Early Coding Unit–Splitting Termination Algorithm for High Efficiency Video Coding (HEVC)

Kalyan Goswami, Byung-Gyu Kim, Dongsan Jun, Soon-Heung Jung, and Jin Soo Choi

A new-generation video coding standard, named High Efficiency Video Coding (HEVC), has recently been developed by JCT-VC. This new standard provides a significant improvement in picture quality, especially for high-resolution videos. However, one the most important challenges in HEVC is time complexity. A quadtree-based structure is created for the encoding and decoding processes and the rate-distortion (RD) cost is calculated for all possible dimensions of coding units in the quadtree. This provides a high encoding quality, but also causes computational complexity. We focus on a reduction scheme of the computational complexity and propose a new approach that can terminate the quadtree-based structure early, based on the RD costs of the parent and current levels. Our proposed algorithm is compared with HEVC Test Model version 10.0 software and a previously proposed algorithm. Experimental results show that our algorithm provides a significant time reduction for encoding, with only a small loss in video quality.

Keywords: HEVC, coding tree block, coding unit, CU splitting.

I. Introduction

Recently, ISO-IEC/MPEG and ITU-T/VCEG formed the Joint Collaborative Team on Video Coding (JCT-VC), which developed the next-generation video coding standard called High Efficiency Video Coding (HEVC) [1]. The major goal of HEVC was to achieve a significant improvement in coding efficiency, compared to H.264/AVC [2], especially with high-resolution video content.

The video encoding and decoding processes in HEVC are composed of three units: a coding unit (CU) for the root of the transform quadtree, as well as a prediction mode for the INTER/SKIP/INTRA prediction; a prediction unit (PU) for coding the mode decision, including motion estimation and rate-distortion (RD) optimization; and a transform unit (TU) for transform coding and entropy coding. Initially, a frame is divided into a sequence of its largest non-overlapping coding units, called a coding tree unit (CTU). A CTU can be recursively divided into smaller CUs and made flexible using quadtree partitioning, which is called a coding tree block (CTB).

A CTU has a block structure size of 64 × 64 pixels, which can be decomposed into four 32 × 32 pixels CUs. Further still, each 32 × 32 pixels CU can be divided into four CUs of 16 × 16 pixels. This decomposition process can continue to CUs of up to 8 × 8 pixels blocks. That means the 8 × 8 pixels block is the smallest possible for a CU. Moreover, for the different combinations of CU structures, different CTBs are generated for a single CTU. For each CTB, RD cost value is calculated. The CTB which has the minimum RD cost value is considered as the best one. The illustration of the CTB structure for a CTU is given in Fig. 1(a). In Fig. 1, a 64 × 64 pixels CTU block is shown divided into smaller blocks of CUs. Upon calculating

Manuscript received May 7, 2013; revised Oct. 22, 2013; accepted Nov. 11, 2013.

This research was supported by the Korea Communications Commission, Korea, under the ETRI R&D support program supervised by the Korea Communications Agency (KCA-2012-11921-02001).

Kalyan Goswami (phone: +82 41 530 2271, iit.kalyan@gmail.com) and Byung-Gyu Kim (corresponding author, bg.kim@mpcl.sunmoon.ac.kr) are with the Department of Computer Engineering, Sun Moon University, Asan, Rep. of Korea.

Dongsan Jun (hopeof@mail.kaist.ac.kr), Soon-Heung Jung (zeroone@etri.re.kr), and Jin Soo Choi (chitos@mail.kaist.ac.kr) are with the Broadcasting & Telecommunications Media Research Laboratory, ETRI, Daejeon, Rep. of Korea.

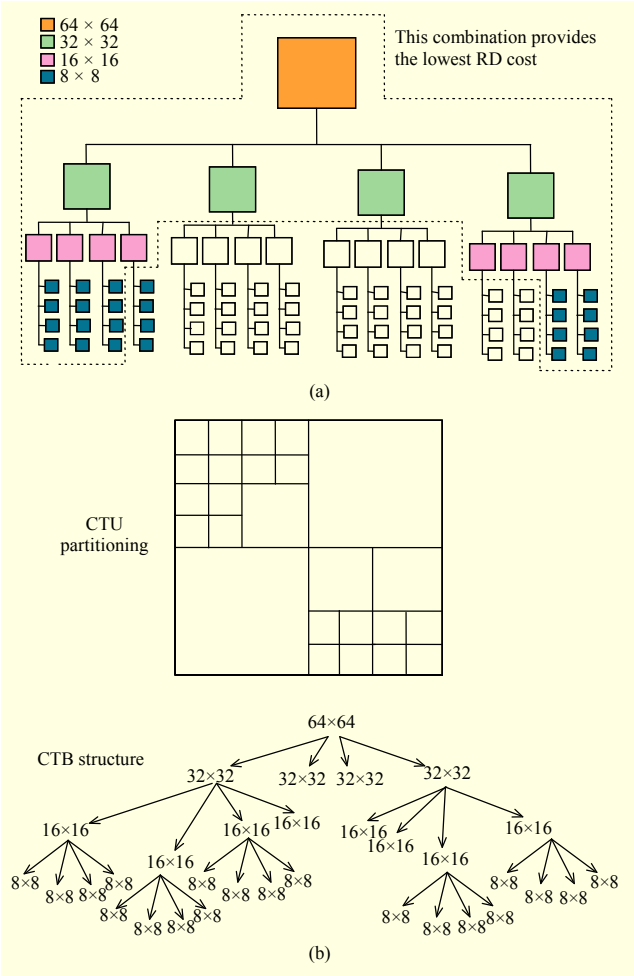


Fig. 1. (a) CTB structure which provides the lowest RD cost for CTU and (b) Corresponding CTU partitioning for the best CTB structure.

the RD cost for every combination, the CUs which are under the red dotted part of Fig. 1(a) give the minimum RD value. The corresponding CTU partitioning and CTB structure for this particular (best) combination is shown in Fig. 1(b).

The CTB is an efficient representation of variable block sizes so that regions of different sizes can be coded with fewer bits while maintaining the same quality. It is possible to encode stationary or homogeneous regions with a larger block size, resulting in a smaller side-information overhead. On the other hand, the CTB structure dramatically increases the computational complexity. As an example, if a frame has dimensions of 704 × 576 pixels, then it will be decomposed into 99 (11 × 9) CTUs, and a separate CTB will be created for each CTU. For each CTB, 85 calculations are involved for different CU sizes. As a result, 8,415 CU calculations are required for the CTB structure, whereas only 1,584 calculations are needed for a 16 × 16 macroblock, as was used in the previous standard (H.264/AVC). From this analysis, it is clear that the new CTB structure in HEVC greatly increases the computational complexity. From the viewpoint of real-time applications, we need faster video encoders to support real-time video services. Hence, it is important to design a HEVC encoder which can encode a video stream so as to achieve a similar bit rate and quality but also a reduction in encoding time, while comparing with the HEVC Test Model (HM) reference software as a benchmark.

Only a few reports have been published regarding reducing the CTB computational complexity. In [3], a proposed novel early-CU termination algorithm, commonly known as ECU, was integrated into the HM reference software. According to ECU, no further processing of sub-trees is required when the current CU selects SKIP mode as the best prediction mode at the current CU depth. In [4], it is shown that when the cost of a current CU is lower than the sum of the costs of CUs belonging to the subtrees of the current CU, then no further processing of subtrees is required. In [5], an early partition decision algorithm is presented that attempts to terminate the mode decision process after checking the INTER mode with each of the PU partition types. Some fast-termination algorithms have been reported to explore other components in HEVC [6]–[9].

In [10], an early CU size-determination algorithm has been reported. In this work, the authors have exploited two fast approaches — adaptive depth-range determination and early termination of unnecessary motion estimation on small CU sizes. Supervised learning–based algorithms have also been used for estimating the CU size using features specified in [11], [12]. A Bayesian decision rule has been used in [11] and the supported vector machine was applied in [12] to determine the CU size before the general RD optimization technique in HM reference software. All the above mentioned algorithms are related with INTER prediction. On the other hand, a fair number of works have been reported in fast INTRA-mode decision. In [13], variance values of coding-mode costs are used to terminate the current CU mode decision as well as TU size selection. A two-stage process has been reported in [14], where in the first stage texture complexity of different CUs are analyzed, followed by an elimination process for small prediction unit candidates for current blocks in INTRA mode. In [15], a coarse INTRA-mode decision algorithm is applied by first using the Hadamard transform. Then, a fine refinement is done to reduce the complexity of the INTRA-mode decision. A gradient-based approach has been used in [16] for fast INTRA prediction. Apart from that, TU splitting approaches are also used for fast mode decision. In [17], a residual quadtree mode decision algorithm has been reported by replacing the original depth-first mode decision process by a merge-and-split process.

In this paper, we are focusing on CU partitioning to reduce the computational complexity of the CTB structure. Our main objective is to create an algorithm that decides whether a CU should be decomposed into four lower-dimension CUs or not. The proposed approach should terminate the coding tree earlier than conventional standard reference software. The proposed algorithm is based on the RD costs of the parent and current CUs in a CTB. Motion activity and local statistics of the RD cost value are also considered in this context. As we have previously mentioned, there have already been a number of attempts at this problem. However, our proposed technique is simple to implement and robust in nature. Moreover, it is possible to incorporate the proposed technique into other early-CU termination algorithms.

In the next section, our proposed approach is discussed in detail. The experimental results are given in section III and finally conclusions are drawn in section IV.

II. Proposed Approach

As we have discussed in the last section, the CTB which provides the minimum RD cost is considered to be the best for a HEVC encoder. According to this technique, before calculating the RD values of all possible combinations of CTB, it is impossible to take any decision regarding CTU partitioning. Since the main objective is to select the lowest RD cost among all combinations in [4], it is shown that no further decomposition is required if the CTB satisfy the condition shown in (1), where the subscript t indicates the current level and t+1 indicates the next level

$$RD_{\text{cost}}(CU_t) < \sum_{i=0}^{3} RD_{\text{cost}}(CU_{t+1}(i)). \tag{1}$$

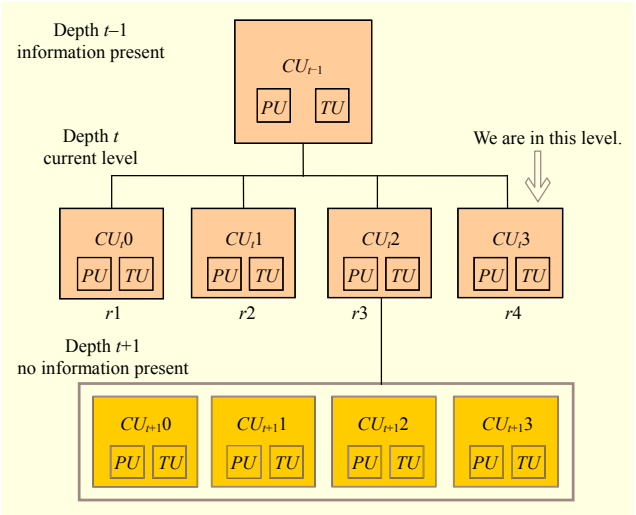


Fig. 2. Available information in a CTB structure.

However, at depth t, it is not possible to get any information about depth t+1 without splitting the CU. We can only use information from the prior level (depth t–1). In Fig. 2, the available information is shown for each level of hierarchy in a CTB. Based on this situation, we use the RD cost values of the current depth t and the previous depth (t–1) levels and propose a ratio function, which is discussed in detail in the next subsection.

1. Ratio Function

Let us consider that CU_{t-1} was split and four CUs were created, which means (1) is not satisfied. So, we can consider that the RD cost of CU_{t-1} is greater than the costs of its child nodes, as shown in (2)

$$\begin{aligned} & RD_{\text{cost}}(CU_{t-1}) \geq \sum_{i=0}^{3} RD_{\text{cost}}(CU_t(i)) \\ \Rightarrow & \frac{\sum_{i=0}^{3} RD_{\text{cost}}\left(CU_t(i)\right)}{RD_{\text{cost}}\left(CU_{t-1}\right)} \leq 1. \end{aligned} \tag{2}$$

The ratio function for a $CU_t(i)$ at depth t can be defined as

$$r_i = \frac{RD_{\text{cost}}(CU_t(i))}{RD_{\text{cost}}(CU_{t-1})}, \text{ where } \sum_{i=0}^{3} r_i \leq 1. \tag{3}$$

This parameter is basically a ratio of the RD costs of the current CU and its parent CU. When a CU is split, then it will create four child CUs and for each newly created CU, we can define a value for its ratio function. From (3), it can be inferred that this parameter should have some upper limit, since the sum of the ratio function has an upper bound of 1. It is also observed that when the ratio function of a child is lower than its siblings, then in the next level it has a low chance of splitting.

To justify our theoretical approach and to make an upper bound for the ratio function, we have performed an experiment on HM reference software for four sequences of different dimensions. In this experiment, we have checked the ratio functions for all encoded CUs for four quantization parameter (QP) values, which have been split. According to our experimental result, all the split CUs have a ratio function below 0.25, as shown in Fig. 3. That means if a child has an RD cost that is less than one quarter of its parent, then at the next level it has a low chance of splitting. Moreover, to make this decision, we do not need any information from depth t+1. Therefore, this parameter can be used as a threshold for making the decision to split a current CU in the next level.

In our proposed algorithm, we have incorporated the ratio function for different motion activities in a video sequence. The details of the motion activity are discussed in the next subsection.

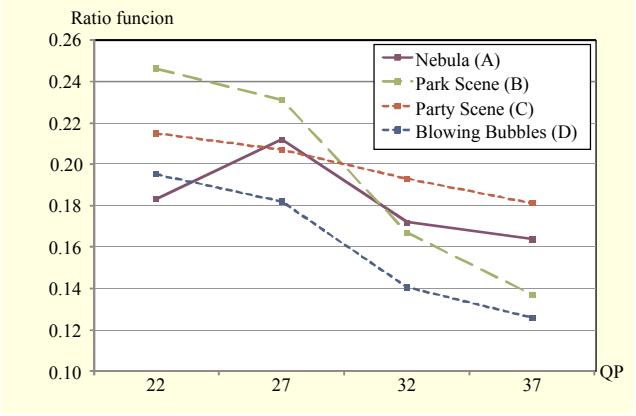


Fig. 3. Ratio function analysis of four video sequences.

2. Motion-Activity Information at PU Level

A CU is consisted of two basic units: PU and TU. All prediction-related calculations and algorithms are under the umbrella of PU. There are three kinds of predictions possible in HEVC. These are SKIP, INTRA and INTER mode, predictions. However, in SKIP mode, there is no need to encode a CU at all — it can be predicted directly from the reference frame. Generally, homogeneous and motionless regions are encoded as SKIP mode. According to the ECU algorithm [3], if a CU is coded as SKIP mode then no further splitting of that CU is required. Motivated by this fact, in this work, we have explored other prediction modes of PU.

As shown in Fig. 4, INTER and INTRA predictions have different kinds of PU modes. In HEVC encoders, SKIP mode is calculated first, followed by INTER and INTRA modes. The RD values for all kinds of modes for both INTER and INTRA (as shown in Fig. 4) are calculated in HEVC encoders to get the best PU mode. After the completion of all mode calculations, the CU is divided into four CUs of lower dimensions. PU mode calculations of each child CU are done recursively.

Generally, complex motion and rich-texture regions are encoded as low dimension INTER or INTRA PU modes. We have classified all the prediction modes according to motion activity (Table 1). However, to classify the prediction modes into different motion activities is not a new concept. This kind of motion activity was explored to a greater extent in the H.264/AVC-based codec system [18]–[20]. In this work, we have classified the PU modes into four motion activities: motionless region, slow-moving region, moderate motion–based region, and complex motion or texture-based region. In Table 1, our classified motion activity and the corresponding PU modes are shown. This is basically a labeling of the PU dimension, which incorporates negligible computational complexity of the overall process [21]. However, we have defined a parameter named as PU-mode weighting factor (PU_WF). This parameter simply represents each group of motion activity–based PU modes, as shown in Table 1.

We have assumed that if a region in a video sequence has relatively slow motion activity compared to other regions, then there is a high chance that the slow-moving region will not split in that hierarchy of the CTB. Motivated by this concept, we have performed an experiment to show as a percentage the number of non-splits of a CU at any level of hierarchy in a CTB. In this experiment, we have tested four sequences with different resolutions for four QP values and observed the CUs which are not split for different PU_WFs. The test sequences have a moderate amount of motion throughout the sequence. The first few frames (20 frames) are considered in this experiment. The result of this experiment is shown in Table 2.

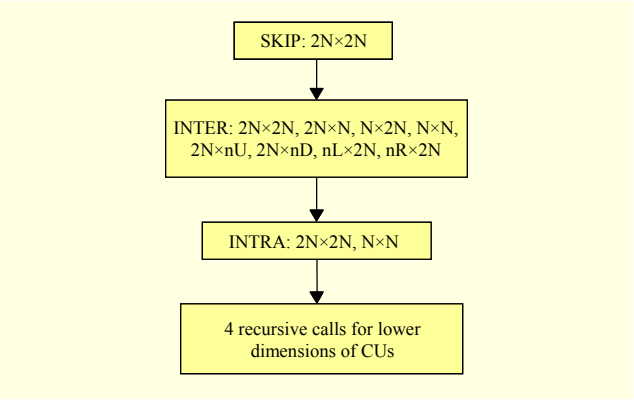


Fig. 4. Four PU partition types in HEVC.

Table 1. Motion activity and PU_WF.

Mode	Motion activity	PU_WF
SKIP	Motionless homogeneous region	0
INTER 2N×2N	Slow motion	1
INTER 2N×N and N×2N	Between slow and moderate	2
Other INTER and INTRA	Complex motion or texture	3

Table 2. Percentage of CUs that are not split at the next level.

Sequence	CUs that are not split (%)			
	PU_WF = 0	PU_WF = 1	PU_WF = 2	PU_WF = 3
Nebula (A)	97.15	63.21	45.95	18.93
Park Scene (B)	91.35	58.98	42.36	16.68
Party Scene (C)	95.16	62.33	43.24	15.29
Blowing Bubbles (D)	93.51	61.32	41.68	11.81
Average	94.29	61.46	43.30	15.67

ETRI Journal, Volume 36, Number 3, June 2014
http://dx.doi.org/10.4218/etrij.14.0113.0458

In this table, the average of four QP values for each sequence is given. Form this table, it is clear that when PU_WF is 0 (SKIP mode), then there is a very high chance that the CU will not split at the next level. Hence, if PU_WF is 0, then we can directly say that there is no need to split the CU, which is basically the ECU algorithm [3]. On the other hand, if PU_WF is 1 or 2, then there is a significant chance that the CU will not split in the next level.

However, the average percentage of non-splitting CUs ranges from 40% to 60%. Hence, it will be not wise to directly make any decision about the splitting of a CU without further investigation. But in this stage, we can at least be confident that a CU with PU_WF of 1 or 2 has a relatively high chance of not splitting at the next level. Finally, for the last group, from this experiment, we can say that it has quite a high chance of splitting at the next level.

3. Local Average of RD Cost Calculation

As we have mentioned in the previous subsection, we need to investigate further CUs that have a PU_WF of 1 or 2. We have calculated a local average of the RD cost values of all encoded CUs. This parameter is not a static value as it takes the values dynamically from the encoded CUs. In this process, after encoding a CU, the final PU mode and the corresponding RD cost values are checked. A running average is calculated for the final RD cost value. The calculation procedure for the local average is shown in Fig. 5 as a flow chart.

This parameter is basically a running average of the RD cost values of different dimension of CUs and the corresponding

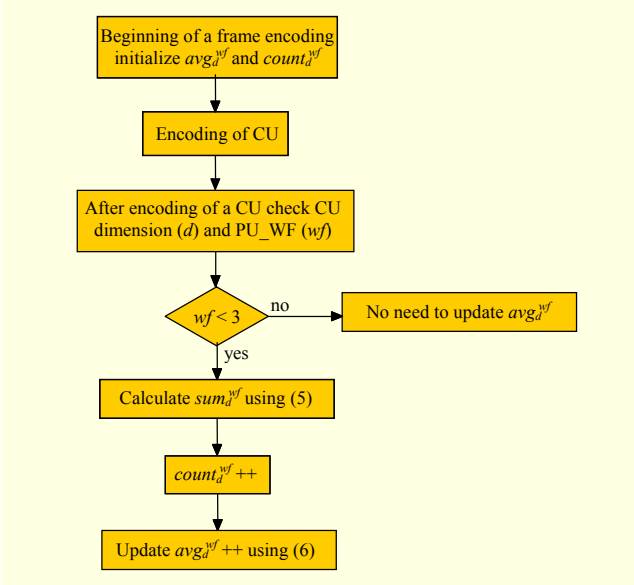


Fig. 5. Flow chart for local average of RD-cost-calculation algorithm.

Table 3. Percentage of CUs not split at the next level if a particular CU's RD cost < avg_d^{wf}.

Sequences	CU size	PU_WF			
		0	1	2	3
Nebula (A)	64×64	72.56	72.96	68.21	2.63
	32×32	73.24	71.36	67.36	7.36
	16×16	78.24	72.36	68.65	5.32
Park Scene (B)	64×64	72.35	69.36	66.32	6.35
	32×32	69.35	68.34	64.23	7.23
	16×16	68.23	67.29	66.37	8.28
Party Scene (C)	64×64	78.25	73.25	68.48	2.21
	32×32	76.35	72.39	71.24	1.58
	16×16	77.84	71.54	72.45	4.23
Blowing Bubbles (D)	64×64	69.32	73.21	71.54	2.23
	32×32	71.25	71.23	71.11	5.41
	16×16	72.23	72.84	69.18	2.71

PU_WFs. This average can be calculated using equations (5) and (6)

$$sum_d^{wf} = (avg_d^{wf} \times count_d^{wf}) + RD\_\text{cost}_d^{wf}, \quad (5)$$

$$avg_d^{wf} = \frac{sum_d^{wf}}{count_d^{wf} + 1}. \quad (6)$$

We have performed an experiment to calculate the probability of a non-splitting CU when the RD cost of the CU is less than its local average RD cost. The result is shown in Table 3. It is clear that for PU_WF values of 0, 1 and 2, the probability is within 65% to 75%. In this experiment, we have used four sequences with different resolutions.

4. Absolute Value of Motion Vector Calculation

An absolute motion vector (MV) is also calculated in our algorithm. This is basically an average of the MVs from both List_0 and List_1 in the HM reference software. To simplify the calculation of the absolute MV, we only use the magnitudes of the x- and y-MV direction, as shown in (7)

$$MV_{abs} = 0.5 \times \left(\sum_{i=0}^{1} |MV_x(i)| + |MV_y(i)| \right). \quad (7)$$

In this context, we have assumed that a CU with low motion should have a high chance of not splitting. Since the MV_{abs} is directly related to the motion of a CU, we can infer that a CU with a low MV_{abs} value has a high chance of not splitting in the current hierarchy of the CTB. To justify our assumption, we

Table 4. Percentage of CUs not split at the next level for different MV_{abs} values.

Sequence	CUs which are not split (%)			
	$MV_{abs}=0$	$1 \le MV_{abs} \le 2$	$3 \le MV_{abs} \le 5$	$MV_{abs}>5$
Nebula (A)	76.15	4.36	2.14	1.22
Park Scene (B)	72.96	3.51	3.26	1.96
Party Scene (C)	71.53	2.79	2.12	2.13
Blowing Bubbles (D)	68.73	3.92	2.73	1.27
Average	72.34	3.64	2.56	1.64

have performed another experiment in the same environment as discussed in section II, subsection 2. In this experiment, we have checked the percentage of CUs which are not split for different values of MV_{abs}. The result is shown in Table 4. From this table, it is clear that for $MV_{abs} = 0$, there is quite a high chance that the corresponding CU will not split. For this reason, we have included this condition when checking for the chance of a CU splitting. Since the probability of a non-split is not so high (65% to 75%), we cannot take any straightforward decision regarding features like avg_d^{wf} or MV_{abs} (see Tables 3 and 4). Hence, we need to combine this with other features in our main algorithm.

5. Proposed CU-Splitting Termination Algorithm

The proposed algorithm can be divided into two stages. In the first stage, only the CTU is considered while we do not have any information from higher levels. Hence, in this stage, the ratio function cannot be calculated. On the other hand, in the second stage, other higher level CUs are considered.

In this algorithm, initially we are checking PU_WF. If PU_WF is 0 (which means SKIP mode), then the decision is taken that there is no need for further splitting. On the other hand, for PU_WF values of 1 and 2, we cannot make any immediate decision. Hence, we need to check some other parameters. Since it is a two-stage algorithm, for both of the stages the parameters which are checked are different. For CTU, only MV_{abs} and local average-RD cost of a suitable dimension of PU are considered for PU_WF values of 1 and 2. Otherwise, we have to check the ratio function for the non-CTU case. In this case, ratio function MV_{abs} and local average-RD cost of suitable dimension of PU are checked for PU_WF values of 1 and 2. The final decision as to whether the CU will be split further it is taken based on the above mentioned checking of the parameters. The pseudocode of the proposed algorithm is given below. The PU_WF is used as an input and the output is CU_split flag.

```
Algorithm : CU-Splitting Termination (CST) Algorithm
input: PU_WF, CU structure
output: CU_split.
if PU_WF = 0 then
    CU_split = 0
end if
else
    calculate RD cost (RD) and MV_abs of Current CU
    if CTU then
        if PU_WF= 1 then
            if RD < avg RD cost of 2N×2N
             and MV_abs= 0 then
                CU_split = 0.
            end if
        end if
        else if PU_WF = 2 then
            if RD < (avg RD cost of 2N×N, avg RD cost
             of N×2N and avg RD cost of SKIP)
             and MV_abs= 0 then
                CU_split = 0.
            end if
        end if
        else CU_split = 1.
    end if
    else
        calculate ratio function (R)
        if PU_WF = 1 then
            if RD < avg RD cost of 2N×2N
             and R<0.25 then
                CU_split = 0.
            end if
        end if
        else if PU_WF= 2 then
            if RD < (avg RD cost of 2N×N,
             avg RD cost of N×2N) and MV_abs= 0 and
             R < 0.25
             then
                CU_split = 0.
            end if
        end if
        else CU_split = 1.
        end else
    end else
    Update avg RD cost of 2N×2N, avg RD cost of 2N×N,
     avg RD cost of N×2N and avg RD cost of SKIP.
```

III. Experimental Results

1. Test Condition

The proposed CST algorithm has been implemented in HM reference software, version 10.0. All simulation experiments

Table 5. Test sequence description.

Sequence name	Class	Dimension (pixels)	Frame rate per second (fps)	Description
Blowing Bubbles	D	416×240	50	Medium motion with zoom out
BQ Square	D	416×240	60	Synthetic with camera movement
Basketball Pass	D	416×240	50	High motion with rich texture
Race Horses	D	416×240	30	Medium motion with rich texture
Basketball Drill	C	832×480	50	High motion
Party Scene	C	832×480	50	Medium motion with zoom-in effect
Race Horses C	C	832×480	30	Medium motion with rich texture
BQ Mall	C	832×480	60	Medium motion with camera movement
Basketball Drive	B	1,920×1,080	50	High motion with rich texture
Cactus	B	1,920×1,080	50	Medium motion with rich texture
Kimono	B	1,920×1,080	24	Medium motion with rich texture
Park Scene	B	1,920×1,080	24	Medium motion with rich texture
BQ Terrace	B	1,920×1,080	60	Medium motion with camera movement
Traffic	A	2,560×1,600	30	Medium motion with rich texture
People on Street	A	2,560×1,600	30	Medium motion with rich texture
Nebula	A	2,560×1,600	60	Medium motion with rich texture and camera movement
Steam Locomotive	A	2,560×1,600	60	Medium motion with rich texture

are conducted on a PC with Intel Core (TM) i7-2600K processor having 3.4 GHz clock speed and 16 GB RAM. The test conditions are set as follows:

1. For each test sequence, the first 100 frames are encoded with group of pictures, size 8.
2. The QP is set at 22, 27, 32, and 37.
3. All experiments are performed in random-access mode.
4. Fast-encoder setting and fast decision for merge RD cost are on.
5. Cbf fast-mode setting and early SKIP detection options are off.
6. The test sequences have different resolution and belong to different classes. The description of different test sequences is given in Table 5. We have performed our experiments in random-access mode. Hence, Class-E sequences are not considered.

2. Performance Evaluation in HM 10.0

First of all, we have evaluated our algorithm with HM reference software, version 10.0. In this experiment, we have checked three parameters: required time to encode the video sequence, the number of bits in the encoded bit stream, and the corresponding peak signal-to-noise ratio (PSNR) value. In Table 6, the time reduction of the proposed algorithm compared with the original HM 10.0 is shown. In Table 5, QP values are shown for each sequence and the parameter ΔT is calculated using the following equation:

$$\Delta T = \frac{Time_{original} - Time_{proposed}}{Time_{original}} \times 100\%, \quad (8)$$

where $Time_{original}$ is the required time to encode the video sequence in HM 10.0 under the given test condition in section III, subsection 1. $Time_{proposed}$ is the required time after implementing the proposed CST algorithm in HM 10.0 in the same experimental environment.

From this table, it is quite clear that our proposed algorithm

Table 6. Time reduction in proposed algorithm.

Sequence name	ΔT%				Avg for all QPs
	QP=22	QP=27	QP=32	QP=37	
Blowing Bubbles	–25.24	–37.74	–59.21	–47.45	–42.41
BQ Square	–31.23	–40.42	–52.05	–59.74	–45.87
Basketball Pass	–26.40	–38.73	–36.30	–46.07	–36.87
Race Horses	–12.95	–17.11	–23.48	–34.67	–22.05
Basketball Drill	–31.35	–34.01	–41.69	–49.66	–39.18
Party Scene	–25.53	–35.76	–51.21	–45.37	–39.47
Race Horses C	–18.62	–27.98	–37.81	–45.56	–32.50
BQ Mall	–24.82	–31.11	–38.38	–45.56	–34.97
Basketball Drive	–30.73	–38.61	–45.59	–52.16	–41.77
Cactus	–32.62	–43.86	–49.79	–55.73	–45.50
Kimono	–21.36	–29.37	–39.10	–49.72	–34.89
Park Scene	–31.45	–42.54	–45.32	–52.21	–42.88
BQ Terrace	–25.54	–37.65	–41.43	–51.34	–38.99
Traffic	–24.43	–34.85	–42.32	–51.67	–38.32
People on Street	–28.32	–33.54	–41.67	–50.21	–37.18
Nebula	–20.32	–29.56	–37.32	–48.56	–33.94
Steam Locomotive	–26.32	–31.43	–40.43	–50.75	–36.48
Average	–26.05	–34.36	–43.24	–49.02	–38.03

Table 7. Performance and quality degradation of proposed algorithm.

Sequence name	$\Delta PSNR$	$\Delta Bit\,\%$	BD Rate
Blowing Bubbles	−0.084	−0.199	2.0
BQ Square	−0.115	−0.407	2.3
Basketball Pass	−0.065	0.222	1.8
Race Horses	−0.093	0.962	3.1
Basketball Drill	−0.056	−0.017	1.4
Party Scene	−0.110	0.901	2.7
Race Horses C	−0.055	−0.040	1.3
BQ Mall	−0.097	0.094	2.5
Basketball Drive	0.016	−0.201	0.4
Cactus	−0.038	−0.430	1.2
Kimono	−0.031	−0.365	0.5
Park Scene	−0.036	−0.430	1.2
BQ Terrace	−0.056	−0.018	1.4
Traffic	−0.061	0.224	1.7
People on Street	−0.039	−0.510	1.3
Nebula	−0.101	−0.410	2.4
Steam Locomotive	−0.040	−0.520	1.4
Average	−0.064	0.067	1.68

gives on average a 38.03% time reduction, with a maximum value of 59.74%. For some sequences (like Race Horses), it gives a relatively low time-reduction factor, due to the presence of high motion complexity and rich texture in the sequence.

To evaluate the performance and quality degradation of the proposed algorithm, we have checked the encoded bits and PSNR. We have defined two parameters, $\Delta PSNR$ and ΔBit, to calculate the quality degradation. These two parameters are calculated using the following equations, (9) and (10), respectively:

$$\Delta PSNR = PSNR_{original} - PSNR_{proposed}, \tag{9}$$

$$\Delta Bit = \frac{Bit_{original} - Bit_{proposed}}{Bit_{original}} \times 100\%. \tag{10}$$

The performance degradation in our proposed algorithm is shown in Table 7. In this table, the average value of each sequence is given in the row. Also, the Bjontegaard Delta (BD) rate is shown in Table 7 for each sequence. The BD rate includes both BD-PSNR and BD-Bitrate [22].

For higher-resolution video sequences, the proposed algorithm gives a better result. To justify this, we have tested the same sequence (Race Horses) for different resolutions (416 pixels × 240 pixels and 832 pixels × 480 pixels). The results are given in Table 6 and Table 7. For low resolution (416 × 240), the proposed CST algorithm produces a 22.05% decrease in time with 3.1 BD rate for Race Horses. On the other hand, for higher resolutions (832 × 480), it gives only 1.3 BD rate degradation with a 32.5% time-reduction factor.

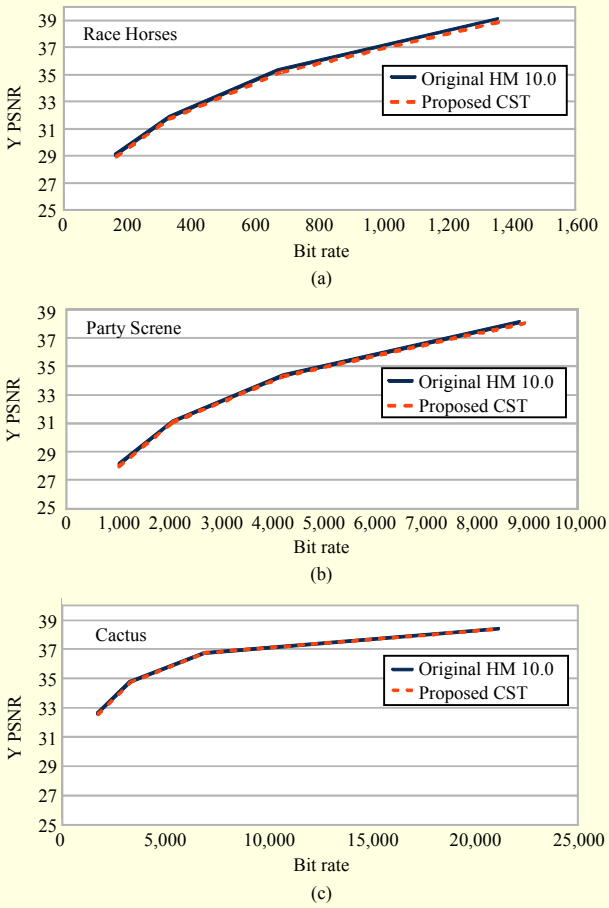


Fig. 6. RD curves: (a) Race Horses, (b) Party Scene, and (c) Cactus sequences.

The RD curves for the three sequences (Race Horses, Party Scene, and Cactus), with different resolutions for our method and the original HM software 10.0, are shown in Fig. 6. Our method achieved performance that is similar to the original HM encoder, especially at a low bit rate. At a high bit rate, our method suffers a small loss in quality.

3. Performance Comparison

We have compared the proposed CST algorithm with ECU [3]. Both of the algorithms are implemented in HM 10.0 under the same test conditions as discussed in section III, subsection 1. In both of the algorithms, we have compared PSNR, number of bits, and time consumption. The parameters which are considered in this comparison are calculated using equations

Table 8. Performance comparison with ECU [3].

Sequence Name	$\Delta PSNR_{ECU}$	ΔBit_{ECU}%	ΔBD_{ECU}	ΔT_{ECU}%
Blowing Bubbles	−0.039	0.566	−1.6	−9.37
BQ Square	−0.058	0.772	−2.1	−8.37
Basketball Pass	−0.015	0.974	−1.3	−8.50
Race Horses	−0.044	1.597	−2.5	−8.70
Basketball Drill	−0.017	0.794	−1.2	−9.93
Party Scene	−0.050	0.842	−2.1	−8.80
Race Horses C	−0.016	0.525	−1.0	−10.57
BQ Mall	−0.039	0.779	−1.9	−8.30
Basketball Drive	0.000	0.108	−0.1	−13.07
Cactus	−0.005	0.124	−0.4	−15.10
Kimono	0.004	0.137	0	−13.07
Park Scene	−0.016	0.432	−0.8	−11.21
BQ Terrace	−0.017	0.573	−0.9	−12.65
Traffic	−0.009	0.624	−0.7	−12.89
People on Street	−0.016	0.741	−0.8	−13.59
Nebula	−0.018	0.834	−1.1	−12.19
Steam Locomotive	−0.009	0.847	−0.7	−11.67
Average	−0.021	0.662	−1.12	−11.05

(11)–(13). In Table 8, the performance comparison is given. Form this result, it is inferred that our proposed algorithm is superior in time reduction over ECU, on average 11% of the time. Apart from that, the quality (PSNR and bit) in both of the algorithms are very similar

$$\Delta PSNR_{ECU} = PSNR_{ECU} - PSNR_{proposed}, \quad (11)$$

$$\Delta Bit_{ECU} = \frac{Bit_{ECU} - Bit_{proposed}}{Bit_{ECU}} \times 100\%, \quad (12)$$

$$\Delta T_{ECU} = \frac{Time_{ECU} - Time_{proposed}}{Time_{proposed}} \times 100\% . \quad (13)$$

IV. Conclusion

We have proposed a new, early CU-splitting termination (CST) algorithm for fast HEVC encoding, based on a ratio function. The RD costs for different CU dimensions and the PU-level motion complexity are also considered in our two-stage algorithm. In the first stage, only the CTU is considered. Other dimensions are considered in the second stage. Our algorithm achieved, on average, a 38.03% time reduction over the original HM 10.0 software, with a 1.68% BD loss.

Table 9. CST combined with other algorithms.

Sequences	CST+[10]		CST+[12]		CST+[13]	
	ΔT%	ΔBD	ΔT%	ΔBD	ΔT%	ΔBD
Blowing Bubbles	−40.38	0.8	−50.91	2.8	−48.41	2.1
BQ Square	−42.12	0.3	−51.22	3.1	−47.53	2.4
Basketball Pass	−33.24	0.8	−45.28	2.4	−42.89	1.8
Race Horses	−22.78	0.9	−41.56	3.4	−31.51	3.0
Basketball Drill	−35.81	0.9	−48.27	1.9	−42.91	1.4
Party Scene	−35.78	1.2	−42.93	2.9	−44.81	2.5
Race Horses C	−30.12	0.8	−41.87	1.8	−38.64	1.7
BQ Mall	−32.12	1.1	−45.78	2.6	−41.78	2.6
Basketball Drive	−39.83	0.1	−51.56	0.8	−46.32	0.7
Cactus	−42.16	0.4	−57.84	1.6	−49.37	1.2
Kimono	−31.53	0.2	−48.78	0.7	−38.72	0.9
Park Scene	−39.11	0.6	−52.91	1.6	−48.61	1.2
BQ Terrace	−35.12	0.6	−49.78	1.5	−43.21	1.3
Traffic	−37.58	0.5	−48.51	1.9	−45.63	1.8
People on Street	−36.15	0.9	−46.21	1.4	−42.67	1.3
Nebula	−31.91	0.6	−49.51	2.9	−38.62	2.6
Steam Locomotive	−31.41	0.4	−51.93	2.1	−41.75	1.5
Average	−35.12	0.65	−48.52	2.08	−43.14	1.76

Compared with ECU [3], our method achieves a better than 11.05% time-reduction factor.

Apart from that, we have combined our algorithm with other related algorithms. There are three algorithms that have been reported which are related with INTER mode decision [10]–[12]. Among them, the performance of the combined algorithms of CST and [11] are not very impressive. Hence in Table 9, we have shown the performance of the combined algorithms CST+[10] and CST+[12]. While we combine the proposed algorithm with [10], it gives a more robust performance in terms of quality loss. In this implementation, we have included the adaptive CU depth-range estimation [10]. Moreover, the early termination techniques proposed in [10] use motion homogeneity; and RD cost–based and SKIP-based checking, which are incorporated here. However, the time-saving factor is degraded in this case, but it gives lower BD loss. On the other hand, [12] and CST give better time reduction with insignificant high-BD loss. In this implementation, we have included the features of coded block flags (CBF) of the INTER 2N × 2N mode (x_{cbf}), the side information in RD cost (x_{si}), the CU depth of the co-located

CU (x_{tp}), the RD cost difference between SKIP and INTER 2N × 2N mode (x_{drc}), and the sum of absolute transformed difference between prediction and original pixel values (x_{std}), which are discussed in [12]. Moreover, we have combined a related work for CU splitting in a INTRA mode decision algorithm [13]. In this implementation, we have considered only the first part of [13], which is related with CU splitting. However, we have implemented in this work the early termination of CU cost calculation, which is discussed in [13] in detail. In Table 9, the performance of CST+[13] is given. This combined algorithm provides a slight improvement in time-saving factor, with similar BD loss.

References

[1] G.J. Sullivan et al., "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Trans. Circuits Syst. Video Technol*., vol. 22, no. 12, Dec. 2012, pp. 1649–1668.

[2] T. Wiegand and G.J. Sullivan, "The H.264/AVC Video Coding Standard," *IEEE Signal Process. Mag*., vol. 24, Mar. 2007, pp. 148–153.

[3] JCT–VC document, JCTVC–F092, *Coding Tree Pruning Based CU Early Termination*, Torino, Italy, July 2011.

[4] K. Choi and E.S. Jang, "Fast Coding Unit Decision Method Based on Coding Tree Pruning for High Efficiency Video Coding," *Opt. Eng. Lett*., no. 51, vol. 3, Mar. 20, 2012.

[5] H.L. Tan et al., "On Fast Coding Tree Block and Mode Decision for High-Efficiency Video Coding (HEVC)," *IEEE Int. Conf. Acoust*., *Speech Signal Process*., Kyoto, Japan, Mar. 25–30, 2012, pp. 825–828.

[6] J. Kim et al., "Adaptive Coding Unit Early Termination Algorithm for HEVC," *IEEE Int. Conf. Consum. Electron*., Las Vegas, NV, USA, Jan. 13–16, 2012, pp. 261–262.

[7] J. Leng et al., "Content Based Hierarchical Fast Coding Unit Decision Algorithm for HEVC," *Int. Conf. Multimedia Signal Process*., Guilin, China, May 14–15, 2011, pp. 56–59.

[8] G.V. Wallendael et al., "Improved Intra Mode Signaling for HEVC," *IEEE Int. Conf. Multimedia Expo*, Barcelona, Spain, July 11–15, 2011, pp. 1–6.

[9] W.J. Chen et al., "Reversed Intra Prediction Based on Chroma Extraction in HEVC," *Int. Symp. Intell. Signal Process. Commun. Syst*., Chiang Mai, Thailand, Dec. 7–9, 2011, pp. 1–5.

[10] L. Shen et al., "An Effective CU Size Decision Method for HEVC Encoders," *IEEE Trans. Multimedia*, vol. 15, no. 2, Feb. 2013, pp. 465–470.

[11] X. Shen, L. Yu, and J. Chen, "Fast Coding Unit Size Selection for HEVC Based on Bayesian Decision Rule," *Picture Coding Symp*., Krakow, Poland, May 7–9, 2012.

[12] X. Shen and L. Yu, "CU Splitting Early Termination Based on Weighted SVM," *EURASIP J. Imag. Video Process*., Jan. 2013.

[13] H. Zhang and Z. Ma, "Early Termination Schemes for Fast Intra Mode Decision in High Efficiency Video Coding," *IEEE Int. Symp. Circuits Syst*., Beijing, China, May 19–23, 2013, pp. 45–48.

[14] G. Tian and S. Goto, "Content Adaptive Prediction Unit Size Decision Algorithm for HEVC Intra Coding," *Picture Coding Symp*., Krakow, Poland, May 7–9, 2012, pp. 405–408.

[15] H. Zhang and Z. Ma, "Fast Intra Prediction for High Efficiency Video Coding," *Advances Multimedia Inf. Process*., Springer, 2012, pp. 568–577.

[16] W. Jiang, H. Ma, and Y. Chen, "Gradient Based Fast Mode Decision Algorithm for Intra Prediction in HEVC," *IEEE Int. Conf. Consum. Electron*., *Commun. Netw*., Hubei, China, Apr. 21–23, 2012.

[17] S.-W. Teng, H.-M. Hang, and Y.-F. Chen, "Fast Mode Decision Algorithm for Residual Quadtree Coding in HEVC," *IEEE Visual Commun. Imag. Process*., Tainan City, Tiwan, Nov. 6–9, 2011, pp. 1–4.

[18] H. Zeng, C. Cai, and K.-K. Ma, "Fast Mode Decision for H.264/AVC Based on Macroblock Motion Activity," *IEEE Trans. Circuits Syst. Video Technol*., vol. 19, no. 4, Apr. 2009, pp. 491–499.

[19] P.I. Hosur and K.K. Ma, "Motion Vector Field Adaptive Fast Motion Estimation," *Int. Conf. Inf*., *Commun. Signal Process*., Sydney, Australia, Nov. 9–11, 1999.

[20] B. Hilmi et al., "Fast Inter-mode Decision Algorithm for H.264/AVC Using Macroblock Correlation and Motion Complexity Analysis," *IEEE Int. Conf. Consum. Electron*., Las Vegas, NV, USA, Jan. 13–16, 2012, pp. 90–91.

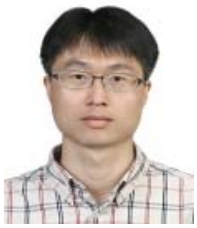
[21] J.-H. Lee et al., "Novel Fast PU Decision Algorithm for the HEVC Video Standard," *IEEE Int. Conf. Imag. Process*., Melbourne, Australia, Sept. 15–18, 2013, pp. 1982–1985.

[22] X. Li, M. Wien, and J.-R. Ohm, "Rate-Complexity-Distortion Evaluation for Hybrid Video Coding," *IEEE Int. Conf. Multimedia Expo*, Suntec City, Singapore, July 19–23, 2010, pp. 685–690.

Kalyan Goswami is a PhD student at the Department of Computer Engineering, Sun Moon University, Asan, Rep. of Korea. He received his BTech degree in electronics and telecommunication engineering from Kalyani University, Calcutta, India, in 2004, and his MS in Advanced Technology Development Center from IIT Kharagpur, India, in 2011. Before joining IIT, he was working as a programmer analyst in Cognizant Technology Solutions, Kolkata, India. His research interests include algorithm development in the field of Video Processing.

Byung-Gyu Kim received his BS degree from Pusan National University, Busan, Rep. of Korea, in 1996 and his MS degree from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Rep. of Korea in 1998. In 2004, he received a PhD degree at the Department of Electrical Engineering and Computer Science from Korea Advanced Institute of Science and Technology (KAIST). In March 2004, he joined in the real-time multimedia research team at the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Rep. of Korea, where he was a senior researcher. In February 2009, he joined the Division of Computer Science and Engineering at Sun Moon University, Asan, Rep. of Korea, where he is currently a professor. In 2007, he served as an editorial board member of the International Journal of Soft Computing, Recent Patents on Signal Processing, Research Journal of Information Technology, Journal of Convergence Information Technology, and Journal of Engineering and Applied Sciences. Also, he is serving as an associate editor of Circuits, Systems and Signal Processing (Springer), International Journal of Image Processing and Visual Communication (IJIPVC), and The Scientific World Journal (Hindawi). He was an organizing committee member of CSIP2013 in Shenzhen, China. He also served as program committee member of CSIP 2011, CUTE2012, EMC 2012, FCC2014, and EMC2014. He has published over 90 international journal and conference papers in his field. His research interests include image and video object segmentation for content-based image coding, wireless multimedia sensor networks, real-time multimedia communication, and intelligent information systems for image signal processing. He is a member of IEEE, ACM, and IEICE.

Dongsan Jun received his BS degree in electrical engineering and computer science from Pusan University, Busan, Rep. of Korea in 2002 and his MS and PhD degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Rep. of Korea in 2004 and 2011, respectively. He has been a senior researcher at Electronics and Telecommunications Research Institute (ETRI), Daejeon, Rep. of Korea, since 2004, and an adjunct professor of the Mobile Communication and Digital Broadcasting Engineering Department at the University of Science and Technology (UST), Daejeon, Rep. of Korea, since 2011. His research interests include image computing systems, pattern recognition, video compression, and realistic broadcasting systems.



Soon-Heung Jung received his BS degree in electronic engineering in 2001 from Pusan National University, Busan, Rep. of Korea. He received his MS degree in electronic engineering in 2003 from Korea Advanced Institute in Science and Technology (KAIST), Daejeon, Rep. of Korea. From 2003 to 2005, he was a research engineer at LG Electronics, Rep. of Korea. Since 2005, he has been a senior member of the engineering staff at ETRI and he is also working toward a PhD in electronic engineering at KAIST, Daejeon, Rep. of Korea. His research interests are in the area of visual communication, video signal processing, video coding, and realistic broadcasting systems.

Jin Soo Choi received his BE, ME, and PhD in electronic engineering from Kyungpook National University, Daegu, Rep. of Korea, in 1990, 1992, and 1996, respectively. Since 1996, he has been a principal member of the engineering staff at ETRI, Daejeon, Rep. of Korea. He has been involved in developing the MPEG-4 codec system, data broadcasting systems, and UDTV. His research interests include visual signal processing and interactive services in the field of digital broadcasting technology.