Very Low Complexity MPEG-2 to H.264 Transcoding Using Machine Learning

Gerardo Fernández
Escribano
Instituto de Investigación en Informática de Albacete.
Universidad de Castilla-La Mancha
Avenida de España, s/n.
02071 Albacete, SPAIN
+ 34 967 599200 Ext. 2664
gerardo@dsi.uclm.es

Hari Kalva
Department of Computer Science and Engineering.
Florida Atlantic University
777 Glades Road,
Boca Raton, FL 33431, USA
+ 1 561 297 0511
hari@cse.fau.edu

Pedro Cuenca
Instituto de Investigación en Informática de Albacete.
Universidad de Castilla-La Mancha
Avenida de España, s/n.
02071 Albacete, SPAIN
+ 34 967 599200 Ext. 2492
pcuenca@dsi.uclm.es

Luis Orozco Barbosa
Instituto de Investigación en Informática de Albacete.
Universidad de Castilla-La Mancha
Avenida de España, s/n.
02071 Albacete, SPAIN
+ 34 967 599200 Ext. 2467
lorozco@dsi.uclm.es

ABSTRACT

This paper presents a novel macroblock mode decision algorithm for inter-frame prediction based on machine learning techniques to be used as part of a very low complexity MPEG-2 to H.264 video transcoder. Since coding mode decisions take up the most resources in video transcoding, a fast macro block (MB) mode estimation would lead to reduced complexity. The proposed approach is based on the hypothesis that MB coding mode decisions in H.264 video have a correlation with the distribution of the motion compensated residual in MPEG-2 video. We use machine learning tools to exploit the correlation and derive decision trees to classify the incoming MPEG-2 MBs into one of the 11 coding modes in H.264. The proposed approach reduces the H.264 MB mode computation process into a decision tree lookup with very low complexity. The proposed transcoder is compared with a reference transcoder comprised of a MPEG-2 decoder and an H.264 encoder. Our results show that the proposed transcoder reduces the H.264 encoding time by over 95% with negligible loss in quality and bitrate.

Categories and Subject Descriptors: I.4.2 [**Image Processing and Computer Vision**]: Compression (Coding) – *Approximate methods.*

General Terms: Algorithms, Performance, Design.

Keywords: H.264, MPEG-2, Transcoding, Inter-frame, Machine Learning.

1. INTRODUCTION

During the past few years, technological developments, such as novel video coding algorithms, lower memory costs, and faster processors, are facilitating the design and development of highly efficient video encoding standards. Among the recent works in this area, the H.264 video encoding standard, also known as MPEG-4 AVC occupies a central place [1].

MM'06, October 23–27, 2006, Santa Barbara, California, USA.

The H.264 standard is highly efficient by offering perceptually equivalent video quality at about 1/3 to 1/2 of the bitrates offered by the MPEG-2 format. However, these gains come with a significant increase in encoding and decoding complexity [2].

Though H.264 is highly efficient compared to MPEG-2, the wide and deep penetration of MPEG-2 creates a need for co-existence of these technologies and hence creates an important need for MPEG-2 to H.264 transcoding technologies. However, given the significant differences between both encoding algorithms, the transcoding process of such systems is much more complex compared to the other heterogeneous video transcoding processes [3-6]. The main elements that require to be addressed in the design of an efficient heterogeneous MPEG-2 to H.264 transcoder are [7]: the inter-frame prediction, the transform coding and the intra-frame prediction. Each one of these elements requires to be examined and various research efforts are underway. In this paper, we focus our attention on a part of the inter-frame prediction: the *macroblock mode decision*, one of the most stringent tasks involved in the transcoding process.

A video transcoder is comprised of a decoding stage followed by an encoding stage. The decoding stage of a transcoder can perform full decoding to the pixel level or partial decoding to the coefficient level. Partial decoding is used in compressed domain transcoding where the transform coefficients in the input format are directly transcoded to the output format. This transformation is straightforward when the input and output formats of the transcoder use the same transform (e.g., MPEG-2 to MPEG-4 transcoding) [5]. When these transforms differ substantially, the compressed domain transcoding becomes computationally expensive. The utility of this compressed domain transcoding is limited to intra MB transcoding. For predicted MBs, the transcoding in compressed domain becomes prohibitively expensive. The substantial differences in MPEG-2 and H.264 make even intra transcoding in the compressed domain relatively expensive [8]; pixel domain transcoding is shown to produce better results [9].

This work was supported by the Ministry of Science and Technology of Spain under CICYT project TIC2003-08154-C06-02, the Council of Science and Technology of Castilla-La Mancha under project PAI06-0106 and FEDER.

Pixel domain transcoders have a full decoding stage followed by a reduced complexity encoding stage. The complexity reduction is achieved by reusing the information gathered from the decoding stage. It is assumed that the input video is encoded with reasonable RD optimization. The MPEG-2 to H.264 complexity reduction techniques reported in the literature fall into two categories: 1) MB mode mapping in H.264 based on the MB modes of the incoming video [10] and 2) Selective evaluation of MB modes in H.264 based on heuristics [11]. Because of the large number of inter and intra MB coding modes supported by H.264, there is no one-to-one mapping between MPEG-2 and H.264 MB modes. A direct mapping leads to either a sub-optimal decision if the mapped mode is the final MB mode or an increase on complexity if additional evaluations have to be made to improve the mode decision. Selective evaluation is based on the observation that certain MB modes are less likely to occur for a class of videos and bitrates. If the selective evaluation is aggressive in limiting the number of allowed modes, the performance is sub-optimal. On the contrary, increasing the number of allowed modes increases the complexity.

We have developed an innovative approach that is not limited by the inefficiencies of mode mapping or selective evaluation approaches. The proposed approach is based on the hypothesis that MB coding mode decisions in H.264 video have a correlation with the distribution of the motion compensated residual in MPEG-2 video. Exploiting this correlation together with the MB coding modes of MPEG-2 could lead to a very low complexity transcoder. Figure 1 shows a plot of the mean and variance of the MPEG-2 MB residual in the input video and the H.264 MB coding mode of the corresponding MB in the transcoded video. As the coding mode changes, the shift in the mean and variance of the corresponding MB can be clearly seen. This correlation can be effectively exploited using machine learning approaches. Thus, the H.264 MB mode computation problem is posed as a data classification problem where the MPEG-2 MB coding mode and residual have to be classified into one of the several H.264 coding modes. The proposed transcoder is developed based on these principles and reduces the H.264 MB mode computation process into a decision tree lookup with very low complexity.

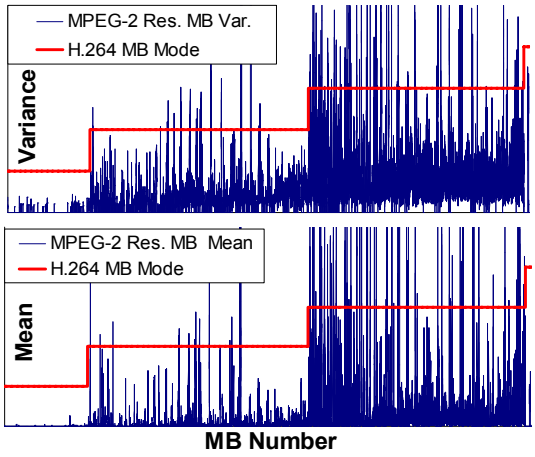


Figure 1. Relationship between MPEG-2 MB residual and H.264 MB coding mode.

The rest of the paper is organized as follows. Section 2 reviews the principles of operation of the prediction of inter-coded macroblocks in p-slices used by the H.264 encoding standard. Section 3 introduces our macroblock mode decision algorithm for inter-frame prediction based on machine learning techniques, specifically designed for MPEG-2 to H.264 transcoders. In Section 4, we carry out a performance evaluation of the proposed algorithm in terms of its computational complexity and rate-distortion results. We compare the performance of our proposal to the reference transcoder with the encoding stage using the H.264 reference implementation. Finally, Section 5 draws our conclusions and outlines our future research plans.

2. MACROBLOCK MODE DECISION AND MOTION ESTIMATION IN H.264

In the H.264 standard, the macroblock decision mode and motion estimation are the most computationally expensive processes. H.264 uses block-based motion compensation, the same principle adopted by every major coding standard since H.261. Important differences from earlier standards include the support for a range of block sizes (down to 4x4) and fine sub-pixel motion vectors (1/4 pixel in the luma component). H.264 supports motion compensation block sizes ranging from 16x16 to 4x4 luminance samples with many options between the two. The luminance component of each macroblock (16x16 samples) may be split up in 4 ways: 16x16, 16x8, 8x16 or 8x8. Each of the sub-divided regions is a *macroblock partition*. If the 8x8 mode is chosen, each of the four 8x8 macroblock partitions within the macroblock may be further split in 4 ways: 8x8, 8x4, 4x8 or 4x4 (known as *sub-macroblock partitions*). These partitions and sub-partitions give rise to a large number of possible combinations within each macroblock (see Figure 2). This method of partitioning macroblocks into motion compensated sub-blocks of varying size is known as *tree structured motion compensation*.

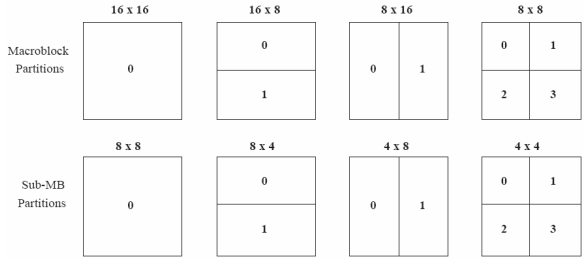


Figure 2. Macroblock partitions, sub-macroblock partitions and partition scans.

A separate motion vector (previously calculated in the motion estimation module) is required for each partition or sub-partition. Each motion vector must be coded and transmitted; in addition, the choice of partition(s) must be encoded in the compressed bitstream. Choosing a large partition size (e.g. 16x16, 16x8, 8x16) means that a small number of bits are required to signal the choice of motion vector(s) and the type of partition; however, the motion compensated residual may contain a significant amount of energy in areas with high detail. Choosing a small partition size (e.g. 8x4, 4x4, etc.) may give a lower-energy residual after motion compensation but requires a larger number of bits to signal the motion vectors and choice of partition(s). The choice of partition size therefore has a significant impact on compression performance. In general, a large partition size is appropriate for homogeneous areas of the frame and a small partition size may be beneficial for areas with high detail.

The resolution of each chroma component in a macroblock (Cr and Cb) is half that of the luminance (luma) component. Each chroma block is partitioned in the same way as the luma component, except that the partition sizes have exactly half the horizontal and vertical resolution (an 8x16 partition in luma corresponds to a 4x8 partition in chroma; an 8x4 partition in luma corresponds to 4x2 in chroma; and so on). The horizontal and vertical components of each motion vector (one per partition) are halved when applied to the chroma blocks.

Each partition in an inter-coded macroblock is predicted from an area of the same size in a reference picture. The offset between the two areas (the motion vector) has ¼-pixel resolution (for the luma component). If the video source sampling is 4:2:0, 1/8 pixel samples are required in the chroma components (corresponding to ¼-pixel samples in the luma). The luma and chroma samples at sub-pixel positions do not exist in the reference picture and so it is necessary to create them using interpolation from nearby image samples. Sub-pixel motion compensation can provide significantly better compression performance than integer-pixel compensation, at the expense of increased complexity. Quarter-pixel accuracy outperforms half-pixel accuracy.

Encoding a motion vector for each partition can take a significant number of bits, especially if small partition sizes are chosen. Motion vectors for neighboring partitions are often highly correlated and so each motion vector is predicted from vectors of nearby, previously coded partitions. The method of forming the prediction MVp depends on the motion compensation partition size and on the availability of nearby vectors.

In H.264, the macroblock mode decision is the most computationally expensive process. Mode decision is a process such that for each possible block-size a ***cost*** is evaluated. The encoder selects the coding-modes for the macroblock, including the best macroblock partition (sub-macroblock partition) and mode of prediction for each macroblock partition, such that the cost is optimized. In the JM reference code (version 10.2) [12], the motion estimation and the mode decision are executed together. This implies that for each macroblock partition (sub-macroblock partition) within the MB, motion estimation is done first and the resulting cost is used for the mode decision.

In the H.264, two methods have been defined to evaluate the cost for MB mode decision: *RD-cost* and *SAE-cost*. In the following, we describe these two methods.

2.1 The RD-Cost

The *Rate-Distortion* (RD) optimization method is based on a *Lagrange* multiplier [13] [14]. The H.264 standard can make use of this optimization method to choose the best macroblock mode decision. Different from evaluating the cost of coding a macroblock on a pixel by pixel basis (SAE cost), the RD-cost consists of making the selection based on a *Lagrange* function. In this way, the H.264 standard selects the macroblock mode exhibiting the minimum *Lagrange* cost. This implies that for each existing macroblock partition (sub-partition) within the MB, *bit-rate* and *distortion* are calculated by actually encoding and decoding the video. Therefore, the encoder can achieve the best Rate-Distortion performance results, at the expense of calculation complexity.

For evaluating the RD-cost, the standard has to obtain the *encoding rate*, ***R***, and the *distortion*, ***D***, of each macroblock partition (sub-macroblock partition). The former is obtained by first computing the difference between the original macroblock and its predictor. Thereafter, a 4x4 *Hadamard Transform* (HT) has to be applied followed by a quantization process. The distortion, ***D***, is obtained by performing an inverse quantization process followed by its inverse HT and then comparing the original macroblock to the reconstructed one. The H.264 standard chooses then the decision mode having the minimum cost, ***J***. The cost is evaluated using the *Lagrange* function $J=D+\lambda \times R$, where λ is the *Lagrange* multiplier. Figure 3 depicts the overall process.

One of the main drawbacks of this method is its excessive computational cost. On the contrary, the encoder can achieve the best Rate-Distortion performance results. However, for many applications, the use of the *Lagrange* multiplier may be prohibitive. This is the case when developing a transcoding architecture aimed to work in real-time.

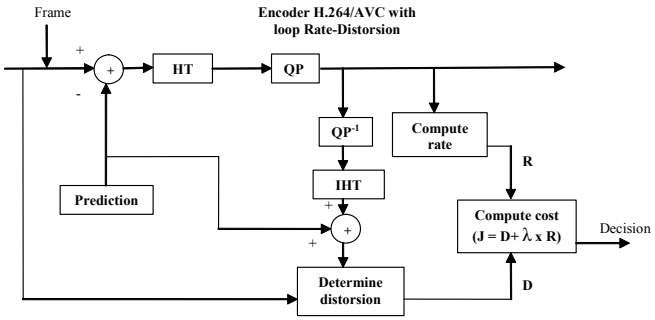


Figure 3. RD-cost method in the H.264 encoder.

2.2 The SAE-Cost

In this method, the H.264 encoder selects the best macroblock mode by using the *Sum of Absolute Errors* (SAE). This implies that for each existing macroblock partition (sub-partition) within the MB, a predictor within the pixel-domain is created from the motion estimation of the current partition and the SAE costs is evaluated. For each MB and for each color component (Y,Cr,Cb), one prediction mode have to be obtained. The best mode is determined corresponding to the mode exhibiting the minimum SAE cost. One of the main advantages of this method is its low computational cost. On the contrary, the Rate-Distortion performance results are sub-optimal.

2.3 The Fast Motion Estimation Option

Motion estimation is one of the most important tools in H.264 encoder for exploiting the high temporal redundancy between successive frames to improve video coding efficiency. And motion estimation is also the most time consuming part in the H.264 encoder (since it is also used for mode decision). Generally motion estimation is conducted into two steps: first is integer pel motion estimation; and the second is fractional pel motion estimation around the position obtained by the integer pel motion estimation.

Algorithms on Fast Motion Estimation (FME) are always hot research spot, especially fast integer pel motion estimation has achieved much more attention because traditional fractional pel

motion estimation only take a very few proportion in the computation load of whole motion estimation. Fast motion estimation algorithms such as *EPZS* [15], *UMHexagonS* [16], and *SEA* [17] have been proposed to reduce the number of searching points in motion estimation.

The *UMHexagonS* algorithm proposed by *Tsinghua University* was adopted by the H.264/MPEG-4 Part 10 (AVC) reference software implementation [12]. This algorithm uses the hybrid and hierarchical motion search strategies. It includes four steps with different kinds of search pattern: 1) Predictor selection and prediction mode reordering; 2) Unsymmetrical-cross search; 3) Uneven multi-hexagon-grid search; 4) Extended hexagon-based search. With the second and third step, the motion estimation accuracy can be nearly as high as that of full search. But the computation load and operations can be reduced even more. Unsymmetrical-cross search uses prediction vector as the search center and extends in the horizontal and vertical directions respectively. Uneven multi-hexagon-grid search includes two sub-steps: first a full search is carried out around the search center. And then a 16-HP multi-hexagon-grid search strategy is taken. Extended hexagon-based search is used as a center based search algorithm, including hexagon search and diamond search in a small range.

In the H.264 reference software, the Fast Motion Estimation (FME) algorithm (based in the *UMHexagonS* algorithm) can be employed for the motion estimation in addition to the original Full Search (FS) algorithm.

3. MACHINE LEARNING

Machine learning refers to the study of algorithms and systems that "learn" or acquire knowledge from experiences. Deductive machine learning deduces new rules/knowledge from existing rules and inductive machine learning uses the analysis of data sets for creating a set of rules to take decisions. These rules can be used, in the machine learning, to build a tree decision using a set of experiments or examples, named the training data set. This set of data must have the following properties [18]:

1. Each attribute or variable can take nominal or numerical values, but the number of attributes cannot vary from an example to another. This is to say, all the samples in the training data set used for training the model must have the same number of variables.
2. The set of categories that the examples can be assigned to must a priori be known to enable supervised learning.
3. The set of categories must be finite and must be different from one another.
4. Since the inductive learning consists of obtaining generalization from examples, it is supposed the existence of a sufficiently great number of examples.

Machine learning uses statistics with different kinds of algorithms to solve a problem by studying and analyzing the data. Machine learning has been used in an extensive range of applications including search engines, medical diagnosis, stock market analysis, classifying DNA sequences, speech and handwriting recognition, object recognition in computer vision, game playing and robot motion, etc.

In this paper, we describe the process of using machine learning to build a decision tree for very low complexity transcoding. The decision tree will be used to determine the coding mode of an MB in P frames of the output H.264 video, based on the information gathered during the MPEG-2 decoding stage. Figure 4 depicts the process for building the decision trees to be used in the MPEG-2 to H.264 transcoding process. The incoming MPEG-2 video is decoded and during the decoding stage, the MB coding mode, the coded block pattern (CBPC), and the mean and variance of the residual information for this MB (calculated for its 4x4 sub-blocks – resulting in 16 means and 16 variances for each MB) are saved. The decoded MPEG-2 video is then encoded using the standard H.264 encoder. The coding mode of the corresponding MBs in H.264 is also saved. Based on the MPEG-2 data and the corresponding H.264 coding mode decision for each MB, a machine learning algorithm is used to create decision trees that classify an MB into one of the 11 H.264 MB coding modes.

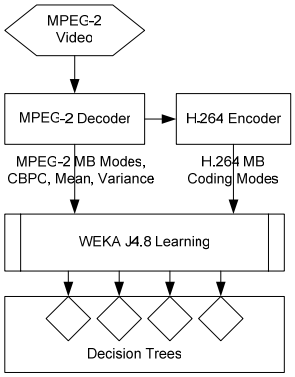


Figure 4. Process for building decision trees for MPEG-2 to H.264 transcoding.

3.1 Creating the Training Files

A decision tree is made by mapping the observations about a set of data to a tree made of arcs and nodes. The nodes are the variables and the arcs the possible values for that variable. The tree can have more than one level; in that case, the nodes (leafs of the tree) represent the decisions based on the values of the different variables that drive the decision from the root to the leaf. These types of trees are used in the machine learning processes for discovering the relationships in a set of data. The tree leafs are the classifications and the branches are the features that lead to a specific classification. A tree decision is a classifier based on a set of attributes allowing us to determine the category of an input data sample.

The decision tree for the transcoder was made using the WEKA data mining tool [18]. The files that are used for the WEKA data mining program are known as *Attribute-Relation File Format* (ARFF) files. An ARFF file is written in ASCII text and shows the relationship between a set of attributes. Basically, this file has two different sections:1) the header which contains the name of the relation, the attributes that are used, and their types; and 2) the section containing the data.

The training sets were made using MPEG-2 sequences encoded at higher than the typical broadcast encoding rates for the same quality, since the B frames are not used. The H.264 decisions in the training set were obtained from encoding the MPEG-2

decoded sequence with a quantization parameter of 25 and RD optimization enabled. After extensive experimentation, we found that sequences that contain regions varying from homogenous to high-detail serve as good training sets. Good sample sequences could be Flower and Football. The goal is to develop a single, generalized, decision tree that can be used for transcoding any MPEG-2 video.

Figure 5 shows the decision trees built using the process depicted in Figure 4. As shown in Figure 4, the *Decision Tree* for the proposed transcoder is a hierarchical decision tree with three different WEKA trees: 1) classifier for Intra, Skip, Inter 16x16, and Inter 8x8, 2) classifier to classify inter 16x16 into one of 16x16, 16x8, and 8x16 MBs and 3) classifier to classify inter 8x8 into one of 8x8, 8x4, 4x8, or 4x4. This paper focuses on the Inter MB mode computation and the further classification and processing for Intra MBs is not discussed in this paper.

For creating the first WEKA tree (Figure 5 node 1), the first training data set uses the mean and variance of each one of the sixteen 4x4 residual sub-blocks, the MB mode in MPEG-2 (skip, intra, and three non-intra modes, labeled as 0, 1, 2, 4 and 8 in the code shown below), the coded block pattern (CBPC) in MPEG-2, and the corresponding H.264 MB coding mode decision for that MB as determined by the standard reference software. The header section of the ARFF files has the attribute declaration depicted herein:

```
@RELATION mean-variance_4x4

@ATTRIBUTE mean0 Numeric
@ATTRIBUTE variance0 Numeric
@ATTRIBUTE mean1 Numeric
@ATTRIBUTE variance1 Numeric
……
@ATTRIBUTE mean15 Numeric
@ATTRIBUTE variance15 Numeric
@ATTRIBUTE mode_mpeg2 {0,1,2,4,8}
@ATTRIBUTE CBPC0 {0,1}
……
@ATTRIBUTE CBPC6 {0,1}
@ATTRIBUTE class {0,1,8,9}
```

The supposed dependent variable, namely *class* in the example, is the variable that we are trying to understand, classify, or generalize. The other attributes are the variables that determine the classification. The ARFF data section has the instance lines, which are the samples used to train our model. Each macroblock sample is represented on a single line. In this case the variable class can take four values (skip, 16x16, 8x8 or Intra labeled as 0, 1, 8 and 9 in the code).

The second training data set, used for creating the second WEKA tree (Figure 5 node 2), was made using the samples (MBs) that were encoded as 16x16 MBs in the H.264 reference encoder. It uses the mean and variances of each one of the sixteen 4x4 residual sub-blocks, the MB mode in MPEG-2 (in this case only the three non-intra modes), the coded block pattern (CBPC) in MPEG-2, and the corresponding H.264 MB coding sub-mode decision in the 16x16 mode, as determined by the standard reference software: 16x16, 16x8 or 8x16. This tree determines the final coding mode of the MBs classified as inter 16x16 by the first tree.

The third and last training data set, was used to create the third WEKA tree (Figure 5 node 3) and was made using the samples (MBs) that were encoded as inter 8x8 MBs in the H.264 reference encoder. It uses four means and four variances of 4x4 residual sub-blocks, the MB mode in MPEG-2 (the three non-intra modes), the coded block pattern (CBPC) in MPEG-2, and the corresponding H.264 MB sub-partition decision in the 8x8 mode, as determined by the standard reference software: 8x8, 8x4, 4x8 or 4x4. Since this decision is made separately for each 8x8 sub-block, only the four means and four variances of 4x4 residual sub-blocks are used in each sample for training the model.

Based on these training files, the J48 algorithm implemented in the WEKA data mining tool was used to create the three decision trees. The J48 algorithm is an implementation of the C4.5 algorithm proposed by Ross Quinlan [19]: the algorithm widely used as a reference for building decision trees.

The decision tree, that is proposed to solve the inter-prediction problem, is a model of the data that encodes the distribution of the class label in terms of the attributes. The final goal of this decision tree is to help find a simple structure to show the possible dependences between the attributes and the class.

3.2 The Decision Tree

This sub-section discusses the proposed macroblock mode decision algorithm aiming to accelerate the inter-frame prediction. This goal is achieved by making use of the MPEG-2 MB coding modes, the coded block pattern (CBPC), and the mean and variance of the residual information for this MB calculated for its 4x4 sub-blocks. MPEG-2 uses 16x16 motion compensation (MC) and does not temporally decorrelate an image fully. The MC residual can thus be exploited to understand the temporal correlation of variable block sizes in H.264. The open source WEKA data mining tool is used to discover a pattern of the mean, variance, MPEG-2 coding modes, and the coded block pattern in MPEG-2 (CBPC) for H.264 coding mode decisions. Figure 5 shows the decision tree used in the proposed transcoder.

The decision tree consists of three WEKA decision trees, shown in Figure 5 with grey balls. The first WEKA tree is used to check for the skip, Intra, 8x8 and 16x16 MBs modes. If an MB is 8x8 or 16x16, a second and a third decision tree is used for selecting the final coding mode of the MB. The WEKA tool determined the mean and variance thresholds for each of the three WEKA trees in the decision tree. Due to space constraints we cannot show all the rules being evaluated in the WEKA decision nodes. The process described in herein should be sufficient for interested people to develop the decision trees and repeat these experiments. The decision tree works as follows:

Node 1. The inputs for this node are all the MPEG-2 coded MBs. In this node a tree decision generated with WEKA is used to decide whether the MB should be coded in H.264. This tree examines whether the MB has a very high residual or a medium residual. The output of this node is a first level decision mode that should be used for coding the MB: skip, Intra, 8x8 or 16x16. The intra decision process is not discussed in this paper. In the other cases, the algorithm has to make a second level decision based in the first decision. For example, the following rules were given by WEKA:

- If the MPEG-2 MB was “MC not coded”, (non-zero MV present, none of the 8x8 block has coded coefficients), then

the MB will be coded as 16x16 in H.264. Again, a second decision level will be made to select the best choice in this case (see node 2).

- If the MPEG-2 MB was coded in intra mode, the MB will be coded as intra or inter 8x8 mode in H.264. In some cases the algorithm will propose Intra, and the algorithm will end, and in other cases the algorithm will propose 8x8 mode, so a second level decision will be done (see node 3).
- If the MPEG-2 MB was coded in skip mode, then the H.264 decision mode should be skip. The decision will be made in node 4.

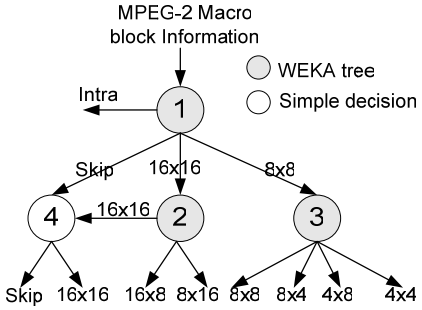


Figure 5. The Decision Tree.

Node 2. The inputs for this node are the 16x16 MBs classified by the node 1. In this node we use again a decision tree generated with WEKA to decide whether the MB should be coded in H.264 (16x16, 16x8 or 8x16). This tree examines if there are continuous 16x8 or 8x16 sub-blocks that might result in a better prediction. The output of this node is the 16x16 sub-mode decision mode that should be used for coding the MB: 16x16, 16x8 or 8x16. When the node decision is 16x8 or 8x16 the coding mode is finalized. In the other case, the evaluation continues in node 4, where the final decision will be made.

Node 3. The inputs for this node are the MBs classified by the node 1 as 8x8. This node evaluates only the H.264 8x8 modes using the third WEKA tree and selects the best option: 8x8, 8x4, 4x8 or 4x4. As explained in the previous section, this tree is run 4 times, once for each of the four sub-macroblocks in the MB. This tree is different from the others because this one only uses four means and four variances to make the decision.

Node 4. The inputs for this node are skip-mode MBs in the MPEG-2 bitstream classified by the node 1, or the 16x16 MBs classified by the node 2. This node evaluates only the H.264 16x16 mode (without the sub-modes 16x8 or 8x16). Then, the node selects the best option, skip or inter 16x16.

Since the MB mode decision, and hence the thresholds, depend on the quantization parameter (QP) used in the H.264 encoding stage, the mean and variance threshold will have to be different at each QP. The two solutions here are: 1) develop the decision trees for each QP and use the appropriate decision tree depending on the QP selected and 2) develop a single decision tree and adjust the mean and variance threshold used by the trees based on the QP. The first option is complex as we have to develop and switch between 52 different decision trees resulting in 156 WEKA trees in a transcoder. Since the QP used by H.264 is designed to change the quantization step size and the relationship between the QPs is well defined, this relationship can be used to adjust the mean and variance thresholds. The proposed transcoder uses a single decision tree developed for a mid-QP of 25 and then adjusted for other QPs. Since the quantization step size in H.264 doubles when QP increases by 6, the thresholds are adjusted by 2.5% for a change in QP of 1. For QP values higher than 25, the thresholds are decreased and for QP values lower than 25 thresholds are proportionally increased.

Figure 6 shows an example of the results obtained by applying our proposed algorithm. Figure 6a illustrates the residual for the MPEG-2 encoded Tempete sequence. Figures 6b and 6c show the mean and variance of the residual. Figures 6.e and 6.f show the differences between the inter mode selection made by the H.264 standard (with the RD-optimized option enabled), and the proposed algorithm, with a value of 10 for QP. From these figures, it is clear that our algorithm obtains very similar results to those obtained using the full estimation of the H.264 standard.

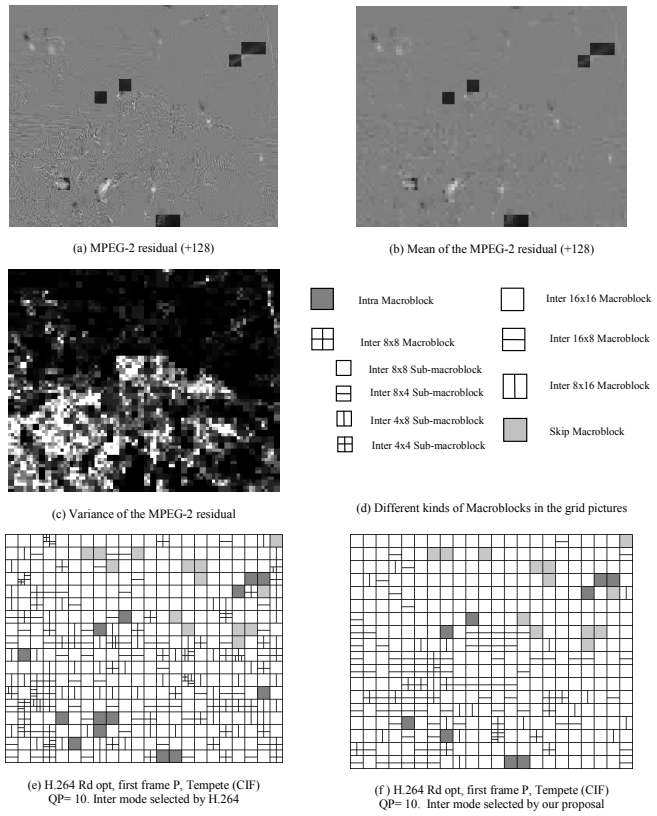


Figure 6. Macroblock partitions generated by the proposed algorithm for the first P-frame in the Tempete sequence.

4. PERFORMANCE EVALUATION

The proposed low complexity MB coding mode decision algorithm is implemented in the H.264/AVC reference software, version JM 10.2 [12]. Figure 7 shows the overall operation of the proposed transcoder. The MPEG-2 video is decoded and the information required by the decision trees is gathered in this stage. The additional computation here is the computation of the mean and variance of the 4x4 sub-blocks of the residual MBs. The MB coding mode decision determined by the decision trees is used in the low complexity H.264 encoding stage. This is an

H.264 reference encoder with the MB mode decision replaced by simple mode assignment from the decision tree. The H.264 video encoder takes as input the decoder MPEG-2 video (pixel data) and the MB mode decision from the decision tree and encodes the H.264 video. The MPEG-2 motion vectors are not used and the encoder performs the motion estimation just for the final MB mode determined by the decision tree.

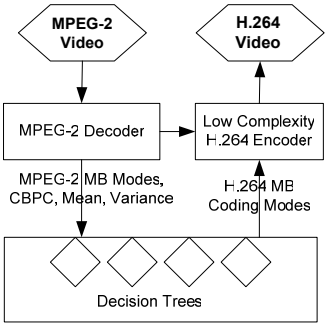


Figure 7. Proposed transcoder.

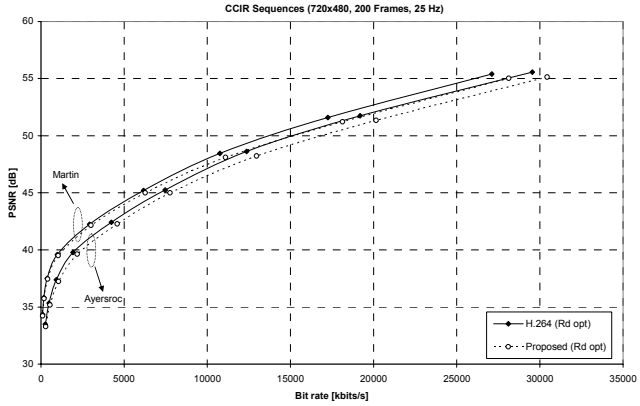
The performance of the proposed very low complexity transcoder is compared with a reference transcoder comprised of a full MPEG-2 decoder followed by a full H.264 encoder. We compare the performance of our proposal to the full H.264 encoder when the RD-cost (with and without FME option enabled) and the SAE-cost (with and without FME option enabled) are used. The metrics used to evaluate the performance are the reduction in the computational cost and rate distortion function. The time results reported are for the H.264 encoding component as the MPEG-2 decoding cost is the same for both the proposed and reference encoders.

We have conducted an extensive set of experiments with videos representing wide range of motion, texture, and color. Experiments were conducted to evaluate the performance of the proposed algorithm when transcoding videos at commonly used resolutions: CCIR-601, CIF, and QCIF. The input to the transcoder is a high quality MPEG-2 video. Since the proposed transcoder addresses transcoding P frames in MPEG-2 to H.264 P frames, MPEG-2 bitstreams were created without B frames. Since the B frames, which are much smaller than P frames, are not used in the input video, the video has to be encoded at higher than the typical encoding rates for equivalent broadcast quality. Table 1 shows the bitrates used for the input MPEG-2 video. The experiments have shown that the proposed approach performs extremely well across all bitrates and resolutions.

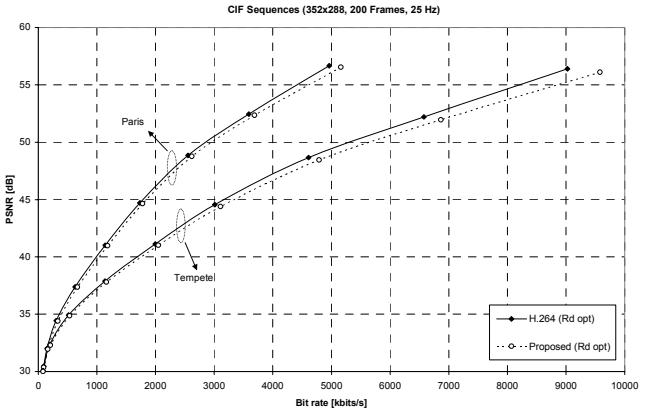
Table 1. Bitrates for the input sequences

Format	Bitrate
CCIR-601 (720x480)	5 Mbps
CIF (352x288)	1.15 Mbps
QCIF (176x144)	0.768 Mbps

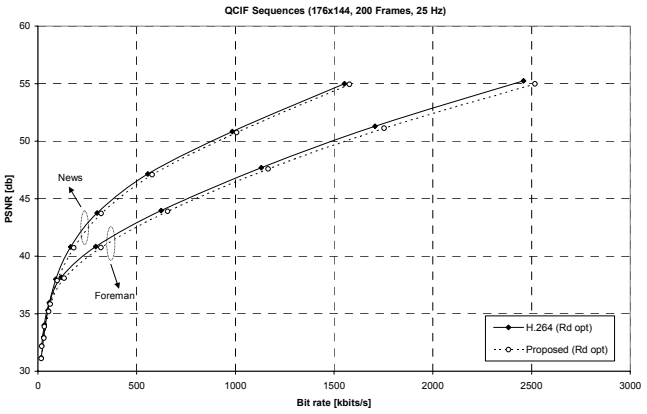
The sequences have been encoded with H.264 using the QP factors ranging from 5 up to 45 in steps of 5. This corresponds to the H.264 QP range used in most practical applications. The size of the GOP is 12 frames; where the first frame of every GOP was encoded as I-frame, and the rest of the frames of the GOP were encoded as a P-frames. The rate control was disabled for all the simulations. The ProfileIDC was set to High for all the simulations, with the FRExt options enabled. The simulations were run on a P4 HT at 3.0 GHz Intel machine with 512 MB RAM. The results are reported for six different sequences: two for each of the three resolutions shown in Table 1.



(a)



(b)

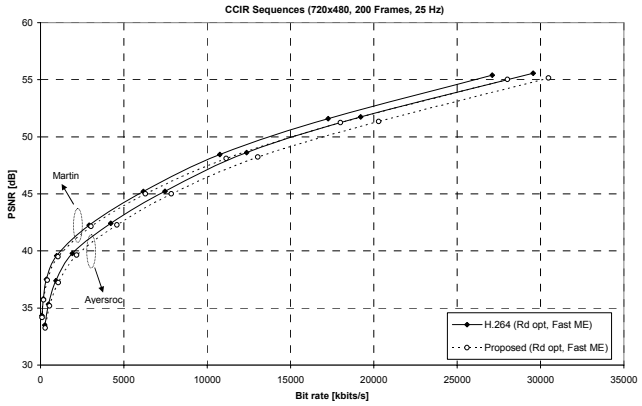


(c)

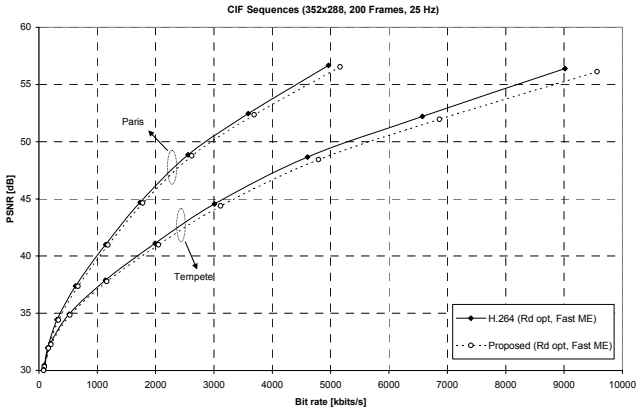
Figure 8. RD results for RD-cost without FME option.

Figure 8 shows the RD results for the reference and proposed transcoder with RD optimization enabled and fast motion estimation (FME) disabled. Figure 9 shows the RD results for the reference and proposed transcoder with RD optimization enabled and fast motion estimation (FME) enabled. As seen from the figures, the PSNR obtained with the proposed transcoder deviates slightly from the results obtained when applying the considerable

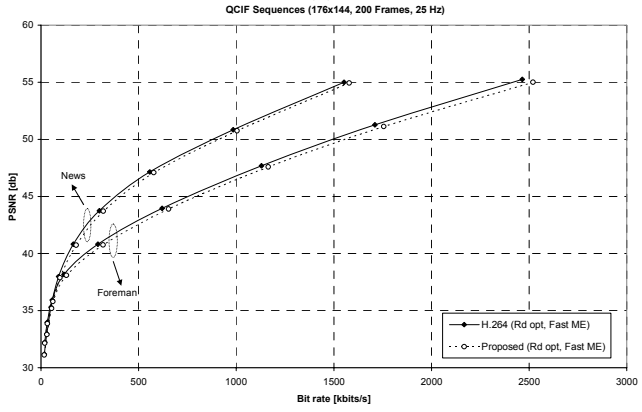
more complex reference transcoder. Compared with the reference transcoder, the proposed transcoder has a PSNR drop of at most 0.3 dB for a given bitrate and bitrate increase of at most 5% for a given PSNR. This negligible drop in RD performance is more then offset by the reduction in computational complexity. Tables 2 and 3 show the average encoding time per frame given in milliseconds. As shown in Table 2 and Table 3, the transcoding time reduces by more than 80% with RD optimization, and more than 90% with FME enabled for both the reference and proposed transcoders.



(a)



(b)



(c)

Figure 9. RD results for RD-cost with FME option.

Figure 10 shows the RD results for the reference and proposed transcoder with SAE-Cost (RD optimization disabled) and fast motion estimation (FME) disabled. Figure 11 shows the RD results for the reference and proposed transcoder with SAE-Cost (RD optimization disabled) and fast motion estimation (FME) enabled. As seen from the figures, in some cases the proposed transcoder have better results than the reference transcoder. This happens because the best solution is obtained by enabling the RD optimization, and in the experiments reported in the figures we are comparing the faster configuration of a H.264 encoder (SAE cost) with our proposed reduced-complexity transcoder. With SAE based encoding (RD-optimization disabled), the proposed transcoder continues to outperform the reference transcoder computationally (Tables 2 and 3). The transcoder still maintains a PSNR drop of less than 0.3 dB and bitrate increase of less than 5%. The computational cost is reduced by over 38% for the SAE case and by over 82% with FME enabled for both the reference and proposed transcoders.

Table 2. Mean encoding time (milliseconds) per frame with the reference transcoder

Sequence	RD Opt	RD Opt + FME	SAE	SAE + FME
Martin	7370	6420	2110	940
Ayersroc	7650	6820	2095	1030
Paris	2305	2020	590	235
Tempete	2360	2050	605	290
Foreman	565	495	155	68
News	550	470	150	55

Table 3. Mean encoding time (milliseconds) per frame with the proposed transcoder

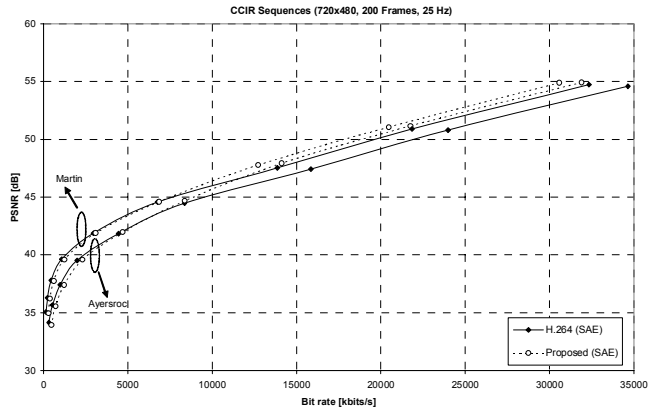
Sequence	RD Opt	RD Opt + FME	SAE	SAE + FME
Martin	1460	470	1190	170
Ayersroc	1620	670	1160	190
Paris	415	95	360	45
Tempete	445	135	360	53
Foreman	102	24	93	12
News	103	21	92	11

Table 4. Mean Time Reduction (%) per frame with the proposed transcoder

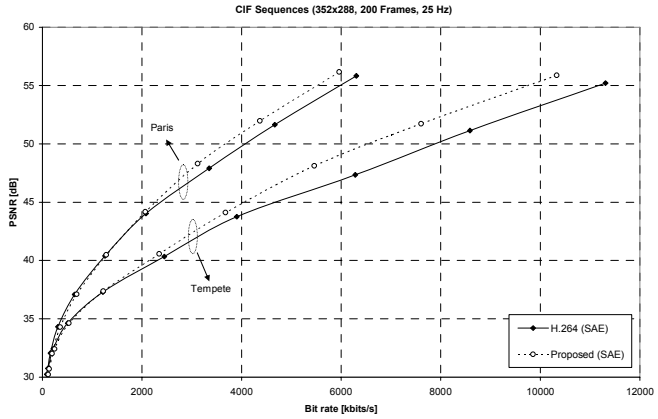
Sequence	RD Opt	RD Opt + FME	SAE	SAE + FME
Martin	80,19	92,68	43,60	81,91
Ayersroc	78,82	90,18	44,63	81,55
Paris	82,00	95,30	38,98	80,85
Tempete	81,14	93,41	40,50	81,72
Foreman	81,95	95,15	40,00	82,35
News	81,27	95,53	38,67	80,00

Based on the results shown in the Tables 2 and 3, the proposed transcoder with SAE and FME has the lowest complexity. The proposed transcoder with RD optimization and FME is still faster than the fastest case of the reference transcoder (SAE + FME). Using FME reduces the complexity substantially. Selecting RD optimization with the proposed transcoder doubles the complexity compared with SAE+FME case. The decision to enable RD optimization can be based on the operating bitrates and sensitivity to the PSNR drop. At higher bitrates, RDOPT + FME option give about 0.6 dB better than the SAE + FME option; this is doubling

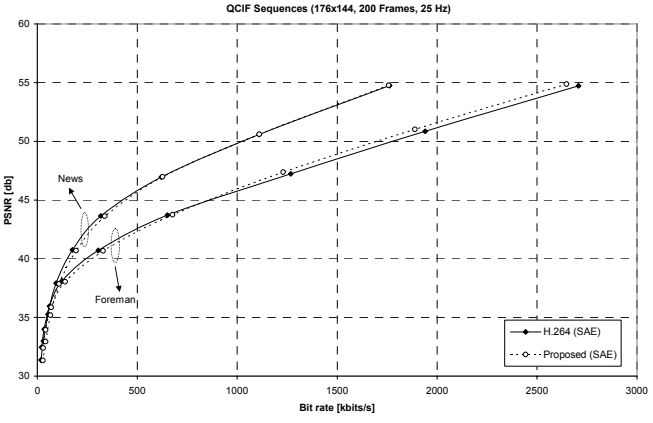
the complexity for a gain of 0.6 dB. However, at lower bitrates, the PSNR gain reduces to about 0.3 dB.



(a)



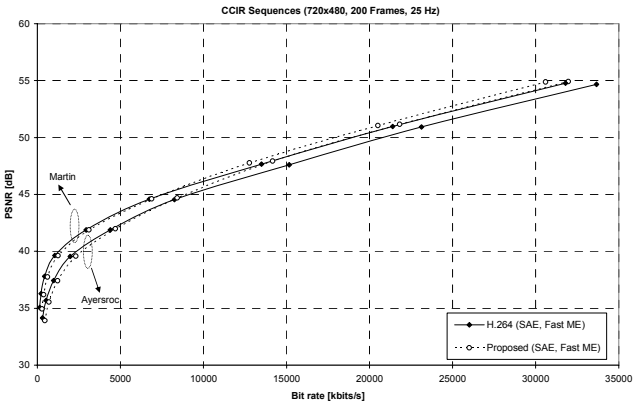
(b)



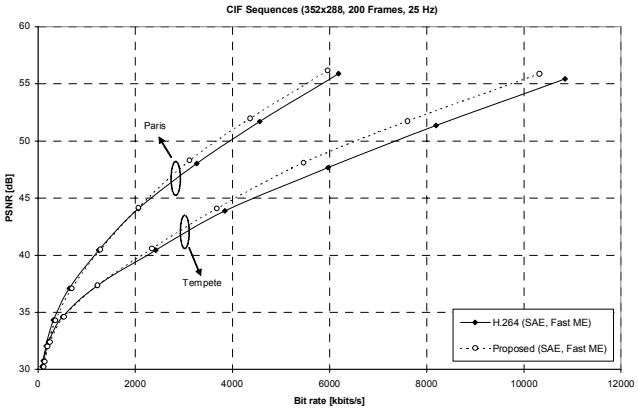
(c)

Figure 10. RD results for SAE-cost without FME option.

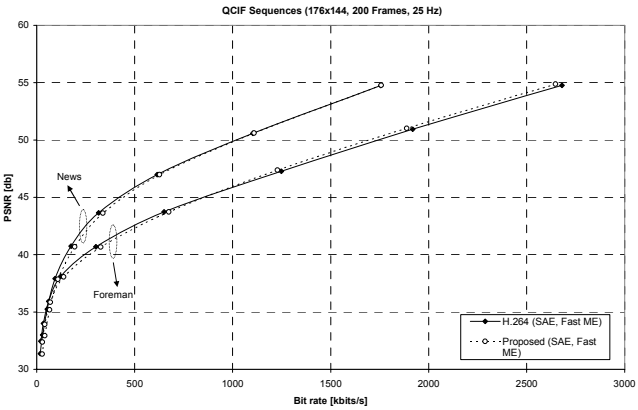
Table 4 summarizes the reduction in the computational cost due to the proposed machine learning based mode decision algorithms in the proposed transcoder. With RD optimization and FME, the computational cost is reduced by over 90%. The cost reduction reaches as high as 95.5% for QCIF sequences. With SAE and FME, the computational cost is reduces by over 80%. The computational cost reduction come at a cost of reduced quality. The quality reduction, however, is very small and negligible for most video applications. Table 5 shows the quality variation versus time reduction of the proposed transcoder with respect the reference transcoder for the same input bitrates shown in Table 1, showing over 96% reduction in the computational complexity characterizing our proposed scheme. As shown in the table, using the proposed transcoder reduces the PSNR by at most 0.3dB with RD optimization enabled and by at most 0.1 dB with SAE cost based transcoder. Our results show that the proposed algorithm is able to maintain a good picture quality while considerably reducing the number of operations to be performed in all the scenarios.



(a)



(b)



(c)

Figure 11. RD results for SAE-cost with FME option.

Table 5. Quality Variation vs Time Reduction (for transcoding rate)

Sequence	MPEG-2 Bit Rate (Mbps)	Quality Variation from Reference Transcoder (dB)				Time Reduction from Reference Transcoder (%)			
		RD OPT	RD FME	SAE	SAE FME	RD OPT	RD FME	SAE	SAE FME
Ayersroc	5.0	- 0.3	- 0.3	0.0	- 0.1	80.0	90.5	43.3	82.3
Martin	5.0	- 0.2	- 0.2	- 0.1	- 0.1	80.5	92.8	42.1	82.0
Tempete	1.15	- 0.2	- 0.2	0.0	0.0	80.0	93.8	41.1	82.5
Paris	1.15	- 0.3	- 0.3	0.0	- 0.1	81.6	95.6	38.5	80.7
Foreman	0.768	- 0.3	- 0.3	0.0	0.0	83.5	95.5	37.4	82.6
News	0.768	- 0.2	- 0.2	0.0	0.0	84.1	96.0	35.1	81.1

5. CONCLUSIONS

In this paper, we proposed a novel macroblock partition mode decision algorithm for inter-frame prediction to be used as part of a high-efficient MPEG-2 to H.264 transcoder. The proposed algorithms use machine learning techniques to exploit the correlation in the MPEG-2 MC residual and the H.264 coding modes. The WEKA tool was used to develop decision trees for H.264 coding mode decision. The proposed algorithm has very low complexity as it only requires the mean and variance of the MPEG-2 residual and a set of rules to compare the mean and variance against a threshold. The proposed transcoder uses a single decision tree with adaptive thresholds based on the quantization parameter selected in the H.264 encoding stage. The proposed transcoder was evaluated using MPEG-2 videos at CCIR, CIF, and QCIF resolutions. Our results show that the proposed algorithm is able to maintain a good picture quality while considerably reducing the computational complexity by as much as 95%. The reduction in computational cost has negligible impact on the quality and bitrate of the transcoded video. The results show that the proposed transcoder maintains its performance across all resolutions and bitrates. The proposed approach to transcoding is novel and can be applied to develop other transcoders as well.

Our future plans will focus on further reducing the complexity of the proposed transcode by reusing the MPEG-2 motion vectors followed by a motion vector refinement. By reusing the motion vector, we believe, real-time transcoding of CIF resolution video at 30 FPS is within reach.

6. REFERENCES

[1] ITU-T RECOMMENDATION H.264 “Advanced Video Coding for Generic Audiovisual Services”. May 2003.

[2] Implementation Studies Group, “Main Results of the AVC Complexity Analysis”. MPEG Document N4964, ISO/IEC JTC11/SC29/WG11, July 2002.

[3] T. Shanableh and M. Ghanbari, “Heterogeneous Video Transcoding to Lower Spatio-Temporal Resolutions and Different Encoding Formats,” *IEEE Transactions on Multimedia*, vol.2, no.2, June 2000.

[4] A. Vetro, C. Christopoulos, and H.Sun “Video Transcoding Architectures and Techniques: An Overview”. *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp.18-29, March. 2003.

[5] H. Kalva, A. Vetro, and H. Sun, “Performance Optimization of the MPEG-2 to MPEG-4 Video Transcoder”. *Proceeding of SPIE Conference on Microtechnologies for the New Millennium, VLSI Circuits and Systems*, May 2003.

[6] S. Dogan, A.H. Sadka and A.M. Kondoz, “Efficient MPEG-4/H.263 Video Transcoder for Interoperability of Heterogeneous Multimedia Networks,” *IEE Electronics Letters*, Vol. 35, No.11. pp. 863-864.

[7] H. Kalva. "Issues in H.264/MPEG-2 Video Transcoding". *Proceedings of Consumer Communications and Networking Conference*, January 2004.

[8] Y. Su, J. Xin, A. Vetro, and H. Sun, “Efficient MPEG-2 to H.264/AVC Intra Transcoding in Transform-Domain,” IEEE International Symposium on Circuits and Systems, 2005. ISCAS 2005. pp. 1234- 1237 Vol. 2, 23-26 May 2005.

[9] B. Petljanski and H. Kalva, "DCT Domain Intra MB Mode Decision for MPEG-2 to H.264 Transcoding" Proceedings of the ICCE 2006. January 2006. pp. 419-420.

[10] Y.-K. Lee, S.-S. Lee, and Y.-L. Lee, “MPEG-4 to H.264 Transcoding using Macroblock Statistics,” Proceedings of the ICME 2006, Toronto, Canada, July 2006.

[11] X. Lu, A. M. Tourapis, P. Yin, and J. Boyce, “Fast Mode Decision and Motion Estimation for H.264 with a Focus on MPEG-2/H.264 Transcoding,” Proceedings of 2005 IEEE International Symposium on Circuits and Systems (ISCAS), Kobe, Japan, May 2005.

[12] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Reference Software to Committee Draft. JVT-F100 JM10.2. 2006.

[13] G. Sullivan and T. Wiegand, “Rate-Distortion Optimization for Video Compression,” IEEE Signal Processing Magazine, vol. 15, no. 6, pp. 74-90, November. 1998.

[14] T. Wiegand et al., “Rate-Constrained Coder Control and Comparison of Video Coding Standards,” IEEE Transactions on Circuits Systems and Video Technology, vol. 13, no. 7, pp. 688-703, July 2003.

[15] A.M. Tourapis, O.C. Au, M.L. Liou, “Highly Efficient Predictive Zonal Algorithms for Fast Block-Matching Motion Estimation,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, Issue 10, Oct. 2002.

[16] Z. Chen, P. Zhou, and Y. He, “Fast Integer Pel and Fractional Pel Motion Estimation for JVT”, 6th Meeting. Awaji, December 2002

[17] M. Yang, H. Cui, K. Tang, “Efficient Tree Structured Motion Estimation using Successive Elimination,” IEE Proceedings-Vision, Image and Signal Processing, Vol. 151, Issue 5, Oct. 2004.

[18] Ian H. Witten and Eibe Frank, “Data Mining: Practical Machine Learning Tools and Techniques”, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[19] J.R. Quinlan, “C4.5: Programs for Machine Learning”, Morgan Kaufmann, 1993.