Garrido-Cantos et al. EURASIP Journal on Advances in Signal Processing 2013, 2013:82
http://asp.eurasipjournals.com/content/2013/1/82

EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH Open Access

Low-complexity transcoding algorithm from H.264/AVC to SVC using data mining

Rosario Garrido-Cantos[1*], Jan De Cock[2], Jose Luis Martínez[1], Sebastian Van Leuven[2], Pedro Cuenca[1] and Antonio Garrido[1]

Abstract

Nowadays, networks and terminals with diverse characteristics of bandwidth and capabilities coexist. To ensure a good quality of experience, this diverse environment demands adaptability of the video stream. In general, video contents are compressed to save storage capacity and to reduce the bandwidth required for its transmission. Therefore, if these compressed video streams were compressed using scalable video coding schemes, they would be able to adapt to those heterogeneous networks and a wide range of terminals. Since the majority of the multimedia contents are compressed using H.264/AVC, they cannot benefit from that scalability. This paper proposes a low-complexity algorithm to convert an H.264/AVC bitstream without scalability to scalable bitstreams with temporal scalability in baseline and main profiles by accelerating the mode decision task of the scalable video coding encoding stage using machine learning tools. The results show that when our technique is applied, the complexity is reduced by 87% while maintaining coding efficiency.

Keywords: Low-complexity algorithms, Scalable video coding, H.264/AVC, Transcoding, Temporal scalability, Data mining

1. Introduction

Everyday mobile media services are being introduced into the market place in response to increasing user demand for ubiquitous media services and applications. Media contents are now being delivered over a wide variety of wireless/wired networks to mobile/fixed devices ranging from smartphones and tablets to powerful laptops and TVs. Some network technologies have even been deployed specifically to deliver this content, such as mobile digital TV networks: the newest one, Advanced Television Systems Committee - Mobile/Handheld [1], was standardized in October 2009 and provides mobile digital TV service in the USA, Canada, part of South America, and parts of Asia. Another network technology is Digital Video Broadcasting - Handheld [2], which was standardized in November 2004 and adopted by the European Commission in March 2008 as the preferred technology to deliver mobile digital TV. This technology is used in Europe, parts of Africa, Asia, and Oceania. Both network technologies are extensions of terrestrial network technologies aimed at delivering terrestrial digital TV. Another technology is Multimedia Broadcast Multicast Services [3] which uses GSM/UMTS networks.

Unlike some years ago, we are confronted daily with video fragments and movies either on the Internet or TV. The progress has been made possible, to a large extent, by efficient image/video compression techniques that allow a reduction in the amount of data to be stored and transmitted; therefore, fewer resources were required for this while maintaining image quality. MPEG-2 [4] video, which was standardized in the early 1990s, and the MPEG-4 Visual [5] format (the breakthrough of which was reinforced by the DivX [6] and XviD [7] implementations) have fostered the proliferation of video fragments and digitized movies. More recently, H.264/AVC [8,9] has been standardized. The H.264/AVC standard further reduces the video bitrate at a given quality when compared to previous specifications and can be considered as the reference in video compression.

In the encoding of media streams, it is important to take into account the huge diversity of decoders and players. Multiple devices such as PCs, laptops, smartphones, PDAs, or TVs are often used to play a single video file. Obviously, these devices have widely varying characteristics,

* Correspondence: charo@dsi.uclm.es
[1]Albacete Research Institute of Informatics, University of Castilla-La Mancha, Campus Universitario s/n, 02071 Albacete, Spain
Full list of author information is available at the end of the article

which should be considered when sending media contents. Moreover, reliable reception of video contents by the mobile devices poses additional constraints because of the dynamic nature of the links and the limited resources of the mobile reception devices. In order to be able to deliver the media streams to the widest possible audience, a media communication system should be able to adapt the media streams to the transmission constraints and characteristics of the end-user devices on-the-fly in order to ensure the continuity of high quality image. Such adaptive media communications services are highly relevant for the development of efficient media consumer applications.

Additionally, the diversity of coding standards and formats used in production environments, distribution networks, and broadcast channels necessitates efficient media manipulation techniques. This diversity explains the necessity of media adaptation techniques. One way for easy stream adaptation is using scalable coding schemes. Although provisions for scalable coding were already available for MPEG-2 and MPEG-4 Visual, they are rarely used in practice. Recently, MPEG and VCEG have standardized a new scalable extension of H.264/AVC, which is denoted as *scalable video coding* (SVC) [8,10]. SVC allows the creation of scalable streams with minimal quality loss for the same bitrate when compared to single-layer H.264/AVC, providing different types of scalability such as temporal, spatial, and quality or a combination of them in a flexible manner. This scalability is possible by creating a layered representation of the media stream during the encoding process in which the video is encoded as one base layer and one or more enhancement layers. The base layer contains the lowest frame rate (temporal scalability), the lowest resolution (spatial scalability), and the lowest quality (quality scalability). The enhancement layers provide information for increasing frame rate, resolution or details, and fidelity. To remove redundancy between layers, inter-layer prediction mechanisms are applied. This scalable media coding is an important mechanism not only to provide several types of end-user devices with different versions of the same encoded media stream but also to enable its transmission at various bitrates. The bitstream is adaptable to the channel bandwidth or the terminal's capabilities by truncating the undesired enhancement layers.

Despite these scalability tools, most of the video streams today are still created in a single-layer format (most of them in H.264/AVC), so these existing video streams cannot benefit from the scalability tools in SVC. Due to the fact that the migration from H.264/AVC to SVC is not trivial, given the relatively high computational complexity of the SVC encoding process, it is likely that the dominance of single layer encoders will continue to exist in the near future. However, transcoding techniques exist to make this process more efficient. Transcoding can be regarded as a process for efficient adaptation of media content, in order to match the properties and constraints of transmission networks and terminal devices, by efficiently (re)using information from the incoming bitstream, while at the same time minimizing the quality loss due to the adaptation.

With this challenge in mind, the goal is to develop an efficient H.264/AVC-to-SVC transcoder [11] that is able to transform an H.264/AVC bitstream without temporal scalability into an SVC bitstream with this scalability (which makes it possible to vary the frame rate of the bitstream) faster than a cascade transcoder in baseline and main profiles. Its efficiency is obtained by reusing as much information as possible from the original bitstream, such as mode decisions and motion information, to reduce the encoding SVC time focusing on the mode decision process. One possible application of this proposed transcoder could be on the broadcaster side of a mobile digital TV network to transform already encoded content in H.264/AVC into SVC content (see Figure 1).

The remainder of this paper is organized as follows: in section 2, technical background is described. Section 3 shows the state-of-the-art of H.264/AVC-to-SVC transcoding. In section 4, our approach is depicted. In section 5, the implementation results are shown. Finally, in section 6, conclusions are presented.

2. Technical background

2.1 Scalable video coding

Scalable video coding is an extension of H.264/AVC. SVC streams are composed of layers which can be removed to adapt the streams to the needs of end users or the capabilities of the terminals or the network conditions. The layers are divided into one base layer and one or more enhancement layers which employ data of lower layers for efficient coding. SVC supports three types of scalability:

(1) *Temporal scalability*. The base layer is coded at a low frame rate. By adding enhancement layers, the frame rate of the decoded sequence can be increased.
(2) *Spatial scalability*. The base layer is coded at a low spatial resolution. By adding enhancement layers, the resolution of the decoded sequence can be increased.
(3) *Quality (SNR) scalability*. The base layer is coded at a low quality. By adding enhancement layers, the quality of the decoded sequences can be increased.

Since our proposal focuses on temporal scalability, a brief explanation about this type of scalability is given in this section. For a comprehensive overview of the whole scalable extension of H.264/AVC, the reader is referred to [10].

In a sequence with temporal scalability, the base layer represents the lowest frame rate (with an identifier equal

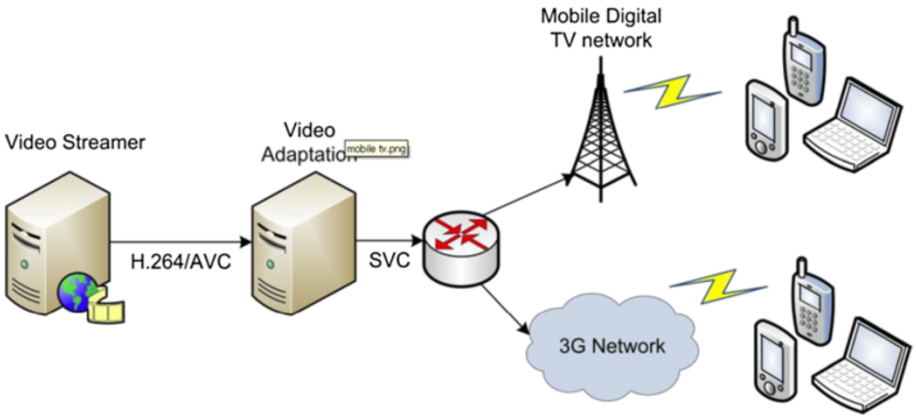


Figure 1 Example of an SVC transcoder for mobile environments.

to 0). With one or more temporal enhancement layers (with identifiers that increase by 1 in every layer), a higher frame rate can be achieved. Figure 2 shows a sequence encoded as four temporal layers. The base layer (layer 0) consists of frames 0 and 8, and provides 1/8 of the original frame rate. Frame 4 lies within the first enhancement temporal layer and, decoded together with layer 0, produces 1/4 of the frame rate of the full sequence. Layer 2 consists of frames 2 and 6; together with layers 0 and 1, it provides a frame rate that is 1/2 of the frame rate of the whole sequence.

Temporal scalability can be achieved using P and B coding tools that are available in H.264/AVC and by extension in SVC. Flexible prediction tools make possible to mark any picture as reference picture so that it can be used for motion-compensated prediction of following pictures. This feature allows coding of picture sequences with arbitrary temporal dependencies. In this way, to achieve temporal scalability, SVC links its reference and predicted frames using hierarchical prediction structures [12,13] which define the temporal layering of the final structure. In this type of prediction structures, the pictures of the temporal base layer are coded in regular intervals using only previous pictures within the temporal base layer as references. The set of pictures between two successive pictures of the temporal base layer together with the succeeding base layer picture is known as a *group of pictures* (GOP). As was mentioned previously, the temporal base layer represents the lowest frame rate that can be obtained. The frame rate can be increased by adding pictures of the enhancement layers.

There are different structures for enabling temporal scalability, but the one used by default in the *Joint Scalable Video Model* (JSVM) reference encoder software [14] is

Figure 2 Sequence with temporal scalability. Distribution of the eight first frames per every layer.

based on hierarchical pictures with a dyadic structure, where the number of temporal layers is thus equal to 1+ $\log_2$[GOP size].

2.2 Mode decision process

In H.264/AVC and its extension SVC, the pictures are divided into *macroblocks* (MBs) that are further split in MB and sub-MB partitions. For every partition, a prediction is created from previously encoded data which is subtracted from the current partition to form a residual. By selecting the best prediction options for an individual MB, an encoder can minimize the residual size to produce a highly compressed bitstream.

H.264/AVC and SVC support both intra-prediction and inter-prediction. Intra-prediction only requires data from the current picture, while inter-prediction uses data from a picture that has previously been coded and transmitted (a reference picture) and is used for eliminating temporal redundancy in P and B frames.

SVC supports motion compensation block sizes ranging from 16 × 16, 16 × 8, and 8 × 16 to 8 × 8, where each of the sub-divided regions is an MB partition. If the 8 × 8 mode is chosen, each of the four 8 × 8 block partitions within the MB may be further split in four ways: 8 × 8, 8 × 4, 4 × 8, or 4 × 4 which are known as sub-MB partitions. Moreover, SVC also allows intra-predicted modes and a skipped mode in inter-frames for referring to the 16 × 16 mode, where no motion and residual information is encoded. Therefore, both H.264/AVC and SVC allow not only the use of the MBs in which the images are decomposed but also the use of smaller partitions by dividing the MBs in different ways. MB and sub-MB partitions for inter-prediction are shown in Figure 3.

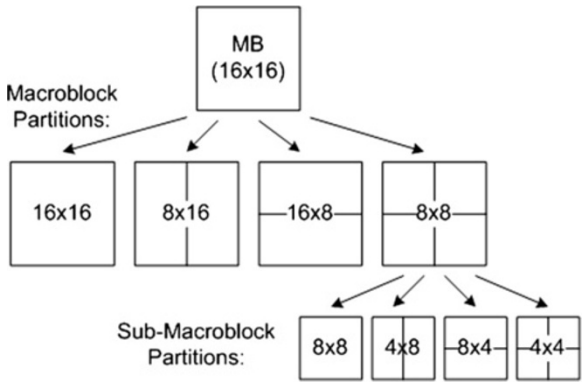


Figure 3 Macroblock and sub-macroblock partitions for inter-prediction.

3. State-of-the-art in H.264/AVC-to-SVC transcoding

As was mentioned above, the scalable extension of H.264/AVC (SVC) was standardized in 2007, so transcoding proposals that involve this standard are recent. Different techniques for transcoding in this framework have been proposed. Most of the proposals are related to quality-SNR scalability, although there are several related to spatial and temporal scalability.

For quality-SNR scalability, the first one [15] was presented in 2006 and performs a transcoding from H.264/AVC to *fine grain scalability* streams. Although it was the first work on this type of transcoding, it does not have much relevance since this technique for providing quality-SNR scalability was removed from the following versions of the standard due to its high computational complexity. In 2007, a transcoding approach from a single layer H.264/AVC bitstream to SNR scalable SVC streams with *coarse grain scalability* (CGS) layers was presented by De Cock et al. in [16]. They proposed an architecture for transcoding to an SVC bitstream with two layers, where depending on the slice and MB type, a distinction is made between spatial and temporal transform-domain compensation. Furthermore, two buffers are provided, one for requantization error values from the current frame and another one for the temporal compensation of inter-predicted MBs. In 2008, De Cock et al. [17] presented a proposal, where the normative bitstream rewriting process implemented in the SVC standard is used to reduce the computational complexity of H.264/AVC to SVC transcoding compared to [16]. It is based on combining the forward and inverse quantization processes. Later, in 2009, De Cock et al. [18] presented different open-loop architectures for transcoding from a single-layer H.264/AVC bitstream to SNR-scalable SVC streams with CGS layers. In 2010, Van Wallendael et al. [19] proposed a simple closed-loop architecture that reduces the time of the mode decision process. This is done using two sources of information: the mode information from the input H.264/AVC video stream and the base layer of SVC that provides information for accelerating the encoding of the enhancement layers. This method is based on the relation between the modes of H.264/AVC, the base layer of SVC, and the enhancement layers of SVC. Then, in 2011, Van Leuven et al. proposed two techniques to improve the previous proposals [20,21]. These methods are based on the same concepts as [19], but improved the results. The first one does so by exploiting more information from the input H.264/AVC bitstream, and the second one by combining open- and closed-loop architectures.

For spatial scalability, a proposal was presented by Sachdeva et al. [22] in 2009. They presented an algorithm for converting a single layer H.264/AVC bitstream to a multi-layer spatially scalable SVC video bitstream, containing layers of video with different spatial resolutions. Using a full-decode full-encode algorithm as its starting point, some modifications are made to reuse the information available after decoding a H.264/AVC bitstream for

motion estimation (ME) with refinement processes on the encoder. Scalability is achieved by an Information Downscaling Algorithm which uses the top enhancement layer (this layer has the same resolution as the original video output) to produce the different spatial layers of the output SVC bitstream.

Finally, for temporal scalability, in 2008, a transcoding method from an H.264/AVC P-picture-based bitstream to an SVC bitstream was presented in [23] by Dziri et al. In this approach, the H.264/AVC bitstream was transcoded to two layers of P-pictures (one with reference pictures and the other with non-reference ones). Then, this bitstream was transformed to an SVC bitstream by syntax adaptation. In 2010, Al-Muscati and Labeau [24] proposed another technique for transcoding that provided temporal scalability. The method presented was applied in baseline profile and reused information from the mode decision and ME processes from the H.264/AVC stream. During that year, we presented an H.264/AVC to SVC video transcoder that efficiently reuses some motion information of the H.264/AVC decoding process in order to reduce the time consumption of the SVC encoding algorithm by reducing the motion estimation process time. The approach was developed for main profile [25] and dynamically adapted for several temporal layers. Later, in 2011, the previous algorithm was adjusted for the baseline profile and P frames [26]. In 2012, Yeh et al. proposed another technique [27] for transcoding from H.264/AVC to SVC using probability models and Markov chains, and we presented another work [28,29] focusing on accelerating the mode decision algorithm, while our previous approaches focused only on motion estimation process. The present work is an extension of these last ones.

4. Proposed low-complexity video transcoding

4.1 Observations and motivation

In H.264/AVC and its extension SVC, the pictures are divided into MBs that are further split into MB and sub-MB partitions. For every partition, a prediction is created from previously encoded data, which is subtracted from the current partition to form a residual. By selecting the best prediction options for an individual MB, an encoder can minimize the residual size to produce a highly compressed bitstream. So, in the encoding process, the encoder has to check all MBs and sub-MBs to determine the best option. SVC supports *motion compensation* block sizes ranging from 16 × 16, 16 × 8, and 8 × 16 to 8 × 8, where each of the sub-divided regions is an MB partition. If the 8 × 8 mode is chosen, each of the four 8 × 8 block partitions within the MB may be further split in four ways: 8 × 8, 8 × 4, 4 × 8, or 4 × 4, which are known as sub-MB partitions. Moreover, SVC also allows intra-predicted modes and a skipped mode in inter-frames for referring to the 16 × 16 mode, where no motion and residual information is encoded. This process was explained in more detail in [1,9,10].

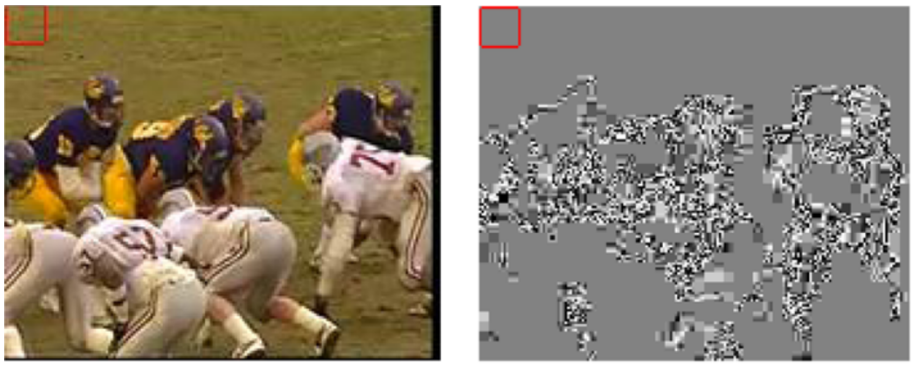
To search all inter- and intra-modes exhaustively in order to select the best for each MB, the SVC encoder part of the H.264/AVC-to-SVC transcoder takes a large amount of time; therefore, this is one of the tasks that can be accelerated to reduce the transcoding time. Although the prediction structures (and, as a result, the frames used as a reference) of H.264/AVC without temporal scalability and SVC are not the same, some data generated by H.264/AVC and transmitted in the encoded bitstream can help us find the best partitioning structure. For example, Figure 4 shows the correlation between the residual and *motion vector* (MV) length calculated in H.264/AVC with respect to the MB coded partition performed in SVC. In this case, we observed that stationary areas or objects with slow motion are often coded in MBs without sub-blocks (such as 16 × 16, 16 × 8, or 8 × 16) or even as skipped, where the MB is copied from the reference one. On the other hand, the regions with sudden changes (scene, light, an object that appears) are coded in inter-modes with lower MB mode partitions (such as 4 × 8, 8 × 4, and 4 × 4) or even in intra-mode. Moreover, we also found a high correlation between the length of the MVs calculated by H.264/AVC and the final MB mode decision, where long MVs suggest a more complicated MB partition such as 4 × 4, while shorter MVs lead to simpler MB partitions. These relationships can be observed in Figure 4 as well.

Taking this into account, it is possible to exploit this correlation using *machine learning* (ML) techniques [30] to build a decision tree which decides the SVC decision mode depending on the values of certain information extracted from the H.264/AVC decoding stage. Thus, the SVC mode decision task becomes a lookup into a decision tree with very low complexity.

To build this decision tree, the informations that need to be extracted from the H.264/AVC decoder process are as follows:

- Residual. The amount of residual of every block of 4 × 4 pixels is used by the decoder to reconstruct the decoded MB, so this information will be available in the decoding process. For our purpose, only the residual data of the luma component was extracted.
- MVs. This information is also available in the decoding process. The MVs of each MB were extracted.
- Mode decision of H.264/AVC. The MB partitioning of each MB in H.264/AVC is related to the residual and the MVs, and can give us valuable information.

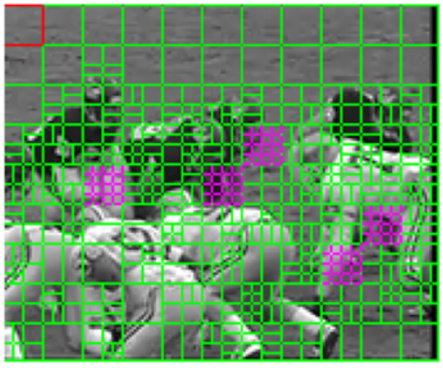
The main goal of this proposal was to reduce the time spent by this mode decision process, trying to narrow down the set of MB partitions to be checked by the encoder using a decision tree generated by data mining techniques.



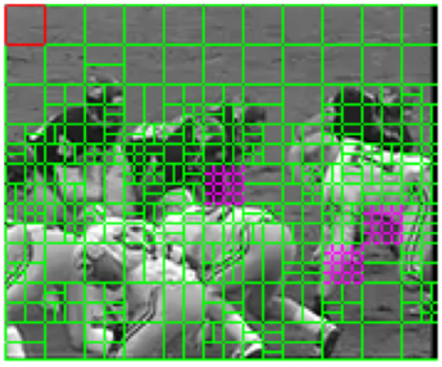
(a) Original frame

(b) Residual H.264/AVC

(c) MVs in H.264/AVC



(d) MB mode decision in H.264/AVC



(e) MB mode decision in SVC

Figure 4 Correlation among residual, MVs, and MB mode decision. (**a**) Original frame. (**b**) Residual H.264/AVC. (**c**) MVs in H.264/AVC. (**d**) MB mode decision in H.264/AVC. (**e**) MB mode decision in SVC.

4.2 Machine learning

ML refers to the study of algorithms and systems that learn or acquire knowledge from experiences. It uses statistics with different kinds of algorithms to solve a problem by studying and analyzing the data. There are two types of learning: inductive and deductive learning. In inductive learning, a synthesis of the knowledge is carried out, while in deductive learning an analysis of existing knowledge is performed in order to improve this knowledge and transform it into a form that is easier or more efficient to use. This information can be used to build a decision tree for taking decisions, which is built using the training data mentioned previously.

The decision tree is made by mapping the observations about a set of data and applying a divide-and-conquer approach to the problem. It is composed of nodes represented by circles and branches which are represented by segments connecting the nodes. Routing down the tree,

the end nodes are named leafs. The nodes involve testing a particular attribute. Leaf nodes give a classification that applies to all instances that would reach the leaf. To classify an unknown instance, the tree is routed down according to the values of the attributes tested in successive nodes, and when a leaf is reached, the instance is classified according to the class assigned to the leaf.

ML techniques have been used in an extensive range of applications including web mining, medical diagnosis, marketing and sales, speech and writing recognition, automation, identifying the genes within a new genome, etc. The use of these techniques in the areas of image and video has focused on detection of hazards or certain characteristics. Moreover, in some transcoding approaches, ML has been used [31,32], although these approaches focus on transcoding from several different standards to H.264/AVC.

In this paper, ML has been used to reduce the complexity of the mode decision process in the H.264/AVC-to-SVC transcoding proposed. In this framework, ML tools were used in order to create rules from the relationships between certain data extracted from the H.264/AVC decoding process and the MB mode partitioning of SVC (this could be seen as the variable to understand). Using these rules instead of the MB partition algorithm of the SVC encoder, this process can be speeded up. In this paper, a decision tree with three levels of decision is presented. This decision tree narrows down the mode decisions that can be chosen by the standard.

Figure 5 depicts the process for building the decision trees to be used in the H.264/AVC-to-SVC transcoding process. The H.264/AVC video is decoded and some information such as the residual, MV lengths, and MB modes are saved. The decoded H.264/AVC video is then encoded using the SVC standard, and the coding mode of the corresponding MB is also saved. Using these data, an ML algorithm is run to create decision trees that classify an MB into one of the several SVC MB coding modes.

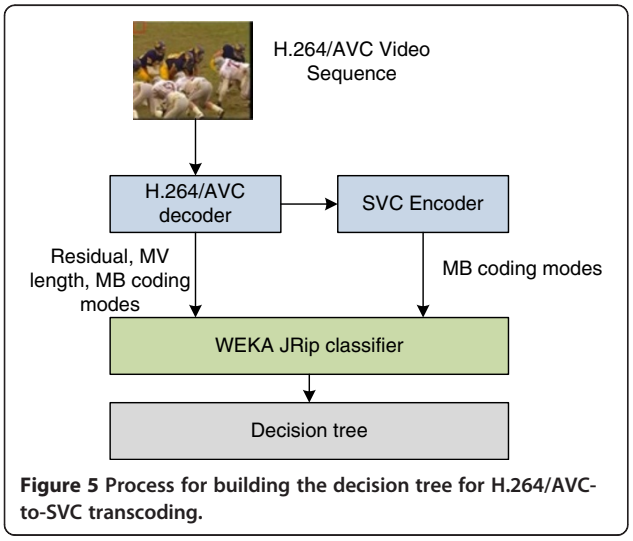


Figure 5 Process for building the decision tree for H.264/AVC-to-SVC transcoding.

```
@relation 'football-2ndlevel-trainning'

@attribute vectorlengthL0 numeric
@attribute residual16x16 numeric
@attribute MBtypeAVC {0,1,2,3,8,9,10,11}
@attribute meansofvariances4x4 numeric
@attribute varianceofmeans4x4 numeric

@attribute class {0,1}

@data
16.40,1687.00,8,33.80,57.87,1
0.71,1715.00,2,95.19,68.55,1
1.00,132.00,1,1.86,0.73,0
```

Figure 6 ARFF file format example.

In this case, the Waikato Environment for Knowledge Analysis (WEKA) software [33] was used. WEKA is a collection of ML algorithms for data mining tasks and contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.

The information gathered from H.264/AVC together with the SVC encoder mode decision was introduced in WEKA; then, an ML classifier was run. The way to introduce the datasets in WEKA is using the *Attribute-Relation File Format* (ARFF) files. An example of an ARFF file is shown in Figure 6. This text file contains the dataset to be classified, and the relationship between a set of attributes is shown. This file has two parts:

- The header with the information about the name we give to the relation (@relation) and the definition of the attributes that are used and their types (@attribute). Nominal attributes are followed by the set of values they can take, while numeric values are followed by the keyword *numeric*.
- The data section which starts with @data signals the starts of the instances in the dataset. The instances are written one per line, with values for each attribute, separated by commas.

In Figure 6, the variable to classify is the attribute class (@attribute class {0,1}), which represents a set of possible MB coding modes for SVC. In this case, the decision tree developed for accelerating mode decision will be a binary tree (this decision will be explained in the following section), so the possible values of the attribute class are '0' or

'1'. The rest of the attributes will be used to decide the value of the variable class, and the lines below the label @data represent the values of the variables in each MB (one line for each one). The final goal is to find a simple structure to show the possible dependencies between the attribute class and the others for building a decision tree with these relationships. More details about the values of the attributes included in the ARFF files of the proposal will be provided in the next section.

This data mining procedure has to be carried out just once in an off-line training process. Once the knowledge has been extracted as a decision tree, it will be implemented in the proposed H.264/AVC-to-SVC transcoder.

4.3 Low-complexity transcoder

The main idea of this low-complexity transcoder is to build a decision tree that uses information from the decoding process of H.264/AVC and, depending on these values, narrows down the number of MB types to be checked by the SVC encoder in both baseline and main profiles.

As was said above, using ML techniques will make it possible to exploit the correlation between different variables in H.264/AVC and the MB decision mode; so, in this framework, ML is used in order to create rules from these relationships to narrow down the MB types that the SVC encoder has to check. A scheme of the proposal is shown in Figure 7.

For every MB, the extracted information is used to generate the decision tree (and then to decide the MB partitioning). Some operations and statistics are calculated for this data. The steps for generating the decision tree are the following:

1. Extract information for each MB in the decoder process: residual, MV length, and MB type.
2. Calculate operations and statistics for these data:

- Residual of the whole MB: The residual of all the 4 × 4 blocks of pixels (*res4x4*) within the MB are added.

$$\text{Residual } 16\times 16 = \sum_{i=1}^{16} \text{Res4}\times 4_i \tag{1}$$

- Length of the average of the MVs of an MB: First of all, the mean of each component of all the MVs in the H.264/AVC MB and sub-MB is calculated. This MV is the MV of the MB that we will use. Then, the length of the resulting MV is calculated.

$$\text{MV}_{x\text{mean}} = \frac{1}{n}\sum_{i=1}^{n} \text{MV}_{xi} \tag{2}$$

$$\text{MV}_{y\text{mean}} = \frac{1}{n}\sum_{i=1}^{n} \text{MV}_{yi} \tag{3}$$

$$\text{Vector length} = \sqrt{\text{MV}_{x_{\text{mean}}}^2 + \text{MV}_{y_{\text{mean}}}^2} \tag{4}$$

- Variance of means of the residual of 4 × 4 blocks within an MB: For every block of 4 × 4 pixels, the mean of the residuals of its 16 pixels (*respixel*) is calculated (*mean4x4*). Then, the variance of these means with respect to the mean of the residual of the whole MB (*residual16x16*) is calculated.

$$\text{Mean4}x4_i = \frac{1}{16}\sum_{j=1}^{16} \text{Respixel}_j \quad \forall i \in [1,16] \tag{5}$$

$$\text{Variance of means } 4\times 4 = \frac{1}{16}\sum_{i=1}^{16} \left(\text{Mean } 4\times 4_i - \overline{\text{Residual } 16\times 16}\right)^2 \tag{6}$$

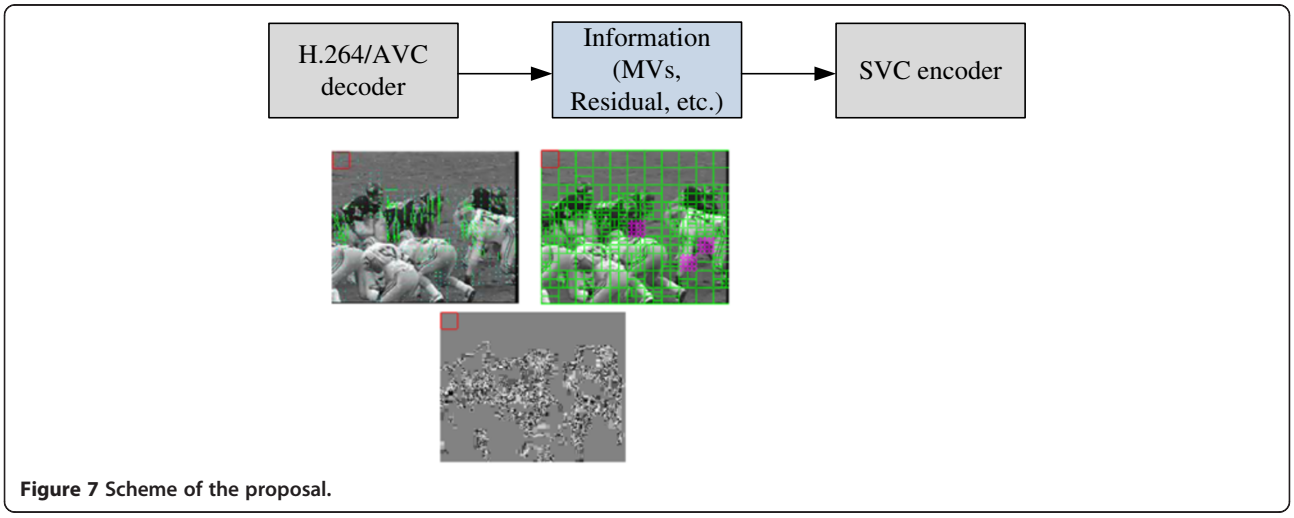


Figure 7 Scheme of the proposal.

- Mean of variances of the residual of 4 × 4 blocks within an MB: For every block of 4 × 4 pixels, the variance of the residuals of its pixels (*respixel*) with respect to the mean of the residuals of this 4 × 4 block (*mean4x4*) is calculated. Then, the mean of the variances resulting from this process is calculated.

$$\text{Variance } 4\times 4_i = \frac{1}{16}\sum_{j=1}^{16}\left(\text{Respixel}_j - \text{Mean } 4\times 4_i\right)^2 \quad \forall i \in [1,16] \tag{7}$$

$$\text{Mean of variances } 4\times 4 = \frac{1}{16}\sum_{i=1}^{16}\text{Variance } 4\times 4_i \tag{8}$$

3. Extract the final MB partition of SVC as a variable. As the decision tree will be a binary tree, this value will be transformed into a '0' or a '1' to indicate to which group of each level the MB type belongs. The choice of a binary tree is due to the possibility of exploiting the similarity between groups of partitions, using the decision tree to narrow down the partitions that the encoder has to check, but not deciding the mode of the MB exactly.

All this information was put together in an ARFF file (as in Figure 6) where the different variables needed were defined as attributes and each line after the @data label represents the information relating to an MB. This file is called the training file and serves to generate a decision tree. For this purpose, after constructing the ARFF file with the necessary data, a classifier algorithm from the ones implemented in WEKA was run to obtain the decision tree. After extensive experimentation, sequences that contain regions varying from homogeneous to high-detail serve as good training sets. In this case, the *Football* QCIF sequence was used to build the training file, and the classifier algorithm chosen was the JRIP algorithm [30] because it was the algorithm that obtained the major quantity of correct decisions. Owing to the differences between the prediction structure of H.264/AVC without temporal scalability and the SVC prediction structure explained above in section 2.1, the decision tree was built by only using the information contained in frames within the enhancement temporal layer with the highest identifier because the structure in the two bitstreams (H.264/AVC without temporal scalability and SVC) is very similar in this part.

This tree was generated with the information available after the decoding process and does not focus on the final MB partition but reduces the set of final MBs that can be chosen by the SVC encoder. This is shown in Figure 8, where the white circles represent the set of MB partitions that the reference standard can choose from.

This final decision tree was generated by levels, taking into account the similarity between groups of partitions as was mentioned previously. It has three levels divided as follows:

- First level. Discriminates between 'low' {SKIP, 16 × 16, 16 × 8, 8 × 16} and 'high complexity' {INTRA, 8 × 8, 8 × 4, 4 × 8, 4 × 4} modes.
- Second level. Inside the 'low complexity' bin, a decision between {SKIP, 16 × 16} or {16 × 8, 8 × 16} is made.
- Third level. Inside the 'high complexity' bin, a decision between {8 × 8, 8 × 4, 4 × 8} or {4 × 4, INTRA} is made.

It was necessary to develop a decision tree for baseline profile and another for main profile because the prediction structure changes (IPPP in baseline profile and IBBP in main profile in H.264/AVC), and in the baseline profile, only the vectors in list 0 were used, while in main profile, a new component (MVs in list 1) is included.

As example of the whole decision tree, the first level for baseline and main profiles is shown in Figure 9. This first level and the other ones (second and third) were implemented in the SVC encoder part of the transcoder, efficiently replacing the more complex MB coding mode decision of SVC.

This decision tree was used for mode decision task with different sequences (*Hall*, *City*, *Foreman*, *Soccer*, *Harbor*, and *Mobile*) and classified correctly in about 87% of cases in the first level, 80% in the second level, and 93% in the third level in baseline profile, and 91% of the cases in the first, 84% in the second, and 89% in the third level in main profile, as is shown in Table 1.

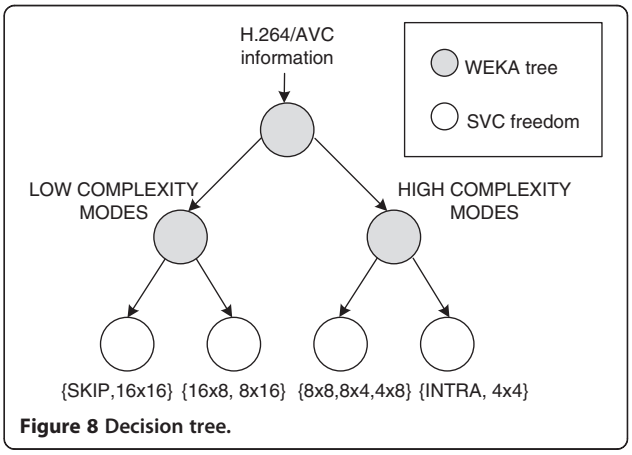
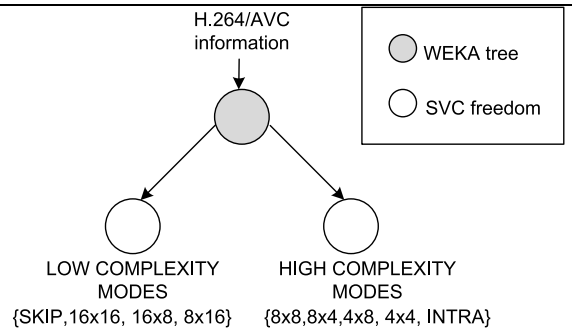


Figure 8 Decision tree.

1st level



Baseline: 1st level

```
if ((residual16x16 >= 1963) && (MBtypeAVC == I4MB))
                return HIGH_COMPLEXITY_MODES;

if ((varianceofmeans4x4 >= 7.71) && (MBtypeAVC == 8x8))
                return HIGH_COMPLEXITY_MODES;

if ((varianceofmeans4x4 >= 5.16) && (vectorlengthL0 <= 13.89) && (MBtypeAVC == 8x8))
                return HIGH_COMPLEXITY_MODES;

if ((residual16x16 >= 1016) && (MBtypeAVC == I4MB))
                return HIGH_COMPLEXITY_MODES;

if ((residual16x16 >= 2076) && (varianceofmeans4x4 >= 265.07))
                return HIGH_COMPLEXITY_MODES;

if ((residual16x16 >= 588) && (MBtypeAVC == I4MB) && (residual16x16 <= 676))
                return HIGH_COMPLEXITY_MODES;

if ((varianceofmeans4x4 >= 6.78) && (residual16x16 >= 2278) && (meansofvariances4x4
                                <= 60.09))
                return HIGH_COMPLEXITY_MODES;

if ((varianceofmeans4x4 >= 4.47) && (residual16x16 >= 1016) && (meansofvariances4x4
                >= 119.55) && (vectorlengthL0 >= 13.45))
                return HIGH_COMPLEXITY_MODES;

                        In any other case:
                return LOW_COMPLEXITY_MODES;
```

Main: 1st level

```
if ((varianceofmeans4x4 >= 27.82) && (residual16x16 >= 2373) && (MBtypeAVC == I4MB))
                return HIGH_COMPLEXITY_MODES;

if ((varianceofmeans4x4 >= 15.09) && (residual16x16 >= 2006) && (MBtypeAVC == 8x8))
                return HIGH_COMPLEXITY_MODES;

                if (varianceofmeans4x4 >=119.18)
                return HIGH_COMPLEXITY_MODES;

if ((MBtypeAVC == 8x8) && (varianceofmeans4x4 >= 38.02))
                return HIGH_COMPLEXITY_MODES;

if ((residual16x16 >= 1591) && (MBtypeAVC == I4MB) && (meansofvariances4x4 <= 42.27)
                && (varianceofmeans4x4 >= 39.47))
                return HIGH_COMPLEXITY_MODES;

if ((meansofvariances4x4 >= 9.39) && (residual16x16 >= 1063) && (vectorlengthL1 >=
                4713.57) && (MBtypeAVC == I4MB))
                return HIGH_COMPLEXITY_MODES;

                        In any other case:
                return LOW_COMPLEXITY_MODES;
```

Figure 9 First level decision tree for baseline (top) and main (down) profiles.

Table 1 Percentage of correct choice of MB group

Sequence	Percentage of classification in the correct MB group					
	Baseline profile			Main profile		
	First level	Second level	Third level	First level	Second level	Third level
Hall	96.83	97.27	93.25	96.10	96.56	95.94
City	92.34	82.35	88.60	96.49	86.87	96.36
Foreman	87.84	79.46	93.00	92.26	81.83	82.14
Soccer	88.25	86.50	88.88	86.52	83.15	74.64
Harbor	80.23	66.86	94.55	88.75	77.33	91.24
Mobile	79.14	67.00	99.24	86.00	78.96	99.11
Average	87.44	79.91	92.92	91.02	84.12	89.91

This decision tree, as shown in Figure 9, is composed of a set of thresholds for the H.264/AVC residual and for the statistics related to it. Since the MB mode decision, and hence the thresholds, depends on the quantification parameter (QP) used in the H.264/AVC stage, the residual, the mean and the variance thresholds will be different at each QP. At this point, there are two different solutions:

- Develop different decision trees for each QP and use the corresponding tree for each case.
- Develop a single decision tree and adjust the thresholds based on the QP.

The first option is rather complex because it leads to the implementation of a lot of WEKA decision trees. The solution adopted was the second one, to develop a single decision tree for a QP and adjust the mean and the variance threshold used by the trees on the basis of the QP.

Since the relationship between the quantization step size and the QP is well known (see Figure 10), an adjustment in the decision tree can be performed. The proposed transcoder uses a single decision tree developed for a mid-QP of 28 which is later adjusted for other QPs (32, 36, and 40). Since the quantization step size doubles when QP increases by 6, the thresholds are adjusted by 12.5% for a change in QP of 1.

5. Performance evaluation

In order to evaluate the fast MB mode decision approach described above, the proposal has been implemented in a SVC encoder based on JSVM software [14]. The results of this implementation are shown in this section.

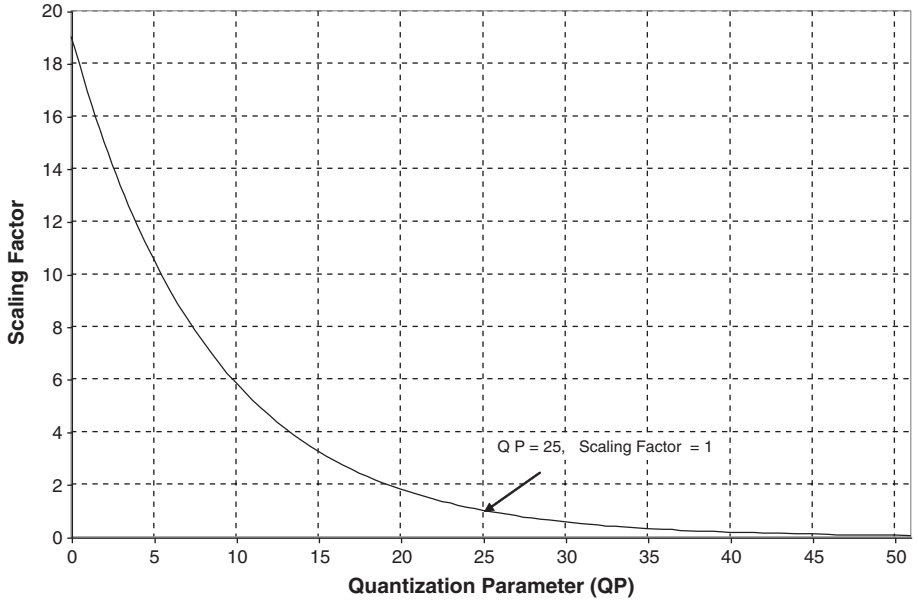


Figure 10 Scaling factor for the decision tree [34].

Table 2 RD performance and time savings of the approach for GOP = 2 and different resolutions

RD performance and time savings of the H.264/AVC-to-SVC transcoder								
GOP = 2 (baseline profile)								
Sequence	QCIF (15 Hz)				CIF (30 Hz)			
	ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)		ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)	
			Full seq.	Partial			Full seq.	Partial
Hall	0.042	−0.05	57.96	85.38	0.055	−0.08	58.94	86.64
City	0.026	0.92	57.16	84.16	0.055	0.25	58.24	85.61
Foreman	0.077	1.21	56.20	82.70	−0.059	1.51	58.12	85.46
Soccer	0.036	1.45	54.34	79.86	0.021	1.28	56.28	82.85
Harbor	0.022	−0.13	52.91	77.95	0.047	−0.35	56.12	80.58
Mobile	0.033	−0.15	52.28	76.93	0.080	−1.10	54.51	80.09
Average	0.039	0.54	55.14	81.16	0.033	0.25	57.03	83.54

5.1 Scenario and metrics

Experiments were conducted to evaluate the performance of the proposed algorithm when transcoding videos using test sequences of varying characteristics namely Hall, City, Foreman, Soccer, Harbor, and Mobile in CIF (30 Hz) and QCIF resolutions (15 Hz). These sequences were encoded using the H.264/AVC *Joint Model* (JM) reference software [35], version 16.2, with an IPPP and IBBP pattern with a fixed QP = 28 in a trade-off between quality and bitrate.

Then, for the reference results, the encoded bitstreams are decoded and re-encoded using the JSVM software, version 9.19.3 [14] with temporal scalability, baseline and main profiles, and different values for QP (28, 32, 36, 40). For the results of the proposal, encoded bitstreams in H.264/AVC are transcoded using the technique described in the previous section, and different GOP lengths (2, 4, 8, 16, and 32) were used.

Since most of the SVC encoding time is spent on the temporal enhancement layers with the two highest identifiers as shown in [25,26], our approach will be applied to these temporal layers and the remaining temporal layers will be decoded and re-encoded completely. If there is only one temporal enhancement layer, it will be applied only to this one to avoid changes in the base temporal layer. In a mathematical way, our technique will be applied to the temporal layers that satisfy the condition:

$$n = \log_2(\mathrm{GOP}_{\mathrm{size}}) - k, \text{with } n > 0 \text{ and } k \in \{0, 1\}, \quad (9)$$

where n is the identifier of the temporal layer, and k varies between 0 and 1.

The metrics used to evaluate the proposed video transcoder are the RD function (bitrate vs. peak signal-to-noise ratio (PSNR)), Δbitrate (%), ΔPSNR (dB), and time saving (%). These metrics are defined below:

- RD function. Rate distortion gives theoretical bounds on the compression rates that can be achieved using different methods. In rate distortion theory, the rate is usually understood as the number of bits per data sample to be stored or transmitted. The notion of distortion is a subject of on-going discussion. In the simplest case (which is actually

Table 3 RD performance and time savings of the approach for GOP = 4 and different resolutions

RD performance and time savings of the H.264/AVC-to-SVC transcoder								
GOP = 4 (baseline profile)								
Sequence	QCIF (15 Hz)				CIF (30 Hz)			
	ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)		ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)	
			Full seq.	Partial			Full seq.	Partial
Hall	0.219	0.04	74.58	85.80	0.328	−0.45	74.69	86.45
City	0.064	1.93	75.69	86.04	0.200	0.66	76.30	86.96
Foreman	0.251	2.34	72.68	83.55	−0.112	3.01	74.63	85.65
Soccer	0.043	2.24	72.11	81.83	0.021	2.37	72.35	83.05
Harbor	0.107	−0.68	68.30	78.88	0.175	−1.22	71.75	81.57
Mobile	0.142	0.15	65.37	76.51	0.229	−1.69	69.83	80.37
Average	0.138	1.00	71.46	82.10	0.140	0.45	73.26	84.01

Table 4 RD performance and time savings of the approach for GOP = 8 and different resolutions

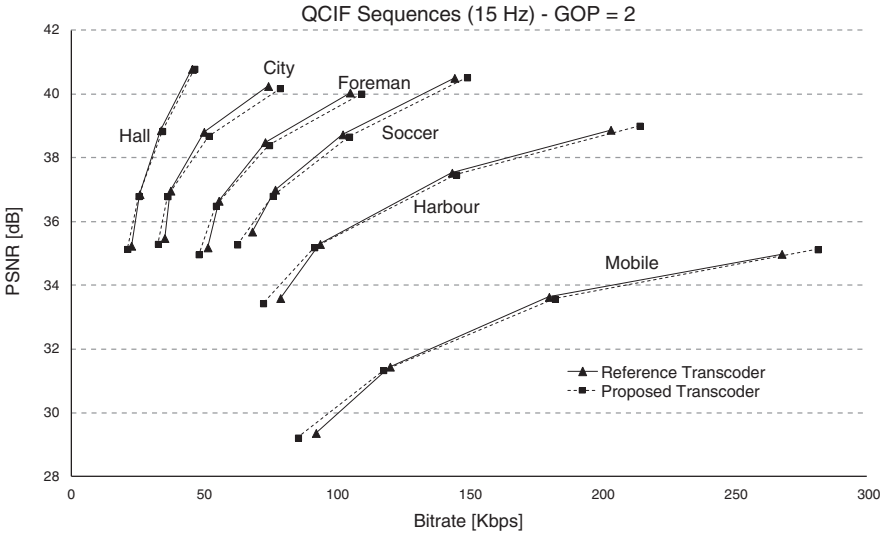
RD performance and time savings of the H.264/AVC-to-SVC transcoder								
GOP = 8 (baseline profile)								
Sequence	QCIF (15 Hz)				CIF (30 Hz)			
	ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)		ΔPSNR (dB)	ΔBitrate (%)	Time Saving (%)	
			Full seq.	Partial			Full seq.	Partial
Hall	0.158	0.37	70.59	86.28	0.025	0.47	70.69	86.83
City	−0.008	2.67	70.16	85.70	0.175	1.32	70.10	86.16
Foreman	0.210	3.22	66.89	82.89	−0.001	3.58	69.96	85.91
Soccer	0.074	2.61	65.19	80.63	−0.001	2.99	68.07	83.55
Harbor	0.048	0.15	64.60	79.54	0.072	−0.18	65.54	80.60
Mobile	0.031	0.87	64.82	79.36	0.233	−0.84	65.81	81.10
Average	0.086	1.65	67.04	82.40	0.084	1.22	68.36	84.02

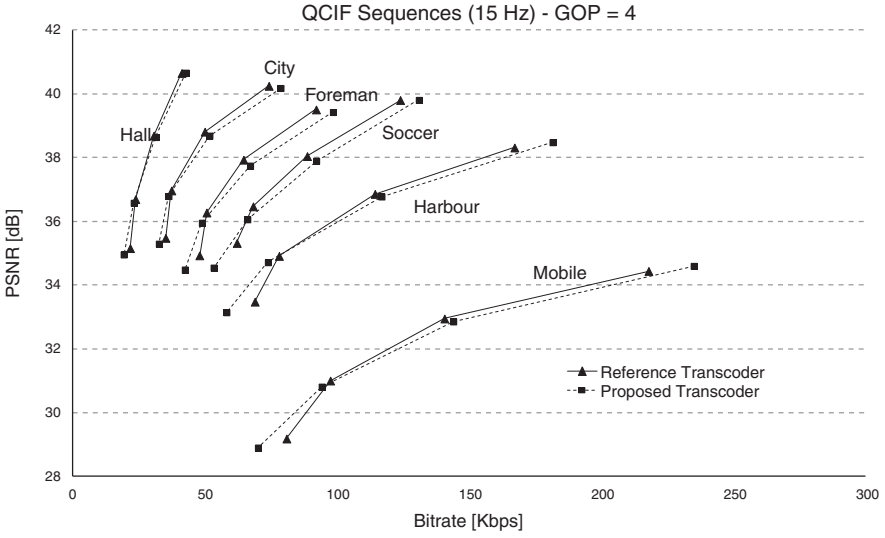
Table 5 RD performance and time savings of the approach for GOP = 16 and different resolutions

RD performance and time savings of the H.264/AVC-to-SVC transcoder								
GOP = 16 (baseline profile)								
Sequence	QCIF (15 Hz)				CIF (30 Hz)			
	ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)		ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)	
			Full seq.	Partial			Full seq.	Partial
Hall	0.325	0.58	69.47	85.97	−0.673	1.90	67.61	86.89
City	−0.040	3.14	69.01	85.45	−0.140	1.96	67.16	86.39
Foreman	−0.333	3.36	65.30	82.47	−0.104	4.86	66.74	85.88
Soccer	0.068	3.03	66.02	81.60	0.031	3.60	65.29	83.83
Harbor	0.199	0.99	65.13	80.51	0.280	2.43	62.78	81.15
Mobile	0.024	1.18	63.31	79.07	0.218	0.17	63.41	81.66
Average	0.041	2.05	66.37	82.51	−0.065	2.49	65.50	84.30

Table 6 RD performance and time savings of the approach for GOP = 32 and different resolutions

RD performance and time savings of the H.264/AVC-to-SVC transcoder								
GOP = 32 (baseline profile)								
Sequence	QCIF (15 Hz)				CIF (30 Hz)			
	ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)		ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)	
			Full seq.	Partial			Full seq.	Partial
Hall	0.291	1.16	66.84	86.94	0.756	1.17	66.62	86.39
City	−0.192	3.64	66.89	85.93	−0.104	2.77	66.27	85.99
Foreman	−0.116	5.51	63.75	82.56	−0.264	5.31	66.05	85.59
Soccer	0.073	4.53	63.92	81.96	0.019	3.97	64.59	83.52
Harbor	0.122	2.41	61.86	80.05	−0.009	2.46	62.64	81.25
Mobile	0.039	2.25	61.59	79.70	0.158	1.62	62.65	81.46
Average	0.036	3.25	64.14	82.86	0.093	2.88	64.80	84.03





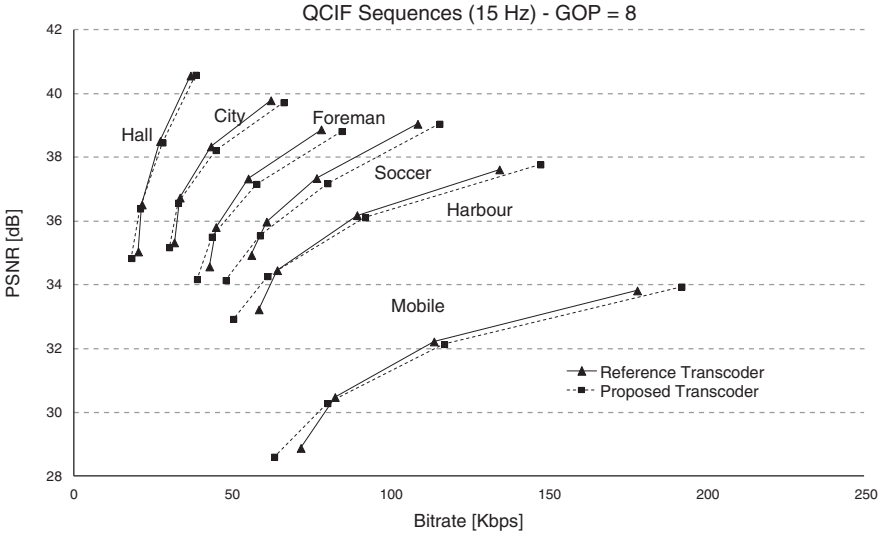


Figure 11 RD performance for QCIF sequences with different GOP sizes (baseline profile).

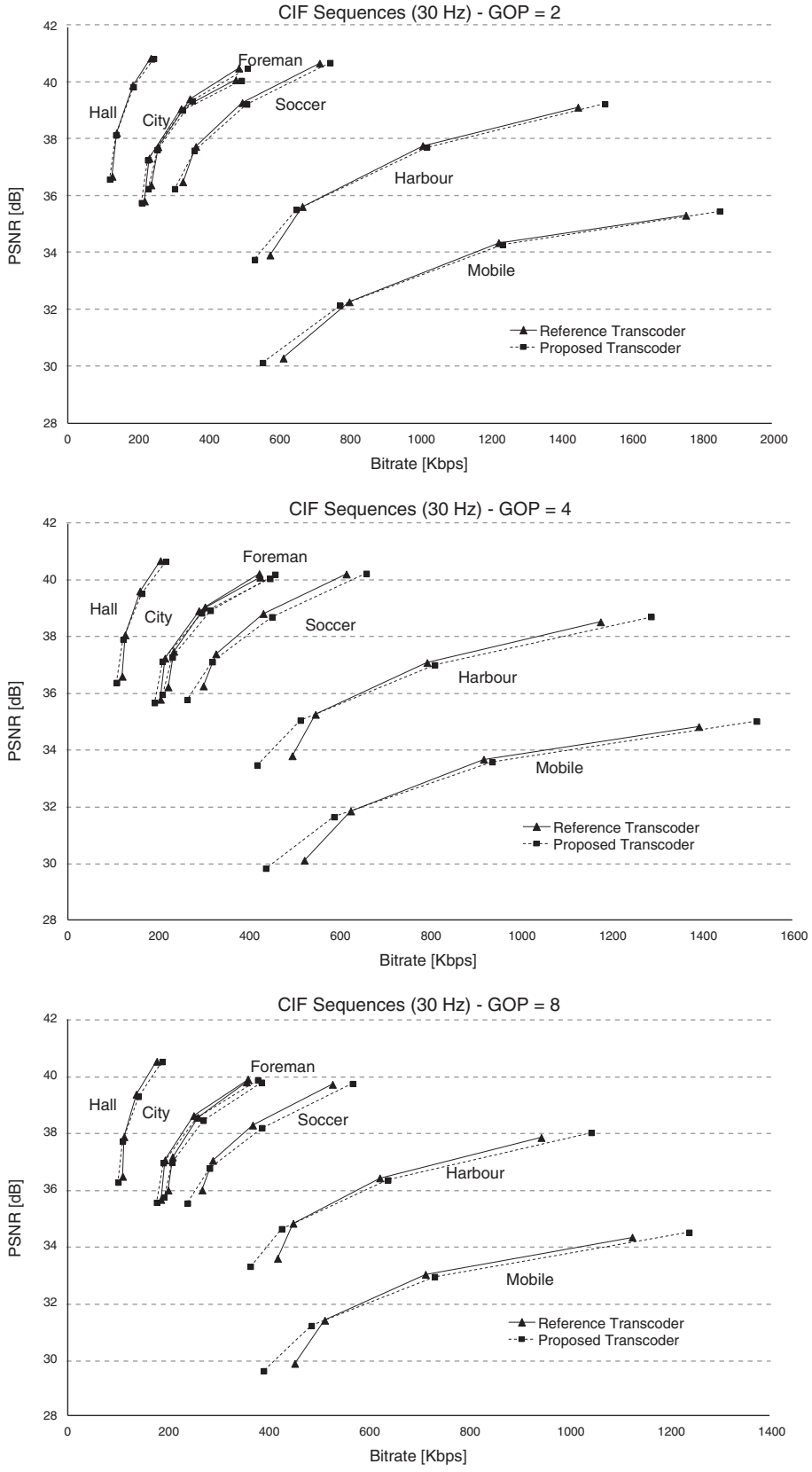
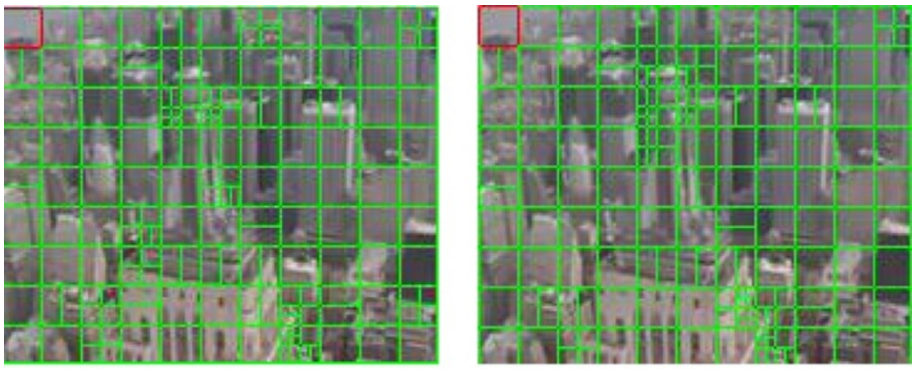
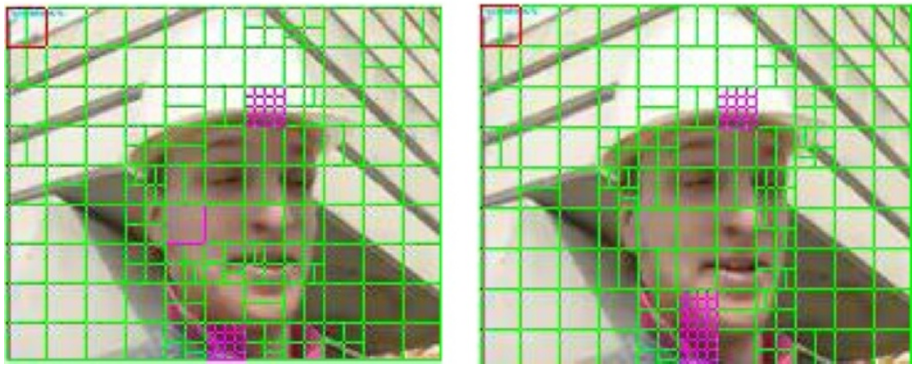


Figure 12 RD performance for CIF sequences with different GOP sizes (baseline profile).



(a) 2nd frame of City sequence (1st P-frame)



(b) 2nd frame of Foreman sequence (1st P-frame)

Figure 13 Comparison of the MB partitioning in baseline profile: proposed (left) and reference (right). (**a**) Second frame of City sequence (first P-frame). (**b**) Second frame of Foreman sequence (first P-frame).

used in most cases), the distortion is defined as the variance of the difference between the input and the output signals (i.e., the mean squared error of the difference). In the definition of the RD function used to show the performance results, PSNR is the distortion for a given bitrate. The averaged PSNR values of luminance (Y) and chrominance (U, V) are used in the RD function graphs to see the general performance. The averaged global PSNR is based on Equation 10.

$$\overline{\mathrm{PSNR}} = \frac{4 \cdot \mathrm{PSNR}_Y + \mathrm{PSNR}_U + \mathrm{PSNR}_V}{6} \tag{10}$$

- ΔPSNR (dB) and ΔBitrate (%). The detailed procedures for calculating these differences can be

Table 7 RD performance and time savings of the approach for GOP = 2 and different resolutions

RD performance and time savings of the H.264/AVC-to-SVC transcoder								
GOP = 2 (main profile)								
Sequence	QCIF (15 Hz)				CIF (30 Hz)			
	ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)		ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)	
			Full seq.	Partial			Full seq.	Partial
Hall	0.064	−0.39	58.13	87.06	0.069	−0.48	58.00	85.84
City	0.061	0.34	59.70	87.92	0.049	−0.41	59.39	88.15
Foreman	−0.004	1.51	57.17	86.76	−0.039	1.42	59.74	88.30
Soccer	−0.016	3.46	62.28	81.24	0.133	1.16	59.36	88.10
Harbor	0.068	−0.52	58.18	85.36	0.055	−0.77	57.20	84.90
Mobile	0.016	−0.13	57.83	85.05	−0.005	1.71	58.89	86.90
Average	0.032	0.71	58.88	85.57	0.044	0.44	58.76	87.03

Table 8 RD performance and time savings of the approach for GOP = 4 and different resolutions

RD performance and time savings of the H.264/AVC-to-SVC transcoder								
GOP = 4 (main profile)								
Sequence	QCIF (15 Hz)				CIF (30 Hz)			
	ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)		ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)	
			Full seq.	Partial			Full seq.	Partial
Hall	0.166	0.15	77.92	88.55	0.230	−1.30	76.24	88.17
City	0.141	2.07	77.47	88.49	0.017	0.36	76.29	88.23
Foreman	0.048	3.46	77.21	88.08	−0.016	3.20	76.51	88.35
Soccer	−0.095	5.42	74.86	85.97	0.079	3.45	75.47	87.06
Harbor	0.171	−0.40	75.56	86.49	0.136	−0.83	74.56	86.35
Mobile	0.041	0.76	74.49	85.66	0.097	−0.63	74.20	85.94
Average	0.079	1.91	76.25	87.21	0.091	0.71	75.55	87.35

found in a *Joint Video Team* document authored by Sullivan and Bjøntegaard [36]. This mechanism is proposed for finding numerical averages between RD curves as part of the presentation of the results. ΔPSNR represents the difference in quality (negative means quality loss), and Δbitrate represents the bitrate increment (positive means that bitrate increases). For these metrics, only the values of the luminance are used as indicated in [36].

- Time saving (%). In order to evaluate the complexity reduction achieved by the proposal compared to the reference transcoder, the following calculation is defined to find the time differences. Let T_{ref} denote the coding time used by the H.264/AVC reference software, and T_{prop} be the time taken by the algorithm proposed or the mechanism that has been evaluated. Time saving is defined in Equation 11. In T_{prop} the full computational cost for the operations needed to prepare the information for the approach is also included.

In the proposal presented in this paper, there are two different time savings calculated:

Full sequence. This is the time reduction for the whole sequence when our proposal is applied.

Partial. This is the time reduction for the temporal layers which the proposal is applied to.

$$\text{Time saving } (\%) = \frac{T_{\text{ref}} - T_{\text{prop}}}{T_{\text{ref}}} \cdot 100. \tag{11}$$

This performance evaluation includes a metric which allows a visual comparison of the MB mode decision chosen by the decision tree and the MB decision generated by the SVC encoder. A grid image showing the MB modes overlaid on a corresponding frame is used for this comparison.

5.2 Results

5.2.1 Baseline profile

Tables 2, 3, 4, 5, and 6 summarize the results (time saving, ΔPSNR, and Δbitrate) of applying the proposal to

Table 9 RD performance and time savings of the approach for GOP = 8 and different resolutions

RD performance and time savings of the H.264/AVC-to-SVC transcoder								
GOP = 8 (main profile)								
Sequence	QCIF (15 Hz)				CIF (30 Hz)			
	ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)		ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)	
			Full seq.	Partial			Full seq.	Partial
Hall	0.668	−0.07	71.66	87.93	0.443	−1.09	71.20	88.23
City	0.063	1.81	71.12	87.99	0.018	0.31	71.29	88.31
Foreman	0.040	3.39	72.65	88.41	−0.171	3.21	71.47	88.42
Soccer	−0.027	5.52	69.99	86.09	0.105	3.46	70.57	87.24
Harbor	0.361	−0.46	70.30	86.57	0.244	−0.73	69.98	86.48
Mobile	0.022	0.68	70.39	86.31	0.212	−0.23	71.63	87.28
Average	0.188	1.81	71.02	87.22	0.142	0.82	71.02	87.66

Table 10 RD performance and time savings of the approach for GOP = 16 and different resolutions

RD performance and time savings of the H.264/AVC-to-SVC transcoder								
GOP = 16 (main profile)								
Sequence	QCIF (15 Hz)				CIF (30 Hz)			
	ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)		ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)	
			Full seq.	Partial			Full seq.	Partial
Hall	0.632	−0.06	68.34	86.72	0.337	−0.89	66.81	87.92
City	−0.009	1.92	68.91	87.18	−0.008	0.34	68.75	88.37
Foreman	0.096	2.72	67.62	86.39	−0.102	2.99	68.93	88.46
Soccer	−0.029	5.06	67.15	85.06	0.117	3.66	68.13	87.36
Harbor	0.172	−0.42	66.72	85.27	0.168	0.54	67.80	87.09
Mobile	0.033	0.53	67.29	85.28	0.185	0.10	68.39	87.22
Average	0.149	1.63	67.67	85.98	0.116	1.12	68.14	87.74

the different sequences in the baseline profile with different GOP sizes (2, 4, 8, 16, and 32) and resolutions using QP factors between 28 and 40 according to [36]. As can be seen in these tables, the algorithm presents negligible loss of video quality on average with only a slight increment in bitrate. This negligible drop in rate distortion performance is sufficiently compensated for by the reduction in computational complexity (around 84%).

Figures 11 and 12 show some resulting RD curves for the SVC bitstreams with several GOP sizes. In these curves, it can be seen that the proposal presented for transcoding is able to get close to the RD optimal transcoded (re-encoded) reference without any significant loss.

Finally, Figure 13 shows the difference between the MB partitioning carried out by the reference transcoder and the proposed algorithm, with a QP value of 28 in the sequences Foreman and City. Both encoding processes were run under the same conditions.

It can be observed that the partitioning is not exactly the same, but they are very similar, and the penalty in bitrate and PSNR is minimal while maintaining the coding efficiency but significantly reducing the time needed.

5.2.2 Main profile

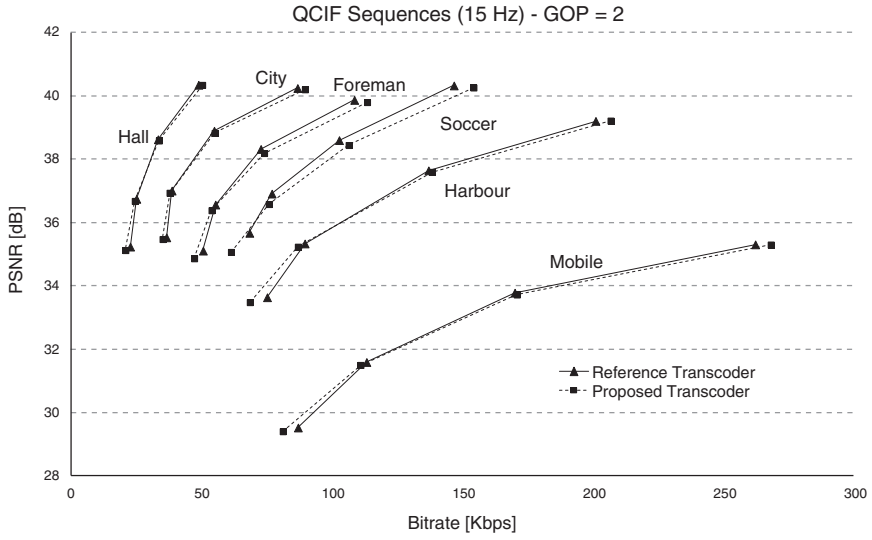
Tables 7, 8, 9, 10, and 11 summarize the results (time saving, ΔPSNR, and Δbitrate) of applying the proposal to the different sequences in the main profile with different GOP sizes (2, 4, 8, 16, and 32) and resolutions using QP factors between 28 and 40.

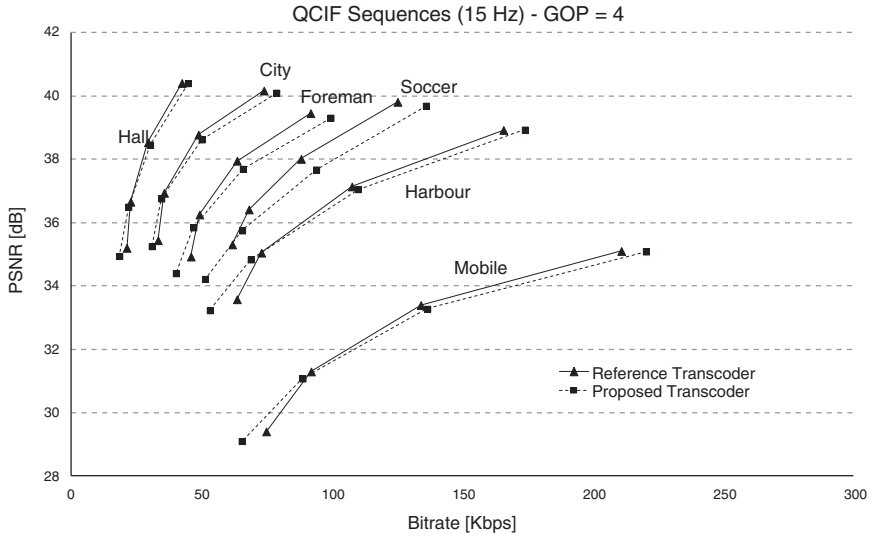
As in the previous results for baseline profile, it can be seen in these tables that the algorithm presents negligible loss of video quality on average with only a slight increment in bitrate. This negligible drop in rate distortion performance is sufficiently compensated for by the reduction in time (around 87%).

Some resulting RD curves for the SVC bitstreams with several GOP sizes are shown in Figures 14 and 15, where it can be seen that our proposal for transcoding is able to approach the performance RD optimal transcoded (re-encoded) reference without any significant loss.

Table 11 RD performance and time savings of the approach for GOP = 32 and different resolutions

RD performance and time savings of the H.264/AVC-to-SVC transcoder								
GOP = 32 (main profile)								
Sequence	QCIF (15 Hz)				CIF (30 Hz)			
	ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)		ΔPSNR (dB)	ΔBitrate (%)	Time saving (%)	
			Full seq.	Partial			Full seq.	Partial
Hall	0.626	0.29	70.31	88.83	0.166	−0.34	66.90	87.78
City	−0.020	1.68	68.97	88.61	−0.011	0.46	67.60	87.78
Foreman	−0.024	3.41	69.16	88.32	0.001	3.19	67.93	87.93
Soccer	0.018	5.79	66.12	86.12	0.089	3.62	67.17	86.92
Harbor	0.371	0.16	68.13	87.39	0.229	−0.15	66.96	86.79
Mobile	0.102	1.03	67.08	86.74	0.144	0.46	66.63	86.59
Average	0.179	2.06	68.30	87.67	0.103	1.21	67.20	87.30





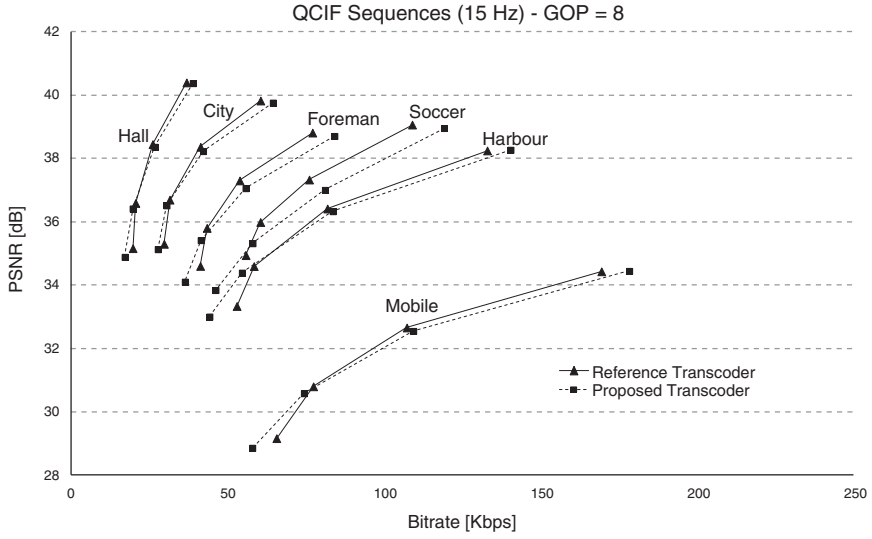
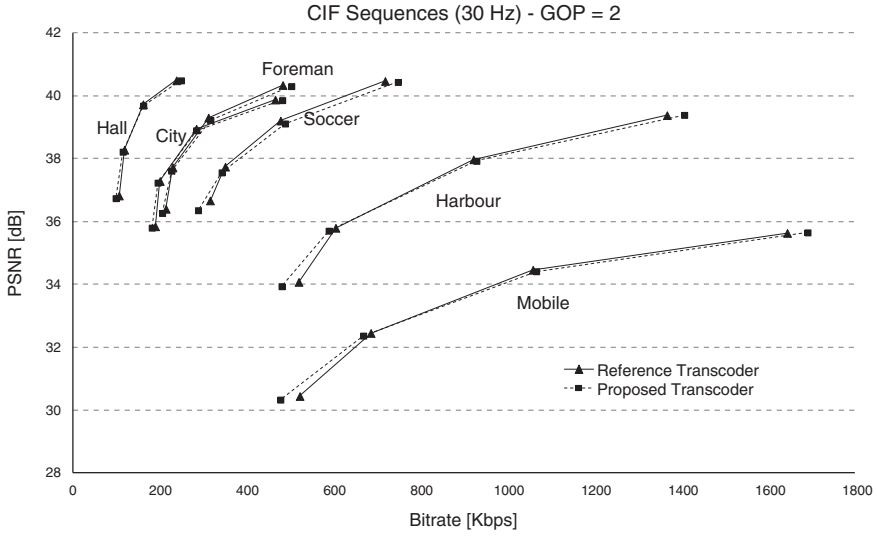
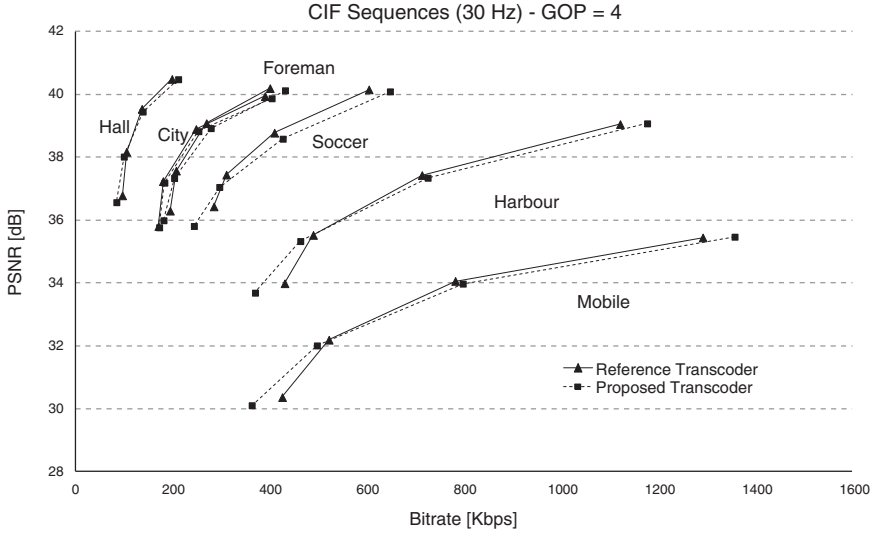


Figure 14 RD performance for QCIF sequences with different GOP sizes (main profile).





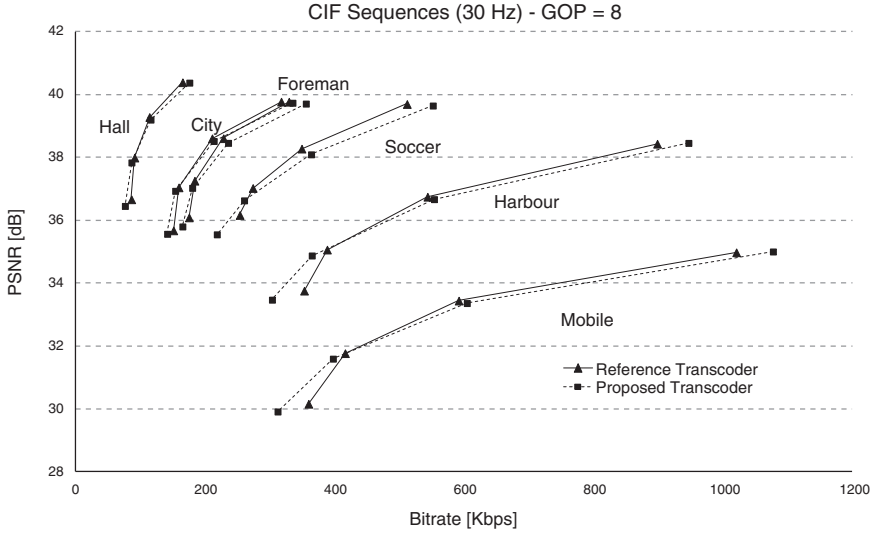


Figure 15 RD performance for CIF sequences with different GOP sizes (main profile).

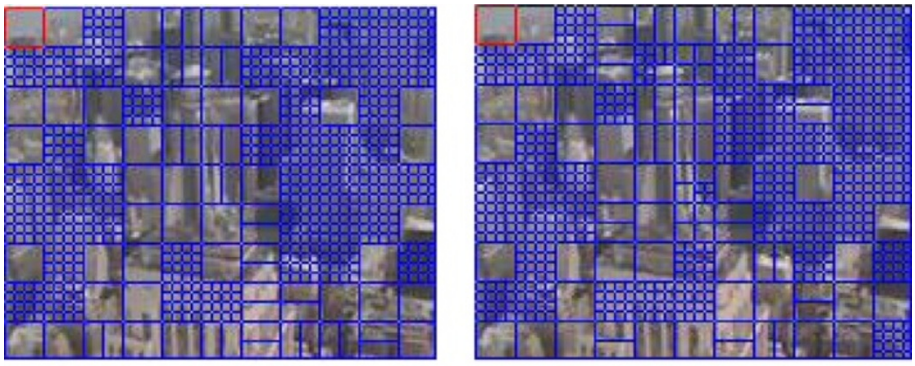
The values of PSNR and bitrate obtained with the proposed transcoder are very close to the results obtained when applying the reference transcoder (re-encoder) while a significant reduction in computational complexity is achieved (around 86% where the proposal is applied).

As in the baseline profile, the difference between the MB partitioning carried out by the reference transcoder and the proposed algorithm, with a QP value of 28 in sequences Foreman and City is shown in Figure 16. It can be observed that, in both cases, the MB partitioning is very similar.

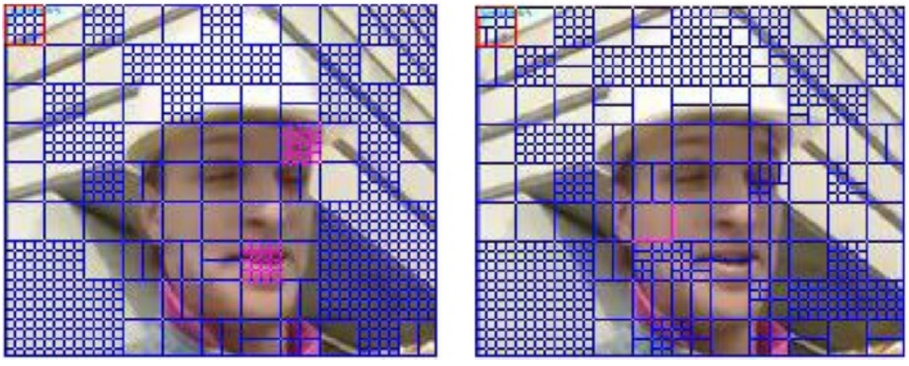
5.2.3 Analysis

Analyzing the results shown previously, some conclusions can be drawn. Both in baseline and main profiles the reduction in computational complexity is appreciable. The partial time saving (the time reduction measured only in the temporal layers where the proposal is applied) achieved is around 84% for baseline and 87% for the main profile. Regarding the total time saving (the time reduction measured over the whole sequence), a reduction of 65% for baseline and 68% for main profile is achieved. These time reductions are obtained without any significant increment in bitrate (in baseline profile between 0.28% in the best case and 3.25% in the worst one, and in main profile between 0.44% and 2% in the worst case). Regarding PSNR, the presented algorithm improves upon the PSNR obtained by the reference transcoder. This is possible because both the reference and proposed transcoder are encoded with the *rate distortion optimization* disable; so, the encoding performed by the reference is not the most optimized one.

The performance results also show that the algorithm works properly with different sequences with varying characteristics and resolutions, although there are some differences between sequences with regard to the increment in bitrate. For example, the increment in bitrate is smaller in Hall or Harbor than in Soccer. This is due to the high movement of the Soccer sequence. Since the prediction structure in H.264/AVC without scalability and SVC is different, the reference frames from the same frame number are usually different. As the information collected from the decoding stage for each frame (residual, MVs, mode decision) is used for the decision tree for deciding the MB type, if the scene has little movement, the different prediction structure has less impact than if the sequence has high movement.



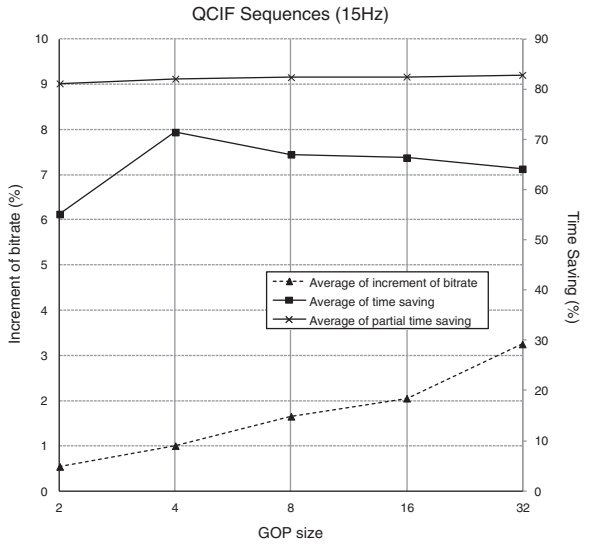
(a) 2nd frame of City sequence (1st B-frame)



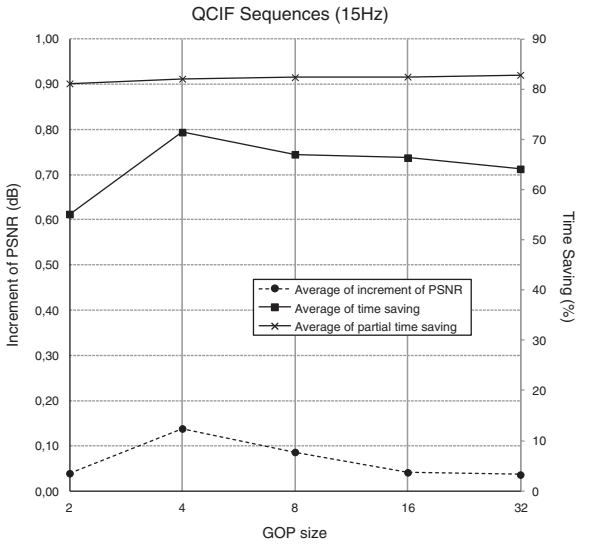
(b) 2nd frame of Foreman sequence (1st B-frame)

Figure 16 Comparison of the MB partitioning in main profile: proposed (left) and reference (right). (**a**) Second frame of the City sequence (first B-frame). (**b**) Second frame of the Foreman sequence (first B-frame).

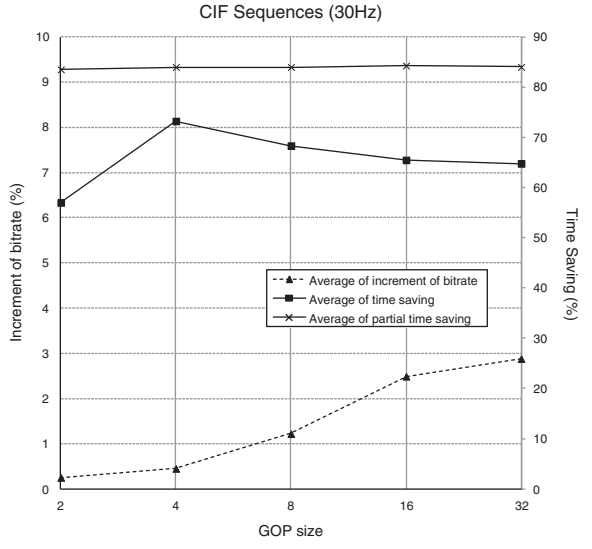
Another thing that can be observed is that the proposal can be applied to different GOP sizes, and the results are very similar in all cases. The average of the time saving, Δbitrate, and ΔPSNR results for every GOP sizes is represented in a graphical way in Figures 17 and 18. Both the tables and graphs show that, although the values of Δbitrate and ΔPSNR vary with GOP, the reduction in time achieved over the whole sequence is always greater than 55% and reaches 70% with a GOP size of 4. This variation is due to the fact that the technique presented is applied to only two enhancement temporal layers, but in the case of the GOP of length 2, there is only one enhancement temporal layer; so, the time reduction is smaller. However, when the transcoding technique is applied to sequences encoded with a GOP size of 4, the time reduction achieves its maximum value which is 70%. This is due to the fact that in this case, the technique is applied only to two out of the three temporal layers, and only the temporal base layer is encoded completely. The partial time saving is constant (around 84% for baseline profile and 87% for main profile).



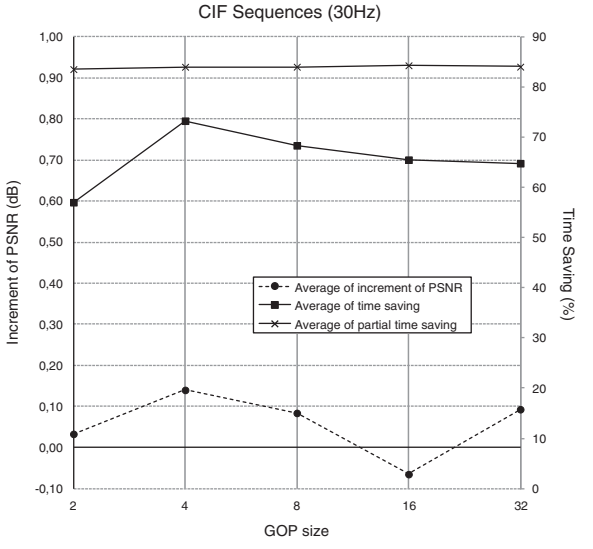
(a) ΔBitrate vs. GOP size vs. Time Saving – QCIF resolution



(b) Loss of PSNR vs. GOP size vs. Time Saving – QCIF resolution

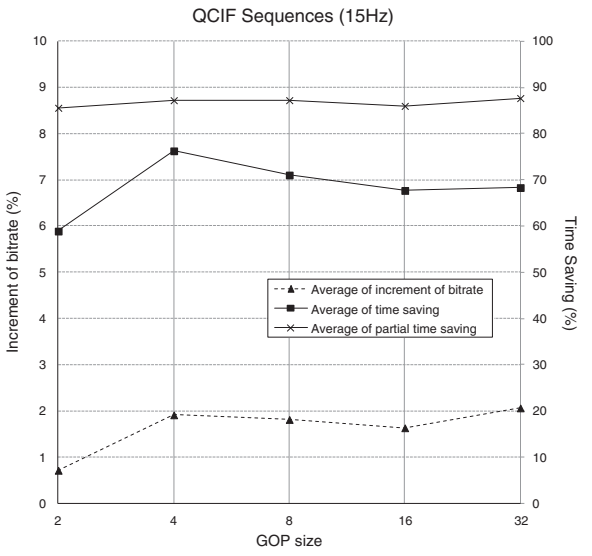


(c) ΔBitrate vs. GOP size vs. Time Saving – CIF resolution

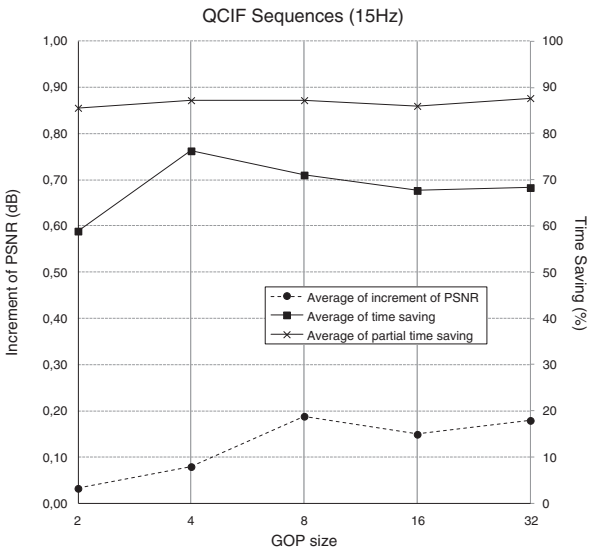


(d) Loss of PSNR vs. GOP size vs. Time Saving – CIF resolution

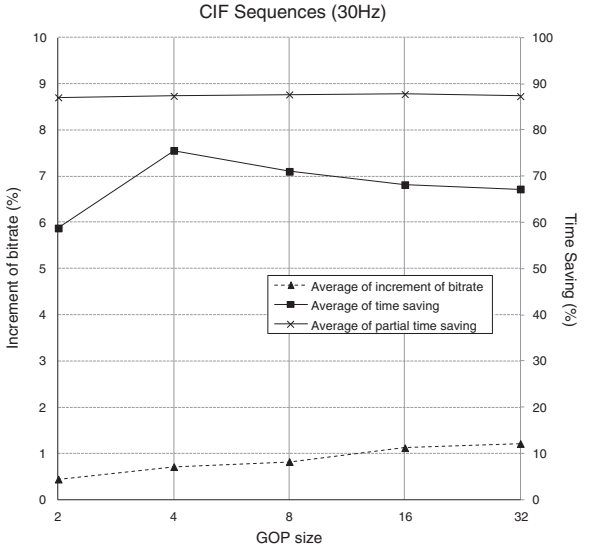
Figure 17 Overall results depending on the GOP size for QCIF and CIF resolutions (baseline profile). (**a**) ΔBitrate vs. GOP size vs. time saving (QCIF resolution). (**b**) Loss of PSNR vs.GOP size vs. time saving (QCIF resolution). (**c**) ΔBitrate vs. GOP size vs. time saving (CIF resolution). (**d**) Loss of PSNR vs. GOP size vs. time saving (CIF resolution).



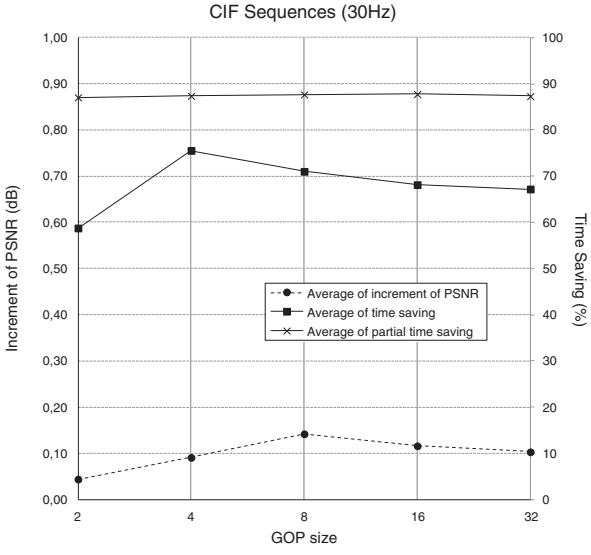
(a) ΔBitrate vs. GOP size vs. Time Saving – QCIF resolution



(b) Loss of PSNR vs. GOP size vs. Time Saving – QCIF resolution



(c) ΔBitrate vs. GOP size vs. Time Saving – CIF resolution



(d) Loss of PSNR vs. GOP size vs. Time Saving – CIF resolution

Figure 18 Overall results depending on the GOP size for QCIF and CIF resolutions (main profile). (**a**) **Δ**Bitrate vs. GOP size vs. time saving (QCIF resolution). (**b**) Loss of PSNR vs.GOP size vs. time saving (QCIF resolution). (**c**) **Δ**Bitrate vs. GOP size vs. time saving (CIF resolution). (**d**) Loss of PSNR vs. GOP size vs. time saving (CIF resolution).

The **Δ**bitrate varies between less than 0.2% for a GOP size of 2 and 3% for a GOP size of 32). Regarding *Δ*PSNR, this varies from a gain of 0.2 dB to a loss of almost 0.10 dB. In conclusion, the proposal can be applied to different GOP sizes and works properly in all of them.

6. Conclusions

As was said previously, the reference transcoder completely decodes the video received and then encodes it into SVC. The most complex part of the transcoder is the encoder stage, in which the inter-prediction process accounts for most of the resources consumed. Focusing on the inter-prediction, the other task that lends itself to being accelerated, apart from ME, is the mode decision process.

In this paper, an improved H.264/AVC-to-SVC transcoder is presented that reduces complexity by around 84% and 87% in the temporal layers to which it is applied in

baseline and main profiles, respectively. This improvement is achieved by choosing the MB types to be checked in the encoder stage depending on the information collected in the decoder stage. The specific MB types checked are selected by a decision tree built using ML tools.

The experimental results presented show that it is capable of reducing coding complexity as was mentioned above while maintaining coding efficiency. Moreover, it is valid for different profiles, GOP sizes, and resolutions.

Competing interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by the Spanish MEC and MICINN funds, under the grants TIN2009-14475-C04 and TIN2012-38341-C04-04. The first author would also like to thank the Spanish Public Employment Service for its funding support.

Author details

[1]Albacete Research Institute of Informatics, University of Castilla-La Mancha, Campus Universitario s/n, 02071 Albacete, Spain. [2]Department of Electronics and Information Systems, Multimedia Lab, Ghent University-IBBT, Gaston Crommenlaan 8, bus 201, B-9050 Ledeberg, Ghent, Belgium.

Received: 30 October 2012 Accepted: 15 March 2013
Published: 18 April 2013

References

1. Advanced Television System Committee, *ATSC-Mobile DTV Standard: A/153 ATSC Mobile Digital Television System, Document A/153 Part 1* (ATSC, Washington DC, USA, 2009), pp. 1–20
2. Digital Video Broadcasting (DVB), *Transmission System for Handheld Terminals (DVB-H), ETSI EN 302 304 V1.3.1* (DVB, Geneva, Switzerland, 2009), pp. 1–14
3. 3GPP, *TS 23.246 V9.4: Multimedia Broadcast/Multicast Service (MBMS); Architecture and functional description* (3GPP, Sophia-Antipolis, France, 2010), pp. 1–66
4. ISO/IEC, *Generic Coding of Moving Pictures and Associated Audio, SC29 WG11 13818* (ISO/IEC, New York, 1994)
5. ISO/IEC, *Coding of Audio-Visual Objects – Part 2: Visual, 14496–2* (ISO/IEC, New York, 2001)
6. Rovi Corporation: DivX, 2012. http://www.divx.com. Accessed 09 April 2013
7. XviD: XviD, , 2013. http://www.xvid.org. Accessed 09 April 2013
8. ITU-T, ISO/IEC JTC 1, *Advanced Video Coding for Generic Audiovisual Services, ITU-T Rec. H.264/AVC and ISO/IEC 14496 (including SVC extension)* (International Telecommunications Union, Geneva, Switzerland, 2009)
9. R Schäfer, T Wiegand, H Shwarz, *The emerging H.264/AVC Standard, Technical Review* (European Broadcasting Union, Grand-Saconnex, Switzerland, 2003), pp. 1–12
10. H Schwarz, D Marpe, T Wiegand, Overview of the scalable video coding extension of the H.264/AVC standard. IEEE Trans. Circuits Syst. Video Technol. **17**(9), 1103–1120 (September 2007)
11. A Vetro, C Christopoulos, H Sun, Video transcoding architectures and techniques: an overview. IEEE Signal Process. Mag. **20**(2), 18–29 (2003)
12. S Wenger, Temporal scalability using P-pictures for low-latency applications, in *IEEE Second Workshop on Multimedia Signal Processing*, ed. by PW Wong, A Alwan, A Ortega, CCJ Kuo, SLM Nikias (IEEE, New York, 1998), pp. 559–564
13. H Schwarz, D Marpe, T Wiegand, *Analysis of Hierarchical B pictures and MCTF Paper presented at the IEEE international conference on multimedia and expo* (Toronto, Canada, 2006)
14. J Reichel, H Schwarz, M Wien, *Joint Scalable Video Model (JSVM) Reference Software*. http://www.hhi.fraunhofer.de/de/kompetenzfelder/image-processing/research-groups/image-video-coding/svc-extension-of-h264avc/jsvm-reference-software.html. Accessed 09 April 2013
15. H Shen, S Xiaoyan, F Wu, H Li, S Li, *Transcoding to FGS streams from H.264/AVC hierarchical B-Pictures. Paper presented at proceedings of 2006 IEEE international conference on image processing* (, Atlanta, GA, USA, 2006)
16. J De Cock, S Notebaert, P Lambert, R Van de Walle, Bridging the gap: transcoding from single-layer H.264/AVC to scalable SVC video streams, in *Proceedings of the 9th International Conference on Advanced Concepts for Intelligent Vision Systems*, ed. by W Philips, D Popescu, P Scheunders (Springer, New York, 2007), pp. 652–662
17. J De Cock, S Notebaert, P Lambert, R Van de Walle, *Advanced bitstream rewriting from H.264/AVC to SVC. Paper presented at the Proceedings of 15th IEEE international conference on image processing* (San Diego, CA, USA, 2008)
18. J De Cock, S Notebaert, P Lambert, R Van de Walle, Architectures of fast transcoding of H.264/AVC to quality-scalable SVC streams. IEEE Transactions on Multimedia **11**(7), 1209–1224 (2009)
19. G Van Wallendael, S Van Leuven, R Garrido-Cantos, J De Cock, JL Martinez, P Lambert, P Cuenca, R Van de Walle, *Fast H.264/AVC-to-SVC transcoding in a mobile television environment. Paper presented at the proceedings of 6th mobile multimedia communications conference* (Lisbon, Portugal, 2010)
20. S Van Leuven, J De Cock, G Van Wallendael, R Van de Walle, R Garrido-Cantos, JL Martinez, P Cuenca, *A low-complexity closed-loop H.264/AVC to quality-scalable SVC transcoder. Paper presented at the proceedings of 17th international conference on digital signal processing* (Corfu, Greece, 2011)
21. S Van Leuven, J De Cock, G Van Wallendael, R Van de Walle, R Garrido-Cantos, JL Martinez, P Cuenca, *Combining open- and closed-loop architectures for H.264/AVC-to-SVC transcoding. Paper presented at the proceedings of 18th IEEE international conference on image processing* (Brussels, Belgium, 2011)
22. R Sachdeva, S Johar, E Piccinelli, *Adding SVC spatial scalability to existing H.264/AVC video. Papere presented at the proceedings of 8th IEEE/ACIS international conference on computer and information science* (Shanghai, China, 2009)
23. A Dziri, A Diallo, M Kieffer, P Duhamel, *P-picture based H.264 AVC to H.264 SVC temporal transcoding., Paper presented at the proceedings of international wireless communications and mobile computing conference* (Crete Island, Greece, 2008)
24. H Al-Muscati, F Labeau, *Temporal transcoding of H.264/AVC video to the scalable format. Paper presented at the proceedings of 2nd international conference on image processing theory tools and applications* (France Paris, 2010)
25. R Garrido-Cantos, J De Cock, JL Martínez, S Van Leuven, P Cuenca, A Garrido, R Van de Walle, Video transcoding for mobile digital television. Telecommunication Syst. (2011). 10.1007/s11235-011-9594-1
26. R Garrido-Cantos, J De Cock, JL Martínez, S Van Leuven, P Cuenca, Motion-based temporal transcoding from H.264/AVC-to-SVC in baseline profile. IEEE Trans. Consum. Electron. **57**, 1 (2011)
27. C-H Yeh, W-Y Tseng, S-T Wu, Mode decision acceleration for H.264/AVC to SVC temporal video transcoding. EURASIP J. Adv. Signal Process. (2012). doi:10.1186/1687-6180-2012-204
28. R Garrido-Cantos, J De Cock, JL Martínez, S Van Leuven, P Cuenca, A Garrido, *H.264/AVC-to-SVC temporal transcoding using machine learning. Paper presented at the proceedings of 16th international conference on knowledge-based and intelligent information & engineering systems* (San Sebastian, Spain, 2012)
29. R Garrido-Cantos, J De Cock, S Van Leuven, P Cuenca, A Garrido, R Van de Walle, Fast mode decision algorithm for H.264/AVC-to-SVC transcoding with temporal scalability, in *Proceedings of 18th International Conference on Multimedia Modeling*, ed. by K Schoeffmann, B Mérialdo, AG Hauptmann, CW Ngo, Y Andreopoulos, C Breiteneder (Springer, New York, 2012), pp. 585–596
30. WW Cohen, *Fast effective rule induction. Paper presented at the 20th international conference on machine learning* (Tahoe City, CA, USA, 1995)
31. G Fernández-Escribano, J Bialkowski, JA Gámez, H Kalva, P Cuenca, L Orozco-Barbosa, A Kaup, Low-complexity heterogeneous video transcoding using data mining. IEEE Trans. Multimedia **10**(2), 286–299 (2008)
32. JL Martinez, G Fernandez-Escribano, H Kalva, WAC Fernando, P Cuenca, Wyner-Ziv to H.264 video transcoder for low cost video communications. IEEE Trans. Consum. Electron. **55**(3), 1453–1461 (2009)
33. M Hall, E Frank, G Holmes, B Pfahringer, P Reutemann, IH Witten, The WEKA data mining software: an update. SIGKDD Explorations **11**(1), 21–35 (2009)
34. G Fernandez-Escribano, *Low complexity MPEG-2 to H.264 transcoding, PhD thesis* (University of Castilla-La Mancha, Albacete, Spain, 2007)
35. Joint Model (JM) reference software, H.264/AVC reference software. http://iphome.hhi.de/suehring/tml/download/. Accessed 09 April 2013
36. G Sullivan, G Bjøntegaard, *Recommended Simulation Common Conditions for H.26L Coding Efficiency Experiments on Low-Resolution Progressive-Scan Source Material, ITU-T VCEG, Doc. VCEG-N81* (International Telecommunications Union, Geneva, Switzerland, 2001)

doi:10.1186/1687-6180-2013-82
Cite this article as: Garrido-Cantos *et al.*: **Low-complexity transcoding algorithm from H.264/AVC to SVC using data mining.** *EURASIP Journal on Advances in Signal Processing* 2013 **2013**:82.