Efficient SVC-to-AVC Conversion at a Media Aware Network Element

(Demo Paper)

Michael Sablatschan[1], Jordi Ortíz Murillo[2], Michael Ransburg[1], and Hermann Hellwagner[1]

[1] Multimedia Communication (MMC) Research Group, Institute of Information Technology (ITEC), Klagenfurt University, Klagenfurt, Austria
{firstname.lastname}@itec.uni-klu.ac.at
[2] Intelligent Systems Group, Department of Computer Science and Artifcial Intelligence, University of Murcia, Murcia, Spain
jordi.ortiz@um.es

Abstract. H.264/SVC, the Scalable Video Coding extension of the H.264/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 H.264/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 H.264/SVC to H.264/AVC conversion by exploiting the similarities of the two codecs. This paper describes a demonstrator showing the advantages of including an improved version of the bit-stream rewriting tool from the existing JSVM H.264/SVC reference software in an H.264/SVC-based multimedia delivery system, by integrating it into a Media Aware Network Element.

Keywords: Multimedia Adaptation, H.264/SVC, SVC-to-AVC rewriting.

1 Introduction

H.264/SVC (SVC) [1] is a block-based hybrid scalable video codec, featuring three different scalability dimensions: temporal scalability, spatial scalability and quality scalability (SNR scalability). These three dimensions allow the incorporation of multiple frame rates, spatial resolutions and different quality variations in a single bit-stream. The scalability of the encoded video bit-stream is achieved by a layered approach. An H.264/SVC bit-stream comprises a base layer which is conformant to H.264/AVC (AVC), representing the video at the lowest quality which can be extracted from the bit-stream. Building on top of the base layer, enhancement layers can be used to refine the video quality, the spatial resolution or the temporal 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.

SVC is an emerging technology, and in case it will further succeed on the market, an efficient mechanism for supporting backward compatibility will be needed. Since the bit-stream syntax and coding tools of the base layer are already backward-compatible to AVC, this part of the bit-stream can be decoded by any legacy AVC device. However, as the base layer only represents the lowest quality of the SVC bit-stream, it is often desirable to provide the higher quality representations to legacy AVC devices. This is achieved by a process named "bit-stream rewriting" [2][3]. It allows to efficiently transform an SVC bit-stream into an AVC bit-stream without loss by exploiting the similarities of the two codecs, i.e., without completely decoding the SVC bit-stream and re-encoding it to AVC. 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. The rewriting process is only defined for the case of quality scalability. The JSVM-rewriter, which implements this process, is available as part of the Joint Software Video Model (JSVM) [4].

The requirement from the European project SCALNET [5] for efficient bit-stream conversion from the scalable SVC video format to the more widespread AVC video format lead to the implementation of an improved version of the JSVM-Rewriter. The improvements comprise the applicability in network environments and the parallel rewriting of different Group of Pictures (GOPs) of the H.264/SVC bit-stream, leading to a better run-time performance. A more detailed description of the improved SVC-to-AVC rewriter can be found in [6], and in Section 4 of this paper, where the evaluation results of the improved rewriter are summarized. The focus of this paper is the description of a demonstrator which was built to show the advantages of including the improved SVC-to-AVC rewriter in an SVC-based multimedia delivery system by integrating it into a Media Aware Network Element (MANE).

Section 2 explains the advantages of using the SVC-to-AVC rewriter and possible application scenarios. After that, in Section 3 the architecture of the demonstrator is described, followed by a summary of the evaluation results for the improved rewriter in Section 4 and finally a conclusion of the demo paper Section 5.

2 Application Scenarios / Advantages

Despite the advantages of SVC, which cover the possibility to include multiple quality variations of a video within one single bit-stream and a very efficient bit-stream adaptation, there are some drawbacks compared to AVC as well. First, the market is largely comprised of AVC legacy devices which do not support SVC. Second, despite the improved compression efficiency of SVC, a multi-layer SVC bit-stream introduces a bit-rate overhead compared to a single layer AVC bit-stream of the same quality because of the additional layer dependency information. These two drawbacks raise the need for an SVC-to-AVC conversion tool like the SVC-to-AVC rewriter.

A first scenario derived from the drawbacks of SVC would be the application in a heterogeneous network which serves different AVC legacy devices. These legacy devices are only able to decode the base layer of the SVC bit-stream, but might want to receive a higher quality if the client's resources allow for it. In this scenario the SVC bit-stream could be rewritten within the network, e.g. at a MANE, in order to fit the client's needs and capabilities.

Another possible application would be in a system where the final link to the client has a limited bandwidth. In this scenario the bit-rate of the transmitted bit-stream needs to be reduced before the final link is reached. One option would be to discard SVC enhancement layers. However, this leads to a decreased quality of the bit-stream. Another possibility would be to rewrite the SVC bit-stream to AVC, reducing the bit rate by avoiding the bit rate overhead of SVC compared to AVC, and, as rewriting is a lossless conversion, keeping the same quality at the same time.

The demo presented in the next section covers both scenarios described above.

3 Demo Architecture and Application Flow

3.1 Demo Components

The architecture of the demo presented in this paper comprises a RTSP streaming server, a MANE containing the SVC-to-AVC rewriter, and different AVC legacy devices.

The RTSP streaming server is a modification of the open source Darwin Streaming Server. The modification allows the streaming of SVC videos. In general, a MANE is located on a network node between the RTSP streaming server and the RTSP Streaming Clients on the AVC legacy devices. The concept of a MANE is introduced in [7], as a "network element, such as a middlebox or application layer gateway that is capable of parsing certain aspects of the RTP payload headers or the RTP payload and reacting to the contents". The MANE used in this demo was implemented in the SCALNET project as an RTSP proxy and provides content adaptation services such as SVC-to-AVC rewriting.

3.2 Demo Scope

This setup accounts for all the advantages of SVC-to-AVC rewriting described in the previous section: The streaming server stores an SVC file containing several quality variations of the same video, instead of multiple AVC files for each quality variation. The conversion to AVC might take place at any point in the network from where the SVC functionalities are no longer needed or desired, e.g. at a home gateway which only serves AVC legacy devices. Before this point in the network is reached, the SVC-feature of simple bit-stream adaptation is still available. After this point, the adaptation of the bit-stream in terms of changing the quality is not as simple as with SVC, but in return you have a decreased bit-rate at the same quality. As already mentioned above, the conversion from SVC to AVC takes place at a MANE between the SVC streaming server and the AVC legacy devices. This way the streaming server is able to store the videos in the SVC format, thus saving disk space, and serve AVC legacy devices with different requirements at the same time, and that with a lower bandwidth consumption compared to SVC. Fig. 1 depicts an overview of the demo architecture.

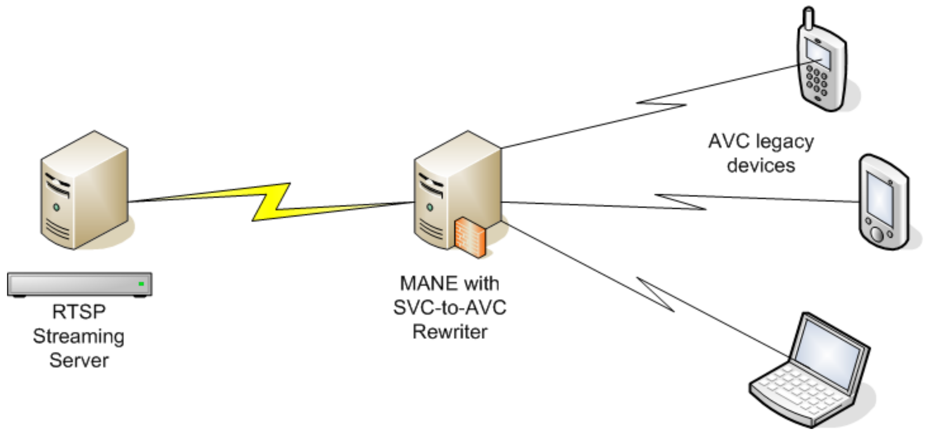

Fig. 1. SVC-to-AVC Rewriter demo architecture

3.3 Further Considerations

As the MANE should be able to serve clients which are capable of playing SVC bit-streams as well, a mechanism to signal the clients' decoding capabilities is required. This signaling takes place during the streaming configuration exchange with RTSP [8]. If a client signals SVC decoding capabilities, the MANE has to disable the SVC-to-AVC rewriting functionality and act as a packet forwarder. Else, if the client is an AVC legacy device, not only the bit-stream has to be rewritten. Additionally all RTP packets have to be de-packetized before, and packetized again after rewriting according to the corresponding RTP payload formats [7][9][10].

3.4 Application Flow

The application flow of the demo is as follows: The AVC client requests a video from the MANE, which forwards this request to the server. The server transmits the SVC video to the client via the MANE, which contains the SVC-to-AVC rewriter. In order to see the differences of using the SVC-to-AVC rewriter or not, two sessions are shown: one session with the SVC-to-AVC rewriter enabled, and one with the SVC-to-AVC rewriter disabled. In the first session, i.e. with the bit-stream rewritten to AVC, the client receives the best quality, while in the second session the client can only obtain the base layer. At the same time, a bandwidth monitor displays the current bandwidth utilization, which demonstrates the bandwidth-saving for the case the SVC-to-AVC rewriter is used.

4 Evaluation of the Improved SVC-to-AVC Rewriter

4.1 Test Setup

The evaluations of the improved rewriter were carried out on an Intel(R) Core(TM)2 Extreme CPU X9650 with four 3.00 GHz cores and 3 GB memory. For the evaluation three different MPEG reference video sequences (Mobcal, Parkrun and Shields) were encoded in different variations. Each sequence was encoded with the spatial resolutions 480x320p, 720x576p and 1280x720p, the temporal resolutions 12.5 fps, 25 fps and 50 fps and three PSNR quality variations. This results in nine different SVC video sequences with 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 different enhancement layer combinations ware extracted from the nine SVC video sequences for the evaluation, i.e. six DTQ layer combinations (001, 011, 021, 002, 012, 022) for each of the nine video sequence have been evaluated. The 54 resulting video sequences were used to compare the JSVM-rewriter (version JSVM_9_18) to the GOP-based parallel rewriter configured with different numbers of threads (1, 2, 4, 8, and 16).

4.2 Delay and Bitrate

A higher number of threads, each rewriting a GOP, results in higher delay. For example, with a GOP size of 16 (as chosen for the test sequences), a frame rate of 25 fps and two GOPs rewritten in parallel, the minimum delay would be 2*16/25 (= 1.28) seconds plus the rewriting duration.

Enabling the rewriting functionality when encoding SVC leads to a 5.8% increase of the bit-rate due to the changes in the SVC decoding process [3]. Rewriting the bit-streams to AVC decreased the bit-rates by 17% to 35% with our SVC encoder settings. 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 scalability. This can be useful in scenarios in which the advantages of SVC are no longer needed, as described in Section 2. The SVC bit-rate overhead (rewriting not enabled) can be reduced to less than 10% with optimized encoder control [1].

4.3 Performance Evaluation

The evaluation results for the spatial resolution 720x576p are presented using a stacked bar chart in Fig. 2. The height of the bars specifies the throughput in frames per second. On the x-axis one can see the different test cases: The JSVM-rewriter on the left, followed by the different configurations of the GOP-based parallel rewriter. For every test case there is a group of three bars, which 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.

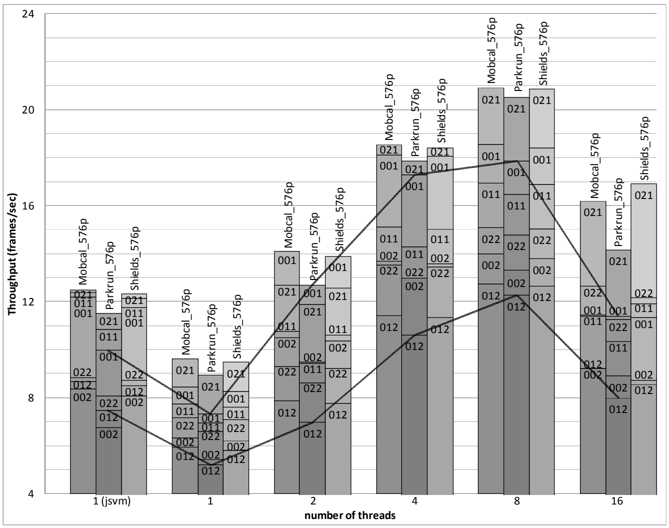

Fig. 2. Improved Rewriter Evaluation for 720x576p Video Sequences

The GOP-based parallel rewriter stores the GOPs which are assembled from the received NAL-Units to disk before it makes a system call to the JSVM-rewriter for each GOP in order to perform the actual rewriting. This approach is not optimized and leads to a decreased performance when comparing the GOP-based parallel rewriter in single threaded mode and the JSVM-rewriter in Fig. 2. The throughput achieved with the GOP-based parallel rewriter increases with the number of threads used, which reflects that our approach is effective. The performance still improves for 8 threads in case of the spatial resolution of 720x576p. 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. Rewriting with a higher number of threads also leads to a higher memory consumption, which can cause the system to run out of memory at high loads, which explains the bad performance of the rewriter with 16 threads. The content of video sequences has no significant impact on the performance of the rewriter, which is expressed through the small differences between the different video sequences in Fig. 2.

For live scenarios it is of interest which qualities can be rewritten in real-time. The upper curve in Fig. 2 outlines the results for the 001 Parkrun sequence. The curves for all the other video sequences and layer combinations progress similarly with different offsets. The lower curve in Fig. 2 for example shows the behavior for variant 012. It can be seen that 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 throughput 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. 2).

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. 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. For further details the reader is referred to [6].

5 Conclusion

This demo paper described the advantages and possible application scenarios of the SVC-to-AVC rewriter tool. The demo setup consists of an RTSP streaming server, a MANE containing the rewriter, and different AVC legacy devices. It comprises all the advantages described in Section 2, namely the saved disk space because of storage in the SVC format including multiple qualities within a single video stream on the streaming server, the support of AVC legacy devices and the saved bandwidth after rewriting to AVC. The evaluation results show that real-time rewriting is possible for 480x320p video sequences at 25 frames per second and for 720x576p at 12.5 frames per second.

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

References

1. Schwarz, H., Marpe, D., Wiegand, T.: Overview of the scalable video coding extension of the H.264/AVC standard. IEEE Transactions on Circuits and Systems for Video Technology 17(9), 1103–1120 (2007)
2. Segall, A.: CE 8: SVC-to-AVC Bit-Stream Rewriting for Coarse Grain Scalability. Joint Video Team, Doc. JVT-V035 (January 2007)
3. Segall, A., Zhao, J.: Bit-Stream Rewriting for SVC-to-AVC Conversion. In: 15th IEEE Int. Conf. on Image Processing (October 2008)
4. Reichel, J., Schwarz, H., Wien, M.: Joint Scalable Video Model 11 (JSVM 11). Joint Video Team (JVT), Doc. JVT-X202 (July 2007)
5. SCALNET project, `http://www.scalnet.info` (retrieved 2009)
6. Sablatschan, M., Ransburg, M., Hellwagner, H.: Towards an Improved SVC-to-AVC Rewriter. In: Proceedings of MMEDIA 2010 (June 2010)
7. Wenger, S., Stockhammer, T., Hannuksela, M.M., Westerlund, M., Singer, D.: RTP payload format for H.264 video. RFC3984, Internet Engineering Task Force (IETF) (February 2005)
8. Schulzrinne, H., Rao, A., Lanphier, R.: Real time streaming protocol (RTSP). RFC2326, Internet Engineering Task Force (IETF) (April 1998)
9. Wang, Y.-K., Even, R., Kristensen, T., Jesup, R.: RTP Payload Format for H.264 Video, draft-ietf-avt-rtp-rfc3984bis-10.txt (April 2010)
10. Wenger, S., Wang, Y.-K., Schierl, T., Eleftheriadis, A.: RTP Payload Format for SVC video, draft-ietf-avt-rtp-svc-21.txt (April 2010)