Novel Video Transcoding System to Efficiently Realize Combinations of Use Cases

Hicham Layachi[1] **and Stéphane Coulombe**[1]
[1]Dept. of Software and IT Engineering, École de technologie supérieure, Université du Québec, Canada

Abstract—*In this paper, we propose a novel heterogeneous video transcoding system that is not only capable of performing the major operations expected of a transcoder, i.e. bit rate adaptation, spatial and temporal resolution scaling, format conversion between the most often used standards on the market, but also any combination of operations. In addition, we propose novel transcoding algorithms to reduce the number of candidate modes and motion estimation refinement operations, leading to improved quality as well as reduced computational complexity. The unified system is implemented using Intel IPP Code Sample. The results show that the proposed unified system can, depending on the use case, be more than 2 times faster than the cascade transcoder, with small video quality degradation.*

Keywords: Video transcoding, bit rate adaptation, format adaptation, H.264, MPEG-4.

1. Introduction

Transcoding consists of modifying and adapting the content of a precompressed bitstream into another video bitstream [1]–[3]. Each bit-stream is characterized by a group of properties: the bit rate, the spatial resolution, the frame rate, and the format used to encode the video bit-stream. The role of the transcoder is becoming crucial in our modern life for maintaining a high level of interoperability in multimedia systems [1], [2] where each component might have its own features and capabilities. The cascaded architecture that consists of decoding the compressed video stream, performing manipulations in the pixel domain (scaling, logo insertion, etc.), and re-encoding to meet the output requirements is the straightforward method for performing any video transcoding use case or group of use cases. Unfortunately, since the computationally intensive motion estimation (ME) and mode decision must be redone, the cascaded transcoder is quite expensive in terms of computational complexity, and is thus problematic for real-time applications and commercial software. In the literature, several efficient video transcoding architectures have been proposed [1]–[3], in the pixel domain or in the DCT domain, to perform bit rate adaptation, spatial resolution adaptation, frame rate adaptation, logo insertion, and format conversion. Nevertheless, the majority of these transcoders are standalone devices and address the implementation of a single use case. Thus, in some works, bit rate reduction has been implemented [4], [5], motion vector (MV) determination and mode mapping being the major problems of interest. Spatial resolution adaptation was presented in [1], [2], [6]–[10], where resampling, MVs and mode mapping were the main issues addressed. The process of MV derivation for the newly generated frames was presented in [8], [11], [12] for adapting the frame rate to the desired temporal resolution. The conversion from one format to another has been studied in [2], [8], [9], [13], [14]. These works addressed the issues of modes and MV mapping, as well as syntax conversion.

Despite the fact that many video transcoding architectures and algorithms have been proposed, only a few works have studied the possibility of integrating the majority of the transcoding use cases in the same transcoder. In [8], the authors implemented a heterogeneous transcoder for format conversion of MPEG-1/2 to lower bit rate H.261/H.263. Even though the algorithm of each use case was detailed in depth in this work, a procedure for performing the proposed use cases in combination was missing. In [9], the authors proposed an architecture for performing transcoding from an interlaced MPEG-2 to MPEG-4 visual simple profile (VSP) with spatio-temporal resolution reduction. However, this work provided a small number of experimental results to validate the proposed transcoder. Unfortunately, these few transcoders were limited to performing a specific group of use cases (spatio-temporal resolution reduction with format conversion), where the format conversion was performed between two specific standards (MPEG-2 to H.263, or MPEG-2 to MPEG-4). In addition, they lack flexibility (they perform either the proposed group of use cases or nothing at all; they cannot select a sub-group of these use cases).

State-of-the-art algorithms use a mode mapping table derived from experiments to determine the most probable mode(s) for each input mode [5], [14]. This table allows reducing computational complexity without significantly affecting quality. But such methods have not been proposed for each transcoding use case. Also, they either reuse directly the MVs (reducing computational complexity as well as quality) or refine every MV (increasing quality and complexity). But a method offering good quality at reduced complexity by performing conditional MV refinement is lacking.

This work was supported in part by Vantrix Corporation and in part by the Natural Sciences and Engineering Research Council of Canada under the Collaborative Research and Developement Program (CRD 326637-05).

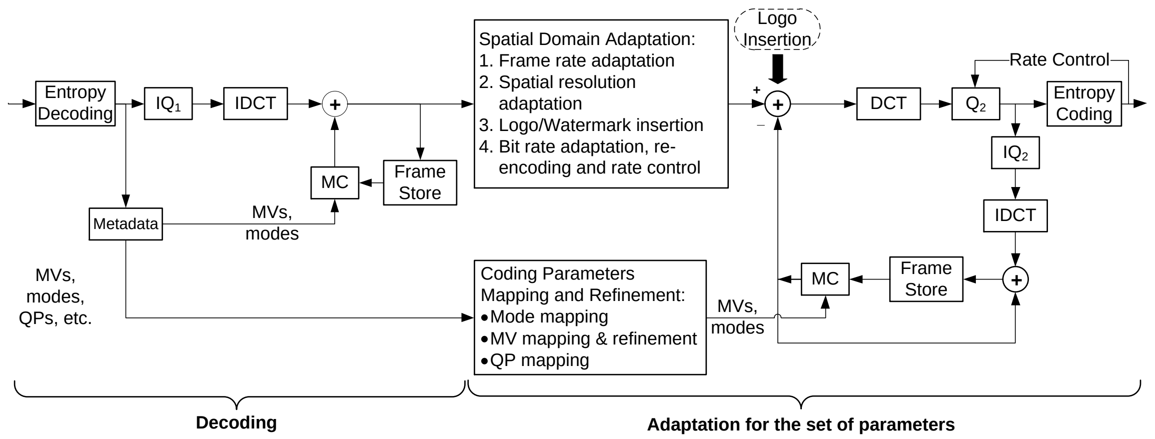

Fig. 1: The architecture of the proposed unified video transcoding system.

In this paper, we propose a new heterogeneous video transcoder capable of efficiently realizing a single video transcoding use case, (or transcoding operation) or a combination of these use cases. The proposed transcoder exploits the incoming metadata retrieved from the decoding process to adapt the video content to the desired output characteristics. We also propose novel or improved mode mapping algorithms for more use cases and new motion estimation (ME) algorithms that refine the MVs under certain conditions to reduce complexity while maintaining good quality. This paper is organized as follows. The proposed unified video architecture for transcoding is introduced in section 2. In section 3, we describe in depth the algorithms used in each combination of operations. The experimental results are presented in section 4. Section 5 summarizes the paper's results and outlines future work.

2. Proposed architecture

As mentioned previously, there is an urgent need to unify the results of the various research efforts and develop an architecture that supports several use cases in the same transcoder. This seems a feasible target, considering that the proposed transcoding architectures use the same approach (1. decoding; 2. video adaptation, and 3. re-encoding) and the same blocks of operations (quantization, transform, entropy coding, ME, etc.). We expect that a unified architecture will lead to more efficient transcoding system development and maintenance, along with improved extensibility. More specifically, the novel unified video transcoding architecture we propose in this paper, and shown in Fig. 1, is aimed at:

1) Efficiently supporting a single transcoding use case, or set of use cases, while providing the best quality. This is achieved by exploiting the metadata extracted from the decoder and reusing them for the adapted video at output. We propose the intelligent reuse of the stored metadata to reduce the computational complexity by reusing the MVs and refining them under certain conditions;

2) Reducing the effort required for software development and maintenance by centralizing several use cases in the same transcoder and improving software component reuse;

3) Achieving extensibility by allowing the easy addition of new transcoding operations and their combinations.

To meet these goals, the proposed unified architecture is developed in the pixel domain. This allows to: 1) avoid the drift error and allow maximum flexibility of the re-encoder; 2) support all the codecs, especially H.264, which cannot implement its deblocking filter in the DCT domain [15]; 3) integrate other use cases that may be added in the future.

In transcoding a video, the incoming video stream is completely decoded in the pixel domain and the necessary metadata are stored. These metadata primarily encompass: frame types (I, P, B, etc.) and slice types, MB modes and sub-modes, MVs, quantization parameters (QP); and coded block patterns (CBP). The decoded video content is then adapted to the desired output properties using the spatial domain adaptation block and the coding parameter mapping and refinement block illustrated in Fig. 1. Finally, the adapted video is re-encoded with rate control.

As shown in Fig. 1, the novel unified video transcoding architecture is similar to the cascaded spatial domain architecture, but with the addition of the following blocks:

1) The metadata block: used to store the metadata extracted from the decoding stage.
2) The spatial domain adaptation block: used to perform the desired use case or group of use cases; it is adapted in accordance with the video's input/output characteristics.
3) The coding parameter mapping and refinement block: used to select the mapping algorithms and refinement methods to be used by looking at the input and output properties of the video. The mapping algorithms take the MVs and modes retrieved from the decoder and map them to the output MVs and modes to be used for the re-encoding following an optional MV refinement process.

Fig. 2 illustrates the proposed approach to implementing a group of video transcoding use cases, the order of which is as follows:

1) Decoding: we decode the input video stream to the pixel domain entirely, and then store the metadata.

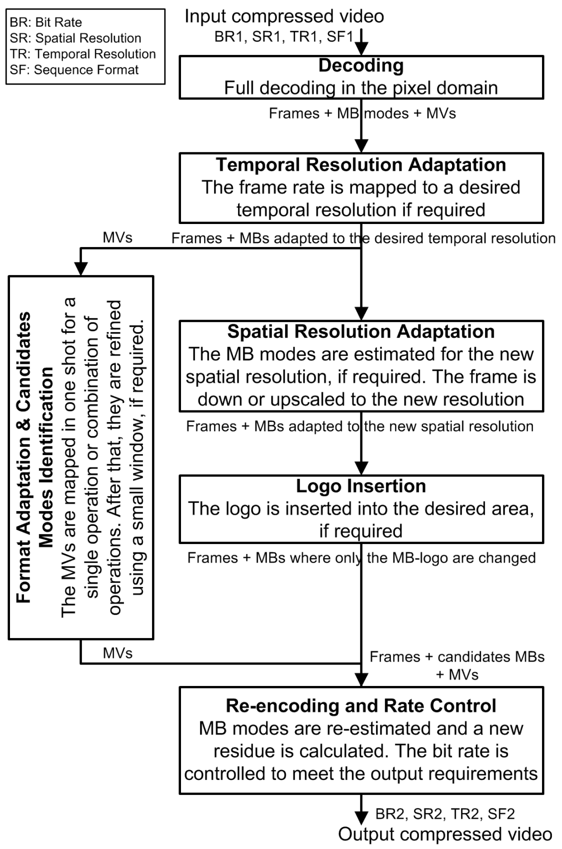
BR: Bit Rate
SR: Spatial Resolution
TR: Temporal Resolution
SF: Sequence Format
Input compressed video
BR1, SR1, TR1, SF1
Decoding
Full decoding in the pixel domain
Frames + MB modes + MVs
Temporal Resolution Adaptation
The frame rate is mapped to a desired temporal resolution if required
MVs
Frames + MBs adapted to the desired temporal resolution
Format Adaptation & Candidates Modes Identification
The MVs are mapped in one shot for a single operation or combination of operations. After that, they are refined using a small window, if required.
Spatial Resolution Adaptation
The MB modes are estimated for the new spatial resolution, if required. The frame is down or upscaled to the new resolution
Frames + MBs adapted to the new spatial resolution
Logo Insertion
The logo is inserted into the desired area, if required
Frames + MBs where only the MB-logo are changed
MVs
Frames + candidates MBs + MVs
Re-encoding and Rate Control
MB modes are re-estimated and a new residue is calculated. The bit rate is controlled to meet the output requirements
BR2, SR2, TR2, SF2
Output compressed video

Fig. 2: Flowchart of the operations of the proposed unified video transcoding system.

2) Temporal resolution adaptation: if a temporal resolution adaptation is required, we map the decoded frames, MVs, and modes to new MVs and modes suitable for the new temp. resolution, in accordance with the input standard.
3) Spatial resolution adaptation: if a spatial resolution adaptation is required, we map the decoded frame resolutions, MVs, and modes to new frame resolutions, MVs and modes, in accordance with the output standard.
4) Logo/watermark insertion: if a logo/watermark is required, only the logo MBs will be affected and will be processed in accordance with the output standard.
5) Bit rate adaptation, re-encoding, and rate control.

This processing order is the result of our analysis of several video transcoding architectures which have been presented in the literature. Each processing step (or use case) can be either invoked or skipped, depending on the desired output characteristics. Thus, any combination of use cases could be performed. A single use case is invoked when all the other use cases are skipped.

If all the use cases are required, the proposed unified architecture behaves as follows. The temporal resolution adaptation is performed first, using the input format, so that we can work on the right number of frames, whether an increase or decrease of the frame rate is desired. The resulting MVs and MB modes constitute the input of the next phase. Then, the remaining operations are performed, and the MV and mode mapping are completed in a single step, because logo/watermark insertion affects only a few MBs in the frame. We adjust the spatial resolution, insert the logo/watermark at the desired locations, and adapt the bit rate. The MV and mode decisions are mapped in accordance with the output format for the entire group of operations. Finally, we re-encode, using the output standard syntax and bit rate. At this final stage, the rate is controlled.

When a combination containing fewer operations is required, the architecture will simply skip the unwanted use cases. For example, if a spatial resolution adaptation with logo insertion is needed, the proposed transcoder decodes the video and extracts the metadata. After that, the transcoder performs a single MV and mode mapping for both operations and performs MV refinement, if needed. Finally, it re-encodes the video. It should be noted that, when all the use cases are needed, MV mapping is achieved in two stages. For the first stage, a one-to-one mapping is performed for the temporal resolution adaptation, in accordance with the input format. For the remaining operations, a one-to-many mapping is performed (i.e. one MV and mode can generate multiple candidate MVs and modes). This reduces the complexity of the proposed architecture. Otherwise, when the temporal resolution adapt. is not needed for a group of use cases, the mapping can be achieved in one stage.

3. Proposed algorithms

In this section, we present the proposed algorithms for the two possible video transcoding scenarios: a single use case, and a set of use cases. For a single use case, we focus, in this work, on bit rate adaptation using the most complicated video standard, H.264 (section 3.1), and the spatial resolution adaptation, also using H.264 (section 3.2). Then, for the set of use cases, we present H.264 Baseline Profile (BP) to the MPEG-4 part 2 Visual Simple Profile (VSP) format adaptation with bit rate adaptation (section 3.3). The proposed algorithms address new use cases or improve over existing ones in terms of speed and quality. These algorithms are implemented using Intel Integrated Performance Primitive Code Samples, version 5.3 [16]. The proposed architecture can be applied to other combinations of operations as well. Because of the limited space in this paper, we could not validate our architecture with more use cases but the reader can verify that it can accommodate various other combinations of use cases.

3.1 Bit rate adaptation

In all standards, bit rate reduction transcoding relies on MV and mode decision mapping from the input to the output bit-stream. The H.264/AVC standard has become the leading standard for different multimedia applications, like video streaming, broadcasting, DVD, Blu-ray, etc. Moreover, its predecessors are considered as a subset of it, and thus any algorithm developed for H.264 can easily be adapted to other

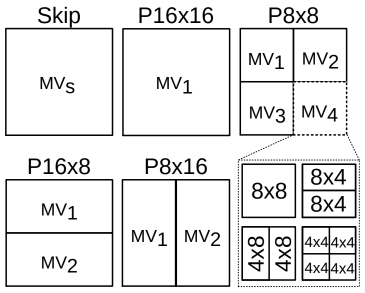

Fig. 3: Inter modes and motion vectors for H.264.

standards. For these reasons, in this work, we implement bit rate adaptation in H.264, and the proposed algorithm can be adapted to VC-1, MPEG-4 part 2, etc.

Compared to the state-of-the-art algorithms [5], ours, called *candidate mapping (CM)* , does not test 8×8 sub-partitions (P8×4, P4×8, and P4×4), because their use is computationally high, with no significant quality gain at lower bit rates. In addition, not all output modes are checked (RDO is not used), owing to the use of Intel IPP Code Sample, which uses predefined thresholds for mode selection. Moreover, we have derived a second algorithm, called *candidate mapping with conditional refinement (CMWCR)*, by further exploiting input sub-pixel precision. The *CM* algorithm for intra and inter frames is shown in Table 1. In this paper, in all the mode mapping tables, we will refer to the inter modes and MV numbering shown in Fig. 3. Inter modes begin with P (e.g. P16×16) and intra modes with I (e.g. I4×4). The input column shows the various modes and MVs received while the output column shows the modes to test, separated by $+$, along with MV parameters to use. Also, the following notations will be used: MV_z denotes the zero MV (i.e. $(0, 0)$), MV_p denotes the predicted MV (typically computed based on neighbouring MVs), MV_s denotes the MV implicitly associated with the skip mode (e.g. MV_z for MPEG-4 part 2 and MV_p for H.264), AVG($MV_{i=1,2,3,4}$) denotes the average of MV_1 to MV_4. When multiple MVs are evaluated (considered) in a candidate partition, we use a comma to separate them (e.g. P16×16(MV_1,MV_2) to test both MV_1 and MV_2 on the P16×16 partition). When MVs are associated to specific partitions, we use a semi-column to separate them (e.g. P16×8(MV_1;MV_2) to test MV_1 on the top P16×8 partition and MV_2 on the bottom P16×8 partition). When a mode is tested conditionally, we specify the condition. For instance, I16×16 + I4×4 if SAD(I16×16) $> T_1$ means that I16×16 is always tested while I4×4 is only tested if the SAD associated with I16×16 exceeds a threshold T_1. These thresholds are either predefined in the Intel IPP codec or compatible with them.

For inter frames, when input modes are I16×16 or I4×4, re-encoding the intra MBs directly as intra MBs resulted in degraded quality. Also, when input modes are P16×16, we could have tested the predicted MV as well but it did not noticeably impact the performance. It is important to note that, for all the methods and all the use cases presented in

Table 1: Proposed H.264 bit rate adapt. mode mapping.

Input	**Output (modes + MVs to check)**
Intra frame	
I16×16	I16×16
I4×4	I16×16 + I4×4 if SAD(I16×16)$>T_1$
Inter frame	
Skip(MV_s)	Skip(MV_s) + P16×16(MV_p)
P16×16(MV_1)	Skip(MV_s) + P16×16(MV_1)
P16×8(MV_1;MV_2)	Skip(MV_s) + P16×16(AVG(MV_1,MV_2)) + P16×8(MV_1;MV_2)
P8×16(MV_1;MV_2)	Skip(MV_s) + P16×16(AVG(MV_1,MV_2)) + P8×16(MV_1;MV_2)
P8×8(MV_1;MV_2; MV_3;MV_4)	Skip(MV_s) + P16×16(AVG($MV_{i=1,2,3,4}$)) + P16×8(AVG($MV_{i=1,2}$);AVG($MV_{i=3,4}$)) + P8×16(AVG($MV_{i=1,3}$);AVG($MV_{i=2,4}$)) + P8×8(MV_1;MV_2;MV_3;MV_4)
I16×16	Skip(MV_s) + P16×16(MV_z) + I16×16 if MIN(SAD(Skip(MV_s)),SAD(P16×16(MV_z)))$>T_2$
I4×4	Skip(MV_s) + P16×16(MV_z) + I16×16 if MIN(SAD(Skip(MV_s)),SAD(P16×16(MV_z)))$>T_2$ + I4×4 if SAD(I16×16)$>T_1$

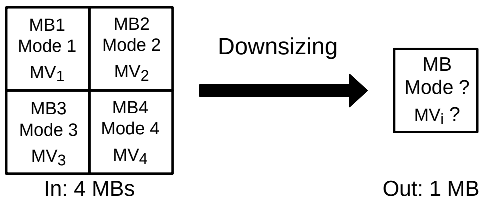

Fig. 4: The problem of mode and MV mapping in spatial resolution reduction by an integer factor of 2.

this paper, the candidate MVs for each candidate mode are tested at integer pel precision (i.e. input MVs are rounded to the nearest integer pel before evaluating the lagrangian cost function). It is also important to note that modes are tested in the following order for H.264: skip, P16×16, P8×8, P16×8, P8×16, I16×16, and I4×4 (for MPEG-4 part 2, the order will be skip, P16×16, P8×8, intra). Also, when testing multiple candidate modes, if the lagrangian cost function of a mode is below a certain threshold, subsequent modes are not tested. Again, throughout this paper, for each candidate mode, once the best candidate MV is identified, we perform sub-pixel refinement using two techniques. In the first *CM*, a ±1 search window (at quarter-pel precision) is tested to determine the final MVs. For the second, *CMWCR*, we use the quarter pixel precision of the input MV corresponding to the best integer pel candidate. However, if the best candidate is the predicted MV, then the predicted MV is refined using ±1 search window (at quarter-pel precision). This has a remarkable effect on the speed-up of the proposed transcoder, while maintaining the same level of quality as our *CM* algorithm.

3.2 Spatial resolution adaptation

Fig. 4 illustrates the spatial resolution reduction problem when downsizing by an integer factor of 2. Our proposed algorithm is developed using H.264, and can easily be adapted to VC-1, MPEG-4 part 2, etc. The filter used for pixel sub-sampling is an integrated filter in Intel IPP.

Compared to the state-of-the-art algorithms [10], our *candidate mapping* algorithm proposes a direct mapping

Table 2: Proposed H.264 resolution adapt. mode mapping.

Input	Output (modes + MVs to check)
Intra frame	
I16×16 or I4×4	I16×16 + I4×4
Inter frame	
4 skip($MV_{s,1}$;$MV_{s,2}$; $MV_{s,3}$;$MV_{s,4}$)	Skip(MV_s) + P16×16$\left(\frac{1}{2}\text{AVG}(MV_{s,i=1,2,3,4})\right)$
4 P16×16(MV_1;MV_2; MV_3;MV_4)	Skip(MV_s) + P16×16$\left(\frac{1}{2}\text{AVG}(MV_{i=1,2,3,4})\right)$
4 inter (heterogeneous inter modes except 4 skip or 4 P16×16) (MV_1;MV_2;MV_3;MV_4)	Skip(MV_s) + P16×16$\left(\frac{1}{2}\text{AVG}(MV_{i=1,2,3,4})\right)$ + P16×8$\left(\frac{1}{2}\text{AVG}(MV_{i=1,2});\frac{1}{2}\text{AVG}(MV_{i=3,4})\right)$ + P8×16$\left(\frac{1}{2}\text{AVG}(MV_{i=1,3});\frac{1}{2}\text{AVG}(MV_{i=2,4})\right)$ + P8×8$\left(\frac{1}{2}MV_1;\frac{1}{2}MV_2;\frac{1}{2}MV_3;\frac{1}{2}MV_4\right)$
nb_mb_intra4×4>1 or nb_mb_intra16×16>1	Same candidates as the 4 inter case (see previous case) + I16×16 + I4×4

when the four input MBs are either four skip or four P16×16, and does not test 8×8 sub-partitions, because they lead to higher complexity without significant quality benefits. As before, not all output modes are checked (RDO is not used). Moreover, we derived two additional algorithms, called *CMWCR* and *candidate mapping with no refinement (CMWNR)*, by further exploiting the input sub-pixel precision. The proposed *CM* algorithm, computed from mode mapping statistics of several sequences, is shown in Table 2. For inter frames, 8×8 input sub-partitions are first transformed into 8×8 blocks, i.e. 8×4, 4×8, and 4×4 sub-partitions are merged into an 8×8 block via MV averaging of sub-partitions. Then, each MB (MB1, MB2, MB3, and MB4 in Fig. 4) is changed to P16×16. This is achieved by merging two P16×8 or P8×16 via MV averaging, or four P8×8 via MV averaging; or by taking the skipped MV (MV_s) or P16×16 MV. The input MVs in Table 2 refer to these merged MVs for each 16×16 partition shown in Fig. 4. It should be noted that all input MVs are downsized by half (divided by 2). For P16×16, the four MVs can be individually tested instead of testing only the average. But the gain in quality in not significant compared to the increase in computationally complexity. Similar conclusions are reached when testing the two corresponding MVs for each partition, for P16×8 and P8×16. For P8×8, sub-partition blocks are not used in the proposed method, because, with a lower bit rate, the encoder uses fewer small partitions.

3.3 Format adaptation with bit rate adaptation

As part of the proposed unified architecture, we propose a novel algorithm for format adaptation from H.264 BP to MPEG-4 VSP. This latter has limited features compared to H.264. For inter MBs, only 16×16 and 8×8 partitions are supported with skip mode limited to the zero MV. For intra MBs, only intra16×16 is supported. In addition, MPEG-4 VSP adopts half-pixel MV precision. So, we first need to alter the input quarter-pel MVs to half-pel MVs.

The *CM* algorithm is a variant of state-of-the-art methods found in [14]. Furthermore, we do not use RDO (not all output modes are checked), owing to the use of Intel IPP, which uses predefined thresholds for mode selection. Moreover, we derived a second algorithm called *CMWCR* by further exploiting input half-pixel precision. The *CM* algorithm is presented in Table 3. As described before, we first convert the sub-partitions to 8×8 blocks using MV averaging. As before in the case of bit rate adaptation, two techniques are used for the sub-pel refinement (but using Table 3 and half-pel instead of quarter-pel precision). The *CM* and the *CMWCR* algorithms can be used for format conversion from H.264 to MPEG-4 part 2 with bit rate adaptation.

Table 3: Proposed format adaptation mode mapping from H.264 to MPEG-4 part 2.

Input	Output (modes + MVs to check)
Intra frame	
I16×16 or I4×4	intra
Inter frame	
Skip(MV_s)	Skip(MV_z) + P16×16(MV_p,MV_z)
P16×16(MV_1)	Skip(MV_z) + P16×16(MV_p,MV_1)
P16×8(MV_1;MV_2)	Skip(MV_z) + P16×16(MV_p,AVG(MV_1,MV_2))
P16×8(MV_1;MV_2)	Skip(MV_z) + P16×16(MV_p,AVG(MV_1,MV_2))
P8×8(MV_1;MV_2; MV_3;MV_4)	Skip(MV_z) + P16×16(MV_p,AVG($MV_{i=1,2,3,4}$)) + P8×8(MV_1;MV_2;MV_3;MV_4)
I16×16 or I4×4	Skip(MV_z) + P16×16(MV_p,MV_z) + intra

4. Experimental results

In order to evaluate the proposed unified video transcoding system, three sets of experiments were performed for the proposed video transcoding operations: 1) bit rate adaptation, 2) spatial resolution adaptation, and 3) format adaptation. These experiments were performed using Intel Performance Primitives (Intel IPP) Code Samples [16], which supports the H.264 BP [17] and the MPEG-4 VSP [18]. Unlike the reference software, JM [19] and MoMuSys [20], that uses a rate distortion optimization RDO loop to obtain the best quality and the lowest bit rate possible for each MB, and are computationally expensive, real-time software and much commercial software use predictive algorithms to rapidly determine the MB mode using the SAD/SATD or other metrics (IPP uses SATD). This provides us with a good approximation of the speed benefits that can be achieved by the proposed architecture and algorithms in real-time transcoders. A total of 9 CIF sequences with low, medium, and high motion were tested (but only 6 are shown in the tables). Only one reference frame was used in the simulations. IPP proposes several ME algorithms for each video codec. Within the scope of this work, the EPZS algorithm (option 3) was used for H.264, and the logarithmic ME algorithm (option 4) was used for MPEG-4 part 2. No full search was used. The speed-up obtained and the quality degradation are given for each method compared with the cascaded method. This latter does not use RDO, but a predictive algorithm based on predefined thresholds for inter or intra modes. We used Intel's default thresholds.

In Tables 4 to 6, we present the experimental results of the bit rate reduction for H.264, the spatial resolution reduction for H.264, and the format adaptation from H.264 BP to MPEG-4 VSP using various CIF sequences (similar results were obtained with QCIF sequences). For the cascaded approach, the PSNR between the input and the transcoded video sequences is computed. For the other methods, the PSNR results show the PSNR differences between each alternative algorithm and the cascaded approach. Speed-up is defined as the ratio between the transcoding time required by the cascaded approach and that required by each algorithm. The sequences were encoded at 512 or 256 kbps with one intra frame in every 100 frames. For all the experiments, in the *statistics* method, only the input modes are transmitted to the encoder, which selects the best output mode using Table 1, 2, or 3, depending on the use case. But unlike *CM* and *CMWCR*, the MVs are obtained from a new ME performed in the pixel domain to show the speed improvements and quality degradation due to the proposed mode mappings only.

For bit rate adaptation, the *statistics* method is, on average, 60% faster than the cascaded method while the *CM* algorithm is more than twice as fast as the cascaded method. The quality obtained by the *CM* algorithm is, on average, similar to that obtained by the cascaded approach for higher input and output bit rates. But the quality difference increases with a reduction of these input and output bit rates. For H.264 spatial resolution reduction, the *statistics* method is, on average, 18% faster than the cascaded method while the *CM* algorithm is approximately 1.5 times faster than the cascaded method with a reasonable PSNR degradation of about 0.3dB. The *CMWCR* algorithm performs 17% faster than the *CM* algorithm, but adds another 0.3dB in quality deterioration. The *CMWNR* algorithm is the fastest algorithm (more than twice as fast), but with a high quality loss of 1dB, showing that refinement affects significantly the quality. For format adaptation from H.264 BP to MPEG-4 VSP, the *statistics* method is, on average, 22 to 32% faster than the cascaded method while the *CM* algorithm is 1.34 to 1.84 times faster than the cascaded method. The reason why the speed-ups obtained are less impressive than those of the previous operations is that MPEG-4 part 2 has only 2 inter modes and 1 intra mode, compared to H.264 (4 inter modes and 4 sub-partitions for the fourth mode, and 2 intra modes with several directional predictions). As Intel IPP is highly optimized, the speed-ups obtained are quite acceptable. Meanwhile, the resulting quality degradation is, on average, close to 1dB, compared with the cascaded approach. Therefore, the *CM* algorithm is significantly faster than the cascaded and *statistics* methods, but this benefit comes with a significant penalty in terms of quality. The *CMWCR* method performs faster than the *CM* method, but degrades quality even more. These results show that the *statistics* method using the proposed mode mappings improves speed without affecting significantly the quality. Further speed-ups can be obtained by refining MVs but such refinements need to be adapted to each use case (e.g. size of refinement region, performing it conditionnally or not) in order to maintain good quality.

5. Conclusion

In this paper, we proposed a unified video system capable of performing a group of video transcoding use cases. Such system would reduce the cost of developing the software dramatically, and decrease the effort required for maintenance and updating. Moreover, it allows to easily integrate new use cases that could be required in the future, instead of developing new stand-alone architectures to support them. New mode and MVs mapping algorithms for several stand-alone operations and groups of operations were also proposed. The results show that the proposed unified system can, depending on the use case, be more than 2 times faster than the cascaded transcoder, with small video quality degradation. Although, for some use cases, MV ref. algorithms need to be improved, this paper showed the capabilities of the proposed system.

References

[1] A. Vetro, C. Christopoulos, and H. Sun, “Video transcoding architectures and techniques: an overview,” *Signal Processing Magazine, IEEE*, vol. 20, no. 2, pp. 18 – 29, Mar. 2003.

[2] I. Ahmad, X. Wei, Y. Sun, and Y.-Q. Zhang, “Video transcoding: an overview of various techniques and research issues,” *Multimedia, IEEE Transactions on*, vol. 7, no. 5, pp. 793 – 804, Oct. 2005.

[3] J. Xin, C.-W. Lin, and M.-T. Sun, “Digital video transcoding,” *Proceedings of the IEEE*, vol. 93, no. 1, pp. 84 –97, Jan. 2005.

[4] H. Sun, W. Kwok, and J. Zdepski, “Architectures for MPEG compressed bitstream scaling,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 6, no. 2, pp. 191 –199, Apr. 1996.

[5] Z. Peng, H. Qing-Ming, and G. Wen, “Key techniques of bit rate reduction for H.264 streams,” in *Proc. Pacific-Rim Conf. Multimedia (PCM 2004)*, vol. 3332/2005, Nov. 2004, pp. 985 –92.

[6] W. Zhu, K. H. Yang, and M. J. Beacken, “CIF-to-QCIF video bitstream down-conversion in the dct domain,” *Bell Labs Technical Journal*, vol. 3, pp. 21 – 29, Feb. 1998.

[7] N. Bjork and C. Christopoulos, “Transcoder architectures for video coding,” *Consumer Electronics, IEEE Transactions on*, vol. 44, no. 1, pp. 88 –98, Feb. 1998.

[8] T. Shanableh and M. Ghanbari, “Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats,” *Multimedia, IEEE Trans. on*, vol. 2, no. 2, pp. 101 –110, June 2000.

[9] X. Jun, S. Ming-Ting, and C. Kangwook, “Motion re-estimation for MPEG-2 to MPEG-4 simple profile transcoding,” in *Int. Packet Video Workshop*, Apr. 2002.

[10] L. Chih-Hung, W. Chung-Neng, and C. Tihao, “A fast downsizing video transcoder based on H.264/AVC standard,” in *Pacific-Rim Conf. Multimedia (PCM) 2004*, Nov. 2004, pp. 215–23.

[11] Y. Jeongnam, S. Ming-Ting, and L. Chia-Wen, “Motion vector refinement for high-performance transcoding,” *Multimedia, IEEE Transactions on*, vol. 1, no. 1, pp. 30 –40, Mar. 1999.

[12] L. Qiang, L. Xiaodong, and D. Qionghai, “Motion information exploitation in H.264 frame skipping transcoding,” in *9th Intl Conf. on Advanced Concepts for Intelligent Vision Systems, ACIVS 2007*, vol. vol. 4678 NCS: Springer Verlag, Aug. 2007, pp. 768–776.

[13] I. Metoevi and S. Coulombe, “Efficient MPEG-4 to H.264 transcoding exploiting MPEG-4 block modes, motion vectors, and residuals,” in *Communications and Information Technology, 2009. ISCIT 2009. 9th Intl Symposium on*, Sept. 2009, pp. 224 –229.

[14] J.-H. Hur and Y.-L. Lee, “H.264 to MPEG-4 transcoding using block type information,” in *TENCON 2005 2005 IEEE Region 10*, Nov. 2005, pp. 1 –6.

[15] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, “Adaptive deblocking filter,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 614 –619, July 2003.

[16] “Intel Integrated Performance Primitives 5.3 - Code Samples.” 2008, [Online] http://software.intel.com/en-us/articles/intel-integrated-performance-primitives-code-samples/.

Table 4: Acceleration and PSNR variation (dB) for H.264 BP bit rate reduction using CIF videos compared to the cascaded method for 512 to 384 kbps, 256 to 192 kbps, and 256 to 128 kbps. Average contains also news, silent and stefan sequences.

		512 to 384 kbps				**256 to 192 kbps**				**256 to 128 kbps**			
CIF videos		Cascade	Stats	*CM*	*CMWCR*	Cascade	Sta-	*CM*	*CMWCR*	Cascade	Stats	*CM*	*CMWCR*
container	Accel.	1	1.6	1.87	1.99	1	1.43	1.75	1.79	1	1.47	1.8	1.76
	PSNR	41.71	0	-0.16	-0.15	39.83	0.15	-0.01	-0.01	36.57	-0.04	-0.23	-0.24
flower	Accel.	1	1.59	2.4	2.55	1	1.7	2.31	2.65	1	1.65	2.41	2.68
	PSNR	30.7	-0.09	0.01	-0.08	28.54	-0.1	-0.25	-0.3	25.56	-0.09	-0.33	-0.51
foreman	Accel.	1	1.69	2.53	2.89	1	1.57	2.31	2.52	1	1.53	2.15	2.45
	PSNR	38.55	-0.14	0.06	0.02	35.76	-0.19	-0.04	-0.11	32.56	-0.08	-0.29	-0.48
mobile	Accel.	1	1.74	2.61	2.77	1	1.63	2.34	2.5	1	1.61	2.26	2.48
	PSNR	30.81	-0.08	-0.2	-0.24	28.33	0.05	-0.26	-0.32	25.14	-0.02	-0.29	-0.49
tempete	Accel.	1	1.72	2.59	2.88	1	1.65	2.44	2.67	1	1.56	2.25	2.45
	PSNR	33.26	-0.07	0	-0.05	30.86	-0.06	-0.16	-0.31	27.95	-0.01	-0.32	-0.56
waterfall	Accel.	1	1.88	2.77	3	1	1.71	2.5	2.74	1	1.73	2.55	2.75
	PSNR	38.8	-0.07	-0.18	-0.2	35.81	0.01	-0.35	-0.39	32.73	0.14	-0.26	-0.39
Average	Accel.	1	1.66	2.39	2.6	1	1.58	2.19	2.42	1	1.53	2.13	2.32
	PSNR	36.91	-0.05	-0.06	-0.12	34.18	0	-0.12	-0.2	30.91	-0.01	-0.27	-0.43

Table 5: Acceleration and PSNR variation (dB) for H.264 BP spatial resolution reduction from CIF to QCIF compared to the cascaded method for 512 to 256 kbps, and 256 to 128 kbps. Average contains also news, silent and stefan sequences.

		512 to 256 kbps					**256 to 128 kbps**				
CIF videos		Cascade	Stats	*CM*	*CMWCR*	*CMWNR*	Cascade	Stats	*CM*	*CMWCR*	*CMWNR*
container	Accel.	1	1.18	1.29	1.43	1.56	1	1.14	1.25	1.32	1.65
	PSNR	40.41	-0.14	-0.19	-0.35	-0.66	37.47	-0.08	-0.27	-0.44	-0.8
flower	Accel.	1	1.19	1.54	1.93	2.24	1	1.19	1.52	1.83	2.18
	PSNR	26.79	-0.15	-0.2	-0.44	-0.79	24.85	-0.13	-0.24	-0.6	-1
foreman	Accel.	1	1.32	1.81	2.33	2.74	1	1.21	1.8	2.22	2.32
	PSNR	35.17	-0.3	-0.64	-1	-1.23	32.98	-0.27	-0.56	-0.97	-1.21
mobile	Accel.	1	1.17	1.68	2.1	2.28	1	1.09	1.58	1.79	2.2
	PSNR	24.2	-0.07	-0.15	-0.44	-1.06	22.81	-0.04	-0.14	-0.51	-1.22
tempete	Accel.	1	1.2	1.65	1.98	2.28	1	1.1	1.53	1.81	2.1
	PSNR	29.12	-0.09	-0.1	-0.31	-0.71	27.45	-0.02	-0.04	-0.32	-0.79
waterfall	Accel.	1	1.17	1.59	1.81	2.19	1	1.09	1.5	1.77	2.09
	PSNR	33.42	-0.06	-0.07	-0.34	-1.05	31.8	-0.06	-0.12	-0.44	-1.2
Average	Accel.	1	1.23	1.6	1.91	2.14	1	1.14	1.52	1.78	2.04
	PSNR	33.5	-0.2	-0.32	-0.57	-0.95	30.94	-0.16	-0.3	-0.62	-1.04

Table 6: Acceleration and PSNR variation (dB) for format conversion from H.264 BP to MPEG-4 VSP using CIF sequences from 512 to 256, 256 to 256 kbps, and 256 to 128 kbps. Average contains also news, silent and stefan sequences.

		512 to 256 kbps				**256 to 256 kbps**				**256 to 128 kbps**			
CIF videos		Cascade	Stats	*CM*	*CMWCR*	Cascade	Stats	*CM*	*CMWCR*	Cascade	Stats	*CM*	*CMWCR*
container	Accel.	1	1.37	1.74	1.91	1	1.37	1.64	1.95	1	1.29	1.62	1.65
	PSNR	35.81	-0.04	-0.22	-1.89	36.87	-0.07	-0.25	-1.97	33.54	-0.05	-0.36	-2.54
flower	Accel.	1	1.36	1.74	1.79	1	1.44	1.84	1.89	1	1.44	1.83	2
	PSNR	25.92	-0.12	-1.09	-1.59	27.29	-0.22	-0.71	-1.62	24.94	-0.02	-0.17	-0.51
foreman	Accel.	1	1.2	1.48	1.76	1	1.12	1.59	1.7	1	1.1	1.52	1.5
	PSNR	33.77	-0.19	-1.77	-2.29	34.95	-0.25	-1.29	-2.07	31.12	-0.14	-1.12	-1.78
mobile	Accel.	1	1.34	1.84	1.94	1	1.43	1.9	2	1	1.28	1.46	1.72
	PSNR	24.34	-0.06	-0.58	-0.84	25.35	-0.06	-0.48	-1.05	24.15	0	-0.1	-0.32
tempete	Accel.	1	1.32	1.64	1.71	1	1.27	1.62	1.65	1	1.17	1.47	1.69
	PSNR	28.46	-0.03	-0.61	-1.07	29.7	-0.07	-0.41	-1.19	26.89	-0.02	-0.32	-0.73
waterfall	Accel.	1	1.4	1.77	1.87	1	1.34	1.66	1.83	1	1.12	1.38	1.65
	PSNR	32.36	-0.11	-0.43	-1.65	33.16	-0.1	-0.28	-1.67	30.36	-0.05	-0.21	-1.54
Average	Accel.	1	1.32	1.7	1.82	1	1.3	1.66	1.79	1	1.22	1.54	1.7
	PSNR	31.37	-0.09	-0.99	-1.58	32.44	-0.14	-0.79	-1.61	29.34	-0.06	-0.53	-1.23

[17] "ISO/IEC 14496-10:2009- Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding," ISO, 2009.

[18] "ISO/IEC 14496-2:2004 - Information technology – Coding of audio-visual objects – Part 2: Visual," ISO, 2004.

[19] "H.264/AVC Reference Software JM 16.1," 2009, [Online] http://iphome.hhi.de/suehring/tml/.

[20] "ISO/IEC 14496-5:2005- Information technology – Coding of audio-visual objects – Part 5: Reference Software," ISO, 2005.