and fast motion estimation because the time saving is significant. For the fast mode decision, the relationship between the input block and the current block was explored in [3]. Jing et al. determined the optimal coding mode for the re-encoding process [4]. Zhang et al. determined the best PU by the proposed power spectrum model [5]. As for the fast motion estimation, MV refinement was proposed to have a

DOI : 10.5121/sipij.2014.5208

better MV predictor [3]. Garrido-Cantos et al. proposed to reduce the motion search range for H.264/SVC with the information of H.264/AVC [6]. In [7], the authors proposed a fast mode decision algorithm for HEVC to H.264/AVC intra frame transcoding.

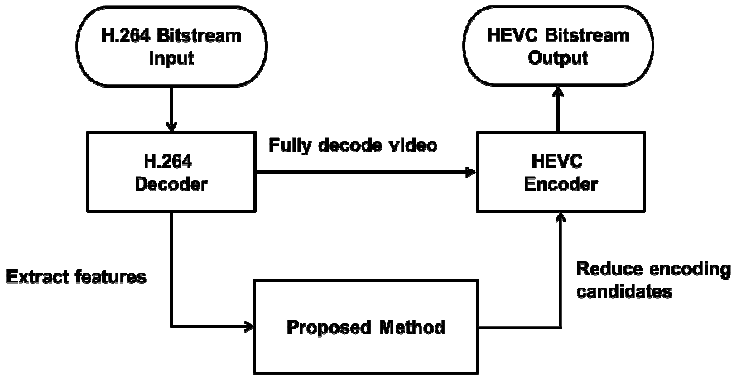


Figure 1. Typical fast transcoder architecture

Although promising results have been accomplished, their transcoders are applied for the same coding size between two standards. The coding unit (CU) sizes in depths 0 and 1 of HEVC are larger than the macroblock (MB) size of H.264/AVC. Thus, to design the transcoding from H.264/AVC to HEVC, various procedures are required for the different coding size prediction. A fast mode decision algorithm was proposed to explore the relationship between the input block and the current block for each depth of CU recently [8]. However, the PU was not considered in this method. Thus, a fast transcoding from H.264/AVC to HEVC focusing on PU is proposed.

The proposed algorithm consists of the fast mode decision and the fast motion estimation. Because of various coding sizes between H.264/AVC and HEVC, depths 0 and 1 and depths 2 and 3 of CU are applied various methods. The MVs, residuals, and modes information from H.264/AVC are reused for the PU encoder. As for the fast motion estimation, the MVs from H.264/AVC are used to decide the search range of PU.

The rest of this paper is organized as follows. In Section 2, the proposed transcoding algorithm is described. Section 3 shows the simulation results, and the conclusion is in Section 4.

2. PROPOSED ALGORITHM

The proposed algorithm consists of the fast mode decision and the fast motion estimation. Before discussing the proposed algorithm, the preprocessing of the MV normalization is described.

2.1. Motion Vector Normalization

The MV extracted from various temporal references or various spatial sizes of H.264/AVC must be normalized to be the parameter predictor for HEVC. H.264/AVC applies multiple frames for the prediction, and the extracted MV may come from various temporal references. The MV normalization, denoted as mv_{norm}, is derived using (1):

$$mv_{norm} = \frac{1}{\alpha} mv_{n-\alpha}, \tag{1}$$

where n is the current frame and n-α is the reference frame. Equation (1) shows that the MV normalization is to divide the frame number between the reference frame and the current frame. In other words, the extracted MV from the reference frame has a larger weighting if the reference frame is closer to the current frame.

H.264/AVC applies variable block sizes for the prediction, and the extracted MV may come from various sizes of the block. The MV normalization is to compare the area of the reference block with the area of the 4×4 block, which is derived using (2):

$$mv_{norm} = \frac{area\ of\ the\ reference\ block}{area\ of\ the\ 4\times 4\ block} mv\ . \tag{2}$$

Equation (2) shows that the extracted MV from the reference block has a larger weighting if the reference block has a larger area.

2.2. Fast Mode Decision Algorithm

In the proposed transcoder, the MVs, residuals, and modes information from H.264/AVC are reused to predict the PU modes of HEVC. However, the CU sizes in depths 0 and 1 of HEVC are larger than the MB size in H.264/AVC. Figure 2 illustrate the block mapping from H.264/AVC to HEVC CU depths 0 and 2. In CU depths 0 and 1, we can not map the mode directly from H.264/AVC. Thus, the proposed fast mode algorithms for the transcoder to predict PU in depths 0 and 1 and PU in depths 2 and 3 are different.

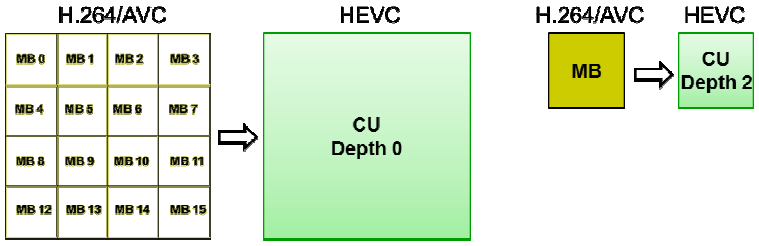


Figure 2. Block mapping from H.264/AVC to HEVC

2.2.1. Fast Mode Decision Algorithm for Depths 0 and 1

The proposed method for the transcoder to predict PU in depths 0 and 1 is described as follows. For the 2N×2N mode of PU, if all MVs in H.264/AVC are the same, i.e. the variance of these MVs is zero, the corresponding blocks in HEVC can be merged into a 2N×2N mode of HEVC. Table 1 shows that the accuracy rate under this condition is over 90%.

Table 1. Hit rates of mode 2N×2N when the variance of H.264/AVC MVs is zero.

Sequence	P(mode=64×64 | *VAR* = 0)	P(mode=32×32 | *VAR* = 0)
ClassB_Kimono1	0.83	0.92
ClassC_BQMall	0.93	0.88
ClassD_BasketballPass	0.94	0.93
ClassE_vidyo1	0.97	0.91
Average	**0.92**	**0.91**

For 2N×N and N×2N modes of PU, the MV variance is defined as:

$$VAR = \frac{1}{N}\sum_{n=1}^{N}(MV_n - \mu_{MV})^2\ , \tag{3}$$

where N is the total number of MVs in the reference block, and μ_{MV} is the average value of these MVs. According to (3), the calculations of the MV variances of the block 2N×N and block N×2N are as follows:

$$\overline{VAR}_{2N\times N} = \frac{1}{2}(VAR_{2N\times N_{(U)}} + VAR_{2N\times N_{(D)}})$$

$$\overline{VAR}_{N\times 2N} = \frac{1}{2}(VAR_{N\times 2N_{(L)}} + VAR_{N\times 2N_{(R)}}),$$

where $\overline{VAR}_{2N\times N}$ is the average MV variance of upper and bottom blocks of the mode 2N×N, and $\overline{VAR}_{N\times 2N}$ is the average MV variance of left and right blocks of the mode N×2N. The block segment is drawn in Figure 3. If $\overline{VAR}_{X\,2N\times N} \geq 2\times\overline{VAR}_{X\,N\times 2N} \cap \overline{VAR}_{Y\,2N\times N} \geq 2\times\overline{VAR}_{Y\,N\times 2N}$, the mode 2N×N in PU is skipped. On the other hand, if $\overline{VAR}_{X\,N\times 2N} \geq 2\times\overline{VAR}_{X\,2N\times N} \cap \overline{VAR}_{Y\,N\times 2N} \geq 2\times\overline{VAR}_{Y\,2N\times N}$, the mode N×2N in PU is skipped. This is because a reference block with a larger MV variance is with less chance to be the candidate for the prediction. Table 2 and Table 3 show the error rates of the proposed algorithm. The modes skipped by our algorithm are rarely chosen as the best PU mode.

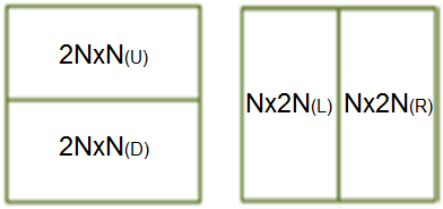


Figure 3. The block segment of the blocks 2N×N and N×2N

Table 2. Error rates of the proposed method for mode 2N×N.

Sequence	P(mode=64×32 | $\overline{VAR}_{2N\times N} \geq 2\cdot\overline{VAR}_{N\times 2N}$)	P(mode=32×16 | $\overline{VAR}_{2N\times N} \geq 2\cdot\overline{VAR}_{N\times 2N}$)
ClassB_Kimono1	0.0345	0.0580
ClassC_BQMall	0.0000	0.0000
ClassD_BasketballPass	0.0000	0.0000
ClassE_vidyo1	0.0833	0.0435
Average	**0.0295**	**0.0254**

Table 3. Error rates of mode the proposed method for N×2N.

Sequence	P(mode=32×64 | $\overline{VAR}_{N\times 2N} \geq 2\cdot\overline{VAR}_{2N\times N}$)	P(mode=16×32 | $\overline{VAR}_{N\times 2N} \geq 2\cdot\overline{VAR}_{2N\times N}$)
ClassB_Kimono1	0.0909	0.0535
ClassC_BQMall	0.0256	0.0000
ClassD_BasketballPass	0.0000	0.0000
ClassE_vidyo1	0.0000	0.0833
Average	**0.0291**	**0.0342**

For the asymmetric motion partition (AMP) mode in HEVC, the residual of each MB in H.264/AVC is applied to determine the candidate of the prediction. The *residual* is denoted as the sum of all the coefficients in the MB, derived using (4):

$$residual = \sum_i \sum_j \left| coeff(x_i, y_j) \right| \qquad (4)$$

where (x_i, y_j) is the coefficient position of the *x*-axis and *y*-axis in an MB, respectively. If the *residual* of block $2N\times N_{(U)}$ is greater than the *residual* of $2N\times N_{(D)}$, the 2N×nD mode is skipped. Otherwise, the 2N×nU mode is skipped. This is because a reference block with a larger *residual* implies that the block is with more content details and has a worse prediction, and this block requires further partition to achieve better prediction. The four proposed conditions for AMP in CU depths 0 and 1 are listed as follows. Table 4 and Table 5 show the error rates of the proposed algorithm for AMP. The low error rates demonstrate the efficiency of the proposed algorithm.

- AMP Cond. 1: When the residual of block $2N\times N_{(U)} > 2N\times N_{(D)}$ in H.264/AVC, we skip 2N×nD mode in HEVC.
- AMP Cond. 2: When the residual of block $2N\times N_{(U)} < 2N\times N_{(D)}$ in H.264/AVC, we skip 2N×nU mode in HEVC.
- AMP Cond. 3: When the residual of block $N\times 2N_{(L)} > N\times 2N_{(R)}$ in H.264/AVC, we skip nR×2N mode in HEVC.
- AMP Cond. 4: When the residual of block $N\times 2N_{(L)} < N\times 2N_{(R)}$ in H.264/AVC, we skip nL×2N mode in HEVC.

Table 4. Error rates of the proposed method for AMP in depth 0.

Sequence	P(mode=2N×nD | AMP Cond. 1)	P(mode=2N×nU | AMP Cond. 2)	P(mode=nR×2N | AMP Cond. 3)	P(mode=nL×2N | AMP Cond. 4)
ClassB_Kimono1	0.0290	0.0305	0.0310	0.0254
ClassC_BQMall	0.0153	0.0155	0.0389	0.0380
ClassD_BasketballPass	0.0000	0.0000	0.0632	0.0000
ClassE_vidyo1	0.0152	0.0176	0.0411	0.0176
Average	**0.0149**	**0.0170**	**0.0436**	**0.0203**

Table 5. Error rates of the proposed method for AMP in depth 1.

Sequence	P(mode=2N×nD | AMP Cond. 1)	P(mode=2N×nU | AMP Cond. 2)	P(mode=nR×2N | AMP Cond. 3)	P(mode=nL×2N | AMP Cond. 4)
ClassB_Kimono1	0.0282	0.0338	0.0296	0.0301
ClassC_BQMall	0.0262	0.0327	0.0309	0.0452
ClassD_BasketballPass	0.0815	0.0691	0.0453	0.0590
ClassE_vidyo1	0.0267	0.0253	0.0344	0.0461
Average	**0.0407**	**0.0402**	**0.0351**	**0.0451**

For the intra mode, it is selected only if the inter prediction has a poor prediction. Thus, if a 2N×2N block contains no any intra mode in H.264/AVC, the intra mode in HEVC is skipped. Table 6 shows the low error rates of the proposed algorithm for intra mode selection.

2.2.2. Fast Mode Decision Algorithm for Depths 2 and 3

Table 6. Error rates of the proposed method for intra mode.

Sequence	P(mode=intra64 | H.264/AVC!=intra)	P(mode=intra32 | H.264/AVC!=intra)
ClassB_Kimono1	0.0007	0.0127
ClassC_BQMall	0.0007	0.0003
ClassD_BasketballPass	0.0000	0.0006
ClassE_vidyo1	0.0000	0.0002
Average	**0.0004**	**0.0035**

As for the depths 2 and 3, the MB size in H.264/AVC is the same as the CU size in HEVC depth 2, and the subMB size is the same as the CU size in depth 3. We can easily predict the candidate PU modes from the modes in H.264/AVC due to their similar block structure. The main concept is to skip the PU modes which are much different from the mode in H.264/AVC. For example, if the mode in H.264/AVC is with horizontal direction, the probability to select the vertical partition modes in HEVC will be low. And the residuals from H.264/AVC also provide information to further reduce the PU modes. An H.264/AVC mode with zero residual implies that this specific mode achieves very good prediction, thus the reliability of this mode is quite high for our algorithm development. When the residual of the H.264/AVC mode is zero, we only search the most similar PU modes in HEVC. As a matter of course, the dominant mode, i.e. 2N×2N, will always be examined. The proposed transcoder for fast algorithms in depths 2 and 3 are described as follows, respectively.

- Depth 2 fast mode decision algorithm:
  - If H.264/AVC is skip => just do 2N×2N
  - If H.264/AVC is 16×16 => do 2N×2N, 2N×N and N×2N
    ➔If residual=0 => just do 2N×2N
  - If H.264/AVC is 16×8 => do 2N×2N, 2N×N, 2N×nU and 2N×nD
    ➔If residual=0 => just do 2N×2N and 2N×N
  - If H.264/AVC is 8×16 => do 2N×2N, N×2N, nL×2N and nR×2N
    ➔If residual=0 => just do 2N×2N and N×2N
  - If H.264/AVC is subMB => skip intra mode
  - If H.264/AVC is intra16 => do 2N×2N and intra
  - If H.264/AVC is intra4 => do 2N×2N, 2N×N, N×2N and intra mode
- Depth 3 fast mode decision algorithm:
  - If H.264/AVC is skip or 16×16 => just do 2N×2N
  - If H.264/AVC is 16×8, 8×16 or 8×8 => skip intra
    ➔If residual=0 => just do 2N×2N
  - If H.264/AVC is 8×4 => skip intra N×N
    ➔If residual=0 => skip N×2N and intra mode
  - If H.264/AVC is 4×8 => skip intra N×N
    ➔If residual=0 => skip 2N×N and intra mode
  - If H.264/AVC is 4×4 => skip intra N×N
    ➔If residual=0 => skip intra mode
  - If H.264/AVC is intra16 => do 2N×2N and intra 2N×2N
    ➔If residual=0 => just do 2N×2N
  - If H.264/AVC is intra4 => do 2N×2N and intra

2.3. Fast Motion Estimation Algorithm

In general, the procedure of the motion estimation spends the most encoding time. Two methods are applied for the proposed transcoder to reduce the search range of the motion estimation. One is to apply the adaptive motion search, and the other is to adjust the search range by the transition probability of the current encoding PU.

An adaptive motion search is proposed to reduce the search range. The reference search range (*refSR*) for each PU is defined using (5):

$$refSR = \max_{i} (\max(|MV_{X_i} - AMVP_X|, |MV_{Y_i} - AMVP_Y|)) . \quad (5)$$

Equation (5) shows that to predict each PU, the search range is replaced by the *refSR* which is the maximum difference between the advance motion vector prediction (AMVP) of HEVC and each MV from H.264/AVC. Comparing with the fixed search range by HEVC, the proposed *refSR* is more flexible and effective. This is because the *refSR* adopts both information of MVs from H.264/AVC and HEVC. If the *refSR* is small, the MV prediction is with high accuracy. So, the search range can be small. On the other hand, if the *refSR* is large, the prediction is with low accuracy. The maximum difference between the predicted MVs as the search range shown in (5) can cover the worst condition. The difference between the H.264/AVC MV and the HEVC AMVP with respect to the *x* and *y* direction is drawn Figure 4.

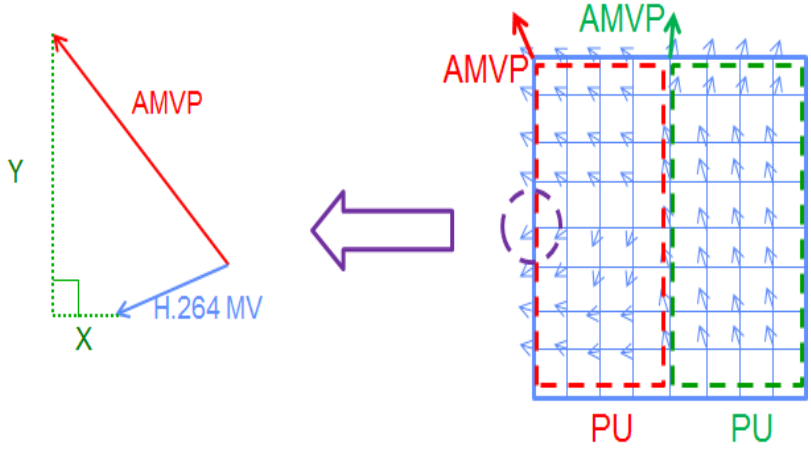


Figure 4. The difference between H.264/AVC MV and HEVC AMVP

In addition to the *refSR*, the mode transition probability from H.264/AVC to HEVC is also considered. If a PU in HEVC is with high probability to occur, the search range can be longer. Otherwise, the search range can be shorter if this PU rarely occurs.

For depths 2 and 3, the transition probability is directly measured from the mode in H.264/AVC to the PU in HEVC because the MB size in H.264/AVC is the same as the CU size in HEVC. Tables 7 and Table 8 show the transition probability of depths 2 and 3, respectively. The left column in Table 7 or Table 8 lists each mode in H.264/AVC and the above row lists the final PU by the HEVC encoder. For each row, it shows the transition probability of a mode in H.264/AVC to a PU in HEVC. The sum of the transition probability of each row is 1.

For depths 0 and 1 of CU, the CU size in HEVC is larger than the MB size in H.264/AVC. If a block has been encoded as a subMB by H.264/AVC, the chance for this block to be merged into a larger size of CU, encoded by HEVC, in depths 0 or 1 is low. In other words, if the current encoding CU contains more subMBs, already encoded by H.264/AVC, the chance for this CU to be encoded by HEVC in depths 0 and 1 is lower. The subMB denotes as a block with an 8×8 size or less. Thus, the transition probability can be obtained by counting the number of subMB within a CU. Counting the subMB is based on each 8×8 block, where an 8×8 block or all the smaller blocks within an 8×8 block are counted by 1.

Table 7. The transition probability for depth 2.

H.264 \ HEVC	16x16	16x8	8x16	16x4	16x12	4x16	12x16	intra16
skip	0.9772	0.0064	0.0082	0.0007	0.0013	0.0023	0.0012	0.0026
16x16	0.8264	0.0395	0.0464	0.0189	0.0158	0.0222	0.0183	0.0108
16x8	0.5681	0.2551	0.0514	0.0393	0.0315	0.0196	0.0174	0.0177
8x16	0.5427	0.0425	0.2778	0.0173	0.0155	0.0447	0.0352	0.0243
8x8	0.3983	0.1674	0.1820	0.0491	0.0457	0.0670	0.0613	0.0291
8x4	0.3528	0.1674	0.1771	0.0819	0.0745	0.0622	0.0557	0.0283
4x8	0.3342	0.1568	0.1827	0.0468	0.0399	0.1066	0.0972	0.0358
4x4	0.3109	0.1706	0.1819	0.0619	0.0597	0.0936	0.0892	0.0323
intra16	0.6367	0.0073	0.0108	0.0032	0.0010	0.0014	0.0027	0.3369
intra4	0.2657	0.0414	0.0443	0.0054	0.0091	0.0237	0.0269	0.5835

Table 8. The transition probability for depth 3.

H.264 \ HEVC	8x8	8x4	4x8	intra8	intra4
skip	0.9944	0.0020	0.0027	0.0009	0.0000
16x16	0.9215	0.0259	0.0339	0.0156	0.0031
16x8	0.8856	0.0391	0.0444	0.0249	0.0061
8x16	0.8764	0.0350	0.0485	0.0311	0.0091
8x8	0.8495	0.0468	0.0613	0.0324	0.0101
8x4	0.5961	0.2714	0.0722	0.0424	0.0179
4x8	0.5615	0.0515	0.3150	0.0515	0.0205
4x4	0.5018	0.1857	0.2395	0.0455	0.0275
intra16	0.8359	0.0018	0.0086	0.1494	0.0042
intra4	0.4604	0.0173	0.0359	0.3541	0.1322

Tables 9 and Table 10 show the transition probability for depths 0 and 1, respectively. The left column in Table 9 or Table 10 lists the number of subMB for the current encoding CU, and the above row lists the final PU by the HEVC encoder. For each row, it shows the transition probability of the PU. The sum of the transition probability of each row is 1.

The search range of a PU is adjusted by a weighting factor which is assigned based on the PU transition probability. The weighting assignment is listed in Table 11. Finally, the search range is defined using (6):

$$Search\ Range = \min(W \times refSR, original\ SR). \quad (6)$$

The final search range is *refSR* multiplied by the weighting, but it cannot be greater than the default value, 64, of HEVC. Figure 5 shows the CDF of the difference between the proposed search range and the best search range (i.e. motion vector difference, MVD) for BQSquare. The difference with positive value denotes that the proposed search range is larger than the best search range, and we do not lose any prediction performance in this situation with just less time saving.

From Figure 5 we can observe that the differences are almost near zero and only a small part is smaller than zero. The proposed search range can efficiently save most of the complexity in motion estimation.

Table 9. The transition probability for depth 0.

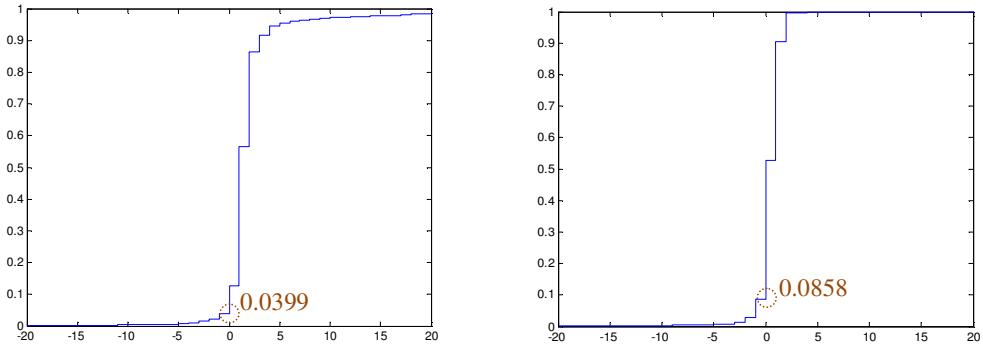
HEVC / subMB	64x64	64x32	32x64	64x16	64x48	16x64	48x64	intra64
0	0.6661	0.1002	0.1417	0.0172	0.0106	0.0163	0.0328	0.0151
1-4	0.2508	0.2675	0.2930	0.0307	0.0233	0.0452	0.0786	0.0108
5-8	0.1544	0.2897	0.3683	0.0337	0.0296	0.0500	0.0565	0.0179
9-12	0.1379	0.3040	0.3894	0.0172	0.0259	0.0402	0.0439	0.0415
13-16	0.0455	0.3482	0.4442	0.0417	0.0720	0.0152	0.0000	0.0333

Table 10. The transition probability for depth 1.

HEVC / subMB	32x32	32x16	16x32	32x8	32x24	8x32	24x32	intra32
0	0.6195	0.0979	0.1159	0.0343	0.0273	0.0440	0.0371	0.0240
1	0.3201	0.1640	0.2084	0.0619	0.0626	0.0836	0.0752	0.0242
2	0.2106	0.1787	0.2346	0.0765	0.0722	0.1037	0.1018	0.0219
3	0.1662	0.1845	0.2342	0.0938	0.0795	0.1152	0.1078	0.0188
4	0.1471	0.1859	0.2507	0.0840	0.0653	0.1250	0.1278	0.0141

Table 11. The weighting table to the search range.

Transition Probability	Weighting, *W*
$P \leq 0.1$	0.6
$0.1 < P \leq 0.3$	0.8
$0.3 < P \leq 0.5$	1
$P > 0.5$	1.2



(a) CDF of depths 0 and 1 (b) CDF of depths 2 and 3

Figure 5. CDF of the difference between the proposed search range and the best search range for BQSquare. The horizontal-axis is the difference between the proposed search range and the best search range, and the vertical-axis is cumulative distribution

3. EXPERIMENTAL RESULTS

Simulations were taken by both JM, the software of H.264/AVC, version17.2 [9] with a QP value of 24, and HM, the software of HEVC, version 9.2 [10] with QP values 24, 27, 31, and 35. The input sequences were IPPP…P with different classes, and the default test zone (TZ) fast search in HEVC was enabled [10]. The test hardware was a PC with Intel I7-3770, 3.4GHz CPU, 16G RAM, and Windows 7 the operating system. Experimental results include the fast mode decision, fast motion estimation, and the overall performance. The overall performance is also compared with the performance proposed by [8].

3.1. Results for the Fast Mode Decision Algorithm

Table 12 lists the performance for the fast mode decision. The proposed algorithm saves 32-35% of the total encoding time from the test sequences under four various QPs. In particular, the E class of sequences can save more encoding time because their contents are homogeneous, with smooth contents. Thus, larger size of blocks or more skip modes can be adopted by the proposed algorithm.

Table 12. Experimental results for fast mode decision.

		QP=24			**QP=27**		
Sequence		ΔPSNR (dB)	ΔBitrate	ΔEncTime	ΔPSNR (dB)	ΔBitrate	ΔEncTime
ClassC	BasketballDrill	-0.041	1.64%	-35.71%	-0.047	1.78%	-34.57%
	RaceHorses	-0.020	1.26%	-27.35%	-0.030	1.49%	-25.30%
ClassD	BasketballPass	-0.026	1.28%	-29.14%	-0.035	1.26%	-28.38%
	BlowingBubbles	-0.019	0.44%	-27.14%	-0.018	0.52%	-26.40%
ClassE	vidyo1	-0.042	0.41%	-49.00%	-0.034	0.90%	-48.79%
	vidyo4	-0.033	0.73%	-43.89%	-0.040	0.71%	-43.18%
Average		**-0.030**	**0.96%**	**-35.37%**	**-0.034**	**1.11%**	**-34.44%**
		QP=31			**QP=35**		
Sequence		ΔPSNR (dB)	ΔBitrate	ΔEncTime	ΔPSNR (dB)	ΔBitrate	ΔEncTime
ClassC	BasketballDrill	-0.050	1.91%	-33.27%	-0.070	1.62%	-32.50%
	RaceHorses	-0.042	1.72%	-23.61%	-0.043	1.73%	-21.39%
ClassD	BasketballPass	-0.047	1.55%	-27.79%	-0.087	1.91%	-27.18%
	BlowingBubbles	-0.033	0.62%	-24.74%	-0.033	0.88%	-23.27%
ClassE	vidyo1	-0.038	0.60%	-48.98%	-0.031	0.74%	-49.41%
	vidyo4	-0.049	0.74%	-43.18%	-0.046	1.03%	-43.35%
Average		**-0.043**	**1.19%**	**-33.60%**	**-0.052**	**1.32%**	**-32.85%**

3.2. Results for the Fast Motion Estimation Algorithm

Table 13 lists the performance of the fast motion estimation. The proposed algorithm saves approximately to 15% of the total encoding time and maintains the R-D performance from the test sequences. In particular, the test sequence RaceHorses in class C has better performance than other sequences. This is because this sequence is a high motion sequence, and the proposed adaptive search range can be more effective than the method of fixed search range. On the other

hand, the test sequence Vidyo 1 in class E is a low motion sequence, the TZ search in HEVC can quickly find the best MV so that the time saving by the proposed algorithm is reduced.

Table 13. Experimental results for fast motion estimation.

		QP=24			QP=27		
Sequence		ΔPSNR (dB)	ΔBitrate	ΔEncTime	ΔPSNR (dB)	ΔBitrate	ΔEncTime
ClassC	BasketballDrill	-0.007	0.25%	-15.70%	-0.008	0.38%	-16.94%
	RaceHorses	-0.001	0.34%	-22.39%	-0.006	0.46%	-22.67%
ClassD	BasketballPass	-0.018	0.41%	-17.14%	-0.017	0.25%	-16.98%
	BlowingBubbles	-0.006	0.11%	-12.34%	-0.005	-0.21%	-13.37%
ClassE	vidyo1	-0.011	-0.05%	-7.78%	-0.008	-0.11%	-7.46%
	vidyo4	-0.008	-0.04%	-15.73%	-0.012	0.09%	-15.70%
Average		**-0.009**	**0.17%**	**-15.18%**	**-0.009**	**0.14%**	**-15.52%**
		QP=31			**QP=35**		
Sequence		ΔPSNR (dB)	ΔBitrate	ΔEncTime	ΔPSNR (dB)	ΔBitrate	ΔEncTime
ClassC	BasketballDrill	0.002	0.47%	-16.95%	-0.033	-0.16%	-16.29%
	RaceHorses	-0.025	0.42%	-24.48%	-0.020	0.40%	-23.87%
ClassD	BasketballPass	-0.030	0.01%	-16.12%	-0.042	0.33%	-14.84%
	BlowingBubbles	-0.005	0.31%	-13.64%	-0.024	0.49%	-13.47%
ClassE	vidyo1	-0.010	-0.28%	-7.31%	0.002	-0.38%	-7.02%
	vidyo4	-0.004	-0.41%	-14.66%	0.001	0.15%	-13.25%
Average		**-0.012**	**0.09%**	**-15.53%**	**-0.019**	**0.14%**	**-14.79%**

3.3. Results for the Overall Performance

Table 14 lists the overall performance. The proposed algorithm can save 43-45% of the total encoding time. Comparing with [8], the proposed algorithm performs better in the encoding time but increases the bitrate a little. In particularly, as the QP is closer to the original QP by H.264/AVC, the proposed algorithm saves much of the encoding time. Therefore, as these two transcoding standards are with the closer QP assignment, the proposed algorithm can be more effective.

4. CONCLUSIONS

In this work, a fast transcoder from H.264/AVC to HEVC focusing on the PU is proposed. The proposed method consists of the fast mode decision and the fast motion estimation. Various fast mode algorithms are applied to the transcoder for depths 0 and 1, and depths 2 and 3, respectively. As for the fast motion estimation, adaptive search range with the mode transition probability is proposed. Experimental results show the algorithms of the fast mode decision and the fast motion estimation can save 32-35% and 15% of the total encoding time, respectively. The overall encoding time can be reduced about 44% on average. In other words, about half of the encoding time can be saved under the acceptable R-D performance.

Table 14. Experimental results for overall algorithm.

		Reference [8]			Proposed Algorithm		
Sequence		ΔPSNR (dB)	ΔBitrate	ΔEncTime	ΔPSNR (dB)	ΔBitrate	ΔEncTime
ClassB	Cactus	-0.030	0.38%	-35.67%	-0.028	0.83%	-44.41%
	ParkScene	-0.021	0.16%	-32.02%	-0.032	0.65%	-44.97%
	Kimono1	-0.027	0.56%	-43.64%	-0.046	0.47%	-47.11%
ClassC	BasketballDrill	-0.045	0.97%	-36.75%	-0.046	1.89%	-46.57%
	RaceHorses	-0.019	0.94%	-41.24%	-0.019	1.69%	-43.83%
	PartyScene	-0.038	0.63%	-30.93%	-0.025	1.23%	-41.20%
ClassD	BasketballPass	-0.054	1.59%	-39.02%	-0.037	1.53%	-42.09%
	BlowingBubbles	-0.063	1.63%	-33.50%	-0.020	0.67%	-36.41%
	BQSquare	-0.040	1.34%	-21.18%	-0.017	0.70%	-37.76%
ClassE	vidyo1	-0.062	-0.39%	-43.43%	-0.057	0.30%	-53.27%
	vidyo3	-0.056	-0.38%	-41.55%	-0.053	0.64%	-52.24%
	vidyo4	-0.042	-0.07%	-43.61%	-0.038	0.98%	-53.73%
QP24 Average		**-0.041**	**0.61%**	**-36.88%**	**-0.035**	**0.97%**	**-45.30%**
Sequence		ΔPSNR (dB)	ΔBitrate	ΔEncTime	ΔPSNR (dB)	ΔBitrate	ΔEncTime
ClassB	Cactus	-0.015	0.44%	-37.16%	-0.028	1.15%	-43.64%
	ParkScene	-0.022	0.20%	-33.55%	-0.036	0.82%	-44.02%
	Kimono1	-0.031	0.48%	-43.62%	-0.044	0.45%	-45.87%
ClassC	BasketballDrill	-0.028	0.97%	-37.48%	-0.050	2.22%	-45.90%
	RaceHorses	-0.029	1.09%	-41.94%	-0.035	2.20%	-43.45%
	PartyScene	-0.039	0.62%	-31.24%	-0.037	1.67%	-39.99%
ClassD	BasketballPass	-0.050	1.60%	-39.52%	-0.051	1.61%	-41.52%
	BlowingBubbles	-0.068	1.03%	-34.17%	-0.027	0.74%	-36.25%
	BQSquare	-0.044	0.72%	-21.78%	-0.026	0.54%	-35.85%
ClassE	vidyo1	-0.023	0.23%	-45.84%	-0.042	1.13%	-53.28%
	vidyo3	-0.026	-0.30%	-44.41%	-0.034	1.22%	-52.26%
	vidyo4	-0.026	-0.02%	-45.98%	-0.047	0.79%	-53.53%
QP27 Average		**-0.033**	**0.59%**	**-38.06%**	**-0.038**	**1.21%**	**-44.63%**
Sequence		ΔPSNR (dB)	ΔBitrate	ΔEncTime	ΔPSNR (dB)	ΔBitrate	ΔEncTime
ClassB	Cactus	-0.016	0.87%	-38.26%	-0.048	1.27%	-42.88%
	ParkScene	-0.025	0.25%	-34.98%	-0.042	0.89%	-43.04%
	Kimono1	-0.051	0.25%	-43.90%	-0.045	0.24%	-44.43%
ClassC	BasketballDrill	-0.029	1.08%	-38.67%	-0.054	2.35%	-44.99%
	RaceHorses	-0.043	1.18%	-42.41%	-0.059	2.33%	-42.72%
	PartyScene	-0.022	0.62%	-31.97%	-0.046	1.98%	-38.87%
ClassD	BasketballPass	-0.046	1.07%	-40.43%	-0.066	1.78%	-40.68%
	BlowingBubbles	-0.059	1.35%	-35.20%	-0.037	1.12%	-35.25%
	BQSquare	-0.041	1.03%	-22.65%	-0.032	0.79%	-34.22%

ClassE	vidyo1	-0.027	-0.32%	-48.39%	-0.060	0.83%	-53.40%
	vidyo3	-0.013	0.48%	-46.98%	-0.056	2.01%	-52.55%
	vidyo4	-0.024	0.06%	-47.58%	-0.056	1.07%	-53.20%
QP31 Average		**-0.033**	**0.66%**	**-39.29%**	**-0.050**	**1.39%**	**-43.85%**
Sequence		ΔPSNR (dB)	ΔBitrate	ΔEncTime	ΔPSNR (dB)	ΔBitrate	ΔEncTime
ClassB	Cactus	-0.017	1.21%	-39.16%	-0.054	1.75%	-42.10%
	ParkScene	-0.017	0.50%	-36.27%	-0.043	0.84%	-41.96%
	Kimono1	-0.082	0.60%	-44.00%	-0.085	0.50%	-42.74%
ClassC	BasketballDrill	-0.062	0.98%	-39.21%	-0.092	2.31%	-44.17%
	RaceHorses	-0.052	1.12%	-42.82%	-0.063	2.42%	-41.29%
	PartyScene	-0.033	0.47%	-32.40%	-0.058	2.01%	-37.20%
ClassD	BasketballPass	-0.081	0.87%	-41.36%	-0.109	2.12%	-39.43%
	BlowingBubbles	-0.051	0.99%	-35.73%	-0.044	1.41%	-34.05%
	BQSquare	-0.057	0.37%	-24.06%	-0.047	0.70%	-32.90%
ClassE	vidyo1	-0.015	0.30%	-49.58%	-0.042	0.94%	-53.60%
	vidyo3	-0.029	-0.03%	-48.49%	-0.072	2.05%	-52.60%
	vidyo4	-0.013	0.17%	-48.52%	-0.051	1.18%	-52.74%
QP35 Average		**-0.042**	**0.63%**	**-40.13%**	**-0.063**	**1.52%**	**-42.90%**

REFERENCES

[1] Advanced Video Coding, ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), Version 13, Mar. 2011.

[2] JCT-VC, "High Efficiency Video Coding (HEVC) Text Specification Draft 10," JCTVC-L1003, 12th JCTVC meeting, Geneva, Jan. 2013.

[3] E. Peixoto and E. Izquierdo, "A Complexity-Scalable Transcoder from H.264/AVC to the New HEVC Codec," in Proc. IEEE International Conference on Image Processing (ICIP), Sep. 2012, pp. 737-740.

[4] X. Jing, W. C. Siu, L. P. Chau, and A. G. Constantinides, "Efficient Inter Mode Decision for H.263 to H.264 Video Transcoding Using Support Vector Machines," in Proc. IEEE International Symposium on Circuits and Systems (ISCAS), May 2009, pp. 2349-2352.

[5] D. Zhang, B. Li, J. Xu, and H. Li, "Fast Transcoding from H.264/AVC to High Efficiency Video Coding," in Proc. IEEE International Conference on Multimedia & Expo. (ICME), July 2012, pp. 651-656.

[6] R. Garrido-Cantos, J. D. Cock, J. L. Martínez, S. V. Leuven, and P. Cuenca, "Motion-Based Temporal Transcoding from H.264 AVC-to-SVC in Baseline Profile," IEEE Trans. Consum. Electron., vol. 57, no. 1, pp. 239-246, Feb. 2011.

[7] J. Zhang, F. Dai, Y. Zhang, and C. Yan, "Efficient HEVC to H.264/AVC Transcoding with Fast Intra Mode Decision," in Proc. the 19th International Conference on Multimedia Modeling, Jan. 2013, Lecture Notes in Computer Science vol. 7732, 2013, pp. 295-306.

[8] Z. Y. Chen, C. T. Tseng, and P. C. Chang, "Fast Inter Prediction for H.264 to HEVC Transcoding," in Proc. the 3rd International Conference on Multimedia Technology (ICMT), Guangzhou, China, Nov. 2013, Atlantis Press, pp. 1301-1308.

[9] Joint Video Team software JM17.2. Available at: https://iphome.hhi.de/suehring/tml/download/

[10] HM Reference Software 9.2. Available at: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoft-ware