2

Mobile Video Communications Based on Fast DVC to H.264 Transcoding

Alberto Corrales Garcia[1], Gerardo Fernandez Escribano[1], Jose Luis Martinez[2] and Francisco Jose Quiles[1]
[1]*Instituto de Investigación en Informática de Albacete, University of Castilla-La Mancha Albacete,*
[2]*Architecture and Technology of Computing Systems Group, Complutense University, Madrid, Spain*

1. Introduction

Nowadays, mobile devices demand multimedia services such as video communications due to the advances in mobile communications systems (such us 4G) and the integration of video cameras into mobile devices. However, these devices have some limitations of computing power, resources and complexity constraints for performing complex algorithms. For this reason, in order to establish a video communications between mobile devices, it is necessary to use low complex encoding techniques. In traditional video codecs (such as H.264/AVC (ISO/IEC, 2003)) these low complexity requirements have not been met because H.264/AVC is more complex at the encoder side. Then, mobile video communications based on H.264/AVC low complexity imply a penalty in terms of Rate – Distortion (RD). However, Distributed Video Coding (DVC) (Girod et al., 2005), and particularly Wyner-Ziv (WZ) video coding (Aaron et al., 2002), provides a novel video paradigm where the complexity of the encoder is reduced by shifting the complexity of the encoder to the decoder (Brites et al., 2008). Taking into account the benefits of both paradigms, recently WZ to H.26X transcoders have been proposed in the multimedia community to support mobile-to-mobile video communications. The transcoding framework provides a scheme where transmitter and receiver execute lower complexity algorithms and the majority of the computation is moved to the network where the transcoder is allocated. This complexity is thus assumed by a transcoder, which has more resources and no battery limitations. Nevertheless, for real time communications it is necessary to perform this conversion from WZ to H.264/AVC with a short delay, and then the transcoding process must be executed as efficiently as possible.

At this point, this work presents a WZ to H.264/AVC transcoding framework to support mobile-to-mobile video communications. In order to provide a faster transcoding process both paradigms involve in the transcoder (WZ decoding and H.264/AVC encoding) are accelerated. On the one hand, nowadays parallel programming is becoming more important to solve high complexity computation tasks and, as a consequence, the computing market is

full of multicore systems, an approach is proposed to execute WZ decoding in a parallel way. On the other hand, at the same time WZ is decoding, some information could be gathered are sent to the H.264/AVC encoder in order to reduce the encoding algorithm complexity. In this work, the search area of the Motion Estimation (ME) process is reduced by means of Motion Vectors (MVs) calculated in the WZ decoding algorithm. In this way, the complexity of the two most complex tasks of this framework (WZ decoding and H.264/AVC encoding) are largely reduced making the transcoding process more efficient.

2. Background

2.1 Wyner-Ziv video coding

The first practical Wyner-Ziv framework was proposed by Stanford in (Aaron et al., 2002), and this work was widely referenced and improved in later proposals. As a result, in (Artigas et al., 2007) an architecture called DISCOVER was proposed which outperforms the previous Stanford one. This architecture provided a reference for the research community and finally it was later improved upon with the VISNET-II architecture (Ascenso et al., 2010), which is depicted in Figure 1. In this architecture, the encoder splits the sequence into two kinds of frames: Key Frames (K) and Wyner-Ziv Frames (WZ) in module (1). K frames are encoded by an H.264/AVC encoder in (2). On the other hand, WZ frames are sent to the WZ encoder, where the information is firstly quantized (3a), and BitPlanes (BPs) are extracted in (3b); in (3c) each BP is independently channel encoded and several parity bits, which are stored in a buffer (3d), are calculated. On the decoder side, initially K frames are decoded by an H.264/AVC decoder (4). From these frames, Side Information (SI) is calculated in (5), which represents an estimation for each non-present original WZ frame. For this estimation, the Correlation Noise Model (CNM) module (6) generates a Laplacian distribution, which models the residual between SI and the original frame. Afterwards, SI and CNM are sent to the turbo decoder, which corrects differences of SI and the original frame by means of iterative decoding (requesting several parity bits from the encoder through the feedback channel). Finally, decoding bitplanes are reconstructed in module (7c).

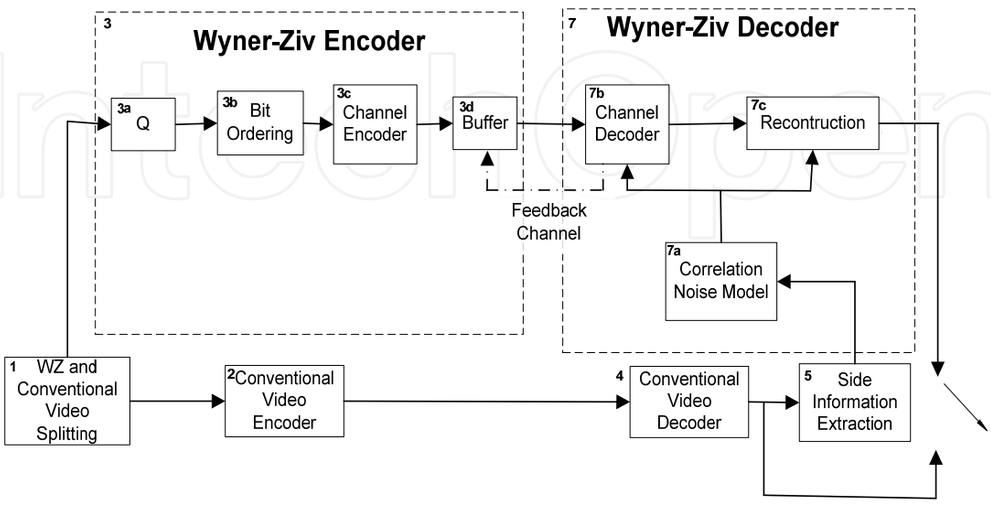


Fig. 1. Block diagram of the reference WZ architecture [Ascenso et al. 2010].

2.2 H.264/AVC

H.264/AVC or MPEG-4 part 10 Advanced Video Coding (AVC) is a compression video standard developed by the ITU-T Video Coding Experts Group (ITU-T VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG). In fact, both standards are technically identical (ISO/IEC, 2003).

The main purpose of H.264/AVC is to offer a good quality standard able to considerably reduce the output bit rate of the encoded sequences, compared with previous standards, while exhibiting a substantially increasing definition of quality and image. H.264/AVC promises a significant advance compared with the commercial standards currently most in use (MPEG-2 and MPEG-4). For this reason H.264/AVC contains a large amount of compression techniques and innovations compared to previous standards; it allows more compressed video sequences to be obtained and provides greater flexibility for implementing the encoder. Figure 2 shows the block diagram of the H.264/AVC encoder.

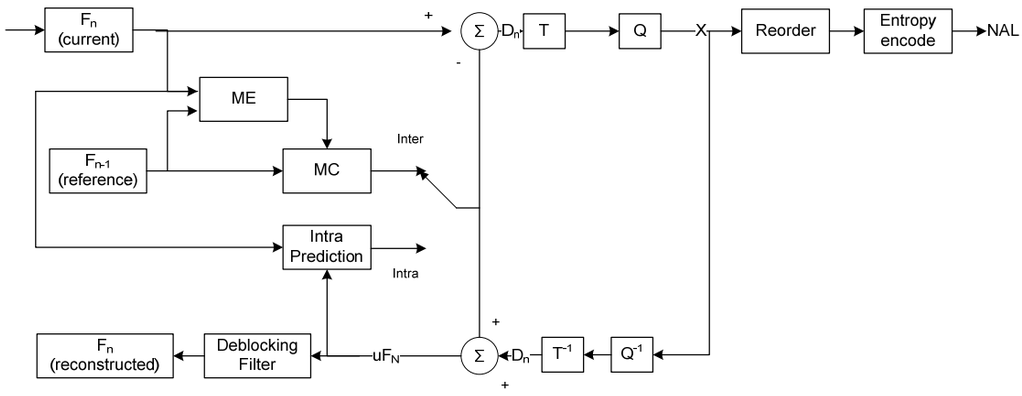


Fig. 2. H.264/AVC encoder diagram

The ME is the most time-consuming task in the H.264/AVC encoder. It is a process which removes the temporal redundancy between images, comparing the current one with previous or later images in terms of time (reference images), looking for a pattern that indicates how the movement is produced inside the sequence.

To improve the encoding efficiency, H.264/AVC allows the use of partitions resulting from dividing the MB in different ways. Greater flexibility for the ME and Motion Compensated (MC) processes and greater motion vector precision give greater reliability to the H.264/AVC encoding process. The ME process is thus carried out many times per each partition and sub-partition. This feature is known as variable block size for the ME.

3. Related work

3.1 Parallel Wyner-Ziv

The DVC framework is based on displacing the complexity from encoders to decoders. However, reducing the complexity of decoders as much as possible is desirable. In traditional feedback-based WZ architectures (Aaron et al., 2002), the rate control is performed at the decoder and is controlled by means of the feedback channel; this is the

main reason for the decoder complexity, as once a parity chunk arrives at the decoder, the turbo decoding algorithm (one of the most computationally-demanding tasks (Brites et al., 2008) is called. Taking this fact into account, there are several approaches which try to reduce the complexity of the decoder, which usually induces a rate distortion penalty. However, due to technological advances, new parallel hardware is beginning to be introduced into practical video coding solutions. These new features of computers offer a new challenge to the research community with regards to integrating their algorithms into a parallel framework; this opens a new door in multimedia research. It is true that, with regards to traditional standards, several approaches have been proposed since multicores appeared on the market, but this chapter focuses on parallel computing applied to the WZ framework.

Having said this, in 2010 several different parallel solutions for WZ were proposed. In particular, in (Oh et al., 2010) Oh et al. proposed a WZ parallel execution carried out by Graphic Processing Units (GPUs). In this proposal, the authors focus on designing a parallel distribution for a Slepian-Wolf decoder based on rate Adaptative Low Density Check Code (LDPC) with Accumulator (LDPCA). LDPC codes are composed of many bit-nodes which do not have many dependencies between each node, so they propose a parallel execution in three kernels (steps): i) kernels for check node calculations, ii) kernels for bit node calculations, and iii) kernels for termination condition calculations. In a GPU they achieve a decoding 4~5 times faster for QCIF and 15~20 for CIF. On the other hand, in (Momcilovic et al., 2010) Momcilovic et al. proposed a WZ LDPC parallel decoding based on multicore processors. In this work, the authors parallelize several LDPC approaches. On a Quad-Core machine, they achieve a speedup of about 3.5. Both approaches propose low-level parallelism for a particular LDPC/LDPCA implementation.

This chapter presents a WZ to H.264/AVC transcoder which includes a higher-level parallel WZ video decoding algorithm implemented on a multicore system. The reference WZ decoding algorithm is adapted to a multicore architecture, which divides each frame into several slices and distributes the work among available cores. In addition, the proposed algorithm is scalable because it does not depend on the hardware architecture, the number of cores or even on the implementation of the internal Wyner-Ziv decoder. Therefore, the time reduction can be increased simply by increasing the number of cores, as technology advances. Furthermore, the proposed method can also be applied to WZ architectures with or without a feedback channel (Sheng et al., 2010).

3.2 WZ to H.26x transcoding

Nowadays, mobile-to-mobile video communications are getting more and more common. Transcoding from a low cost encoder format to a low cost decoder provides a practical solution for these types of communications. Although H.264/AVC has been included in multiple transcoding architectures from other coding formats (such as MPEG-2 to H.264/AVC (Fernandez-Escribano et al., 2007, 2008) or even homogeneous H.264/AVC (De Cock et al., 2010), proposals in WZ to H.26x to support mobile communications are rather recent and there are only a few approaches so far.

In 2008, the first WZ transcoder architecture was introduced by Peixoto et al. in (Peixoto et al., 2010). In this work, they presented a WZ to H.263 transcoder for mobile video

communications. However, H.263 offers lower performance than other codecs based on H.264/AVC and they did not exploit the correlation between the WZ MVs and the traditional ME successfully and only used them to determine the starting centre of the ME process.

In our previous work, we proposed the first transcoding architecture from WZ to H.264/AVC (Martínez et al., 2009). This work introduced an improvement to accelerate the H.264/AVC ME stage using the Motion Vectors (MV) gathered in the WZ decoding stage. Nevertheless, this transcoder is not flexible since it only applies the ME improvement for transcoding from WZ frames to P frames. In addition, it only allows transcoding from WZ GOPs of length 2 to IPIP H.264/AVC GOP patterns, so it does not use practical patterns due to the high bit rate generated neither flexible. Furthermore, this work used a less realistic WZ implementation. For this reason, the approach presented in this chapter improves this part by introducing a better and more realistic WZ implementation based on the VISNET-II codec (Ascenso et al., 2010), which implements lossy key frame coding, on-line correlation noisy modeling and uses a more realistic procedure call at the decoder for the stopping criterion.

4. Transcoding for mobile to mobile communications

4.1 Introduction

The main task of a transcoder is to convert a source coding format into another one. In the case of mobile video communications, the transcoding process should be done as fast as possible. In addition, a flexible transcoder should take into account the conversion between the input and the output patterns. In order to provide a flexible and fast transcoding architecture, it is proposed the architecture displayed in Figure 3.

This architecture is composed of a Wyner-Ziv decoder and a H.264/AVC encoder with several modifications or extra modules. In particular, the WZ decoder is redesigned to parallelize the decoding process and the black modules in Figure 3 have been included or modified to obtain a faster H.264/AVC encoding. Details will be given in the following subsections.

4.2 Parallelization of WZ decoding

WZ video coding accumulates the majority of the complexity on the decoder side. If you study each module inside the decoder scheme (Figure 1), you discover that most of this complexity is concentrated in the Channel Decoder module (Brites et al., 2008). This module receives successive chunks of parity bits. Then, the quantized symbol stream associated to each bitplane is obtained in an iterative process, which is based on the residual statistics calculated by the CNM. This procedure stops when a condition based on probabilities is satisfied. Obviously, the complexity of the decoder increases when more bitplanes (in the pixel domain) or coefficient bands (for the transform domain) are decoded. At this point, as a first stage on the transcoding process, it is proposed a WZ decoding architecture which distributes decoding complexity across several processing units. The proposed architecture is shown in Figure 3. The approach is a flexible and scalable architecture which distributes the parallel decoding between two parallelism levels: GOPs and frames. First, the input bitstream composed of K frames is stored in a K-frame buffer. Then, at the first parallelism level, the WZ frames inside two K frames delimit a GOP structure, and therefore each GOP

decoding procedure is carried out independently by a different core. Additionally, for each WZ frame inside a GOP, an SI is calculated and then split into several parts. Then each portion of the frame is assigned to any core which executes the iterative turbo decoding procedure in order to decode the corresponding part of the WZ reconstructed frame. Therefore, each spatial division of the frame is decoded in an independent way by using the feedback channel to request parity bits from the encoder. When each part of a given frame is decoded, these parts are joined in spatial order and the frame is reconstructed. Finally, a sequence joiner receives each decoded frame and key frames in order to reorganize the sequence in its temporal order.

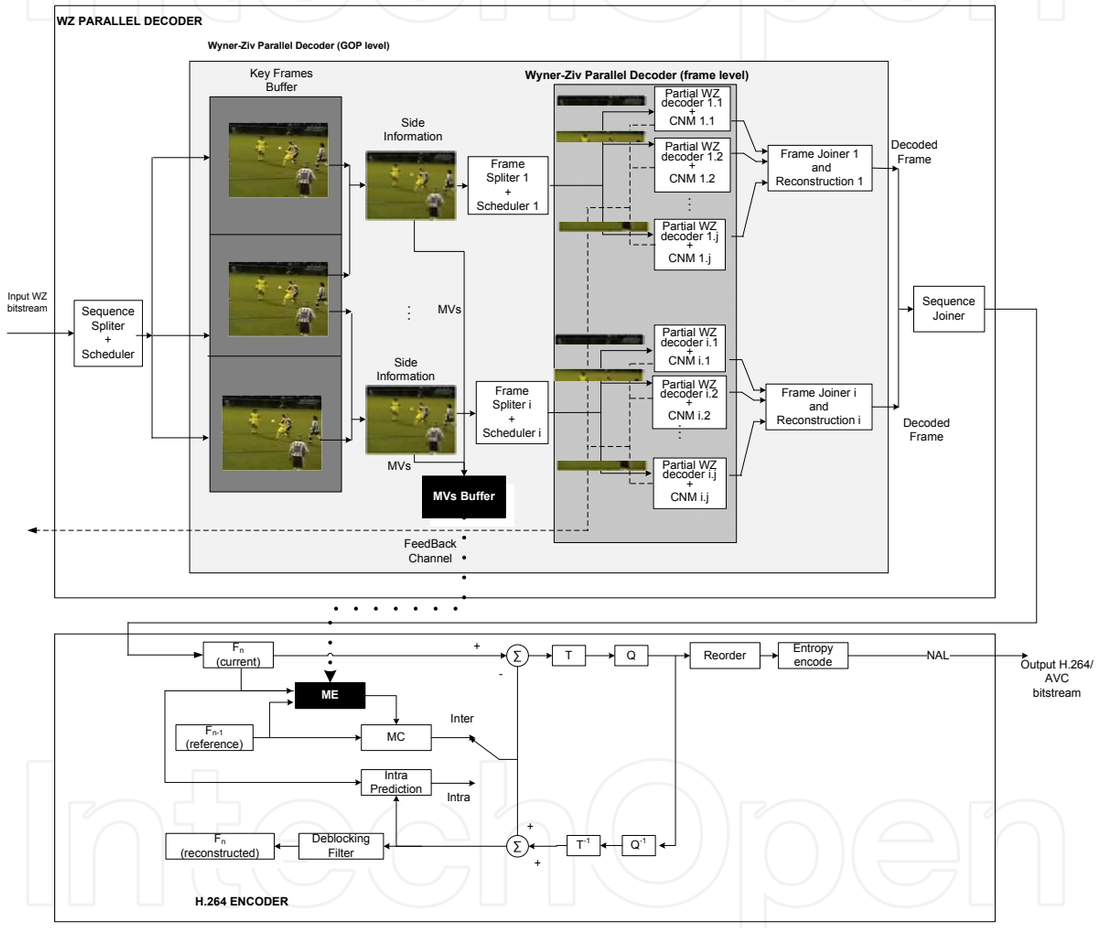


Fig. 3. Proposed WZ-to-H.264/AVC transcoding achitecture

Concerning the scheduler, a dynamic scheduler is implemented. That means that whenever a core is free and there is no pending task, it is assigned to the idle core. The number of tasks is always equal to, or bigger than, the number of cores. So that means there are always tasks in the scheduler queue until the end of the decoding stage is reached. However, partial decoding for each frame requires a synchronization barrier. To illustrate this, Figure 4 shows the decoding time line for a sequence composed of 5 GOPs (with length = 2) on a multicore with four cores. As can be seen, decoder initialization takes some time at the beginning of the decoder process. After that, each core receives a task (defined by a thread) from the

scheduler. When a thread finishes the decoding of a part of a frame, it can continue decoding other parts of the same frame. In the case of there being no more parts of this frame for decoding, this core has to wait until the rest of parts of the same frame are decoded. This is a consequence of the synchronization barrier implicit for each frame to be reconstructed. In Figure 4, when a thread is waiting it is labeled as being in an idle state. In addition, while the sequence decoding process is finishing, there are not enough tasks for available cores, so several cores change their status to idle until the decoding process finishes. Nevertheless, real sequences are composed of many GOPs and decoder initialization and ending times are quite shorter than the whole decoding time.

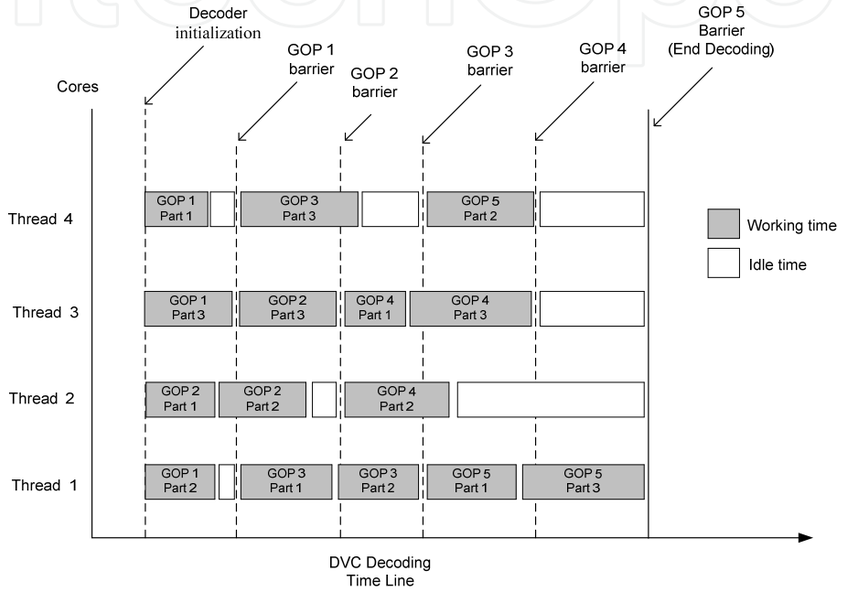


Fig. 4. Timeline for the proposed parallel WZ decoding with a sequence with 5 GOPs (GOP length = 2) and 4 cores.

The size of the K-frame buffer S is defined by Equation 1, where i is the number of GOPs which can be executed in parallel. For example, in the execution in Figure 4, a 4-core processor can execute two GOPs at the same time, so three stored K frames are providing enough tasks for four cores. In addition, it is not necessary to fill the buffer fully and it could be filled progressively during the decoding process. For different GOP lengths, the buffer size would be the same, since every WZ GOP length only needs two K frames to start the first WZ decoding frame.

$$S = i + 1 \tag{1}$$

Finally, considering that the parity bits could be requested to the encoder without following a sequential order, it calculates the Parity Position (PP) which determinates the parity bit position to start to send. PP is calculated by Equation 2, where I is the Intra period, P is the position of the current GOP, Q is the quantification parameter, and W is the width of the image and H the height.

$$PP = (I - 1) * P * Q * \left(\left(\frac{W*H*2}{8}\right) + 1\right) \tag{2}$$

4.3 H.264/AVC transcoding approach

In order to provide fast and flexible transcoding at the H.264/AVC encoder side, we have to study two issues: firstly, how MVs generated during the SI process could help to reduce the time used in ME; secondly, taking into account that DVC and H.264/AVC can build different GOPs, how to map MVs between different GOP combinations in order to provide flexibility.

4.3.1 Reducing motion estimation complexity

Within the WZ decoding process, an important task is the SI generation stage, which is the first step in the process for generating the WZ frames from K frames. VISNET-II performs Motion Compensated Temporal Interpolation (MCTI) to estimate the SI. The first step of this method is shown in Figure 5, which consists in matching each forward frame MB with a backward frame MB inside the search area. The process checks all the possibilities inside the search area and chooses the MV that generates the lowest residual. The middle of this MV represents the displacement for the MB interpolated (more details about the SI generation process in (Ascenso et al., 2005)).

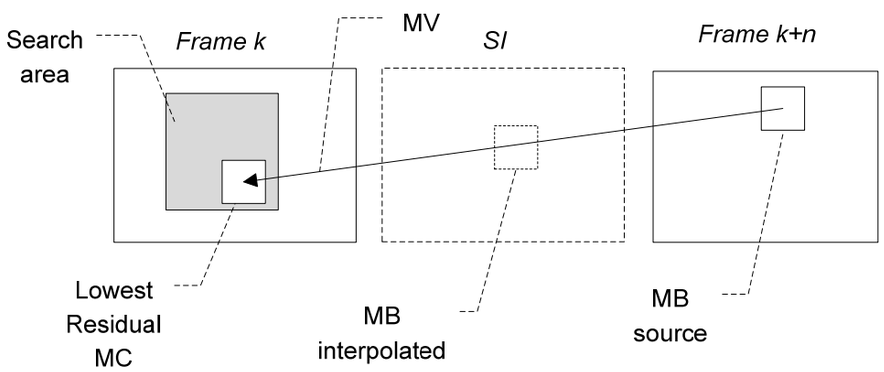


Fig. 5. First step of SI generation process.

Obviously, MVs generated in the WZ decoding stage contain approximated information about the quantity of movement of the frame. Following this idea, the present approach proposes to reuse the MVs to accelerate the H.264/AVC encoding stage by reducing the search area of the ME stage. Moreover, the present reduction is adjusted for every input DVC GOP to every H.264/AVC GOP in an efficient and dynamic way. As is shown in Figure 6, the search area for each MB is defined by a circumference with a radius dependent on the incoming SI MV (*Rmv*). This search area can oscillate between a minimum (defined by *Rmin*) and a maximum (limited by the H.264 search area). In particular, the length will vary depending on the type of frame and the length of the reference frame, as will be explained in section 4.2.2. Furthermore, a minimum area is considered since MVs are calculated from 16x16 MBs in the SI process, and H.264/AVC can even work with smaller partitions than 16x16. Besides, SI is an approximation of the frame, so some changes could occur when the fame is completely reconstructed. For these reasons, this minimum was set at 4 pixels.

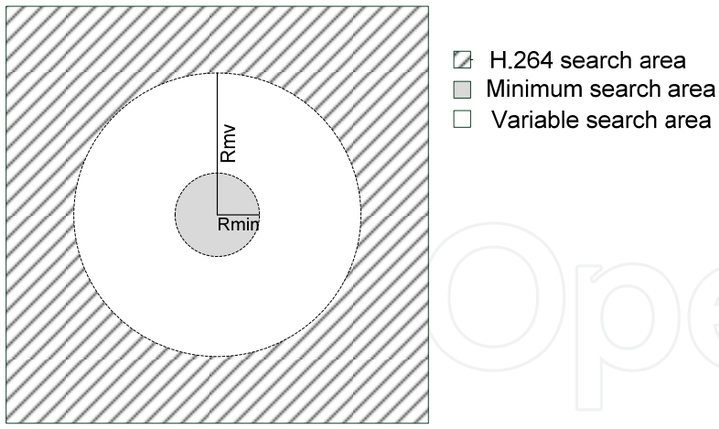


Fig. 6. Search area reduction for H.264 encoding stage.

4.3.2 Mapping GOPs from DVC to H.264

One desired feature of every transcoder is flexibility. To achieve it, an important process is to perform a with care known as GOP mapping. On the second part of the transcoder, it is proposed a DVC to H.264/AVC conversion which allows every mapping combination by performing this task using techniques to improve the time spending by the transcoding process. To extract MVs, first the distance used to calculate the SI is considered. For example, Figure 7 shows the transcoding process for a DVC GOP of length 4 to a H.264/AVC pattern IPPP (baseline profile). In step 1, DVC starts to decode the frame labeled as WZ_2 and the MVs generated in its SI generation are discarded because they are not closely correlated with the proper movement (low accuracy). When the WZ_2 frame is reconstructed (through the entire WZ decoding algorithm, WZ'_2) in step 2, the WZ decoding algorithm starts to decode frames WZ_1 and WZ_3 by using the reconstructed frame WZ'_2. At this point, the MVs $V_{0\text{-}2}$ and $V_{2\text{-}4}$ generated in this second iteration of the DVC decoding algorithm are stored. These MVs will be used to reduce the H.264/AVC ME process. Notice that in the case of higher GOP sizes the procedure is the same. In other words, MVs are stored and reused when the distance between SI and the two reference frames is 1. Finally, $V_{0\text{-}2}$ and $V_{2\text{-}4}$ are divided into two halves because P frames have the reference frame with distance one and MVs were calculated for a distance of two during the SI process.

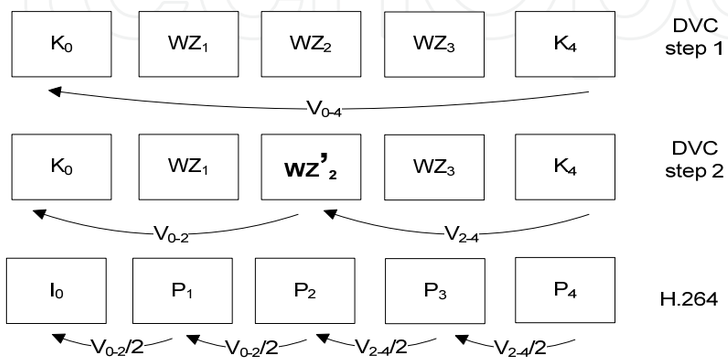


Fig. 7. Mapping from DVC GOP of length 4 to H.264 GOP IPPP.

For more complex patterns, which include mixed P and B frames (main profile), this method can be extended in a similar way with some changes. Figure 8 shows the transcoding from a DVC GOP of length 4 to a H.264 pattern IBBP. MVs are also stored by always following the same procedure. However, in this case the way to apply them in H.264/AVC changes.

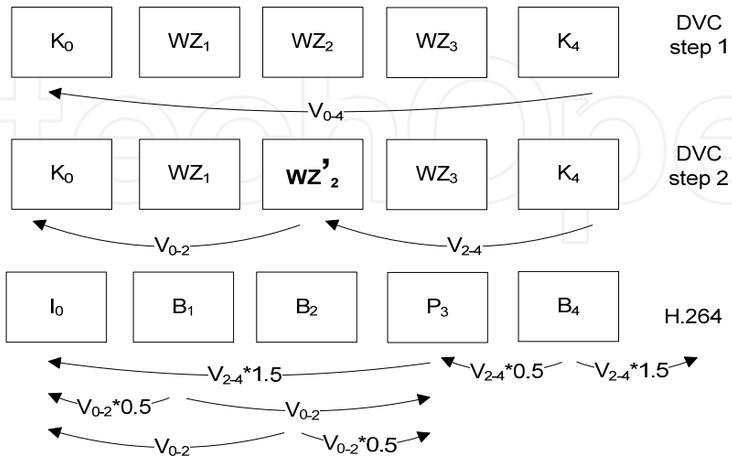


Fig. 8. Mapping from DVC GOP of length 4 to H.264 GOP IBBP.

For P frames, MVs are multiplied by a factor of 1.5 because MVs were calculated for a distance of 2 and P frames have their references with a distance of 3. For B frames, it depends on the position that they are allocated and it changes for backward and forward searches.

As can be observed, this procedure can be applied to both K and WZ frames. Therefore, following this method the proposed transcoder can be used for transcoding from every DVC GOP to every H.264/AVC GOP.

5. Experimental results

The proposed transcoder has been evaluated by using four representative QCIF sequences with different motion levels were considered. These sequences were coded at 15 fps and 30 fps using 150 frames and 300 respectively. In the DVC to H.264/AVC transcoder applied, the DVC stage was generated by the VISNET II codec using PD with BP = 3 as quantification in a trade-off between RD performance and complexity constraints but with whatever BP could be used. In addition, sequences were encoded in DVC with GOPs of length 2, 4 and 8 to evaluate different patterns. The parallel decoder was implemented by using an Intel C++ compiler (version 11.1) which combines a high-performance compiler as well as Intel Performance Libraries to provide support for creating multi-threaded applications. In addition, it provides support for OpenMP 3.0 (OpenMP, 2011). In order to test the performance of parallel decoding, it was executed over an Intel i7-940 multicore processor (Intel, 2011), although the proposal is not dependent on particular hardware. For the experiments, the parallel decoding was split into 9 parts where each core has thus a ninth part of the frame. This value is a good selection for QCIF frames (176x144), 16x16 macroblocks (this is the size of the block in the SI generation and thus a QCIF frame has 99 16x16 blocks) and 4 processors (4 cores, 8 simultaneous processes with hyper-threading).

During the decoding process, the MVs generated by the SI generation stage were sent to the H.264/AVC encoder; hence it does not involve any increase in complexity. In the second stage, the transcoder performs a mapping from every DVC GOP to every H.264/AVC GOP using QP = 28, 32, 36 and 40. In our experiments we have chosen different H.264/AVC patterns in order to analyze the behavior for the baseline profile (IPPP GOP) and the main profile (IBBP pattern). These patterns were transcoded by the reference and the proposed transcoder. The H.264/AVC reference software used in the simulations was theJM reference software (version 17.1). As mentioned in the introduction, the framework described is focused on communications between mobile devices; therefore, a low complexity configuration must be employed. For this reason, we have used the default configuration for the H.264/AVC main and baseline profile, only turning off the RD Optimization. The reference transcoder is composed of the whole DVC decoder followed by the whole H.264/AVC encoder. In order to analyze the performance of the proposed transcoder in detail we have taken into account the two halves and global results are also presented.

Furthermore, the performance of the proposed DVC parallel decoding is shown in Tables 1 (for 15 and 30fps sequences). PSNR and bitrate (BR) display the quality and bitrate measured by the reference WZ decoding. To calculate the PSNR difference, the PSNR of each sequence was estimated before transcoding starts and after transcoding finishes. Then the PSNR of the proposed transcoding was subtracted from the reference one for each H.264/AVC RD point, as defined by Equation 3. However, Table 1 do not include results for ΔPSNR because the quality obtained by DVC parallel decoding is the same as the reference decoding, it iterates until a given threshold is reached (Brites et al., 2008).

$$\Delta PSNR(db) = PSNR_{reference} - PSNR_{porposed} \tag{3}$$

Equation 4 was applied in order to calculate the Bitrate increment (ΔBR) between reference and proposed DVC decoders as a percentage. Then a positive increment means a higher bitrate is generated by the proposed transcoder. As the results of Table 1 show, when DVC decodes smaller and less complex parts, sometimes the turbo decoder (as part of the DVC decoder) converges faster with less iterations and it implies less parity bits requested and thus a bitrate reduction. However, generally speaking the turbo codec yields a better performance for longer inputs. For this reason, the bitrate is not always positive or negative. Comparing different GOP lengths, in short GOPs most of the bitrate is generated by the K frames. When the GOP length increases, the number of K frames is reduced and then WZ frames contribute to reducing the global bitrate in low motion sequences (like Hall) or increasing it in high motion sequences (Foreman or Soccer). Generally, decoding smaller pieces of frame (in parallel) works better for high motion sequences, where the bitrate is similar or even lower in some cases.

$$\Delta BR(\%) = 100 * \frac{(BR_{porposed} - BR_{reference})}{BR_{reference}} \tag{4}$$

Concerning the time reduction (TR), it was estimated as a percentage by using Equation 5. In this case, negative time reduction means decoding time saved by the proposed DVC decoding. As is shown in Table 1, DVC decoding time is reduced by up to 70% on average. TR is similar for different GOP lengths, but it works better for more complex sequences.

$$TR(\%) = 100 * \frac{(Time_{porposed} - Time_{reference})}{Time_{reference}} \quad (5)$$

		15 fps sequences				30 fps sequences			
		Reference DVC decoder		Proposed DVC parallel decoder		Reference DVC decoder		Proposed DVC parallel decoder	
Sequence	GOP	PSNR (dB)	BR (kbps)	ΔBR (%)	TR (%)	PSNR (dB)	BR (kbps)	ΔBR (%)	TR (%)
Foreman	2	30.25	295.58	0.66	-75.99	32.41	504.93	-2.37	-74.29
	4	29.73	450.59	-0.05	-70.88	31.95	648.69	-1.91	-74.7
	8	28.96	571.31	-1.04	-76.98	30.87	804.73	-0.88	-73.23
Hall	2	32.81	222.24	-2.52	-75.14	36.4	439.74	1.33	-70.89
	4	33.1	224.09	7.99	-70.47	36.34	412.8	1.27	-70.83
	8	33.06	224.13	9.05	-69.34	36.03	384.31	6.49	-68.96
Coast Guard	2	30.14	289.84	1.11	-72.8	33.84	592.64	4.69	-71.88
	4	30.13	371.62	1.38	-74.46	33.32	608.2	5.88	-72.51
	8	29.65	437.85	1.91	-74.73	32.24	661.11	4.72	-70.92
Soccer	2	29.56	377.15	-2.67	-74.17	30.52	532.94	1.2	-74.86
	4	29.05	593.66	-2.82	-75.81	30.21	855.23	-0.58	-75.41
	8	28.34	735.48	-3.2	-73.94	29.53	1069.21	-1.19	-74.65
mean				0.82	-73.73			1.55	-72.76

Table 1. Performance of the proposed DVC parallel decoder for 15 and 30 fps sequences (first stage of the proposed transcoder).

Results for the second stage of the transcoder are shown in Tables 2 and 3. In this case, both H.264/AVC encoders (reference and proposed) start from the same DVC output sequence (as DVC parallel decoding obtains the same quality as the reference DVC decoding), which is quantified with four QP values. For these four QP values, ΔPSNR and ΔBRare calculated as specified in Bjøntegaard and Sullivan's common test rule (Sullivan et al., 2001). TR is given by Equation 5. In Table 2, DVC decoded sequences are mapped to an IPPP pattern. In this case RD loss is negligible and TR is around 40%. For 30 fps sequences, the accuracy of the proposed method works better and RD loss is even lower. In addition, Figure 9 displays each plot for each of the four 4 QP values simulated. As can be observed, all RD points are much closer. For the IBBP pattern (Table 3), the conclusions are similar. Comparing both patterns, the IBBP pattern generates a slightly higher RD loss, but H.264/AVC encoding is performed faster (up to 48%). This is because B frames have two reference frames, but dynamic ME search area reduction is carried out in both of them. Figure 10 displays plots for each of the four QP points when an IBBP pattern is performed. As can be observed, the RD drop penalty is negligible.

IPPP H.264 pattern							
		15fps			30fps		
Sequence	GOP	$\Delta PSNR$(db)	ΔBR(%)	TR (%)	$\Delta PSNR$(db)	ΔBR(%)	TR (%)
Foreman	2	-0.02	0.57	-41.57	-0.01	0.31	-42.54
	4	-0.03	0.86	-44.62	0.00	0.18	-41.81
	8	-0.04	1.11	-45.85	-0.01	0.36	-43.71
Hall	2	-0.01	0.41	-30.04	0.00	0.05	-30.50
	4	0.00	0.12	-30.77	0.00	0.06	-29.28
	8	0.00	0.17	-27.21	0.00	0.01	-27.81
Coast Guard	2	-0.01	0.27	-47.46	-0.01	0.19	-46.39
	4	-0.01	0.33	-46.15	0.00	0.08	-45.59
	8	-0.01	0.20	-47.61	0.00	0.09	-45.23
Soccer	2	-0.01	0.19	-38.85	-0.01	0.15	-37.86
	4	-0.04	1.18	-43.35	-0.03	0.84	-40.61
	8	-0.05	1.63	-44.98	-0.03	0.90	-42.35
mean		-0.02	0.59	-40.70	-0.01	0.27	-39.47

Table 2. Performance of the proposed transcoder mapping method for IPPP H.264 pattern with 15fps and 30fps sequences.

IBBP H.264 pattern							
		15fps			30fps		
Sequence	GOP	$\Delta PSNR$(db)	ΔBR(%)	TR (%)	$\Delta PSNR$(db)	ΔBR(%)	TR (%)
Foreman	2	-0.06	1.51	-48.86	-0.08	2.31	-48.17
	4	-0.08	2.00	-51.11	-0.07	2.18	-49.56
	8	-0.07	2.18	-52.19	0.00	0.00	-37.71
Hall	2	-0.01	0.21	-39.06	-0.01	0.24	-37.31
	4	-0.01	0.55	-37.11	-0.01	0.13	-35.74
	8	-0.01	0.48	-36.15	0.00	0.01	-52.41
Coast Guard	2	-0.04	1.13	-49.44	-0.01	0.26	-51.14
	4	-0.04	1.24	-49.90	-0.01	0.37	-50.62
	8	-0.05	1.40	-51.07	-0.05	1.59	-45.19
Soccer	2	-0.02	0.43	-42.93	-0.08	2.63	-46.90
	4	-0.05	1.54	-46.03	-0.08	2.57	-48.52
	8	-0.08	2.52	-48.71	-0.04	1.21	-45.93
mean		-0.04	1.27	-46.05	-0.08	2.31	-48.17

Table 3. Performance of the proposed transcoder mapping method for IBBP H.264 pattern with 15fps and 30fps sequences.

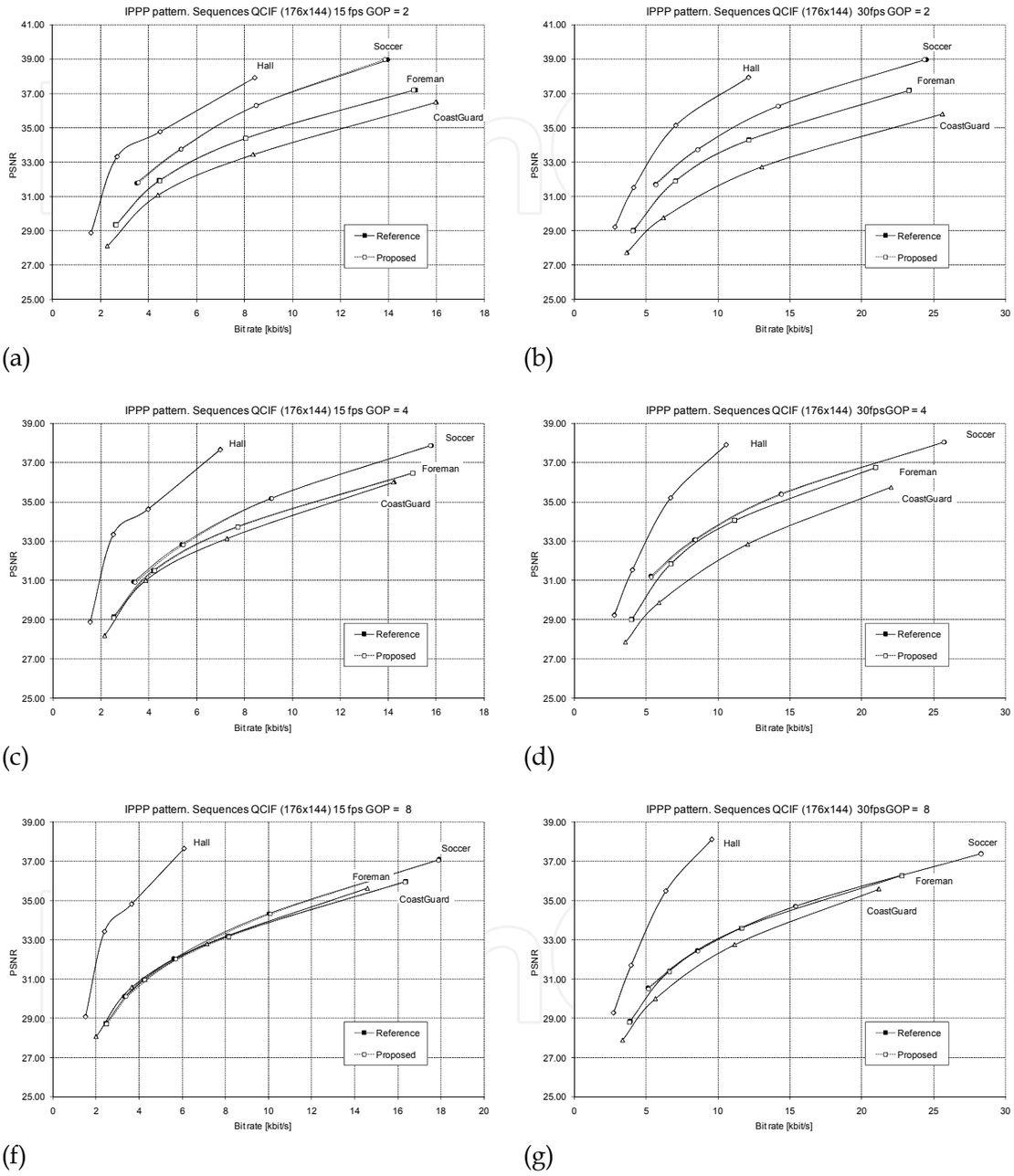


Fig. 9. PSNR/bitrate results transcoding to H.264 IPPP GOP from DVC GOP = 2, 4, and 8 in sequences with 15 and 30 fps. Reference symbols: ■Foreman ♦Hall ▲CoastGuard ●Soccer

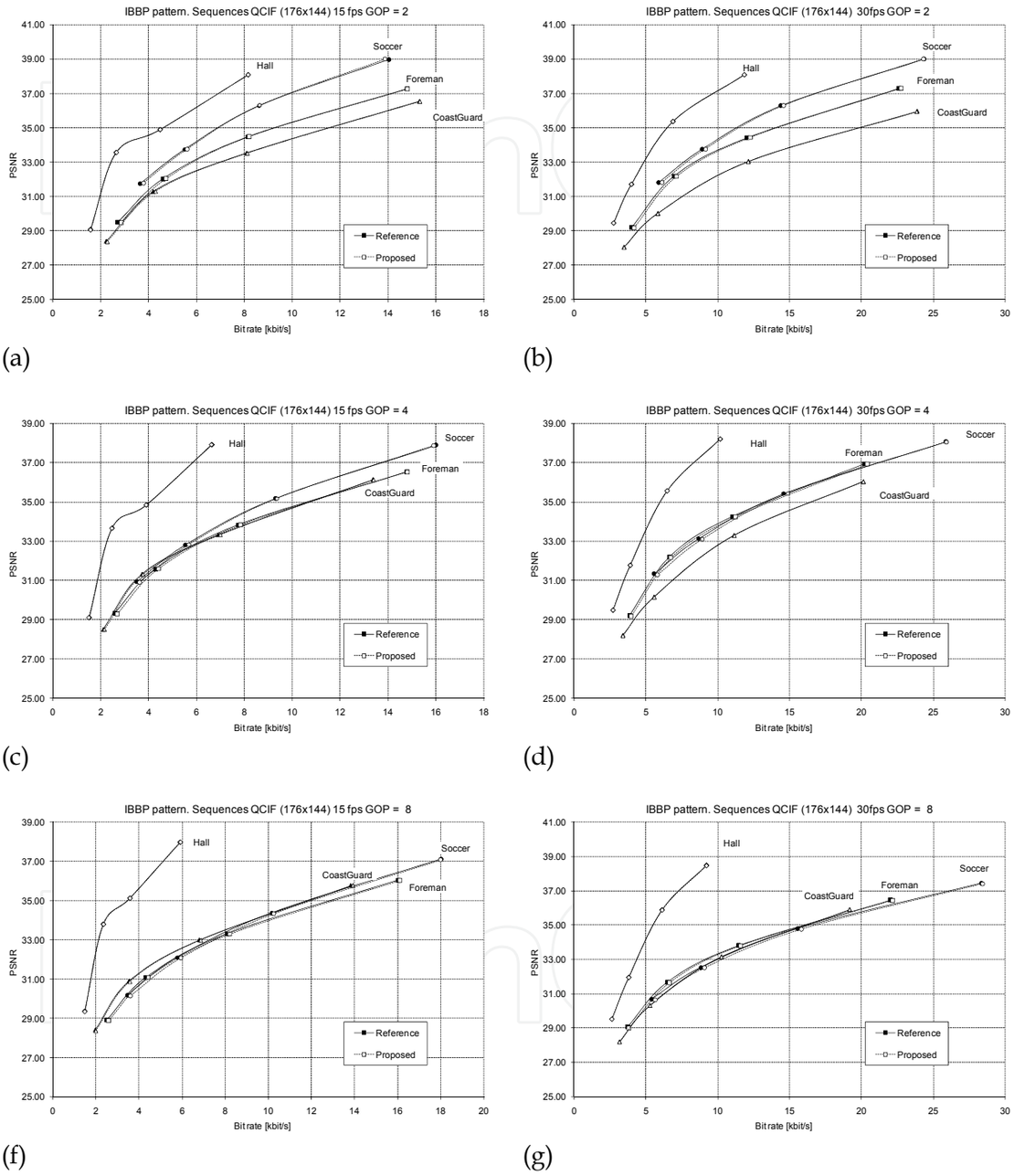


Fig. 10. PSNR/bitrate results transcoding to H.264 IBBP GOP from DVC GOP = 2, 4, and 8 in sequences with 15 and 30 fps. Reference symbols: ■Foreman ♦Hall ▲CoastGuard ●Soccer

15 fps for IPPP H.264 pattern				
Sequence	GOP	$\Delta\overline{PSNR}$ (dB)	$\Delta\overline{BR}$ (%)	$\overline{TR}$ (%)
Foreman	2	-0.02	0.64	-75.50
	4	-0.02	-0.05	-70.66
	8	-0.01	-1.02	-76.78
Hall	2	0	-2.47	-74.48
	4	0	7.86	-70.01
	8	0	8.91	-68.92
Coast Guard	2	0	1.07	-72.41
	4	0	1.35	-74.18
	8	0	1.88	-74.52
Soccer	2	-0.03	-2.62	-73.82
	4	-0.03	-2.78	-75.61
	8	-0.03	-3.15	-73.79
mean		-0.01	0.80	-73.39

Table 4. Performance of the proposed transcoder for 15fps sequences and IPPP pattern.

30 fps for IPPP H.264 pattern				
Sequence	GOP	$\Delta\overline{PSNR}$ (dB)	$\Delta\overline{BR}$ (%)	$\overline{TR}$ (%)
Foreman	2	-0.01	-2.32	-73.72
	4	0	-1.88	-74.34
	8	0	-0.87	-72.99
Hall	2	0	1.31	-70.03
	4	0	1.25	-70.24
	8	0	6.40	-68.48
Coast Guard	2	0	4.59	-71.31
	4	-0.01	5.78	-72.15
	8	-0.01	4.65	-70.65
Soccer	2	-0.02	1.16	-74.39
	4	-0.02	-0.57	-75.15
	8	-0.02	-1.17	-74.45
mean		-0.01	1.53	-72.33

Table 5. Performance of the proposed transcoder for 30fps sequences and IPPP pattern.

15 fps for IBBP H.264 pattern				
Sequence	GOP	$\Delta\overline{PSNR}$ (dB)	$\Delta\overline{BR}$ (%)	$\overline{TR}$ (%)
Foreman	2	-0.03	0.67	-75.12
	4	-0.03	-0.03	-70.51
	8	-0.03	-1.01	-76.63
Hall	2	0	-2.47	-73.99
	4	0	7.86	-69.66
	8	0	8.92	-68.63
Coast Guard	2	-0.01	1.10	-72.00
	4	-0.01	1.37	-73.92
	8	-0.02	1.89	-74.31
Soccer	2	-0.04	-2.61	-73.45
	4	-0.04	-2.77	-75.39
	8	-0.03	-3.15	-73.64
mean		-0.02	0.81	-73.10

Table 6. Performance of the proposed transcoder for 15fps sequences and IBBP pattern.

30 fps for IBBP H.264 pattern				
Sequence	GOP	$\Delta\overline{PSNR}$ (dB)	$\Delta\overline{BR}$ (%)	$\overline{TR}$ (%)
Foreman	2	-0.04	-2.28	-73.25
	4	-0.03	-1.86	-74.06
	8	-0.02	-0.85	-72.80
Hall	2	0	1.31	-69.40
	4	0	1.25	-69.82
	8	0	6.40	-68.13
Coast Guard	2	0	4.60	-70.94
	4	-0.01	5.78	-71.88
	8	0	4.66	-70.46
Soccer	2	-0.04	1.20	-74.01
	4	-0.05	-0.55	-74.92
	8	-0.04	-1.16	-74.29
mean		-0.02	1.54	-72.00

Table 7. Performance of the proposed transcoder for 30fps sequences and IBBP pattern.

Finally, to analyze the global transcoding improvement, Tables 4, 5, 6 and 7 summarize global transcoding performance. In this case, Bjøntegaard and Sullivan´s common test rule (Sullivan et al., 2001) was not used because it is a recommendation only for H.264/AVC. Then, to estimate the PSNR obtained by the transcoder, the original sequences were compared with the output sequences after transcoding. For each four QP points, the PSNR measured is displayed as an average ($\Delta\overline{PSNR}$)). To estimate the BR generated by the reference and the proposed transcoder, the BR generated by both stages (DVC decoding and H.264/AVC encoding) was added. Then equation (1) was applied and it was averaged for each four H.264/AVC QPs ($\Delta\overline{BR}$). As the DVC decoding contributes with most of the bitrate, results are very similar to those in Tables 1. In order to evaluate the TR, total transcoding time was measured for the reference and proposed transcoder. Then Equation 5 was applied and a mean was calculated for each of the four H.264/AVC QPs ($\overline{TR}$). As DVC decoding takes up most of the transcoding time, improvements in this stage have a bigger influence on the overall transcoding time, and so the TR obtained is similar to that in Table 1, reducing the complexity of the transcoding process by up to 73% (on average).

6. Conclusions

In this chapter it is analyzed the transcoding framework for video communications between mobile devices. In addition, it is proposed a WZ to H.264/AVC transcoder designed to support mobile-to-mobile video communications. Since the transcoder device accumulates the highest complexity from both video coders, reducing the time spent in this process is an important goal. With this aim, in this chapter two approaches are proposed to speed-up WZ decoding and H.264/AVC encoding. The first stage is improved by using parallelization techniques as long as the second stage is accelerated by reusing information generated during the first stage. As a result, with both approaches a time reduction of up to 73% is achieved for the complete transcoding process with negligible RD losses. In addition, the presented transcoder performs a mapping for different GOP patterns and lengths between the two paradigms by using an adaptive algorithm, which takes into account the MVs gathered in the side information generation process.

7. Acknowledgements

This work was supported by the Spanish MICINN, Consolider Programme and Plan E funds, as well as European Commission FEDER funds, under Grants CSD2006-00046 and TIN2009-14475-C04-03. It was also supported by JCCM funds under grant PEII09-0037-2328 and PII2I09-0045-9916, and the University of Castilla-La Mancha under Project AT20101802. The work presented was performed by using the VISNET2-WZ-IST software developed in the framework of the VISNET II project.

8. References

Aaron, A., Rui, Z. & Girod, B. (2002). Wyner-Ziv coding of motion video. In: Asilomar Conference on Signals, Systems and Computers, pp. 240-244.

Artigas, X., Ascenso, J., Dalai, M., Klomp, S., Kubasov, D. & Ouaret, M. (2007). The DISCOVER codec: architecture, techniques and evaluation. In: Picture Coding Symposium (PCS), pp. 1-4. Citeseer.

Ascenso, J., Brites, C. & Pereira, F. (2005). Improving frame interpolation with spatial motion smoothing for pixel domain distributed video coding. 5th EURASIP Conference on Speech and Image Processing, Multimedia Communications and Services, Smolenice, Slovak Republic.

Ascenso, J., Brites, C., Dufaux, F., Fernando, A., Ebrahimi, T., Pereira, F. & Tubaro, S. (2010). The VISNET II DVC Codec: Architecture, Tools and Performance. In: European Signal Processing Conference (EUSIPCO).

Brites, C., Ascenso, J., Quintas Pedro, J. & Pereira, F. (2008). Evaluating a feedback channel based transform domain Wyner-Ziv video codec. Signal Processing: Image Communication 23, 269-297.

De Cock, J., Notebaert, S., Vermeirsch, K., Lambert, P. & Van de Walle, R. (2010). Dyadic spatial resolution reduction transcoding for H.264/AVC - Multimedia Systems, 16(2): 139-149.

Fernández-Escribano, G., Kalva, H., Cuenca, P. & Orozco-Barbosa L. (2007). A first approach to speeding-up the inter mode selection in MPEG-2/H.264 transcoders using machine learning. Multimedia Tools Appl. 35(2): 225-240.

Fernández-Escribano, G., Cuenca, P., Orozco-Barbosa, L., Garrido del Solo, A. J. & Kalva, H. (2008). Simple intra prediction algorithms for heterogeneous MPEG-2/H.264 video transcoders. Multimedia Tools Appl. 38(1): 1-25.

Girod, B., Aaron, A.M., Rane, S. & Rebollo-Monedero, D. (2005) Distributed Video Coding. Proceedings of the IEEE 93, 71-83.

Intel Processor Core family. (2011). URL: http://www.intel.com/.

ISO/IEC International Standard 14496-10 (2003). Information Technology – Coding of Audio – Visual Objects – Part 10: Advanced Video Coding.

Martínez, J. L., Kalva, H., Fernández-Escribano, G., Fernando, W.A.C. & Cuenca, P. (2009). Wyner-Ziv to H.264 video transcoder, 16th IEEE International Conference on Image Processing (ICIP), Cairo, Egypt. pp. 2941-2944.

Momcilovic, S., Wang, Y., Rane, S. & Vetro, A. (2010). Toward Realtime Side Information Decoding on Multi-core Processors, IEEE International Workshop on Multimedia Signal Processing (MMSP), Saint-Malo, France.

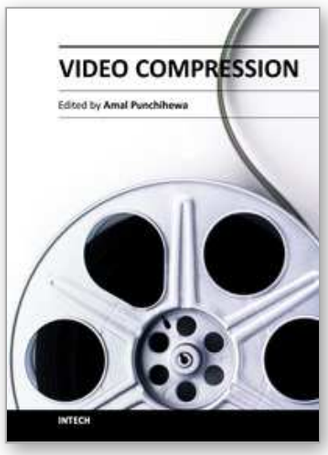
Oh, R., Park, J. & Jeon, B. (2010). Fast implementation of Wyner-Ziv Video codec using GPGPU, IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Shangai, China.

OpenMP. (2011) The OpenMP API specification for parallel programming. URL: http://openmp.org

Peixoto E., Queiroz R. L. & Mukherjee, D. (2010). A Wyner-Ziv Video Transcoder, Circuits and Systems for Video Technology, IEEE Transactions on , vol.20, no.2, pp.189-200.

Sheng, T., Zhu, X., Hua, G., Guo, H., Zhou, J. & Chen, C. (2010). Feedback-free rate-allocation scheme for transform domain Wyner–Ziv video coding - Multimedia Systems, 16(2): 127-137.

Sullivan G. & Bjøntegaard G. (2001). Recommended Simulation Common Conditions for H.26L Coding Efficiency Experiments on Low-Resolution Progressive-Scan Source Material. ITU-T VCEG, Doc. VCEG-N81.



Video Compression

Edited by Dr. Amal Punchihewa

ISBN 978-953-51-0422-3
Hard cover, 154 pages
Publisher InTech
Published online 23, March, 2012
Published in print edition March, 2012

Even though video compression has become a mature field, a lot of research is still ongoing. Indeed, as the quality of the compressed video for a given size or bit rate increases, so does users' level of expectations and their intolerance to artefacts. The development of compression technology has enabled number of applications; key applications in television broadcast field. Compression technology is the basis for digital television. The "Video Compression" book was written for scientists and development engineers. The aim of the book is to showcase the state of the art in the wider field of compression beyond encoder centric approach and to appreciate the need for video quality assurance. It covers compressive video coding, distributed video coding, motion estimation and video quality.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Alberto Corrales Garcia, Gerardo Fernandez Escribano, Jose Luis Martinez and Francisco Jose Quiles (2012). Mobile Video Communications Based on Fast DVC to H.264 Transcoding, Video Compression, Dr. Amal Punchihewa (Ed.), ISBN: 978-953-51-0422-3, InTech, Available from: http://www.intechopen.com/books/video-compression/mobile-video-communications-based-on-fast-dvc-to-h-264-transcoding

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen