Machine Learning Based Fast Angular Prediction In Video Coding

Natarajan, A[1],Saravanan, M[2],Sargunam, T[3] and Baskar, A[4]
[1, 2, 3] Dept of Computer Science and Engineering
[4] Associate professor, Dept of Computer Science and Engineering
[1, 2, 3, 4] E.G.S. Pillay Engineering College,Nagapattinam, Tamilnadu, India.

Abstract- *The aim of this project is to present a unified framework for human action and activity recognition.This is used to estimate an intra-prediction mode from a prediction unit and to reduce encoding time significantly by avoiding the intensive Rate-Distortion optimization of a number of intra-prediction modes. The proposed technique is High Efficiency Video Coding Test Model (HM) video coding standard and Joint Exploration Model (JEM) reference software, by integrating the random forest trained off-line into the codecs.To summarize a set of events and to search for particular events because they contain various pieces of context information. Police department in every country has been trying to decrease crime's number, but in fact, the crime's number always grows every year. Instead of catching the criminal, we can minimize the crime by lowering the opportunity of criminal's action, so we propose the system to detect if there is someone crossing (or) ubnormal situation in the area of Closed-circuit Television (CCTV) to detect using image processing.*

Keywords- Fast intra prediction, fast mode decision, machine learning, random forest, HEVC/H.265, HEVC Test Model (HM), Joint Exploration Model (JEM).

I. INTRODUCTION

Ultra high definition (UHD) videos are widely available and affordable, and displays of even higher resolution such as 8K (i.e., 7680 × 4320) UHD videos are emerging with realistic views. However, data volume of the high spatial resolution video increases significantly, which poses challenges in delivering the contents. The state-of-the-art video coding standard, High Efficiency Video Coding (HEVC) [1] which was developed under the efforts of the Joint Collaborative Team on Video Coding (JCT-VC) of the ITU-T Video Coding Expert Group (VCEG) and ISO/IEC Moving Picture Expert Group (MPEG), has been established to encode high definition (HD) or UHD video contents. HEVC provides a substantial coding gain, i.e., approximately a half of bit-rates saving at the same visual quality, as compared to previous video coding standards. Recently, Joint Video Exploration Team (JVET) is created to start exploration of coding technologies beyond HEVC, based on the Joint Exploration Model (JEM) reference software [2], [3]. The JEM model incorporates several advanced coding tools [4], [5] and shows significantly improved coding efficiency i.e., around 24% and 33% BDrate savings, respectively, in HD and 4K UHD test sequences over HEVC [6]. However, the computational complexity of the model is intractable, i.e., around 20 times of all-intra encoding measurement time, as compared to HEVC reference software [7]. The intensive computational complexity increases power consumptions and hardware costs, which obstructs deployments of the techniques to video coding applications [8].

For an intra-coding, the state-of-the-art video coding standards enhance the granularity of spatial intra-prediction. HEVC adopts two non-directional prediction modes (mode 0 for Planar and mode 1 for DC) and 33 angular prediction modes, and more fine-granular angular predictions using 65 modes are offered in the JEM software [4], [5]. However, while the increased number of the intra-prediction modes improves compression efficiency, it requires expensive computational complexity to perform an Rate-Distortion (R-D) optimization, choosing the best mode among many candidates. Thus, many research works are conducted to facilitate the optimization process [9]–[21]. Most of the fast intra-coding techniques are developed using two approaches. In [9]–[15], fast mode decision algorithms determines fewer candidate modes chosen by a rough mode decision (RMD) to avoid exhaustive search in R-D optimization process. In this approach, it is mattered how to elaborately select probable modes to the best mode, while minimizing the number of the candidates. The proposed technique, which aims to choose an angular prediction mode efficiently, also belongs to this category. In the other approach [16]–[21], the fast mode decisions are jointly combined with other optimizations such as recursive quad-tree block partitioning in HEVC and a quad-tree plus binary tree (QTBT) in JEM. In the second approach, the depths of the block partitioning structures are adaptively determined; however the two approaches can be complement to one another rather than to be exclusively used for a fast intra-coding [17]. Our work can be extended to an ingredient of the state-of-the-arts using the joint optimization to provide further improvements in the mode decision.

Machine learning is able to discover effective representations of high dimensional multimedia data, widely used for computer vision and image processing. Motivated by the capability of machine learning, in video coding, several research works have been conducted [22]–[25]. Kang et al. propose two-layered representations of motion-compensated residual signals for a coding [22]. Correa et al. develop a transcoder based on a learning process on decoding attributes [23]. Classification as a conventional machine learning task is applied in speeding up mode decision process. In [18]–[20], [26], [27] the quad-tree block structure of a coding unit (CU) is configured by classifying textures of sub-blocks, and the corresponding sizes of the block partitions are determined. In [28], [29], statistical correlation between properties of contents in blocks and inter-prediction modes is used for a fast mode decision. In [30]–[32], the intra-prediction modes are predicted by training previously coded blocks.

In this paper we propose a machine learning-based fast intra-prediction mode decision algorithm. Specifically, we use a random forest [33], which is an ensemble model of randomized decision trees, to infer an appropriate prediction mode from a prediction unit. To the best of our knowledge, there are very rare fast video coding studies adopting the random forest, even though the random forest represents a fast and robust machine learning technique, thus widely used for computer vision applications [34]–[36]. In this work, we develop efficient split functions learning directional blockbased features in a randomized tree to infer an intra-prediction mode. The estimation is very fast because only few pixels are used for evaluating a prediction mode. The prediction of each randomized tree is combined to make the ensemble in the forest and defeat an overfitting problem. To integrate the proposed algorithm into the recent video coding standard frameworks, the intra-prediction mode derived from the proposed technique, called an inferred mode (IM), is used to shrink the pool of the candidate modes before carrying out the Rate-Distortion (R-D) optimization. The proposed technique is implemented into the reference software of the state-of-the-art video coding standard, and shows significant encoding time reduction with only slight coding loss.

The rest of the paper is organized as follows. In Sec. II, we review related works. In Sec. III some preliminary of random forest is presented. In Sec. IV, we show the proposed technique. Experimental evaluations are carried out in Sec. V. We conclude with remarks in Sec. VI.

II. RELATED WORKS

We categorize the related works into two approaches, 1) the fast intra-prediction mode decision scheme and 2) the optimization of the block partitioning, respectively shown in subsection II-A and subsection II-B. We also show the machine-learning based fast coding techniques in subsection II-C, followed by the contribution of the paper with random forest.

A. Fast Intra-Prediction Mode Decision

HEVC reference software uses a two-step procedure to expedite the intra-prediction mode decision [9]. In the first step, a lower complexity cost function such as the sum of absolute transform difference (SATD) is used to choose the best candidates from all the available intra-prediction modes.
The step is also known as a rough mode decision (RMD). If the number of the candidates does not fulfill a pre-defined number of RMD, another candidates such as most probable modes (MPM) are added into the pool. In the second step, the Rate-Distortion (R-D) optimization that requires higher computational complexity is applied to accurately choose the final intra-prediction mode among the candidates.

There are several fast HEVC intra prediction mode decision algorithms. They focus on collecting more feasible modes to the best mode while reducing the number of the candidates. Shen et al. and Zhao et al. propose neighboring block-based intra-prediction mode decision techniques by exploiting high spatial correlation between adjacent blocks [10], [16]. Jamali et al. apply edge detection to consider relevant modes from neighboring blocks [37]. Zhang et al. develop progressive rough mode searching technique computed with Hadamard cost to check only fewer number of modes [17]. Hu et al. propose an outlier-based fast intra-mode decision with refining an entropy coding [38]. The candidate modes from RMD process are managed before R-D optimization. Zhang et al. study statistical correlation between the RMD candidates and the best mode as a result of R-D optimization [11]. Quanhe et al. adaptively change the order of the candidates from MPM and RMD by taking account of a texture of a block [14]. Liao et al. utilize a depth of a block partition to choose more probable modes and adjust the order of MPM and RMD [39]. Jaballah et al. cluster a set of intra modes into some clusters and choose the candidate for the R-D optimization [40].

Block-based texture analysis is used for a fast intraprediction mode decision. In [12], edge components of blocks are used for restricting available directional prediction modes to reduce the encoding time. In [15], a gradient-based

fast mode decision algorithm that computes a histogram of gradient components is developed for choosing RMD modes. In [13], a pixel-gradient statistics is used for speeding up the computation, and a chosen mode is refined with mode information of neighboring blocks. Furthermore, compressed domain block analysis is performed for an early mode decision. Kim et al. use correlation between an intra-prediction mode of a current block and that of a block in a lower depth to reuse the mode [41]. Motra et al. employ directional components of co-located neighboring blocks in previous frames [42]. Binary Robust Independent Elementary Feature (BRIEF) descriptor is used for an early decision [43].
JEM software adopts several advanced intra-prediction tools [4], [5], [44], [45] at the expense of computational complexity. The number of the directional intra-prediction modes for a luma block increases to 65, including 33 HEVC angular prediction modes and in-between modes by four-tab interpolation filters [4], [5]. The JEM software also uses a coarse-tofine mode decision algorithm similar to the HEVC reference software while it uses two-step RMD process, followed by the R-D optimization. In the first RMD process, among the original 33 modes in HEVC, the SATD is computed to select the first group of candidates whose size is decided by the block size. In the second RMD process, as the JEM software involves more angular prediction modes, the in-between prediction modes that are adjacent to the angular modes of the first group are added into the candidates. Then, the JEM software computes the SATD to choose the second group as in the first RMD. The final candidates are determined after adding several MPMs if they are not redundant to the second group, and then the R-D optimization is applied.

B. Fast Intra-Coding with Optimized Block Partitioning

Optimization of a block structure such as quad-tree based block partitioning in HEVC is also studied. Zhang and Ma [17] propose a fast intra-coding algorithm in two levels. In a coding unit (CU) level, the R-D cost of sub-CUs is compared with R-D cost of current CU to decide whether or not to split the CU, and in a prediction unit (PU) level, the progressive mode selection algorithm is shown to avoid evaluations of all the modes. Shen et al. consider the both CU size and the mode decision simultaneously by using R-D costs and the correlation of prediction modes in different levels of depths in spatially neighboring CUs [16]. The techniques show considerable encoding time reduction because of some short paths to avoid many recursions in the block partitioning, while sometimes significantly degrading a coding gain, incurred from unexpected block patterns.

A quad-tree plus binary tree (QTBT) structure applied to the JEM software allows for a coding unit to have either a square or a rectangular block shape [46]. Specifically, some leaf nodes in a quad-tree can be further divided into a binary tree, representing a horizontal or vertical splitting. The blocks corresponding to the binary tree go through predictions and transforms without any extra partitioning. As the QTBT supports a more flexible block partitioning, several research works are studied for adaptively deciding the depths of the trees. In [47], a constrained QTBT structure where its size and depth are dynamically limited is proposed to speed up the encoding decisions. In addition to the dynamic partition parameters, a joint classifier using information gain attribute evaluation is further used for speeding up the partitioning decision [48].

C. Machine Learning-Based Fast Coding Techniques

Machine learning-based fast coding techniques are also actively studied in past years. In [20], Thanuja et al. present a content adaptive feature-based CU size prediction algorithm for an inter-coding. They use two classifiers, i.e, one for motion feature-based CU classification and the other for R-D cost threshold-based CU classification. In [26], Correa et al. develop fast HEVC encoding based decision trees obtained by the data mining technique to expedite the partitioning process of a coding unit. In [19], Ruiz-Coll et al. develop a fast partitioning algorithm for HEVC intra frame coding by using a tree-structured classifier. The classifiers go through rule-based training of the associated parameters. However, the rule-based training is easily over-fitted. Hu et al. use Neyman-Pearsonbased rule to balance an R-D cost and complexity reduction, and find nonparametric density estimation of a likelihood function to predict associated parameters [28].
Support vector machine (SVM) usually achieves a fast binary classification by obtaining a decision surface in a high dimensional space. Zhang et al. propose a fast coding method to determine a proper CU depth by using a weighted SVM [18]. Mu et al. [21] and Shen et al. [49] develop SVM-based classifiers with features combined with the distortion and the number of encoded bits to determine the depth of a CU and the split of a CU, respectively. In those algorithms, SVM requires linearly separable feature models using a kernel-trick in a highdimensional space since the nature of video data is hardly linear. However, the larger dimension the model has, the more parameters it needs to optimize. Non-linear machine learning algorithms are also applied to fast encoding techniques. In [27] Du et al. use the random forest to make the binary decision when a CU is further split or not. Ryu et al. employ a randomized tree to save the complexity in a mode decision [50]. In [51], Laude and Ostermann apply a deep convolutional neural network (CNN) to a mode decision, where a block is fed through several learnt filters of

convolutional layers and fully connected layers to discriminate an intra-prediction mode.

In this paper, we use the random forest for a fast intraprediction mode decision efficiently to learn block-based directional components for classification and reduce encoder complexity. The random forest shows high generalization capability, employed in many computer vision and image processing research works [34]–[36]. The learning technique fits naturally to a fast mode decision problem in a codec as well, thus used for the proposed technique. 1) the random forest is effective to a multi-class classification problem such as estimation of the best mode among several intra-prediction modes because the leaf nodes in a tree act as estimators. 2) the ensemble of the tree structure allows reliable decisions in nonlinear data. The decision is made with traversing independently trained trees and aggregating the decisions of the individual trees. Thus, the random forest can correct for a behavior of each tree that often over-fits to its own training set. 3) the decision is computationally efficient because it needs only few nodes to be visited in the traverse. Furthermore the decisions of trees can be parallelized with multiple processors in recent hardware, which using multiple instruction multiple data (MIMD) architectures for the speed-up.

III. PRELIMINARY OF THE STANDARD RANDOM FOREST

Random forest F = {Tt} |T| t=1 is an ensemble classifier constructed by a set of randomized decision trees. A randomized binary decision tree Tt(x) classifies a d-dimensional sample vector x ∈ X by traversing left or right down the tree until a leaf node corresponding to the posterior distribution over the classes is reached as shown in Fig. 1. Each internal node in a tree operates a weak classifier to maximize the classification performance in the traverse, recursively. To this aim, a node vi in the tree needs to make an optimal binary decision, achieved by a binary split function:

$$h\phi(x, \phi i) : R\ d \times \Phi \rightarrow \{0, 1\}, \qquad (1)$$

whereϕi denotes the split function parameters associated with the node vi , and Φ is the set of all split function parameters. The outputs 0 and 1 correspond to the left and the right children nodes to be placed after branching, respectively

The split continues until an input sample arrives at a leaf node from the root node. In the classification a sample x is associated with a label y ∈ Y. Thus, the output of the tree is a prediction of x to a target label y. The posterior probability is usually used for measuring the prediction at the leaf node reached by x. However, the prediction of an individual tree is not quite accurate because the tree sometimes shows high variance and suffers from an overfitting problem. Therefore the predictions from the multiple independent trees are combined into an ensembled output of a random forest to provide a more reliable prediction.

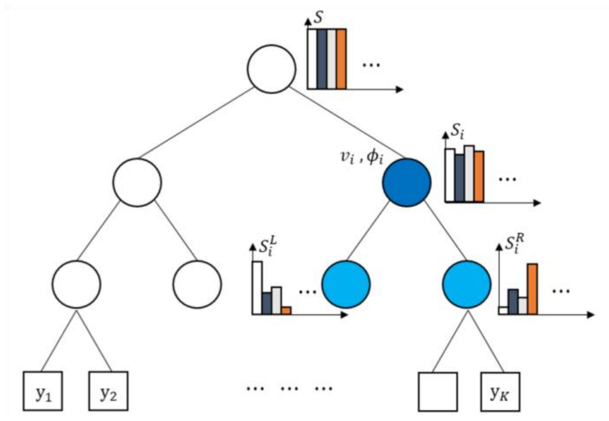


Fig. 1. A structure of a randomized decision tree.

A. Decision Tree Training

Given with a node vi and its training sample set Si ⊂ X × Y, the goal is to obtain parameters ϕi for the split function hϕ(x, ϕi) to achieve the best separation of the classes. The best separation here is measured by the information gain ∆H developed in Information Theory, although more elaborated measurements can be used. The gain is defined as the reduction of uncertainty when Si would be divided into the two subset S L i and S R i in the left and the right child node. Then, the information gain ∆H is mathematically defined as follows:

$$\Delta H = H(Si) - \sum j\in\{L,R\} |S\ j\ i| \ |Si|\ H(S\ j\ i), \qquad (2)$$

where H(S) is the entropy given as − ∑ y py log(py) and py is the probability of samples with a label y in S.

The split process to maximize ∆H is repeated at each internal node, where the split parameters are trained by using the training samples. Fig. 1 shows an example of the split in a node vi . The distributions of the classes are more skewed in the children when ∆H increases. Following the branching mechanism, a tree keeps growing, but the process is finished when a node reaches to a pre-determined depth or it has too few training samples.

B. Testing In testing, an unseen sample x ∈ X traverses the tree down by using the trained split functions with the associated parameters. The prediction is performed when the sample is reached to a leaf node, corresponding to the posterior distribution p(x|yk) over the C class (i.e.,k = 1..C). MaximumA-Posteriori (MAP) is usually used for the estimator, defined as

$$y* = \arg\max_{y_k \in Y} p(y_k|x). \quad (3)$$

Note the final decision is made by collecting the results from all the trees such as a majority voting.

IV. THE PROPOSED TECHNIQUE

A. Intra-Prediction Mode Estimation Using Random Forest

We propose a classification technique using a random forest to infer an intra-prediction direction of a prediction unit (PU) sample. The proposed classifier operates by building multiple independent randomized decision trees during off-line training time and outputting an inferred mode of a block in a codec. A tree Tt in a forest F = {Tt} |T| t=1 is constructed by training a PU sample denoted by x to predict a label y, associated with an intra-prediction mode. 1) Directional Block-based Features: We develop a simple yet effective split function that exploits a block-based feature, reflecting directional characteristics of a block. Fig. 2 shows the four points that are randomly selected in a training sample xi (e.g. an 8 × 8 PU), managed in vi . We randomly choose a set of four points in 2-D coordinates pi = {p 1 i , p 2 i , p 3 i , p 4 i } that are respectively located at each quadrant of xi . The index increases counter-clockwise from the quadrant including the top-left corner of the PU, ranging from 1 to 4. All the three color components including the luminance (Y), and the two color components (U and V) in a PU can be used for selecting the points. However, the luminance component is considered solely because its prediction is more important in an intraprediction, and the choice helps speed up the feature extraction process.

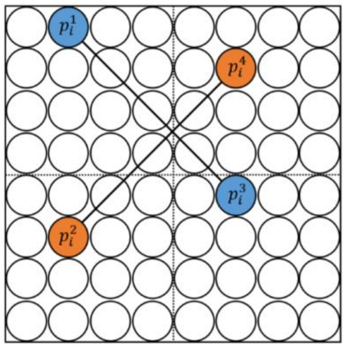


Fig. 2. Four-points that are randomly selected in a 8 × 8 prediction unit, used for directional block-based features.

Our features are the differences between two random points among the selected four points, which is very quickly evaluated. I(p k i) denotes the pixel intensity at a coordinate p k i . Then, given the two indices α and β of the two coordinates, the feature value is defined as

$$f(x_i, p^{\alpha}_i, p^{\beta}_i) = |I(p^{\alpha}_i) - I(p^{\beta}_i)|, \quad (4)$$

where p α i and p β i are the coordinates within xi .

The split functions of the random decision trees use not only the two indices α and β but also the two other indices α ∗ and β ∗ in the remaining two quadrants to complete the directional features. It is more robust to estimate underlying directional properties of blocks. Accordingly, we define the split function incorporating the function parameter set including the four coordinates and τ1 and τ2 to be trained, as follows:

$$h(x_i, \phi_i) = \begin{cases} 0, & f(x_i, p^{\alpha}_i, p^{\beta}_i) < \tau_1 \text{ and } f(x_i, p^{\alpha*}_i, p^{\beta*}_i) < \tau_2 \\ 1, & \text{otherwise}, \end{cases} \quad (5)$$

where the result 0 refers to the left child node, and 1, otherwise. The simple split function can generalize a geometric shape of a block because the function is applied to a series of nodes that are cascaded as weak classifiers. 2) Optimization of a Split Function and Parameters: The procedures of training parameters ϕi in a split function are as follows. Given with a node i, we use a set of training sample Si ⊂ X ×Y, consisting of a prediction unit sample xi ∈ X and a label yi∈ Y that corresponds to an intra-prediction mode in a codec. We choose the optimal parameters ϕ ∗i to maximize the information gain, given as,

$$\phi^*_i = \arg\max_{\phi_i \in \Phi} \{H(S_i) - \sum_{j \in \{L,R\}} \frac{|S^j_i(\phi_i)|}{|S_i|} H(S^j_i(\phi_i))\}, \quad (6)$$

where H(Si) is the entropy value of Si , and H(S L i (ϕi)) and H(S R i (ϕi)) are the entropy values when the samples are divided into the two disjoint subsets S L i (ϕi) and S R i (ϕi) by the parameter ϕi , defined as,

$$S^L_i(\phi_i) = \{S | f(x_i, p^{\alpha}_i, p^{\beta}_i) < \tau_1 \text{ and } f(x_i, p^{\alpha*}_i, p^{\beta*}_i) < \tau_2\}, \quad (7)$$

And

$$S^R_i(\phi_i) = S_i \backslash S^L_i(\phi_i), \quad (8)$$

and a function | . | returns the size of a sample set.

The information gain increases more when a child node contains less diversified classes, thus providing more discriminative capability of the tree. Therefore, the parameters are determined when solving (6). Specifically for updating the threshold parameters τ1 and τ2, we choose the two integer values ranging from 0 to 255 to maximize the gain, when the four coordinate parameters are randomly given. The procedure is conducted four times, and the parameter set to maximize the gain are chosen. When measuring the gain, we use the Gini entropy H(Si) = ∑ k pk(y)(1 − pk(y)) where pk(y) is the proportions of the samples in Si having a label k. The Gini entropy can reduce the computations a bit less than the conventional entropy.

The optimization of the parameter selection in a split function is recursively done at each internal node, using the training prediction units collected in the node. After the optimization, the trained parameters are recorded at the node for the test later. The split is terminated during the training when the depth of the tree exceeds to the pre-determined threshold or when the number of the training blocks are too small. When generating each leaf node vl , the posterior distribution p(y = y ′ |xl) over the set of the prediction mode for a label y ′ ∈ Y is saved, so that the distribution can be used for the decision. The probability is the ratio of the number of samples in class y ′ observed at vl to the total number of samples at vl .

3) Training a Prediction Unit with a Random Forest:

M denotes an entire set of directional intra-prediction modes, i.e., M = {mk} |M| k=1. The size |M| is 33 in HEVC and 65 in Joint Exploration Model (JEM) except for non-directional prediction modes such as Planar and DC modes. As a number of the directional modes increase more, the resolution of the prediction is finer, e.g. 1/32 pixel accuracy in an intraprediction of HEVC. As compared to precision, the resolution of the proposed directional feature is coarse to capture all the underlying directional characteristics of the modes. In a word, the feature is computed using four integer-pels in a PU. Meanwhile, an extraction of a sub-pel based feature may require much computational complexity. Therefore, to resolve the problem, we define a mapping function Q : M → Y to categorize the possible modes into the feasible output space Y as shown in Table I. In the function, several modes in M are quantized into a label y ∈ Y. The modes whose directions are similar to one another are grouped in y, so that the output spaces are eqi-variant. The modes of a group in y have each representative mode yˆ as a median in the modes of HEVC, as shown in Table I. In JEM software, the adjacent angular prediction modes are further included to each class.

TABLE I
THE LABEL y (AND THE REPRESENTATIVE MODE yˆ) IN THE GROUP OF INTRA-PREDICTION MODES IN HEVC. PLANAR (MODE 0) AND DC (MODE 1) ARE EXCLUDED

y ($\hat{y}$)	1 (3)	2 (6)	3 (10)	4 (14)	5 (18)
Prediction mode	2~4	5~7	8~12	13~15	16~20
y ($\hat{y}$)	6 (22)	7 (26)	8 (30)	9 (33)	-
Prediction mode	21 2	24 2	29 31	32 34	-

The procedure to train a prediction unit is shown in Fig. 3. A training sample (xi ,yi) is generated by a codec. Specifically, a decoder produces an intra-prediction mode mi of a prediction unit xi . The mode is mapped to a label yi by Q, and the label is tagged to xi . Then, the training is conducted in a supervised learning manner. For example, the prediction unit samples (xi , ma i), a = 24 ∼ 28 in Fig. 3 choose a label yi = 7 according to Table I to create training samples (xi , 7). The samples traverse into a randomized tree to optimize the parameters of the split functions at nodes, as presented in the previous subsections. The training samples fall into leaf nodes after the traversing. As a result, each leaf node vl presents a posterior probability p(yi |xi) to determine the representative mode, e.g. yˆ = 26. Once the individual trees are trained, they are ensembled by injecting randomness, known as bagging [33], to create a random forest.

We use a block-adaptive training technique. In other words, different random forests are trained with the sizes of their input blocks. HEVC provides 4 × 4, 8 × 8, 16 × 16, and 32 × 32 block-sized PUs in the intra-prediction, and, therefore there are

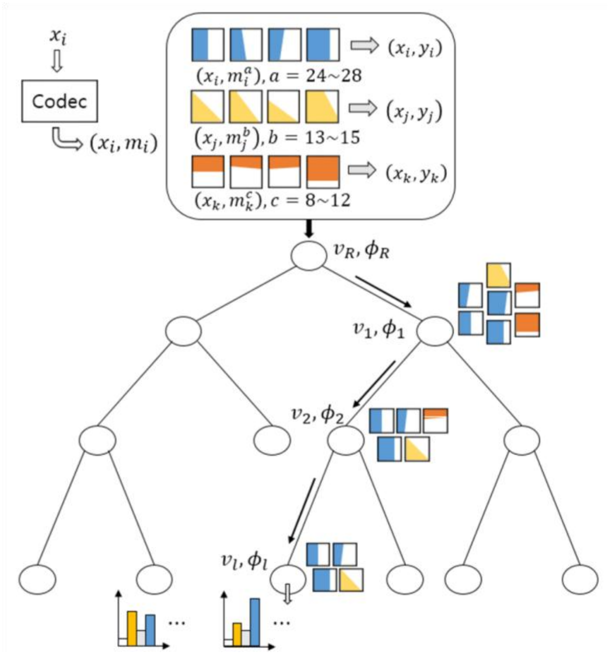


Fig. 3. Training process of a prediction unit using a random forest. A prediction unit is first encoded to create a training sample (xi, yi). A leaf node has a posterior probability of a representative mode yˆi after the training.

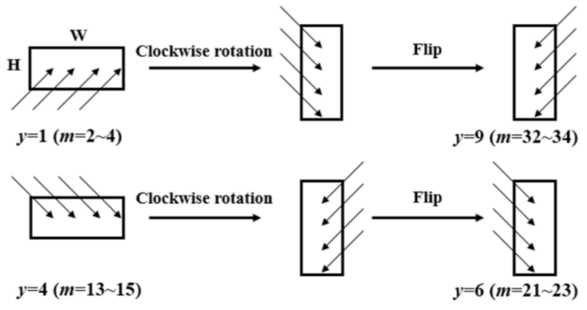


Fig. 4. Training random forests in QTBT partitioning structures.

four different random forests stored in a codec to predict the prediction representative modes. The JEM software adopts a quad-tree plus binary tree (QTBT) block partitioning structure of a coding unit (CU) whose width and the height can vary from 4 to 32 in the software configuration [2], so the blockbased design becomes complicated. To tackle the problem, we reduce the number of the random forests by using the symmetric properties of the blocks in geometry. We train only the blocks whose widths are greater than or equal to the heights and apply the same trained classifiers to the other blocks in certain prediction directions, obtained from rotations and flips of the original directions. Specifically, the directional modes are derived from 90-degree clockwise rotation and horizontal flip operations as shown in Fig. 4. For instance, the random forest trained to predict an index y = 1 in W ×H is also used for the classification of H × W blocks with an index y = 9.

4) Decision of a Prediction Mode with a Random Forest:

In the decision or classification, a prediction unit passes down each independent randomized tree from the root node to a leaf node. The prediction unit is evaluated at internal nodes by performing a series of the split functions with the corresponding trained parameters, and it keeps traversing either to the left or the right child. Fig.5 shows an example of the decision process for a prediction unit in the proposed technique. A prediction unit x falls into leaf nodes presenting conditional posterior probabilities of classes after going through nodes v with the parameters ϕ. It is highlighted that the computational complexity can be significantly reduced in the decision process, owing to the tree structures. If having K leaf nodes in a tree, the decision requires only the ⌈log2 K⌉ nodes to be tested in a balanced tree. Because four points in a prediction unit are evaluated at each test and the features are extracted using simple mathematical calculations as in (4), the overall computational complexity is very low

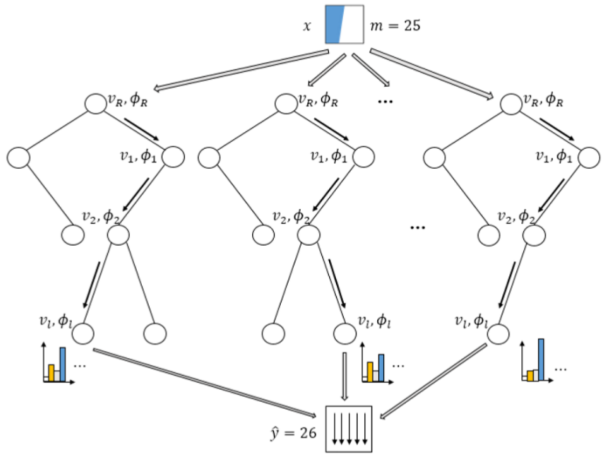


Fig. 5. Decision process of a prediction unit using a random forest. A prediction unit falls into leaf nodes representing posterior probabilities of representative modes, and the results are ensembled.

A random forest yields prediction results by aggregating the outputs of a number of decision trees, as shown in Fig.5. Specifically, the proposed technique uses the sum of the posterior probability obtained from a leaf node of each tree to determine the representative mode. Assuming the number of the trees is N and the sum in a mode yk is larger than N 2 , it will be certainly chosen for the representative mode. However, the number of the trees used for the decision becomes larger as the probability exceeds the threshold, which requires more computational complexities. Therefore, in the proposed technique, we use a parameter γ to control the trade-off between the confidence level to choose yk in the early termination and the complexity. The early termination to choose yk is conditioned as follows:

$$\sum Nci=1\ p(y\ i\ k\ |x) \geq \gamma\ N\ 2\ ,\ (9)$$

where N is the total number of the trees in the forest and Nc is the number of the trees that completed the decision. In our experiments, γ is set to 0.8, which has been empirically found. For example, in Fig.5, the sum of the probabilities of the vertical mode (y = 7) becomes dominant in only few trees, so the final decision is made to the vertical mode immediately.

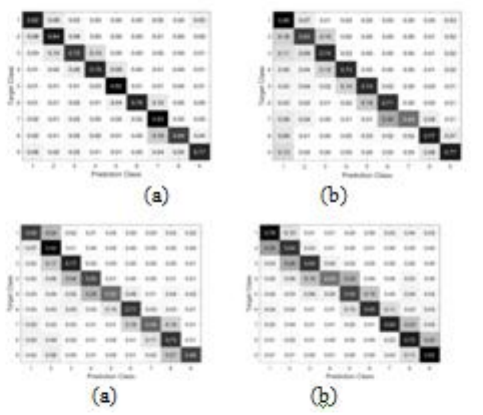


Fig. 6. Confusion matrices of (a) 4 × 4 PU, (b) 8 × 8 PU, (c) 16 × 16 PU, and (d) 32 × 32 PU.

We show detailed procedures in training and testing. We use “Vidyio1”, “Vidyo3”, “Vidyo4”, “Kendo”, “Baloon” for training videos and “Kimono”, “BasketballDrive”, and “Cactus” for test videos. Video frames are randomly chosen in a sequence to disallow similar block patterns observed in adjacent frames as many as possible. The encoder is configured to run with the full R-D optimization, examining the entire intra-prediction modes in the optimization, and then the decoder collects the PU samples with labels (i.e., the best mode chosen by an encoder) as actual classes. Furthermore, we apply preprocessing to avoid duplication of data samples and make the samples with different block patterns equally important in training. For this, we use K-medoids algorithm in [40] to cluster the PU samples belonging to the same group in Table I, and then choose the same number of training samples in each cluster. The number of the clusters is set to the prediction modes of each group shown in Table I. K-fold cross validation is also applied to training data sets to avoid an overfitting problem. In training, we use more than 90,000, 35,000, 20,000, and 10,000 PU samples, respectively for 4×4, 8×8, 16×16, and 32×32 block-sizes and, in testing, around 30% PU samples of the training samples. The testing samples are unseen in the training procedure.

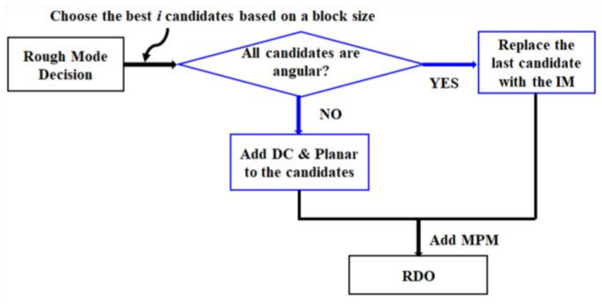
The classification accuracy is estimated with comparing the actual modes and the inferred modes from the random forest. To obtain the classification results, unseen PU samples decoded from the test videos go through the trained classifier. The test videos are coded with an encoder configured with using RMDs, as in the practical scenarios. The same test codec is used for evaluating the performance of the proposed technique in Section V. The classification performance to estimate an intra-prediction mode is quantitatively evaluated as shown in Fig 6, where the columns and the rows of the matrices represent the instances in a predicted class and in an actual class, respectively. The overall classification accuracies are 81.0%, 72.1%, 68.6%, and 66.4% for 4×4, 8×8, 16×16, and 32 × 32 PUs, respectively. The accuracy degrades with a block size because it needs more feature points to describe fine-grained directional patterns of large blocks. We observe enhanced classification performance with more feature coordinates yet higher computational complexity. We empirically determine the same number of features (i.e. the four points) by observing the trade-off. Beside to the classification accuracies, the misclassified samples are frequently observed in adjacent prediction angles, while small errors are evenly distributed at the distant angles. For example, the proposed technique often misclassifies samples in class 9 with those in class 1 and class 8 because of the similar directions, which might be used for an alternative prediction mode.

B. Implementation to a Codec : Fast Intra-Prediction Mode Decision Using a Random Forest

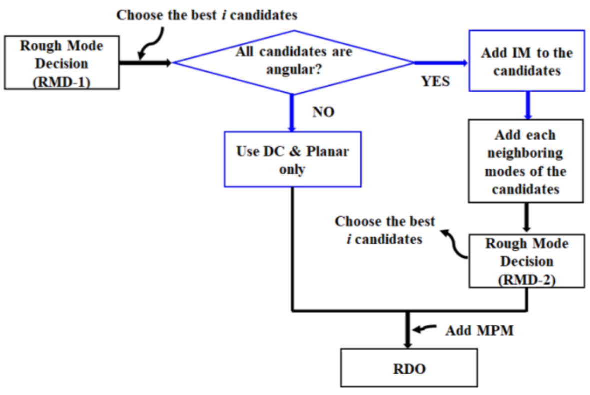
We define an inferred mode (IM) as the representative mode computed from the proposed technique and use it in the conventional mode selection process. The key idea is not only to reduce several prediction mode candidates after RMD for alleviating the complexity but also to add the IM mode for reducing the loss of coding efficiency as much as possible. As a result, the total number of the candidate modes becomes smaller to decrease the overall encoding time with slight coding loss, owing to the IM. We integrate the trained classifier into the same codecs that we have used for learning.
We show more detailed implementation of the proposed technique, which is colored with the blue in Fig. 7(a), based on a HEVC reference model (HM) [52]. The RMD process chooses 3 and 1 prediction mode candidates using the SATD costs for 4 × 4 or 8 × 8 PUs and 16 × 16 or 32 × 32 PUs, respectively, whereas the numbers are 8 and 3 in the original reference software. It is highlighted that the IM is developed for inferring angular prediction modes. Furthermore, we observe Planar and DC modes tend to be chosen when they appear after RMD. Hence, if the prediction mode candidates are only the angular modes, we replace the mode with the

largest SATD among the candidates with the IM. Otherwise, we add both the Planar and DC modes to the candidates. Later, we append the MPM to the candidates if they are not redundant, as in the HM. Lastly, the final mode is determined through the Rate-Distortion optimization.

The proposed technique is used for JEM software similar to HM software, as shown in Fig. 7(b). The JEM software adopts a coarse-to-fine decision process, extended from the HM, where it uses two RMD processes denoted by RMD-1 and RMD-2 in Fig. 7(b). The proposed technique sets the number of the prediction mode candidates remaining after each RMD to 2. When neither DC or Planar modes are after the first RMD, the proposed technique adds the IM to the candidates. The original HEVC prediction modes are considered in RMD-1. However, the additional angular prediction modes neighboring with all of the candidates are evaluated in RMD-2, so that


Choose the best *i* candidates based on a block size
Rough Mode Decision
All candidates are angular?
YES
Replace the last candidate with the IM
NO
Add DC & Planar to the candidates
Add MPM
RDO

(a)


Choose the best *i* candidates
Rough Mode Decision (RMD-1)
All candidates are angular?
YES
Add IM to the candidates
NO
Use DC & Planar only
Add each neighboring modes of the candidates
Choose the best *i* candidates
Rough Mode Decision (RMD-2)
Add MPM
RDO

(a)

Fig. 7. Implementations of the proposed technique (a) in the HM codec and (b) in the JEM codec. The blocks colored with a blue reflect the proposed technique.

the codec tests only the small number of modes among 65 prediction modes.

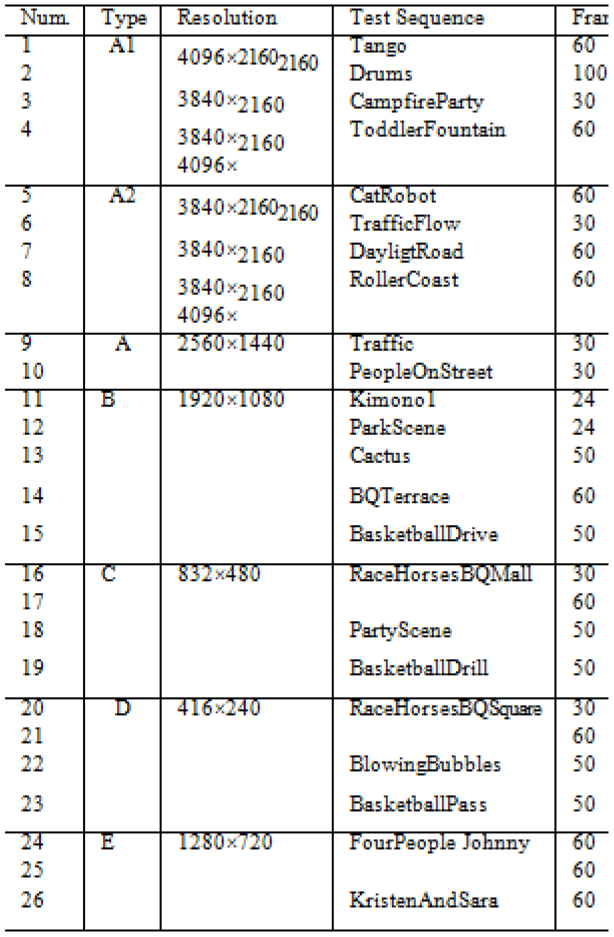
V. EXPERIMENTAL RESULTS

A. Coding Configurations and Test Sequences

The implementations of the proposed technique are based on the recent HEVC reference software, i.e., HM version 16.6 [52] and the JEM reference software version 5.0 [2]. The experiments are performed on PCs with 3.40 GHz Intel CPU and 8.0 GB RAM. The coding configuration of the reference software is set to All-Intra coding, in which all the frames are coded as I frames. Experimental conditions are aligned with the common test conditions recommended by the JCTVC [53] and JVET [54]. Test video sequences used for the evaluation of the proposed technique are shown in Table II. The sequences have various resolutions with their types such as A2 ~ E, as defined in the test conditions [53], [54].

We build the block-based random forest to classify an intraprediction mode from each PU. For this, we use open-source random forest software implemented with C++ [55], integrated to the codecs. The classifiers trained off-line are integrated into an encoder. A number of sample blocks are used for training to avoid over-fitting problems. The number of block samples is

TABLE II TEST SEQUENCES AND PROPERTIES



Num.	Type	Resolution	Test Sequence	Fra
1	A1	4096×21602160	Tango	60
2			Drums	100
3		3840×2160	CampfireParty	30
4		3840×2160 4096×	ToddlerFountain	60
5	A2	3840×21602160	CatRobot	60
6			TrafficFlow	30
7		3840×2160	DayligtRoad	60
8		3840×2160 4096×	RollerCoast	60
9	A	2560×1440	Traffic	30
10			PeopleOnStreet	30
11	B	1920×1080	Kimono1	24
12			ParkScene	24
13			Cactus	50
14			BQTerrace	60
15			BasketballDrive	50
16	C	832×480	RaceHorsesBQMall	30
17				60
18			PartyScene	50
19			BasketballDrill	50
20	D	416×240	RaceHorsesBQSquare	30
21				60
22			BlowingBubbles	50
23			BasketballPass	50
24	E	1280×720	FourPeople Johnny	60
25				60
26			KristenAndSara	60

around 150,000. It is noted that the computational complexity is of less concern during the off-line training.

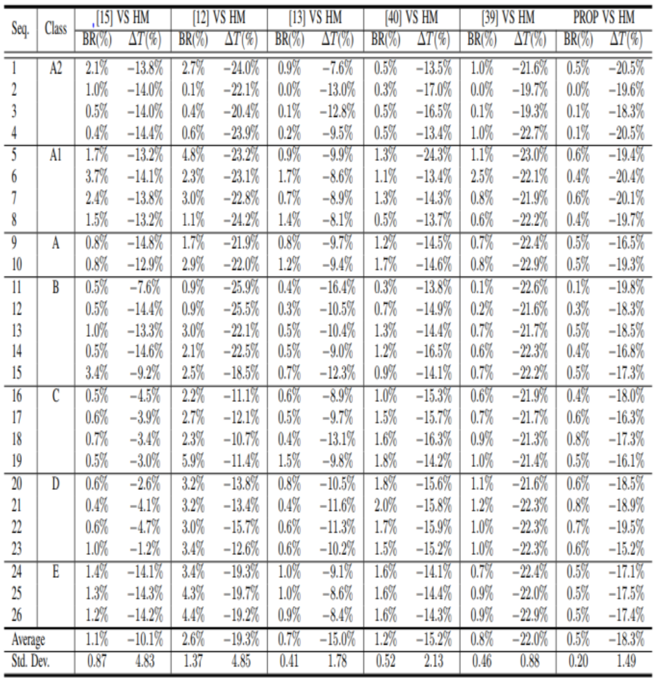
B. Coding Performance Evaluation and Analysis

1) Performance Evaluation in HEVC Reference Software:
literature [12], [13], [15]. Those algorithms try to reduce encoding time by carefully choosing the mode candidates based on edge and texture information of prediction units before carrying out the R-D optimization. The compared algorithms are implemented on the same reference software as an anchor. The Bjontegaard-Delta rate (BR) savings (or loss) are used for measuring the R-D performance, and the encoding time reduction ∆T is calculated as follows:

$$\Delta T(\%) = Ttest - Tori\ Tori \times 100, (10)$$

where Tori is the encoding time of the anchor, and Ttest is the encoding time of the test algorithm. We repeat the same experiments five times and use the averaged measurement time to show the results. The positive numbers in BR and the negative numbers in ∆T refer to coding loss and encoding time reduction, respectively. Table III shows the performance of the proposed algorithm and the conventional algorithms, as compared to HM 16.6. By comparisons in Table III, the proposed technique outperforms the conventional algorithms when considering the tradeoff between coding loss and encoding time reduction. The proposed technique denoted by "PROP" shows the BD-rate increment about 0.5% in the luminance component and the

TABLE III
THE BD-RATE (BR) INCREMENT AND THE ENCODING TIME REDUCTION OF THE TEST ALGORITHMS AS VERSUS HM16.6 [52] IN AI CODING CONFIGURATION



Seq.	Class	[15] VS HM		[12] VS HM		[13] VS HM		[40] VS HM		[39] VS HM		PROP VS HM	
		BR(%)	$\Delta T(\%)$	BR(%)	$\Delta T(\%)$	BR(%)	$\Delta T(\%)$	BR(%)	$\Delta T(\%)$	BR(%)	$\Delta T(\%)$	BR(%)	$\Delta T(\%)$
1	A2	2.1%	−13.8%	2.7%	−24.0%	0.9%	−7.6%	0.5%	−13.5%	1.0%	−21.6%	0.5%	−20.5%
2		1.0%	−14.0%	0.1%	−22.1%	0.0%	−13.0%	0.3%	−17.0%	0.0%	−19.7%	0.0%	−19.6%
3		0.5%	−14.0%	0.4%	−20.4%	0.1%	−12.8%	0.5%	−16.5%	0.1%	−19.3%	0.1%	−18.3%
4		0.4%	−14.4%	0.6%	−23.9%	0.2%	−9.5%	0.5%	−13.4%	1.0%	−22.7%	0.1%	−20.5%
5	A1	1.7%	−13.2%	4.8%	−23.2%	0.9%	−9.9%	1.3%	−24.3%	1.1%	−23.0%	0.6%	−19.4%
6		3.7%	−14.1%	2.3%	−23.1%	1.7%	−8.6%	1.1%	−13.4%	2.5%	−22.1%	0.4%	−20.4%
7		2.4%	−13.8%	3.0%	−22.8%	0.7%	−8.9%	1.3%	−14.3%	0.8%	−21.9%	0.6%	−20.1%
8		1.5%	−13.2%	1.1%	−24.2%	1.4%	−8.1%	0.5%	−13.7%	0.6%	−22.2%	0.4%	−19.7%
9	A	0.8%	−14.8%	1.7%	−21.9%	0.8%	−9.7%	1.2%	−14.5%	0.7%	−22.4%	0.5%	−16.5%
10		0.8%	−12.9%	2.9%	−22.0%	1.2%	−9.4%	1.7%	−14.6%	0.8%	−22.9%	0.5%	−19.3%
11	B	0.5%	−7.6%	0.9%	−25.9%	0.4%	−16.4%	0.3%	−13.8%	0.1%	−22.6%	0.1%	−19.8%
12		0.5%	−14.4%	0.9%	−25.5%	0.3%	−10.5%	0.7%	−14.9%	0.2%	−21.6%	0.3%	−18.3%
13		1.0%	−13.3%	3.0%	−22.1%	0.5%	−10.4%	1.3%	−14.4%	0.7%	−21.7%	0.5%	−18.5%
14		0.5%	−14.6%	2.1%	−22.5%	0.5%	−9.0%	1.2%	−16.5%	0.6%	−22.3%	0.4%	−16.8%
15		3.4%	−9.2%	2.5%	−18.5%	0.7%	−12.3%	0.9%	−14.1%	0.7%	−22.2%	0.5%	−17.3%
16	C	0.5%	−4.5%	2.2%	−11.1%	0.6%	−8.9%	1.0%	−15.3%	0.6%	−21.9%	0.4%	−18.0%
17		0.6%	−3.9%	2.7%	−12.1%	0.5%	−9.7%	1.5%	−15.7%	0.7%	−21.7%	0.6%	−16.3%
18		0.7%	−3.4%	2.3%	−10.7%	0.4%	−13.1%	1.6%	−16.3%	0.9%	−21.3%	0.8%	−17.3%
19		0.5%	−3.0%	5.9%	−11.4%	1.5%	−9.8%	1.8%	−14.2%	1.0%	−21.4%	0.5%	−16.1%
20	D	0.6%	−2.6%	3.2%	−13.8%	0.8%	−10.5%	1.8%	−15.6%	1.1%	−21.6%	0.6%	−18.5%
21		0.4%	−4.1%	3.2%	−13.4%	0.4%	−11.6%	2.0%	−15.8%	1.2%	−22.3%	0.8%	−18.9%
22		0.6%	−4.7%	3.0%	−15.7%	0.6%	−11.3%	1.7%	−15.9%	1.0%	−22.3%	0.7%	−19.5%
23		1.0%	−1.2%	3.4%	−12.6%	0.6%	−10.2%	1.5%	−15.2%	1.0%	−22.3%	0.6%	−15.2%
24	E	1.4%	−14.1%	3.4%	−19.3%	1.0%	−9.1%	1.6%	−14.1%	0.7%	−22.4%	0.5%	−17.1%
25		1.3%	−14.3%	4.3%	−19.7%	1.0%	−8.6%	1.6%	−14.4%	0.9%	−22.0%	0.5%	−17.5%
26		1.2%	−14.2%	4.4%	−19.2%	0.9%	−8.4%	1.6%	−14.3%	0.9%	−22.9%	0.5%	−17.4%
Average		1.1%	−10.1%	2.6%	−19.3%	0.7%	−15.0%	1.2%	−15.2%	0.8%	−22.0%	0.5%	−18.3%
Std. Dev.		0.87	4.83	1.37	4.85	0.41	1.78	0.52	2.13	0.46	0.88	0.20	1.49

encoding time saving about 18.3% on average. Slight coding loss about 0.2% is observed in chroma components as the proposed technique is applied to only the Y-component. However, Jiang's algorithm [15], Da Silva's algorithm [12], Chen's algorithm [13], and Jaballah's algorithm [40] show the BDrate increments about 1.1%, 2.6%, 0.7%, and 1.2% and the encoding time savings about −10.1% , −19.3%, −15.0% and −15.2% on average. Jiang's algorithm and Chen's algorithm yield comparable results in BR with the proposed technique, but the encoding time savings are worse than the proposed technique. In contrast, Da Silva's algorithm and Jaballah's algorithm yield comparable encoding time saving, but the coding loss is significantly degraded. Jaballah's algorithm chooses a subset of RMDs by using K-medoid clustering, while the clustering requires more computational complexity than the random forest. Liao's algorithm [39] show comparable results, which are more encoding time reduction i.e., about −22% but less coding efficiency about i.e., 0.8% than the proposed technique.

We show the robustness of the proposed technique in various configurations and test sequences. For this, we present the standard derivation of the performance in the last row. As shown, the proposed technique provides the lowest standard deviation in BR. The results demonstrate the capability of the proposed technique to adapt to different test sequences. However, the features used for representing edges and gradients in the compared algorithms have difficulties in the adaptation, so the deviation becomes large among the sequences. The proposed technique is also evaluated with various resolutions of test sequences A2 ∼ E, and it shows fairly reliable performance with the factor. Liao's algorithm provides the lowest standard deviation of the encoding time reductions as it reduces the consistent number of the mode candidates with a block depth. However, the standard deviation of the coding performance is large in various video sequences, as compared to the proposed technique

Some commercial HEVC codecs often reduce depths of residual quad-tree (RQT) for speed-up. When the RQT depth is restricted to 1, we observe the proposed technique provides 29.0% encoder time reduction and 1.1% BD-rates increment as compared to the anchor. We show the two R-D curves of the proposed technique and the anchor to examine the R-D performance in different bit-rates and to see the efficiency of the proposed technique during the mode selection. Higher QPs represent lower bitrates, and vice versa. The six sequences such as "Tango," "PartyScene," "KristenAndSara," "CatRobot," "BlowingBubbles," and "BQSquare" yielding less coding efficiency than the other test sequences are used for presenting the R-D curves in Fig. 8 to see variations, if any. The Y-PSNR values are zoomed in different ranges of bit-rates. However, as can be seen, the amount of the loss is likely even in the ranges, and the differences are very small. The observation can be explained with the behaviors of mode selection in R-D optimization and the associated roles of the proposed technique. Typically, the prediction modes are differently chosen in various ranges of bit-rates, considering the minimization of the Lagrangian cost J. The cost J is defined as D + λR where D is the

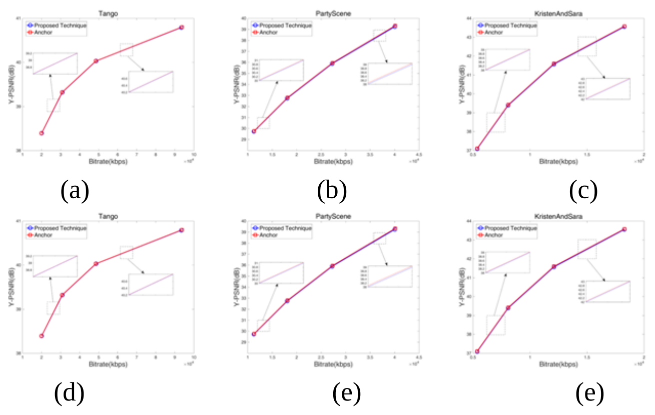


Fig. 8. Rate-Distortion Curves of the proposed technique and the HM software in (a) “Tango,” (b) “PartyScene,” (c) “KristenAndSara,” (d) “CatRobot,” (e) “BlowingBubbles,” and (f) “BQSquare”

distortion and R is the bits to encode a prediction unit. λ is the penalizing term that increases exponentially with an increment of QP. In lower bit-rates, the modes belonging to MPM in the neighboring PUs are preferred since they can save the overhead bits. The proposed technique derives a directional mode by exploiting a pattern in a block, which can be also repeated in the neighbors. In higher bit-rates, λ becomes smaller as the corresponding QP decreases, and the codec is likely to choose a prediction mode that can faithfully represent a pattern of a block to reduce the distortion. Thus, the representative candidate chosen by the proposed technique is important to support the choice. In sum, even though the inferred mode replaces several existing modes to reduce the encoding time, the R-D curves show only the slight changes in the wide ranges of bit-rates. The results tell the efficiency of proposed technique in the mode selection.

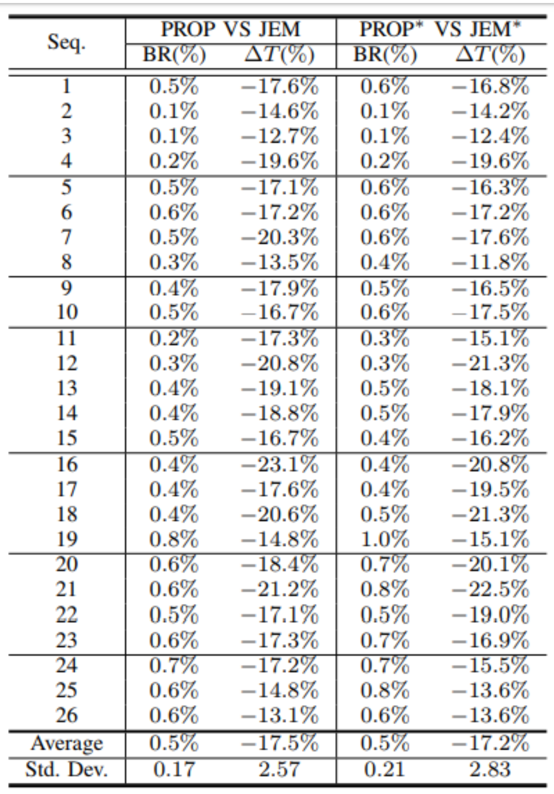
2) Performance Evaluation in JEM software:

We also evaluate the performance of the proposed technique as compared to the JEM software. As shown in Table IV, the proposed technique provides similar trade-off between coding loss and encoding time reduction to that in HM software. The proposed technique denoted by “PROP” shows the BD-rate increment about 0.5% in the luminance component and the encoding time saving about 17.5% on average, as compared to the anchor. There are also changes of coding gains about −0.2% in chroma components as in HEVC. QTBT can affect the JEM software significantly. Thus, we examine the performance when disabling the QTBT both in the proposed technique and the anchor, respectively denoted by “PROP∗ ” and “JEM∗ ” in Table IV. As shown, the proposed technique provides the similar coding loss about 0.5% in the luminance component and encoding time reduction about 17.2% as in the other configurations, and there are slight coding loss about 0.4% in chroma components. The standard deviation in BR is also similar to that in turning on the QTBT. The results show that the performance of the proposed technique is reliable to the use of the QTBT.

3) Performance Analysis with Proportions of DC/Planar Modes:

The proposed technique manages the directional prediction modes and DC/Planar modes separately, by using the two branches in Fig. 7. It is noted that the encoding measurement times are almost same regardless of which branch is used for the decision. The size of the initial candidates before the RMD is 3 in smaller block sizes and 1 in larger block sizes. Therefore, for instance in small block sizes, the left branch examines 3 or 4 modes (3 modes when the both DC and Planar are originally in the list after RMD and 4 modes when only one of DC and Planar is originally in the list). In the other hand, the right branch always examines 3 modes. Thus, the number of the modes in the left branch is rather greater than that in the right branch on average. Actually, we observe the both branches almost equally contribute the acceleration or the right branch requires slightly increasing complexity because of

TABLE IV
THE BD-RATE INCREMENT AND THE ENCODING TIME REDUCTION OF THE TEST ALGORITHMS VS JEM5.0 [2] IN AI CODING CONFIGURATION. PROP∗ AND JEM∗ REFER TO THE CODEC TURNING OFF THE QTBT.



Seq.	PROP VS JEM		PROP* VS JEM*	
	BR(%)	$\Delta T(\%)$	BR(%)	$\Delta T(\%)$
1	0.5%	−17.6%	0.6%	−16.8%
2	0.1%	−14.6%	0.1%	−14.2%
3	0.1%	−12.7%	0.1%	−12.4%
4	0.2%	−19.6%	0.2%	−19.6%
5	0.5%	−17.1%	0.6%	−16.3%
6	0.6%	−17.2%	0.6%	−17.2%
7	0.5%	−20.3%	0.6%	−17.6%
8	0.3%	−13.5%	0.4%	−11.8%
9	0.4%	−17.9%	0.5%	−16.5%
10	0.5%	−16.7%	0.6%	−17.5%
11	0.2%	−17.3%	0.3%	−15.1%
12	0.3%	−20.8%	0.3%	−21.3%
13	0.4%	−19.1%	0.5%	−18.1%
14	0.4%	−18.8%	0.5%	−17.9%
15	0.5%	−16.7%	0.4%	−16.2%
16	0.4%	−23.1%	0.4%	−20.8%
17	0.4%	−17.6%	0.4%	−19.5%
18	0.4%	−20.6%	0.5%	−21.3%
19	0.8%	−14.8%	1.0%	−15.1%
20	0.6%	−18.4%	0.7%	−20.1%
21	0.6%	−21.2%	0.8%	−22.5%
22	0.5%	−17.1%	0.5%	−19.0%
23	0.6%	−17.3%	0.7%	−16.9%
24	0.7%	−17.2%	0.7%	−15.5%
25	0.6%	−14.8%	0.8%	−13.6%
26	0.6%	−13.1%	0.6%	−13.6%
Average	0.5%	−17.5%	0.5%	−17.2%
Std. Dev.	0.17	2.57	0.21	2.83

executing the random forest.

Table V presents the BD-rate increments and the encoding time reductions of the proposed algorithm with proportions of DC and Planar modes observed right after the RMD. The ratios of the DC/Planar modes are mostly determined with video characteristics, and the natural videos used in the common test conditions have 27% ∼ 64% DC and Planar modes in the decision. For example, “BasketBallPass,” “Drums,” and “PartyScene” show DC/Planar ratios less than 35%, while “Kimono,” “Tango,” and “ParkScene,” show DC/Planar ratios more than 60%. We categorize the test sequences into four groups, i.e., G1∼ G4 with the proportions of the DC and Planar modes after the RMD. As shown, the BD-rates are slightly different as 0.4% ∼ 0.5% in the groups whereas the encoding time reductions increase more with the larger percentages of the DC and Planar modes, which caused by executing the random forest. That is, more angular prediction modes, more IM modes. The encoding time varies with the groups about 2.2% on average. We observe similar behaviors in the JEM software.

4) Performance Analysis with the IM mode:

While the branching mechanism facilitates the mode decision process, the IM mode aims to keeps the R-D performance in the proposed technique. To examine how the IM mode affects the performance, we turn off the derivation process of the IM mode and observe the changes of the R-D performance. In other words, the replacements of the IM modes in Fig.7 (a) is skipped while the original prediction modes are maintained in

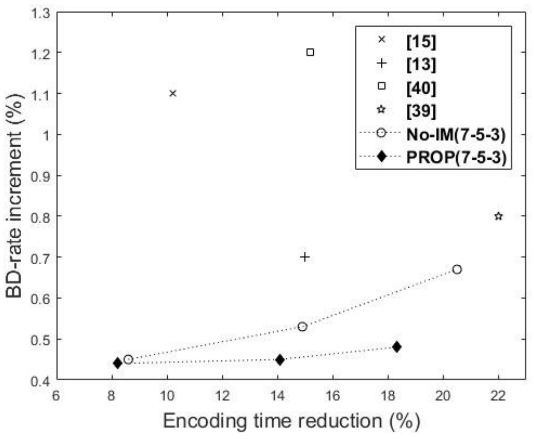


Fig. 9. BD-rate and Complexity comparisons with respect the number of the candidates and IM modes.

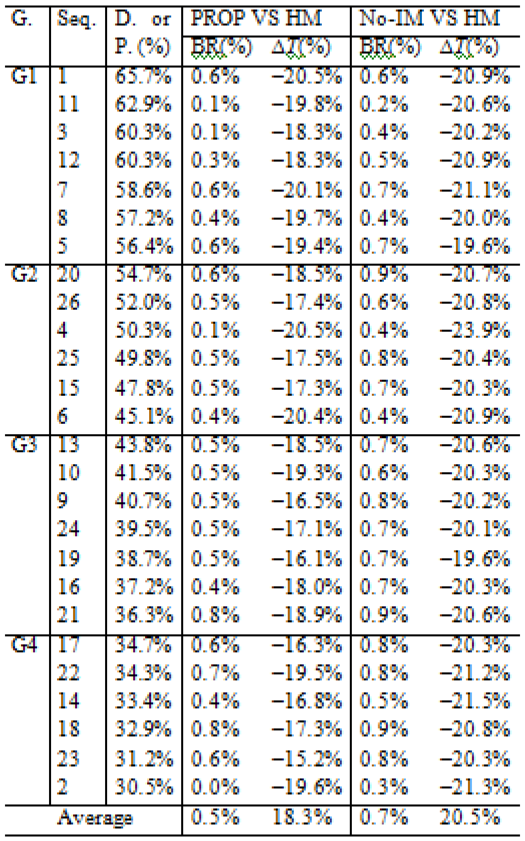
the candidate list. “No-IM” in Table V shows the experimental results of the BD-rate increments and the encoding time saving as compared to the anchor. It is shown that the encoding time saving is around 20.5% and the BD-rate increment is around 0.7% on the average. We also observe the IM modes affects more in G3 and G4 than in G1. For example, the BD-rates in G4 decreases around 0.1% ∼ 0.3% with the extra 3% computational complexity. The IM modes affects the performance relatively less in G1 because there are more DC/Planar modes.

Fig.9 shows the performance changes of the proposed technique when the size of the initial candidates varies with 3, 5, and 7, referred to as “PROP(3-5-7)”. “No-IM(3-5-7)” refers to the same configuration of the size as “PROP(3-5- 7)”, yet the derivation of the IM mode is disabled. As can be seen in Fig.9, the IM mode increase only slight encoding time when the size is equal to 7. However, when the size becomes smaller, the IM mode plays an important role in the mode decision. For instance, “PROP(5)” provides almost same RD performance as No-IM(7), but the encoding time reduces about 6%. In our experiments, “PROP(3)” provides the best trade-off as reported in Table III. “PROP(3)”, as compared to other operating R-D points.

C. Performance Evaluation with Various Configurations in Random Forest

The performance of the proposed technique can depend on several factors of a random forest such as a number of the trees consisting of the random forest and a depth of the individual trees. The effects of the conditions to the overall performance are presented with several test sequences such as “BQMall”, “BQSquare”, “Drums”, and “FourPeople”, as compared to the HM software as an anchor. In Fig.10, the proposed technique shows changes in the BD-rates and the encoding time with respect to the number of the trees, i.e., 3, 5, 7, and 10. The encoding time tends to increase about 1 ~ 2% when the number changes from 3 to 10. The increments are not rapid because of the fast decision scheme in the random forest. Meanwhile, the number of the

TABLE V
THE BD-RATE INCREMENT AND THE ENCODING TIME REDUCTION OF THE PROPOSED ALGORITHM WITH THE PROPORTIONS OF THE DC OR PLANAR MODES AFTER RMD AND THE EVALUATIONS WHEN THE DERIVATION PROCESS OF THE IM-MODE IS DISABLED.



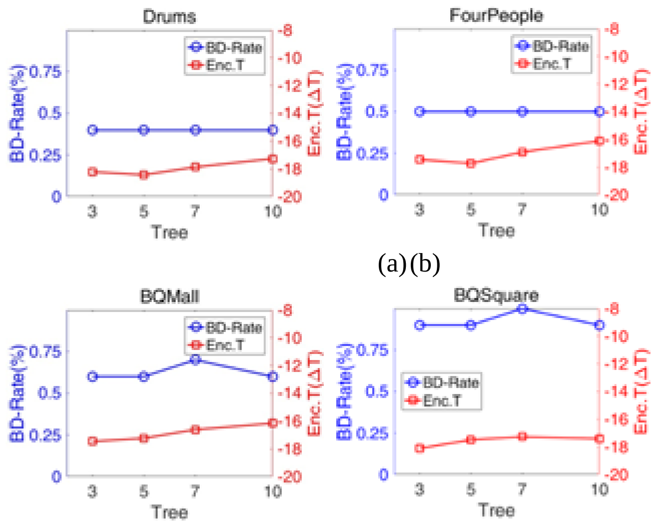
G.	Seq.	D. or P. (%)	PROP VS HM		No-IM VS HM	
			BR(%)	ΔT(%)	BR(%)	ΔT(%)
G1	1	65.7%	0.6%	−20.5%	0.6%	−20.9%
	11	62.9%	0.1%	−19.8%	0.2%	−20.6%
	3	60.3%	0.1%	−18.3%	0.4%	−20.2%
	12	60.3%	0.3%	−18.3%	0.5%	−20.9%
	7	58.6%	0.6%	−20.1%	0.7%	−21.1%
	8	57.2%	0.4%	−19.7%	0.4%	−20.0%
	5	56.4%	0.6%	−19.4%	0.7%	−19.6%
G2	20	54.7%	0.6%	−18.5%	0.9%	−20.7%
	26	52.0%	0.5%	−17.4%	0.6%	−20.8%
	4	50.3%	0.1%	−20.5%	0.4%	−23.9%
	25	49.8%	0.5%	−17.5%	0.8%	−20.4%
	15	47.8%	0.5%	−17.3%	0.7%	−20.3%
	6	45.1%	0.4%	−20.4%	0.4%	−20.9%
G3	13	43.8%	0.5%	−18.5%	0.7%	−20.6%
	10	41.5%	0.5%	−19.3%	0.6%	−20.3%
	9	40.7%	0.5%	−16.5%	0.8%	−20.2%
	24	39.5%	0.5%	−17.1%	0.7%	−20.1%
	19	38.7%	0.5%	−16.1%	0.7%	−19.6%
	16	37.2%	0.4%	−18.0%	0.7%	−20.3%
	21	36.3%	0.8%	−18.9%	0.9%	−20.6%
G4	17	34.7%	0.6%	−16.3%	0.8%	−20.3%
	22	34.3%	0.7%	−19.5%	0.8%	−21.2%
	14	33.4%	0.4%	−16.8%	0.5%	−21.5%
	18	32.9%	0.8%	−17.3%	0.9%	−20.8%
	23	31.2%	0.6%	−15.2%	0.8%	−20.3%
	2	30.5%	0.0%	−19.6%	0.3%	−21.3%
Average			0.5%	18.3%	0.7%	20.5%

trees affects the BD-rates slightly. In sum, the variations of the performance are relatively small in spite of the different number of trees as an hyper-parameter. Based on the results, we claim that the robustness of the classifier over various conditions and practical advantages because subtle changes in the implementation do not affect significant changes in the performance.

We also compare the BD-rates and the time with respect to the depth of each tree in Fig.11. The loss in BD-rates tends to decrease when the depth increases from 5. However, when the depth increases to e.g. 10, the loss becomes large or saturated. This is because of an over-fitting problem in the random forest. In other words, the parameters to be learned are larger significantly when the tree is deeper. Furthermore, the encoding time increases about 7 ~ 9% with the depth because a large number of nodes are visited for the decision. Thus, we empirically set the depth to 5 in experiments.

VI. CONCLUSION

In this paper, a machine learning-based fast intra-prediction mode decision algorithm was proposed. The random forest was used to estimate an intra-prediction mode from a prediction unit. We developed a randomized tree model including parameterized split functions at nodes to learn directional blockbased features. The feature used only four pixels reflecting directional characteristics in a prediction unit, and, thus the evaluation was fast and accurate. We defined the inferred mode to shrink the size of the candidates before carrying out



(a)(b)

(c) (d)

Fig. 10. Changes in BD-rates and encoding time when the random forest is configured with a different number of trees. The number of trees is 3, 5, 7, and 10. Test sequences are (a) “BQMall”, (b)“BQSquare”, (c)“Drums”, and (d)“FourPeople”.

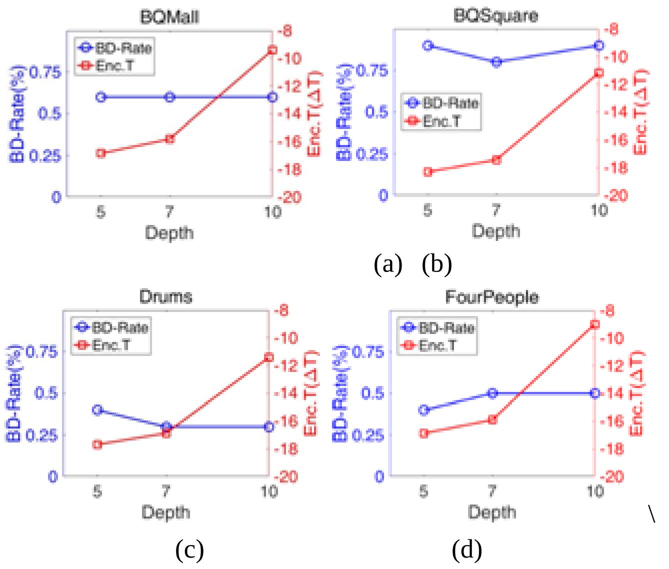


Fig. 11. Changes in BD-rates and encoding time when the random forest is configured with a different depth of trees. The depth is 5, 7, and 10. Test sequences are (a) “BQMall”, (b)“BQSquare”, (c)“Drums”, and (d)“FourPeople”.

the Rate-Distortion optimization. It was demonstrated with experimental results that the proposed technique achieved significant encoding time reduction with only slight coding loss as compared to different reference software models as anchors, used for the developments of state-of-the-art video coding standards.

REFERENCES

[1] G. Sullivan, J. Ohm, W. Han, and T. Wiegand, “Overview of the High Efficiency Video Coding Standard,” IEEE Trans. Circuits Syst. Video Tech., vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[2] Joint Exploration Model (JEM) reference software 5.0, “https://jvet.hhi. fraunhofer.de/svn/svnHMJEMSoftware/tags/HM-16.6-JEM-5.0/.”

[3] J. Chen, E. Alshina, G. Sullivan, J. Ohm, and J. Boyce, “JVET-E1001: Algorithm Description of Joint Exploration Test Model 5 (JEM 5),” in ISO/IEC/JTC1/SC29/WG11 and ITU-T SG16 Q.6, Jan. 2017.

[4] M. Karczewicz, J. Chen, W.-J. Chien, X. Li, A. Said, L. Zhang, and X. Zhao, “Study of coding efficiency improvements beyond HEVC,” in MPEG doc. m37102, Oct. 2015.

[5] E. Alshina, A. Alshin, J. Min, K. Choi, A. Saxena, and M. Budagavi, “VCEG-AZ05: Known tools performance investigation for next generation video coding,” in ITU-T SG16, Jun. 2015.

[6] S. Cho, S.-C. Lim, and J. Kang, “JVET-E0053: Evaluation report of SDR test sequences (4K5-9 and 1080p1-5),” in ISO/IEC/JTC1/SC29/WG11 and ITU-T SG16 Q.6, Jan. 2017.

[7] E. Alshina, A. Alshin, K. Choi, and M. Park, “JVET-B0022: Performance of JEM 1 tools analysis ,” in ISO/IEC/JTC1/SC29/WG11 and ITU-T SG16 Q.6, Feb. 2016.

[8] T. Nguyen and D. Marpe, “Objective Performance Evaluation of the HEVC Main Still Picture Profile,” IEEE Trans. Circuits Syst. Video Tech., vol. 25, no. 5, pp. 790–796, 2015.

[9] Y. Piao, J. Min, and J. Chen, “Encoder improvement of unified intra prediction,” Joint Collaborative Team on Video Coding (JCT-VC), JCTVCC207, 2010.

[10] L. Zhao, L. Zhang, S. Ma, and D. Zhao, “Fast mode decision algorithm for intra prediction in HEVC,” in Visual Communications and Image Processing (VCIP), 2011