

Towards an Improved SVC-to-AVC Rewriter

Michael Sablatschan, Michael Ransburg, and Hermann Hellwagner

Multimedia Communication (MMC) Research Group, Institute of Information Technology (ITEC)

Klagenfurt University

Klagenfurt, Austria

E-mail: *firstname.lastname@itec.uni-klu.ac.at*

Abstract—The Scalable Video Coding (SVC) extension of the H.264/AVC (AVC) video coding standard features spatial, quality and temporal scalability. Backwards compatibility with legacy decoding devices is maintained through an H.264/AVC compliant base layer, which represents the lowest quality of an SVC bit-stream. However, it is often desirable to also provide the higher quality layers to legacy H.264/AVC devices. This is achieved by a process commonly known as “bit-stream rewriting”, which allows for an efficient SVC to AVC conversion by exploiting the similarities of the two codecs. This paper introduces an improved version of the existing JSVM reference software rewriter (JSVM-rewriter). The improvements include a better run-time performance through parallel processing, as well as applicability in streaming scenarios. A detailed evaluation provides performance measurements for the improved rewriter and compares it to the existing JSVM-rewriter. The evaluation shows that notable performance improvements can be achieved using the presented approach. The paper concludes on how the rewriter could be further improved.

Keywords: *SVC-to-AVC Rewriting; H.264/SVC; Transcoding; Media Adaptation*

I. INTRODUCTION

H.264/SVC [1] is a block-based hybrid scalable video codec, which introduces scalability mechanisms in three different scalability dimensions. First, it offers temporal scalability, which refers to the embedding of the video content at different temporal resolutions (frame rates). Second, spatial scalability allows incorporating multiple spatial resolutions (e.g., HD, SD) of the same video in a single bit-stream. Finally, quality scalability (or SNR scalability) refers to the fact that different quality variations of the same video can be embedded into a single bit-stream trading off visual distortion and required video bit-rate.

The scalability of the encoded video bit-stream is achieved by a layered approach. An H.264/SVC bit-stream comprises an H.264/AVC-conformant base layer which represents video at the lowest quality that can be extracted from the bit-stream. Building on top of the base layer, enhancement layers can be used to refine the video quality and/or the spatial resolution of the video. As a consequence, the adaptation of the scalable video bit-stream is as simple as truncating certain enhancement layers or parts thereof from the initial bit-stream.

Since the bit-stream syntax and coding tools of the base layer are backward-compatible to H.264/AVC, this part of

the bit-stream can be decoded by any legacy H.264/AVC device. However, as the base layer only represents the lowest quality of the H.264/SVC bit-stream, it is often desirable to also provide the higher quality representations to legacy H.264/AVC devices. This is achieved by a process named “bit-stream rewriting” [2]. It allows to efficiently transform an H.264/SVC bit-stream into an H.264/AVC bit-stream without loss by exploiting the similarities of the two codecs, i.e., without requiring complete transcoding of the bit-stream. The JSVM-rewriter, which implements this process, is available as part of the Joint Software Video Model (JSVM) [3].

This paper introduces an improved version of the JSVM-rewriter, which is used within the European project SCALNET [4]. The main improvement is the parallel rewriting of different segments of the H.264/SVC bit-stream. Past research work has been carried out on parallel encoding and decoding of bit-streams, including parallelism based on Groups of Pictures (GOPs), e.g. in [5] and [6]. In the approach presented in this paper, this GOP-based parallel approach is applied on the aforementioned bit-stream rewriting process. In the remainder of this paper, H.264/AVC is denoted as AVC, and H.264/SVC as SVC.

Section II details the bit-stream rewriting process. Section III introduces our improvements to the existing JSVM-rewriter. Section IV evaluates the run-time performance of the improved rewriter for different layer configurations of three reference video bit-streams. Finally, Section V concludes this paper and provides an outlook to future research in this area.

II. BIT-STREAM REWRITING

SVC-to-AVC bit-stream rewriting can be subsumed as the low-complexity combination of interdependent layers of a scalable multi-layer SVC bit-stream to a single-layer AVC bit-stream. SVC-to-AVC bit-stream rewriting is only defined for the case of quality scalability; both Coarse-Grain Scalability (CGS) and Medium Grain Scalability (MGS) are supported. It is not possible to convert a spatially scalable SVC bit-stream to an AVC bit-stream. In order to enable SVC-to-AVC bit-stream rewriting, some modifications to the SVC decoding process were required. The two major aspects which have been changed affect the *inter-layer intra-prediction* and the *residual prediction*.

In the *residual prediction* residual data are mapped from the reference layer to the enhancement layer by accumulating transform coefficients. In order to be compatible with the

AVC syntax though, it is necessary to perform the mapping by accumulating the transform coefficient level values, i.e., without rescaling (inverse quantizing) the transform coefficients. This required change is accomplished by scaling the transform coefficient levels by the ratio of the quantization intervals of the reference and the enhancement layers (SVC Scaling).

The *inter-layer intra-prediction* in SVC predicts intra-frame coded blocks in the enhancement layer from reconstructed pixel values of neighboring blocks within the reference layer. However, these pixel values are not available in the enhancement layer. Thus a single layer AVC decoding process cannot be applied. This problem is solved by projecting the prediction data directly to the enhancement layer instead of reconstructing the intensity values in the reference layer for prediction. This way it is possible to perform intra-frame prediction as in single-layer AVC coding.

Fig. 1 depicts the rewriting process. It can be seen that the residual data is accumulated on the transform coefficient level values. Both the reference layer and the enhancement layer are entropy decoded, but the intensity data are not fully reconstructed. The results of the entropy decoding steps are the quantized transform coefficients. In case of the reference layer these are scaled, so that the quantized residuals can be accumulated in the next step. After that the combined coefficients are rewritten to an AVC bit-stream. The rewriting comprises the entropy encoding and the writing of the prediction data into the AVC bit-stream.

Compared to full transcoding, i.e. completely decoding the SVC stream and then encoding to AVC, this rewriting process avoids the computationally expensive inverse quantization and inverse transform steps for decoding the SVC bit-stream, as well as the transform and quantization steps for encoding to AVC.

III. THE GOP-BASED PARALLEL REWRITER

The GOP-based parallel rewriter was developed in the context of the SCALNET project and thus had to fulfill requirements according to the scenarios identified in the project.

A first requirement was that the rewriter needs to accept a streamed video as input, rather than reading the video from a file on the disk. This is needed for several reasons, which correspond to the different video streaming scenarios within the SCALNET project: 1) the SVC video is encoded on the fly, as it is usual, e.g., in live streaming scenarios; 2) on-the-fly filtering of SVC enhancement layers is performed before the SVC stream is rewritten in order to, e.g., react to a changing usage environment; 3) rewriting is performed within the home network, in order to exploit the SVC scalability features in the core network, but still service AVC legacy devices.

Moreover, although the JSVM-rewriter inherently has a lower execution time compared to full transcoding, it needs to be further improved in order to be applicable in real-time scenarios of the SCALNET project with high quality affordances. Our approach to satisfy the requirement of improved performance can be described as parallelization of

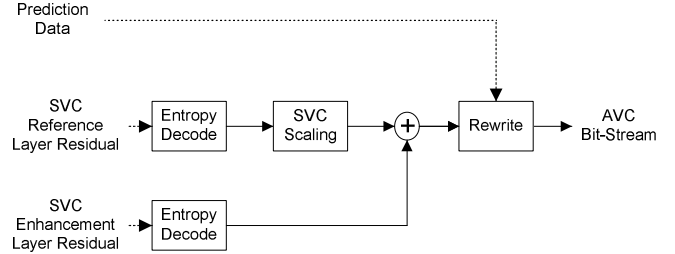


Figure 1. Block diagram of the SVC-to-AVC rewriting process.

the rewriting process based on Group Of Pictures (GOPs). A GOP is a group of successive frames within a video stream. In the context of this paper, the GOP size is considered to be the number of frames between two Instantaneous Decoding Refresh (IDR)-frames plus the IDR-frame at the beginning of the GOP. As IDR-frames can be decoded without taking reference to previous frames and each GOP starts with an IDR-frame, GOPs can be decoded independently.

We apply this GOP-concept to our “GOP-based parallel rewriter”. The idea is to rewrite several GOPs in parallel in order to improve the performance of the JSVM-rewriter. The GOP-based parallel approach also serves the requirement of the video streaming ability. The incoming SVC stream is buffered and assembled into GOPs until a configurable number of N GOPs is received. The assembled GOPs are then stored to disk and rewritten in parallel by N instances of the JSVM-rewriter before they are streamed into the network again. For an efficient exploitation of the GOP-based parallel rewriter, it is installed on a multi-core system which allocates the rewriting threads to different CPU cores. One drawback of buffering GOPs is the increased delay which depends on the number of buffered GOPs. Amongst other things this is explained in more detail in the next section in which these basic improvements to the JSVM-rewriter are evaluated.

IV. EVALUATION

In order to evaluate the performance of the parallel rewriter compared to the JSVM-rewriter (version JSVM_9_18), three different video sequences were encoded in different variations. First, each sequence was encoded with the resolutions 480x320p, 720x576p and 1280x720p, resulting in nine video sequences. The different resolutions were not encoded into a single SVC bit-stream, as it is not possible to rewrite spatially scalable SVC bit-streams. The resolutions were chosen on the basis of existing end-devices. Each of the scalable sequences contains the temporal resolutions 12.5 fps, 25 fps and 50 fps. 25 fps was chosen since it is the frame rate used in the PAL (720x576p) television standard, 50 fps is commonly used for the 1280x720p format and 12.5 fps was picked as an additional scalability option, suitable e.g. for mobile devices. A constraint in choosing the frame rate is imposed by the hierarchical prediction structure with dyadic temporal enhancement layers which only allows doubling the frame rate between different enhancement layers [1].

Moreover two MGS quality enhancement layers are included in each of the SVC bit-streams. MGS was chosen as quality scalability type, but it is noted that CGS leads to

similar results. All test sequences have a GOP size of 16. Each of the nine different SVC video sequences has zero dependency enhancement layers D (i.e., no spatial or CGS enhancement layers), two temporal enhancement layers T and two quality enhancement layers Q, denoted as DTQ 022. The Base Layer was encoded at the SVC Main profile and the enhancement layers at the Scalable High profile. It would be meaningless to rewrite the base layer and its temporal enhancement layers because it is already compatible with AVC. So the focus lies on the quality enhancement layers and their different frame rates which results in six DTQ layer combinations (001, 011, 021, 002, 012, 022) for each of the nine video sequences. On the whole, 54 video sequence variants, originating from three source video sequences, were evaluated. The three source video sequences are the Mobcal, Parkrun and Shields MPEG reference video sequences recorded by SVT Sveriges Television. The quantization parameters QP chosen for the two quality enhancement layers are 36 and 32 for the 320p and 576p encodings and 34 and 30 for the 720p encodings, resulting in the bit-rates depicted in Table I.

The three video sequences comprise contents of different complexity. Especially the Parkrun-sequence contains a large amount of details and high movement and therefore has a notably higher bit-rate compared to the other sequences. Enabling the rewriting functionality when encoding SVC leads to a 5.8% increase of the bit-rate [7] due to the changes in the SVC decoding process introduced in Section II. Rewriting the bit-streams to AVC decreased the bit-rates by 17% to 35% in our tests. This corresponds to the bit-rate overhead introduced by SVC compared to single layer streams, in our case AVC. So up to 35% of bandwidth could be saved at the cost of the scalability of SVC. This can be useful in scenarios in which the advantages of SVC are no longer needed (e.g. because the required quality can be determined at the server). However, the SVC bit-rate overhead as compared to single layer streams can be reduced to less than 10% with optimized encoder control [1].

The evaluations were carried out on an Intel(R) Core(TM)2 Extreme CPU X9650 with four 3.00 GHz cores and 3 GB memory. The JSVM-rewriter is compared to the GOP-based parallel rewriter configured with different numbers of threads (1, 2, 4, 8, and 16). The number of rewriting threads has a linear dependency to the delay introduced by the GOP-based parallel rewriter: Due to the GOP-buffering described in Section III, a higher number of threads results in higher delay. For example, with a GOP size of 16, a frame rate of 25 fps and two GOPs rewritten in parallel, the minimum delay would be $2 \cdot 16 / 25 (= 1.28)$ seconds plus the rewriting duration. Thus, the configuration of the GOP-based parallel rewriter has to be carefully chosen with respect to the delay requirements. The evaluation results are presented using stacked bar charts in Fig. 2-4. The height of the bars specifies the throughput in frames per second. On the x-axis the different tests are outlined: The JSVM-rewriter on the left hand side followed by the different configurations of the GOP-based parallel rewriter. For every test there is a group of three bars, which

TABLE I. SVC BIT-RATES IN MB/S.

	001	011	021	002	012	022
Mobcal_320p	0.545	0.603	0.672	1.111	1.196	1.297
Parkrun_320p	1.342	1.465	1.549	2.569	2.826	2.986
Shields_320p	0.689	0.750	0.822	1.383	1.471	1.573
Mobcal_576p	1.279	1.420	1.592	2.554	2.757	3.002
Parkrun_576p	3.828	4.493	4.793	7.247	8.459	9.103
Shields_576p	1.524	1.670	1.842	3.036	3.241	3.484
Mobcal_720p	3.228	3.469	3.772	6.727	7.081	7.522
Parkrun_720p	11.584	14.258	15.984	21.924	26.348	29.707
Shields_720p	3.713	3.970	4.282	7.913	8.280	8.722

correspond to the different video sequences. Each bar is subdivided into the different DTQ layer combinations from 001 to 022 in ascending order regarding the throughput. This means that the combination for which the throughput is the smallest can be found at the bottom. Fig. 2 depicts the results for the spatial resolution 480x320p.

One noticeable point when looking at the figures is the decreased performance of the GOP-based parallel rewriter in single threaded mode compared to the JSVM-rewriter. This overhead results mainly from our approach of simply storing the assembled GOPs to disk and calling the JSVM-rewriter for each GOP. It can be further observed that the throughput increases with the number of threads used, which reflects that our approach is effective. A strong increase of the throughput from 1 to 4 threads is evident, which is to be expected on the test machine with 4 CPU cores. The performance still improves for 8 and also for 16 threads in case of the spatial resolution of 480x320p. Although employing 4 threads at 4 cores, there are still CPU stalls due to system IO. These stalls are allocated by the additional threads, which leads to the slight improvements. From Fig. 3 and Fig. 4 it can be observed that the performance deteriorates clearly when using 16 threads with higher spatial resolutions. The higher resolutions imply higher memory consumption, which is the reason for the system running out of memory at high loads. This leads to swapping operations and causes the performance of the rewriter to drop rapidly.

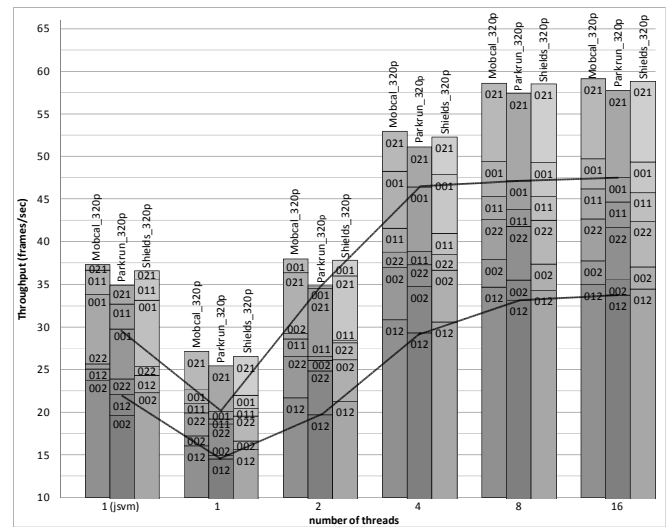


Figure 2. Rewriter evaluation for 480x320p sequences.

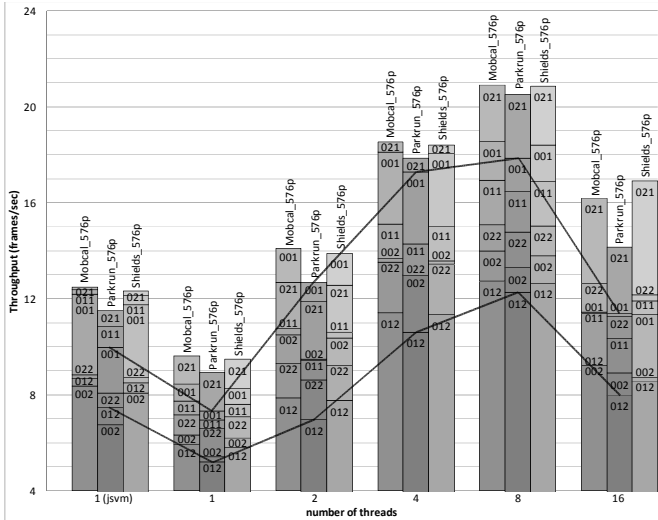


Figure 3. Rewriter evaluation for 720x576p sequences.

Moreover the figures show that there is only a small difference between the three video sequences Mobcal, Parkrun and Shields. This means that the content of video sequences has no significant impact on the performance of the rewriter.

It can be seen from Fig. 2 that all sequences at a spatial resolution of 480x320p and with a frame rate of 25 or 12.5 fps can be rewritten in real-time by the GOP-based parallel rewriter configured with 4, 8, and 16 threads. For example, the JSVM-rewriter manages to rewrite the 012 variant of the Parkrun sequence at 22.0 fps and the single threaded GOP-based parallel rewriter only at 14.5 fps. When configured to rewrite with two threads, the GOP-based parallel rewriter reaches 19.7 fps, with 4 threads 29.4 fps, 33.1 fps with 8 threads, and 33.7 fps with 16 threads. The lower curve in Fig. 2 outlines the results for the 012 Parkrun sequence. The curves for all the other video sequences and layer combinations progress similarly with different offsets. The upper curve in Fig. 2 for example shows the behavior for variant 001. The variants with only one quality enhancement layer show a better throughput compared to the variants with two quality enhancement layers in all tests.

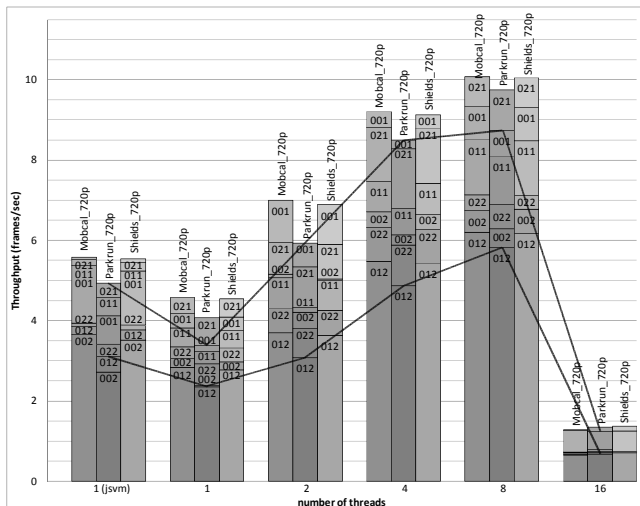


Figure 4. Rewriter evaluation for 1280x720p sequences.

Fig. 3 depicts the results for the video sequences encoded with a spatial resolution of 720x576p. Here, for the layer combination 012 of the Parkrun sequence only 7.5 (JSVM), 5.2 (1 thread), 7.0 (2), 10.6 (4), 12.3 (8) and 8.0 (16) fps are achieved. These results indicate that it is not possible to rewrite PAL video sequences with 25 fps in real-time on the adopted platform. Only by dropping a temporal and a quality enhancement layer (DTQ 001, i.e., frame rate 12.5 fps), real-time rewriting can be accomplished with a throughput of 12.7 fps using 2 rewriting threads, 17.3 fps with 4 threads, and 17.9 fps with 8 threads (the upper curve in Fig. 3).

Fig. 4 shows that real-time rewriting of 1280x720p SVC video sequences is not possible with any configuration of the GOP-based parallel rewriter. For the 001 variant the throughput of 8.7 fps with 8 threads is the maximum.

V. CONCLUSION AND FUTURE WORK

In this paper an improved version of the existing JSVM-rewriter was introduced. The improvements can be subsumed as a parallelization of the rewriting process based on GOPs. The evaluation showed that despite the overhead introduced by the GOP-based parallel rewriter, notable performance improvements are accomplished, especially for the configurations with 4 and 8 rewriting threads. With our test setup, real-time rewriting with the GOP-based parallel rewriter is possible for resolutions of 480x320p at 25 fps and 720x576p at 12.5 fps. Thus, further enhancements of the presented approach are still necessary in order to support higher resolutions. A first improvement of the GOP-based parallel rewriter would be a better integration of the JSVM-rewriter. This way the overhead of invoking the command processor to execute the JSVM-rewriter for each GOP could be avoided and instead of storing each GOP to disk, rewriting could take place in memory. A next step would be a profiling of the existing JSVM-rewriter in order to find out where code optimizations would be reasonable. Profiling would also give the information if the integration of existing improved SVC decoders could bring some performance gain.

VI. ACKNOWLEDGEMENTS

This work is supported by the Österreichische Forschungsförderungsgesellschaft mbH (FFG) in the context of the Celtic SCALNET (CP5-022) project.

REFERENCES

- [1] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard", IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, no. 9, 1103–1120, Sept. 2007.
- [2] A. Segall, "CE 8: SVC-to-AVC Bit-Stream Rewriting for Coarse Grain Scalability," Joint Video Team, Doc. JVT-V035, Jan. 2007.
- [3] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, "Joint Scalable Video Model", Doc. JVT-X202, 2007.
- [4] SCALNET project, <http://www.scalnet.info>, retrieved Nov. 2009.
- [5] A. Rodriguez, A. Gonzalez, M.P. Malumbres, "Hierarchical parallelization of an h.264/avc video encoder", Proc. Int. Symp. on Parallel Computing in Electrical Engineering, 363-368, Sept. 2006.
- [6] A. Bilas, J. Fritts, J.P. Singh, Real-Time Parallel MPEG-2 Decoding in Software, Proc. Int. Symp. on Parallel Processing, 197-203, Apr. 1997.
- [7] A. Segall and J. Zhao, "Bit-Stream Rewriting for SVC-to-AVC Conversion," 15th IEEE Int. Conf. on Image Processing, Oct. 2008.