Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

A Study on Keyframe Injection in Three Generations of Video Coding Standards for Fast Channel Switching and Packet-Loss Repair

Hannes Mareen* · **Martijn Courteaux** · **Pieter-Jan Speelmans** · **Peter Lambert** · **Glenn Van Wallendael**

Received: date / Accepted: date

Abstract It is challenging to enable fast channel switching and packet-loss repair in low-delay live video distribution without negatively influencing the steady-state viewing performance. For example, regularly breaking the inter-frame dependency by introducing intra-predicted keyframes enables random access, but is costly in terms of rate-distortion performance. For this reason, the keyframe-injection method minimizes the impact by sending a compression-efficient normal video stream to all end-users. As accompaniment, a companion stream that solely consists of keyframes is sporadically used for only those users that switch channels or experience packet loss. This paper describes the requirements to implement keyframe injection in three video coding standard generations (H.264/AVC, H.265/HEVC, and H.266/VVC). We evaluated the impact that keyframe injection has on the quality of the video in terms of a decrease in VMAF, PSNR and SSIM. We demonstrate that the quality reduction caused by keyframe insertion is generally low, meaning that keyframe injection typically is imperceptible. However, drift-error artifacts become perceptible over time for rare outliers. Moreover, we pinpointed the cause of this worst-case artifact type to be halfpel interpolation. As a solution, codecs can disable subpel motion estimation, and future standards could design their filters more carefully. Lastly, it should be noted that keyframe injection will only be applied sporadically when users require a random access or experience packet loss, and only to those users. Most interestingly, all other users receive a compression-efficient stream wherein the inter-frame dependency is not artificially broken at regular short intervals. As such, our proposed solution makes low-latency video distribution efficient and viable in multiple coding standards.

Keywords Fast Channel Switching · Random Access · Packet Loss · Error Recovery · H.264/AVC · H.265/HEVC · H.266/VVC

1 Introduction

Low-latency video distribution serving a variety of end-users with different connectivity characteristics [33] and random-access behavior (i.e., channel switching/zapping) [13, 34] is challenging. It is important to minimize the negative impact of end-users performing channel switching or having low-fidelity connections on general usage. Additionally, it is desirable to keep processing and the number of different versions of the same video stream to a minimum to not pollute the cache behavior of the Content Distribution Network (CDN).

The overall negative impact of packet-loss and random-access restrictions becomes clear when considering the prediction structure of a video stream. On one hand, it is desirable to enable numerous inter-predictions to increase compression efficiency, yet, on the other hand, packet loss and random-access restrictions force the regular introduction of inefficient intra-predicted keyframes or other forms of intra refresh. For

This work was funded by the Research Foundation – Flanders (FWO) under Grant 1S55218N, by IDLab (Ghent University – imec), by Flanders Innovation & Entrepreneurship (VLAIO) project LIVE-G (Low-latency video over five-G - HBC.2020.2375), and by the European Union.

H. Mareen*, M. Courteaux, P. Lambert, G. Van Wallendael
Ghent University - imec, IDLab, Department of Electronics and Information Systems, 9000 Ghent, Belgium
E-mail: firstname.lastname@ugent.be

P. Speelmans
THEO Technologies, 3001 Leuven, Belgium
E-mail: pieter-jan.speelmans@theoplayer.com

example, live services may require frequent switching points of 1 second or less [35]. To not burden steady-state viewing end-users having high-fidelity connections, solutions such as the High Efficiency Streaming Protocol (HESP) utilize keyframe injection (also called keyframe insertion in previous work) [36, 39].

To facilitate keyframe injection, a normal stream (NS) is accompanied by a companion stream (CS). The normal stream has minimal facilities for random access and packet loss and mainly focuses on compression efficiency (see Figure 1). Note that the normal stream still contains intra-predicted keyframes in its first frame and when content-specific changes such as scene changes occur. The companion stream, which solely consists of keyframes, remains on the server (or CDN) and interactively solves random access and packet loss whenever the client device requests a single frame from this stream. This is different from well-known intra-refresh techniques that operate directly in the NS, as every client receives a personalized stream multiplexed from the NS and CS. It should be stressed that, in the end, the client receives a single standard-compliant video stream that consists of Network Abstraction Layer Units (NALUs) from the NS and CS. Note that the NS is encoded from the source (NS_{src}), and the CS is generated by encoding the reconstructed/decoded normal stream (CS_{NS}) to reduce the errors introduced by keyframe injection. The frequency of the keyframes in the CS corresponds to the trade-off between random-access latency, packet loss recovery latency, and bitrate overhead.

Figure 1 visualizes the keyframe-injection procedure followed whenever a packet-loss or random-access event occurs. If packets are lost or a channel change occurs, the server or CDN is requested for a CS frame by the end-user client. Because the CS is synchronized with the NS, the first available CS keyframe time is transmitted to the end-user device. Subsequently, after the possibly buffered and packet-loss corrupted frames present at the client device have been played back, the keyframe from the CS will replace the collocated inter-frame. With this action, random access is enabled, and packet-loss errors can be greatly reduced.

Related works introduced keyframe injection in MPEG-2 [19], in H.263 [12], briefly in H.264/AVC [6], and in H.265/HEVC [39]. Additionally, our previously published 1-page abstract briefly discussed the viability of keyframe injection in H.264/AVC, H.265/HEVC, and H.266/VVC [27]. As this paper's novelties, this is the first work:

- To provide thorough technical descriptions and motivations on how to perform keyframe injection on three generations of coding standards, namely H.264/AVC, H.265/HEVC, and H.266/VVC.
- To quantify the bitrate and quality impact (using VMAF, SSIM, and PSNR) on five codecs.
- To identify the worst-case zebra-pattern artifacts and rigorously analyze the cause of their rare appearance. Additionally, we present a workaround and propose a criterion to consider when developing new compression standards.

After summarizing existing techniques competing with or complementing the keyframe-injection technique in Section 2, the necessary modifications to allow keyframe injection in different video compression generations (H.264/AVC, H.265/HEVC, and H.266/VVC) are discussed in Section 3. Section 4 provides a thorough evaluation of the impact on the bitrate and the quality, complemented with a worst-case analysis. Finally, concluding remarks are provided in Section 5.

2 State-Of-The-Art

Three types of solutions can be identified when tackling random access and packet loss: client based, network based, and content based [39]. Client-based techniques such as prefetching [25, 46], playback modification [1, 21], and error concealment [23] do not require modifications of content or network. Network-based techniques encompass in-network caching [2,17], buffering [24], time-shifted replicas [3], transcoding [10, 31], and network-distributed video coding [9, 32].

Techniques that generate additional video streams to accommodate channel switching and packet-loss behavior are classified as content-based techniques. H.264/AVC allowed the concept of Switching Intra (SI) frames and Switching Predicted (SP) frames to solve random-access and packet-loss issues [22]. As a downside, modifying P-frames to become SP-frames introduces an overhead in the normal video stream penalizing steady-state viewing and high-fidelity connections.

In what follows, the different techniques prioritize fast visual feedback on a channel-change request over immediate full-quality display. One of these examples is keyframe injection. Keyframe injection has first been introduced by Farber *et al.* using the H.263 standard [12]. Additionally, an extension to keyframe injection in H.263 has been proposed that introduces a third stream with so-called S-frames [11]. This stream encodes the mismatch that is introduced due to keyframe injection, solving the error propagation at the cost of a bitrate overhead. Later, Boyce and

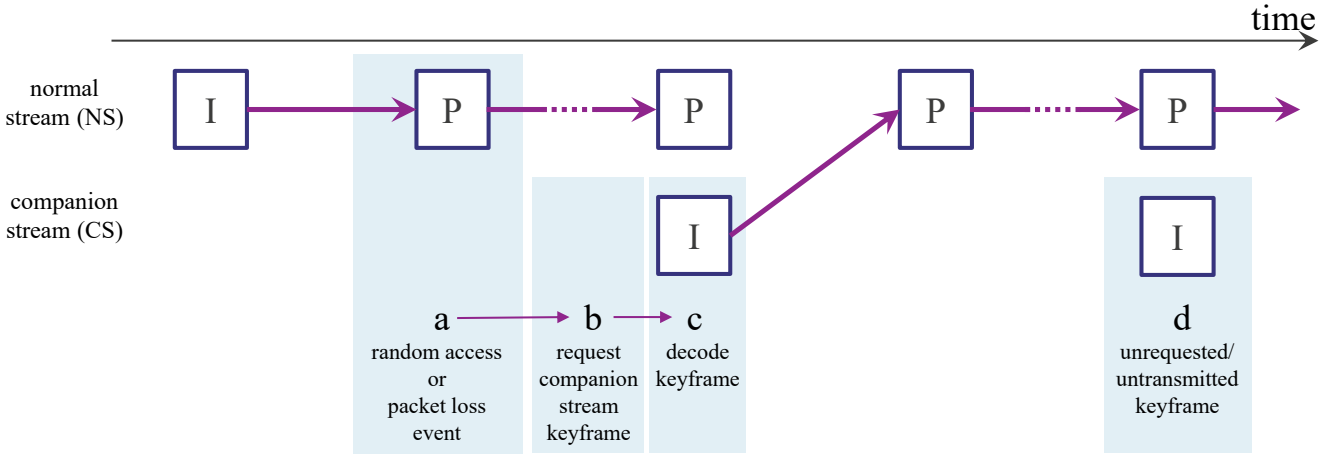


Fig. 1 Only after a (a) random-access or packet-loss event, (b) a companion stream keyframe is requested and (c) it replaces the collocated predicted frame. Note that (d) succeeding unrequested keyframes remain untransmitted, and that the client receives a continuous standard-compliant stream of NALUs.

Tourapis presented the keyframe-injection concept for H.264/AVC [6], and Jennehag and Pettersson further analysed of this method [19]. Nowadays, keyframe injection is used in the HESP [36]. In previous works, the NS [6, 20, 27, 39] has also been named the P-stream [12], main stream [19]. The CS [27, 39, 41] is known as the I-stream [12], synchronization stream [19], the channel change stream [6], or the fast channel change stream [20].

The NS and CS entail the same visual content; therefore these streams can be compressed in a scalable way. Traditionally, the low-quality CS is considered the base layer enhanced by the NS [26]. Such techniques use both the base and enhancement layers during steady-state viewing, penalizing such steady-state viewing and high-fidelity connections with overhead caused by scalable coding. This problem can be solved by flipping this arrangement and using single-loop scalable coding [40,41]. Using single-loop scalable decoding, the NS can function as the base layer, extended with CS keyframes. This enables efficient steady-state viewing of a single base layer and only penalizing channel-switching devices or low-fidelity connections.

3 Keyframe Injection in H.264/AVC, H.265/HEVC & H.266/VVC

To perform keyframe injection and provide clients with a standard-compliant bitstream in the three coding standard generations (H.264/AVC, H.265/HEVC, and H.266/VVC), the following restrictions apply:

Frame number / Picture Order Count (POC): In H.264/AVC, keyframes are required to have a frame number equal to zero. It is essential to synchronize the CS keyframes with the NS frames having frame number zero. Therefore, the Sequence Parameter Set (SPS) parameter called log2_max_frame_num_minus4 decides the periodicity with which keyframes can be included in the CS. For H.265/HEVC, similar restrictions apply if an Instantaneous Decoder Refresh (IDR) frame is used as keyframe. In that case, the SPS-parameter log2_max_pic_order_cnt_lsb_minus4 decides the periodicity of the keyframes. In practice, this restriction can be relaxed for decoders that perform error concealment and are able to deduce the right reference frame numbering.

For H.265/HEVC using Clean Random Access (CRA) keyframes, and for H.266/VVC in general, it is possible to keep the POC of the CS keyframes identical to the POC of the replaced frame in the NS. As such, there is no limitation on how many keyframes can be incorporated in the CS and at what position in time these are present. The synchronization of keeping these POCs identical is preferably performed during the encoding process of the streams.

Parameter Sets: With respect to parameter sets, there are two options for performing keyframe injection, namely identical parameter sets or parameter set injection. When the Video Parameter Set (VPS, used in H.265/HEVC and H.266/VVC only), Sequence Parameter Set (SPS) and Picture Parameter Set (PPS) are identical in both the NS and CS, then the keyframe can replace the non-keyframe of the normal stream. If there is a difference between the sets of both streams, then the injected keyframe should be preceded with the parameter sets of the CS and succeeded again with the parameter sets of the NS.

Additionally, for H.266/VVC, there is the concept of Adaptation Parameter Sets (APS) [8]. APS packets communicate three different parameter sets (aps_params_type: first three bits in APS), namely parameters of the Adaptive Loop Filter (ALF), luma mapping with chroma scaling (LMCS) parameters, and scaling list parameters. For each of these APS types, it is crucial to keep and update a list of the APS packets having unique identification (ID) numbers (aps_adaptation_parameter_set_id: next five bits in APS). As such, the first eight bits of the APS identify the unique APS packets that need to be reintroduced in the stream. The most straightforward solution is to keep and update a dictionary with that eight-bit key during the processing (i.e., decoding) of the NS. Then, after keyframe injection, the APS buffer in the decoder is flushed and the APS dictionary needs to be signaled in the video stream again. This is necessary because predicted frames succeeding the injected keyframe could make use of preceding APS information. Uniqueness of the APS IDs and thus removal of earlier duplicates is crucial to guarantee standard compliance.

Temporal Motion Vector Prediction (TMVP): TMVP enables using motion vectors (MVs) from reference frames, additional to the pixels of the frames themselves [7, 37]. In this way, future frames use MVs from reference frames as MV prediction. Because keyframes lack motion information, frames succeeding the injected companion keyframe will not have such motion information present, resulting in explicit and annoying blocking artifacts. For this reason and similar in any packet loss use case, TMVP must also be disabled for keyframe injection to work well. So in general, it is advised to disable TMVP whenever packet loss is present during transmission because of the artifacts it can produce. To give an idea about the compression performance of this tool, in this paper, enabling TMVP would lead to a Bjontegaard-Delta bitrate (BD-Rate) decrease of -2.57% (Std: 0.86%, Min: -3.92%, Max: -0.22%) of the NS in H.265/HEVC.

4 Experimental Results & Discussion

4.1 Experimental Setup

We tested keyframe injection in three generations of video coding standards. In short, we compressed 22 video sequences with 5 codecs (of 3 coding standards). For each video, we created an NS and CS at 4 quality levels. Then, in each NS, we performed keyframe injection to simulate a random-access or packet-loss event. Most interestingly, we analyzed the influence on the bitrate and quality of inserting a keyframe from the CS into the NS. The rest of this subsection describes each element of the experimental setup in more detail.

The experiments were performed on 22 sequences with resolutions between 416×240 and 2560×1600, namely *BlowingBubbles, BasketballPass, BQSquare, RaceHorses, BasketballDrill, BasketballDrillText, PartyScene, RaceHorses, BQMall, Johnny, FourPeople, KristenAndSara, SlideEditing, SlideShow, ChinaSpeed, BasketballDrive, BQTerrace, Cactus, Kimono, ParkScene, ParkJoy, Traffic,* and *PeopleOnStreet* [5]. These sequences contain between 150 and 600 frames and have a frame rate between 20 and 60 frames per second (fps). Most sequences are 10 seconds (s) long, except for *SlideShow* (25 s), ChinaSpeed (16.7 s), Traffic (5s) and PeopleOnStreet (5s).

To compress these sequences and thoroughly test the practical applicability of keyframe injection, we used a variety of codecs over three generations. Namely, we used the *x264*-encoder [42] and the Joint test Model (JM) [14] v19.0 for H.264/AVC, the *x265*-encoder [43] and the HEVC reference Model (HM) [15] version 16.15 for H.265/HEVC, and the VVC Test Model (VTM) [16] version 10.2 for the H.266/VVC standard.

The NS videos were encoded using a low-delay configuration in which the first frame is a keyframe and all other frames are predicted frames that each take only the preceding frame as reference (IPPP). This configuration is chosen since this paper focuses on low-delay real-time video. For the reader's information, we provide some extra discussion on the hypothetical influence of the coding structure. In a low-delay IPPP configuration, errors introduced by keyframe injection are able to drift until another keyframe (I-frame) is encoded in the NS. In contrast, error drifting in higher-delay coding structures depends on which frame in the structure introduced the errors. Such higher-delay configurations typically consist of a Group of Pictures (GOP) with a hierarchy (e.g. IBPBP or IBBP). In such cases, errors of B-frames deeper in the hierarchy only have short-term drift capabilities, since they will only be referenced by other frames within that GOP. In contrast, P-frames at the bottom of the core of the GOP structure are not only used as reference within that GOP, but also by neighboring GOPs, and hence have long-term drift capabilities. Future work may evaluate the practical impact of using such coding structures.

In the *x264* and *x265*-encoder, the slow preset was used. The CS was encoded with the same configuration as the NS, albeit only encoding keyframes that do not depend on other frames. Each sequence is com-

pressed using four different Quantization Parameters (QPs), namely 22, 27, 32, and 37, which is further denoted as $\mathrm{QP_{NS}}$ for $\mathrm{NS_{src}}$, and $\mathrm{QP_{CS}}$ for $\mathrm{CS_{NS}}$.

In our experiments, the keyframe of the CS is injected in the NS at frame $f = 16$, which is the 17^{th} frame. We further denote this combined stream as Keyframe-Injected stream (KI). Hence, the decoding can start from the injected keyframe as if a random access was performed, or as if *any* frame packet before $f = 16$ had been lost. The exact frame number f is an arbitrary decision, yet by setting log2_max_frame_num_minus4 to 0 in H.264/AVC, frame $f = 16$ is the first occurrence that has a frame number equal to zero in the NS. This is convenient because it enables the usage of strict H.264/AVC decoders, as explained in Section 3. We also experimented with other frame injection numbers than $f = 16$ and observed similar results. In fact, keyframe injection can be performed at any frame position, which is the main advantage of our proposed solution: we can recover from a packet loss, or allow a random access, at *any* frame position, enabling ultra-low-delay live streaming capabilities. Note that the keyframe at $f = 16$ is the only injected frame - all other frames are left unchanged. We would also like to stress that the KI is standard compliant as long as the restrictions given in Section 3 are considered.

In summary, the video sequences were transformed into NSs and CSs of various QPs, and keyframe injection was performed using each combination of NS and CS. The frame sizes were logged, and quality values were measured using the Video Multimethod Assessment Fusion (VMAF) open-source software [29] (more information in Section 4.3). Finally, the obtained csv-files from these experiments were analyzed and visualized using python scripts.

The source code of keyframe injection has been made available for reproducibility[1].

4.2 Impact on Frame Size & Bitrate

In Table 1, the factor of the frame size increase is shown, i.e., the factor with which the size of the injected keyframe is larger than the inter-frame of NS that it replaces. These are the median values calculated over all test sequences, to reduce the influence of outliers. When $\mathrm{QP_{NS}} \neq \mathrm{QP_{CS}}$, the results are shown in gray rather than black, as to simplify its interpretation. The frame size increase values help us understand the bitrate spike that occurs when performing keyframe injection.

[1] Code available on https://github.com/IDLabMedia/NALUProcessing

The keyframe size at equal QPs ($QP_{NS} = QP_{CS}$) is between 2.44 and 20.30 times larger than the corresponding inter-frames in the NS. This can be decreased by injecting a keyframe with a higher QP_{CS}, or further increased by using a lower QP_{CS}. These results give a good indication about the bitrate burst when a CS keyframe is additionally requested over the network. Future work should look into novel strategies to reduce the frame size increase of inserting a CS frame in the NS. For example, instead of inserting a keyframe, a P-frame with a reference further in the past could be inserted instead (which would only allow packet-loss recovery, but would not allow random access).

In Table 1, we additionally observe a larger factor of frame size increase in newer coding standards compared to older generations. For example, for $QP_{\mathrm{NS}} = QP_{\mathrm{CS}} = 22$, the frame size increase factor is 2.44, 5.61, and 6.71 for the JM, HM, and VTM-encoder, respectively. This may suggest that newer codec generations perform more accurate predictions and hence are more efficient in compressing the inter-frames.

In contrast to Table 1 that shows the increase of the P-frame that is replaced by a keyframe, Table 2 shows the overall median bitrate increase in the entire video sequence. In this case, we assume that a single keyframe injection occurs due to a single random-access or packet-loss event. We can observe overall bitrate increases between 0.4% and 3.6%, at equal QPs ($QP_{NS} = QP_{CS}$).

4.3 Impact on Quality

To measure the quality, the Video Multimethod Assessment Fusion or VMAF is mainly used in this paper [29]. The VMAF quality metric results in a score between 0 and 100, where 100 means that the two compared videos are subjectively indistinguishable, and a 6-point difference is a just-noticeable difference (JND) [30]. We utilized the VMAF model *vmaf_4k_v0.6.1*. For completeness, we also calculated the Peak Signal-to-Noise Ratio (PSNR) [18] and Structural SIMilarity (SSIM) values [44]. For each quality measure, we calculated the quality between the uncompressed video and the NS, on the one hand, and the uncompressed video and the KI, on the other hand. As such, we could calculate the decrease in quality from the NS to the KI (ΔQuality).

Table 3 shows the median VMAF decreases (ΔVMAF) for all investigated configurations. The decreases in VMAF scores are first averaged over all frames for each test sequence, and then the median is calculated of all these average VMAF decreases of each test sequence. The median is used to reduce the influence of significant outliers (such as those discussed

Table 1 Median factor of frame size increase due to a keyframe injection.

QP_{CS}	QP_{NS}											
	22	27	32	37	22	27	32	37	22	27	32	37
	JM				HM				VTM			
22	2.44	4.70	9.40	16.25	5.61	15.32	30.95	52.54	6.71	18.83	39.40	68.84
27	1.64	3.24	6.22	10.50	3.43	10.63	21.47	34.47	4.27	12.71	26.75	45.83
32	0.96	1.89	3.90	6.90	2.12	6.30	14.53	23.71	2.48	7.63	17.67	30.91
37	0.54	1.09	2.40	4.53	1.11	3.35	8.21	15.53	1.39	4.40	10.57	20.30
	x264				x265							
22	3.75	8.66	17.54	34.88	4.54	12.11	19.64	43.68				
27	2.59	5.64	11.60	23.29	2.64	7.39	14.66	29.60				
32	1.72	3.34	7.53	14.91	1.57	4.25	9.13	20.31				
37	1.08	2.10	4.59	8.90	1.05	2.49	5.12	12.30				

Table 2 Median percentage (%) of bitrate increase, assuming a single keyframe injection occurs in each video sequence at f=16.

QP_{CS}	QP_{NS}											
	22	27	32	37	22	27	32	37	22	27	32	37
	JM				HM				VTM			
22	0.4	1.0	2.1	4.1	0.9	2.9	6.4	12.7	1.2	3.3	7.3	13.5
27	0.2	0.6	1.4	2.6	0.5	1.7	4.3	8.2	0.6	2.1	4.8	8.9
32	0.0	0.2	0.7	1.6	0.2	1.0	2.8	5.0	0.3	1.3	2.9	5.9
37	-0.1	0.0	0.3	0.9	0.0	0.4	1.5	3.0	0.1	0.6	1.6	3.6
	x264				x265							
22	0.7	1.6	3.1	6.2	0.8	2.1	4.5	9.5				
27	0.4	1.0	2.1	4.0	0.4	1.3	2.9	6.2				
32	0.1	0.5	1.3	2.5	0.2	0.7	1.9	3.8				
37	0.0	0.2	0.6	1.6	0.0	0.4	1.0	2.4				

in Section 4.4). We can observe that, for all codecs, a higher QP_{NS} with an equal QP_{CS} (i.e., the diagonal values) usually results in a larger quality decrease. These cases also result in a larger bitrate increase (see Table 1), i.e., keyframe injection is least efficient in these cases. Moreover, using a higher QP_{CS} (with a constant QP_{NS}) decreases the quality more, which is as expected, since the injected keyframe is of a lower quality. It should be stressed that all reported median ΔVMAF values are well below 6, which indicates that the median impact on the quality is imperceptible.

When we compare the quality decrease of the evaluated codecs, we notice that there are no significant differences: all codecs perform well for keyframe injection. This is a very promising observation, as this indicates a practical applicability of the keyframe-injection method in the future, i.e., it does not become obsolete with the usage of new compression standards.

Finally, for completeness, Table 4 and Table 5 give the median PSNR decrease and median SSIM decrease results. The same observations can be made as from the ΔVMAF results. That is, in general, keyframe injection performs well for all evaluated codecs.

4.4 Outlier Worst-Case Quality-Decrease Analysis

The median ΔVMAF values of Table 3 indicate that the general impact on the quality is very low. However, as shown in Figure 2a, the sequence *BQSquare* performed significantly worse than the others. This is why this section analyzes this outlier sequence in depth.

The left side of Table 6 shows the maximum VMAF decreases (after keyframe injection) for this sequence (*BQSquare*). To limit redundant information, we only show the results where $QP_{NS} = QP_{CS}$ (i.e., corresponding to the black diagonal values in Table 3). From

Table 3 Median decrease in VMAF score.

QP_{CS}	QP_{NS}											
	22	27	32	37	22	27	32	37	22	27	32	37
	JM				HM				VTM			
22	**0.73**	0.92	0.91	0.55	**0.14**	0.40	0.25	0.13	**0.17**	0.37	0.23	0.10
27	1.11	**1.26**	1.01	0.83	0.27	**0.53**	0.33	0.21	0.49	**0.58**	0.45	0.21
32	2.14	2.50	**1.89**	1.27	0.88	1.21	**0.89**	0.47	1.05	1.32	**0.61**	0.37
37	4.60	4.00	3.73	**1.85**	2.32	2.58	1.54	**1.04**	2.43	2.53	1.23	**0.82**
	x264				x265							
22	**0.74**	1.08	0.89	0.54	**0.94**	0.87	0.76	0.48				
27	1.04	**1.31**	1.09	0.68	1.46	**1.20**	1.00	0.69				
32	1.53	2.20	**1.38**	0.93	2.61	2.08	**1.64**	0.93				
37	2.80	3.37	2.45	**1.48**	4.89	3.97	3.12	**1.69**				

Table 4 Median decrease in PSNR.

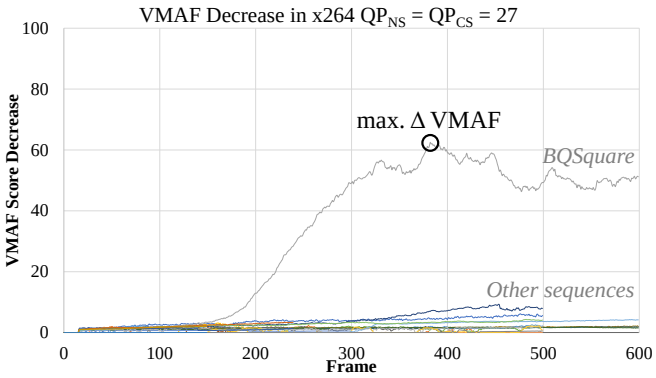
QP_{CS}	QP_{NS}											
	22	27	32	37	22	27	32	37	22	27	32	37
	JM				HM				VTM			
22	**2.20**	0.82	0.33	0.17	**0.19**	0.11	0.05	0.03	**0.35**	0.18	0.09	0.05
27	2.47	**0.91**	0.57	0.22	0.37	**0.18**	0.08	0.04	0.67	**0.28**	0.14	0.08
32	3.13	1.36	**0.75**	0.38	0.74	0.40	**0.16**	0.09	1.01	0.58	**0.24**	0.14
37	4.51	2.32	1.28	**0.56**	1.40	0.84	0.36	**0.17**	2.16	0.90	0.48	**0.24**
	x264				x265							
22	**1.98**	0.87	0.35	0.18	**0.78**	0.34	0.19	0.11				
27	1.95	**0.93**	0.49	0.21	1.05	**0.47**	0.25	0.15				
32	2.42	1.51	**0.64**	0.35	1.52	0.90	**0.43**	0.19				
37	4.01	1.94	1.04	**0.47**	2.85	1.34	0.70	**0.31**				

Table 5 Median decrease in SSIM ($\cdot 10^{-2}$).

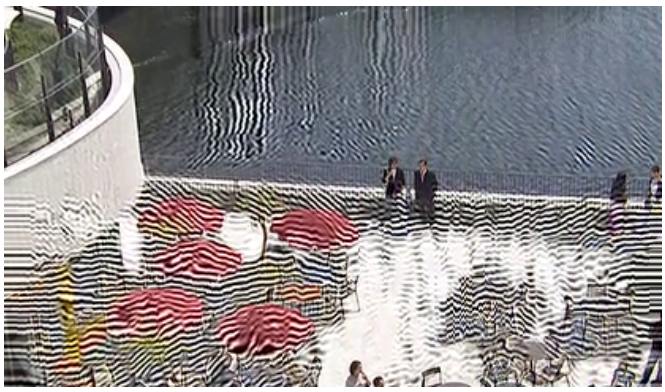
QP_{CS}	QP_{NS}											
	22	27	32	37	22	27	32	37	22	27	32	37
	JM				HM				VTM			
22	**0.34**	0.20	0.14	0.11	**0.04**	0.05	0.04	0.02	**0.04**	0.06	0.05	0.04
27	0.38	**0.26**	0.20	0.16	0.08	**0.10**	0.08	0.05	0.10	**0.10**	0.08	0.06
32	0.59	0.41	**0.32**	0.25	0.20	0.18	**0.13**	0.14	0.22	0.19	**0.17**	0.13
37	1.01	0.81	0.70	**0.39**	0.52	0.55	0.49	**0.29**	0.49	0.40	0.37	**0.20**
	x264				x265							
22	**0.47**	0.29	0.23	0.18	**0.20**	0.21	0.20	0.16				
27	0.53	**0.33**	0.25	0.19	0.25	**0.24**	0.23	0.20				
32	0.68	0.47	**0.32**	0.25	0.34	0.32	**0.28**	0.24				
37	0.97	0.70	0.58	**0.38**	0.57	0.53	0.47	**0.38**				

Table 6 Results for worst-case sequence *BQSquare*.

Codec		Max. ΔVMAF				Min. frames before ΔVMAF ≥ 6			
	$QP_{NS} = QP_{CS} =$	22	27	32	37	22	27	32	37
x264		67.0	62.5	48.6	35.2	155	161	174	188
JM		57.3	48.8	37.7	29.4	168	164	183	187
x265		4.4	4.4	5.3	3.1	-	-	-	-
HM		3.1	2.8	2.6	3.5	-	-	-	-
VTM		7.1	25.0	11.7	3.3	355	194	232	-
x264 - halfpel changed		3.6	2.8	2.4	2.6	-	-	-	-
x264 - ultrafast (no subpel)		1.5	3.1	3.1	4.2	-	-	-	-



(a) VMAF decrease over time.



(b) Frame 383 of KI with max. ΔVMAF.

Fig. 2 (a) The VMAF decrease over time, for all test sequences. We observe that most sequences have a relatively low quality decrease. In contrast, BQSquare shows a very large VMAF decrease. (b) The selected frame with the worst quality decrease of the outlier sequence *BQSquare*. We observe very perceptible zebra-pattern artifacts in this frame.

Table 6, we can observe that the severity of the worst-case drift artifacts is much worse for the H.264/AVC and H.266/VVC standards compared to H.265/HEVC.

Figure 2b visualizes the frame with the largest quality decrease when using *x264* and $QP_{NS} = QP_{CS} = 27$, with clearly perceivable zebra-pattern artifacts. Note that these artifacts are not visible immediately but gradually become perceptible. In the analyzed example, the artifacts are slightly noticeable approx. 100 frames after keyframe injection, and reach their peak at frame 383. Similar artifacts were also observed in *BQSquare* when using JM and VTM. Moreover, the same artifact type appeared in few other test sequences, yet in a much lower quantity and very locally.

In general, the right side of Table 6 shows the minimum number of frames (*after* keyframe injection at $f = 16$) before the VMAF decrease exceeds 6, i.e., before the drift-errors become just noticeable. The reported values lay between 155 and 355, which corresponds to approx. 2.5 to 6 seconds (at 60 fps). Thus, in practice, a new keyframe that stops the drift may already occur in the NS before the worst-case artifacts become perceptible (e.g., due to a scene change).

We researched the origin of the zebra-like drift-error artifact in H.264/AVC, and pinpointed the half-pixel (halfpel) interpolation of the motion estimation (ME) step as its cause. The following explains this phenomenon. During ME, viable blocks of pixels are considered for motion prediction. This is done on a sub-pixel (subpel) level for increased accuracy, i.e. on halfpel and quarter-pixel (qpel) level. In order to estimate these virtual pixel values, they are interpolated from the integer pixel locations. In H.264/AVC, a 6-tap filter with coefficients $(1, -5, 20, 20, -5, 1)/32$ is used to create the halfpel samples using 6 surrounding integer pixel samples [45]. Subsequently, the qpel samples are created using linear interpolation using two neighboring halfpel samples. In this way, the interpolated subpel samples improve the ME performance.

Most interestingly for this paper, the halfpel interpolation filter slightly amplifies certain frequency components [28]. This is visualized in Figure 3a, which plots the filter's frequency response in the Fast Fourier Trans-

form (FFT) domain. The figure demonstrates that the filter functions as a low-pass filter, i.e., it leaves low frequencies untouched, while decreasing the amplitude for high-frequency components. Most notably, the amplitude of the frequency response is greater than one for certain mid-frequencies. In ordinary compression, this slight overshooting is not a perceptible problem because they are automatically corrected in the residual signal.

Note that, in qpel interpolation, these frequencies do not overshoot, because an additional linear filter with coefficients $(0, 0, 1, 1, 0, 0)/2$ is applied on halfpel level. This acts as an additional lowpass filter that lowers the overshot frequency amplitudes below one. Thus, the overshooting of frequencies is only a problem in halfpel interpolation.

When the residuals do not perfectly reconstruct the original frame as it existed in the encoder (i.e., due to keyframe injection), a slight error is introduced. Although this error may initially not be visible, it becomes perceptible when the filter is applied repetitively. That is because the repeated filter application causes this mid-frequency band to be amplified exponentially. This is visualized in Figure 3b, which shows the frequency response when the filter is applied 10 times. In other words, this simulates what happens when this halfpel interpolation filter is applied on the same region of pixels 10 times, e.g., in 10 subsequent frames. After 10 repetitions, the amplitude of the overshot mid-frequencies is close to 2. More extremely, after 100 repetitions, the amplitude is over 300. In other words, these frequencies are significantly amplified and become perceptible. These are the zebra-pattern drift artifacts of Figure 2b.

To confirm our hypothesis that the subpel interpolation is the cause of these zebra-pattern artifacts, we changed the coefficients of the 6-tap filter in the *x264*-encoder and the JM decoder. Note that this makes the encoder and decoder not standard compliant, but this is only done for demonstration purposes. The coefficients are changed to $(1, -4, 19, 19, -4, 1)/32$, because they yield a similar filter, but without any amplification (as visualized in Figure 3c). When we apply keyframe injection using this adapted codec, the zebra-pattern artifacts do not occur anymore and the worst-case maximum VMAF quality decreases are lower than six points. Alternatively, a more practical solution is to use the ultrafast configuration in *x264*, which disables subpel interpolation and hence also prevents the zebra-pattern artifacts. The corresponding results are shown in the bottom rows of Table 6. This confirms our hypothesis that the halfpel interpolation is the problem.

In conclusion, the worst-case artifacts occur in H.264/AVC because the halfpel interpolation amplifies certain frequencies. When a keyframe is injected, and hence this reference frame of motion prediction is slightly wrong, the errors drift and zebra-pattern artifacts become perceptible. Although the problem can be avoided by disabling subpel ME, future coding standards could more carefully design their interpolation filters such that they do not amplify any frequency components. Future work will investigate how to prevent these worst-case perceptible artifacts by tweaking the video encoder implementation without disabling subpel ME. This could be done by limiting the usage of subpel ME repetitively in succeeding frames. Additionally, future work will analyze the H.266/VVC worst cases in more detail.

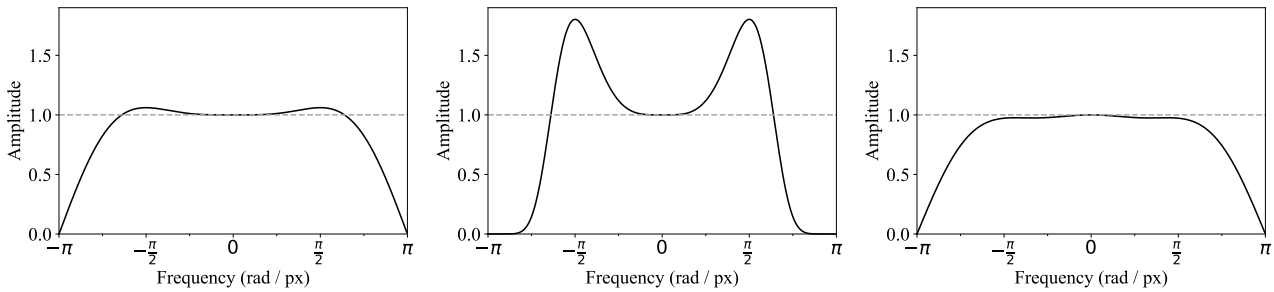
4.5 Discussion

This section briefly discusses how keyframe injection behaves in various packet-loss scenarios, as well as how it compares with a traditional scheme where no keyframe injection is applied.

When packet loss happens frequently but sparsely in one GOP, keyframe injection may be applied after each packet-loss event. Each time a keyframe is injected into that user's NS, the bitrate for that user increases roughly with the values in Table 2. In other words, the overall bitrate increases for that user increases roughly linear with the number of packet-loss events. Hence, if the packet loss happens very frequently, it may be better for the system to decide to instead perform other error-concealment (EC) strategies (e.g. a frame-copy strategy [38] or more advanced strategies [4]). Future work should look into making a system that decides which error-concealment or error-resilience strategy is optimal, which may depend on the video content and each user's network conditions.

When packet loss occurs in a burst, i.e., multiple sequential frames are lost, the system could decide to wait to perform keyframe injection until the packet-loss burst is over. For the keyframe injection solution, it does not matter how many preceeding frames were lost: a single injected keyframe conceals all prior packet losses. This is in contrast to other error-concealment strategies that utilize previous frames (such as a frame-copy strategy [38]). That is, such EC methods work best when they can estimate the current frame using neighboring frames which do not significantly differ from the current one. If many preceding frames are lost, the performance of the EC method will hence drop.

In practice, it is fair to assume that no keyframe injection occurs for most users and/or for most of the time, and they hence suffer no bitrate increase. In contrast, they even receive a more compression-efficient NS than when no keyframe-injection scheme is applied,



(a) Standard filter once. (b) Standard filter 10 times. (c) Modified filter once.

Fig. 3 Frequency response of halfpel interpolation filter in H.264/AVC after applying (a) the standard filter once, (b) the standard filter 10 times, and (c) applying the modified halfpel interpolation filter with our changed coefficients.

since the NS's inter-frame dependency is not artificially broken at regular short intervals. More specifically, when a traditional scheme is applied instead of the the proposed keyframe-injection solution, the intra-period is typically shortened to provide more regular random-access and packet-loss capabilities.

For example, assume that a more traditional scheme is applied and the intra-period of the NS is made half of the length of the test sequences (i.e., 5 seconds for most sequences). In that case, a single extra I-frame is added to each NS, which has an impact on the bitrate that is equivalent to applying keyframe injection once (i.e., a P-frame is replaced by a keyframe). However, the important difference of this traditional scheme with the proposed keyframe-injection scheme is that *all* users have a bitrate increase of approximately 0.4% to 3.6%. If the intra-period is further reduced and more keyframes are forced to be introduced in the NS, the bitrate would increase roughly linearly with the number of introduced keyframes. For example, if the intra-period is halved an extra time (i.e., 2.5 seconds for most of the sequences), two more keyframes are added in each sequence. These keyframes are approximately 2.44 to 20.30 times larger than the keyframes that they replace, and hence the overall bitrate increase that *all* users is approximately tripled compared to the case of adding only a single keyframe in Table 2 (i.e., roughly 1.2% and 10.2% larger). Halving the intra-period yet another time (i.e., 1.25 seconds for most of the sequences) leads to 7 introduced keyframe and hence a bitrate of roughly 2.8% to 25.2%. As such, in the proposed keyframe-injection solution, all users benefit by having a more compression-efficient NS. Only the users that require a random access or packet-loss recovery undergo a temporary bitrate increase.

5 Conclusion

This paper discussed the requirements to create a standard-compliant bitstream using keyframe injection in three generations of video compression standards (H.264/AVC, H.265/HEVC, and H.266/VVC). That is, one has to take care of the POC and parameter sets during keyframe injection, and disable TMVP. Additionally, we studied the performance using five different codecs to thoroughly demonstrate the viability of keyframe injection.

Since the median VMAF quality decrease is below six points for all tested codecs, the drift-error artifacts generally remain imperceptible. This is also confirmed by the PSNR and SSIM measurements. Additionally, keyframe injection is most efficient when using low QP values. In H.264/AVC and H.266/VVC, a worst-case outlier created rare but undesirable, perceptible zebra-pattern artifacts. We explored the root cause for these rare outliers. That is, the halfpel interpolation filter coefficients are not ideal for the proposed use case in H.264/AVC. As a solution, codecs can disable subpel ME, and future coding standards could more carefully design their filters in order to avoid such behavior.

There are several interesting future work directions. First, future research could evaluate the impact of keyframe insertion on other coding structures than the low-delay configuration. Second, novel strategies to reduce the frame size increase of inserting a CS frame in the NS will be explored. Third, future work will investigate how to prevent the rare zebra-pattern artifacts without disabling subpel ME, and it will analyze the H.266/VVC worst cases in more detail. Fourth, a decision system that estimates which error-concealment or error-resilience strategy is optimal for given video content and network conditions could be developed.

In conclusion, we demonstrated the practical applicability of keyframe injection in three generations of

video coding standards. As such, keyframe injection enables fast channel switching capabilities and packet-loss robustness, which is very important in low-latency video distribution in error-prone networks.

Acknowledgements The computational resources (STEVIN Supercomputer Infrastructure) and services used in this work were kindly provided by Ghent University, the Flemish Supercomputer Center (VSC), the Hercules Foundation and the Flemish Government department EWI.

Conflict of interest One of the authors, Pieter-Jan Speelmans, is founder of THEO Technologies, which is part of the HESP Alliance. The HESP Alliance standardizes and promotes the HESP, which utilizes keyframe injection. This could be considered as a potential conflict of interest.

Data Availability Statement The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request. The results presented and used for this study are available for download on https://media.idlab.ugent.be/keyframe-insertion. The source code of keyframe injection has been made available for reproducibility on https://github.com/IDLabMedia/NALUProcessing.

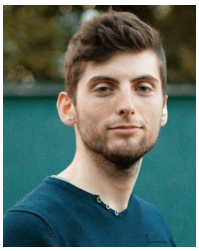
References

1. Akgul, T.: A Client-Based Fast Channel Change Technique Using Multiple Decoder Clocks. IEEE Trans. Consum. Electron. **66**(1), 61–68 (2020)
2. Begen, A.C., Glazebrook, N., Ver Steeg, W.: A Unified Approach for Repairing Packet Loss and Accelerating Channel Changes in Multicast IPTV. In: 2009 6th IEEE Consumer Communications and Networking Conference, pp. 1–6 (2009)
3. Bejerano, Y., Koppol, P.V.: Improving zap response time for IPTV. In: IEEE INFOCOM 2009, pp. 1971–1979 (2009)
4. Benjak, M., Samayoa, Y., Ostermann, J.: Neural network-based error concealment for VVC. In: IEEE International Conference on Image Processing (ICIP), pp. 2114–2118. IEEE (2021)
5. Bossen, F.: Common test conditions and software reference configurations. Tech. Rep. JCTVC-L1100, ITU-T Joint Collaborative Team on Video Coding (JCT-VC) (2013)
6. Boyce, J.M., Tourapis, A.M.: Fast efficient channel change [set-top box applications]. In: 2005 Digest of Technical Papers. International Conference on Consumer Electronics, 2005. ICCE., pp. 1–2 (2005)
7. Bross, B., Helle, P., Lakshman, H., Ugur, K.: Inter-Picture Prediction in HEVC. In: High Efficiency Video Coding (HEVC): Algorithms and Architectures, pp. 113–140. Springer International Publishing, Cham (2014)
8. Bross, B., Wang, Y.K., Ye, Y., Liu, S., Chen, J., Sullivan, G.J., Ohm, J.R.: Overview of the versatile video coding (VVC) standard and its applications. IEEE Trans. Circuits Syst. Video Technol. **31**(10), 3736–3764 (2021)
9. De Praeter, J., Van Wallendael, G., Slowack, J., Lambert, P.: Video Encoder Architecture for Low-Delay Live-Streaming Events. IEEE Transactions on Multimedia **19**(10), 2252–2266 (2017)
10. Erfanian, A., Tashtarian, F., Farahani, R., Timmerer, C., Hellwagner, H.: On Optimizing Resource Utilization in AVC-based Real-time Video Streaming. In: 2020 6th IEEE Conference on Network Softwarization (NetSoft), pp. 301–309 (2020)
11. Farber, N., Girod, B.: Robust h.263 compatible video transmission for mobile access to video servers. In: Proceedings of International Conference on Image Processing, vol. 2, pp. 73–76 vol.2 (1997)
12. Farber, N., Steinbach, E., Girod, B.: Robust h.263 compatible transmission for mobile video server access. In: Proceedings of First International Workshop on Wireless Image/Video Communications, pp. 8–13 (1996)
13. Fati, S.M., Azad, S., Khan Pathan, A.S.: Channel-Zapping Time in IPTV: Challenges and Solutions. In: IPTV Delivery Networks: Next Generation Architectures for Live and Video-on-Demand Services, pp. 151–183. Wiley (2018)
14. Fraunhofer: Advanced Video Coding (H.264/AVC) — JVT. URL https://avc.hhi.fraunhofer.de/
15. Fraunhofer: High Efficiency Video Coding (H.265/HEVC) — JVT-VC. URL https://hevc.hhi.fraunhofer.de/
16. Fraunhofer: Versatile Video Coding (VVC) — JVET. URL https://jvet.hhi.fraunhofer.de/
17. Ghahfarokhi, B.S., Moghim, N., Eftekhari, S.: Reducing channel zapping time in live TV broadcasting over content centric networks. Multimedia Tools and Applications **76**(22), 23239–23271 (2017)
18. Horé, A., Ziou, D.: Image quality metrics: PSNR vs. SSIM. In: International Conference on Pattern Recognition, pp. 2366–2369 (2010)
19. Jennehag, U., Pettersson, S.: On synchronization frames for channel switching in a GOP-based IPTV environment. In: 2008 5th IEEE Consumer Communications and Networking Conference, CCNC 2008, pp. 638–642 (2008)
20. Joo, H., Song, H., Lee, D., Lee, I.: An Effective IPTV Channel Control Algorithm Considering Channel Zapping Time and Network Utilization. IEEE Transactions on Broadcasting **54**(2), 208–216 (2008)
21. Kalman, M., Steinbach, E., Girod, B.: Adaptive media playout for low-delay video streaming over error-prone channels. IEEE Transactions on Circuits and Systems for Video Technology **14**(6), 841–851 (2004)
22. Karczewicz, M., Kurceren, R.: The SP- and SI-frames design for H.264/AVC. IEEE Transactions on Circuits and Systems for Video Technology **13**(7), 637–644 (2003)
23. Kazemi, M., Ghanbari, M., Shirmohammadi, S.: A review of temporal video error concealment techniques and their suitability for HEVC and VVC. Multimedia Tools and Applications (2021)
24. Kokkonis, G., Psannis, K.E., Roumeliotis, M., Schonfeld, D.: Real-time wireless multisensory smart surveillance with 3D-HEVC streams for internet-of-things (IoT). The Journal of Supercomputing **73**, 1044–1062 (2017)
25. Lee, E., Ku, J.Y., Bahn, H.: An efficient hot channel identification scheme for IPTV channel navigation. IEEE Transactions on Consumer Electronics **60**(1), 124–129 (2014)
26. Lee, Y., Lee, J., Kim, I., Shin, H.: Reducing IPTV channel switching time using H.264 scalable video coding. IEEE Transactions on Consumer Electronics **54**(2), 912–919 (2008)
27. Mareen, H., Courteaux, M., Vounckx, J., Lambert, P., Van Wallendael, G.: Keyframe insertion for random access and packet-loss repair in H.264/AVC, H.265/HEVC,

and H.266/VVC. In: 2022 Data Compression Conference (DCC) (2022)
28. Muratori, C.: Stable Filtering - Part 1. URL https://caseymuratori.com/blog_0035
29. Netflix Technology Blog: Toward A Practical Perceptual Video Quality Metric. URL https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652
30. Ozer, J.: Finding the Just Noticeable Difference with Netflix VMAF. URL https://streaminglearningcenter.com/codecs/finding-the-just-noticeable-difference-with-netflix-vmaf.html
31. Pham Van, L., De Praeter, J., Van Wallendael, G., Van Leuven, S., De Cock, J., Van de Walle, R.: Efficient Bit Rate Transcoding for High Efficiency Video Coding. IEEE Transactions on Multimedia **18**(3), 364–378 (2016)
32. Praeter, J.D., Hollmann, C., Sjoberg, R., Wallendael, G.V., Lambert, P.: Network-Distributed Video Coding. arXiv (2021)
33. Psannis, K.E.: HEVC in wireless environments. Journal of Real-Time Image Processing **12**(2), 509–516 (2016)
34. Ramos, F.M.V., Crowcroft, J., Gibbens, R.J., Rodriguez, P., White, I.H.: Reducing channel change delay in IPTV by predictive pre-joining of TV channels. Signal Processing: Image Communication **26**(7), 400–412 (2011)
35. Skupin, R., Bartnik, C., Wieckowski, A., Sanchez, Y., Bross, B., Hellge, C., Schierl, T.: Open GOP resolution switching in http adaptive streaming with VVC. In: 2021 Picture Coding Symposium (PCS), pp. 1–5. IEEE (2021)
36. Speelmans, P.J.: HESP - High Efficiency Streaming Protocol. Internet-Draft draft-theo-hesp-01, Internet Engineering Task Force (2021). URL https://datatracker.ietf.org/doc/html/draft-theo-hesp-01. Work in Progress
37. Sullivan, G.J., Ohm, J., Han, W., Wiegand, T.: Overview of the High Efficiency Video Coding (HEVC) Standard. IEEE Transactions on Circuits and Systems for Video Technology **22**(12), 1649–1668 (2012)
38. Van Wallendael, G., Lambert, P., Speelmans, P.J., Mareen, H.: Mixed-resolution HESP for more efficient fast channel switching and packet-loss repair. In: Picture Coding Symposium (PCS), pp. 319–323 (2022)
39. Van Wallendael, G., Mareen, H., Vounckx, J., Lambert, P.: Keyframe Insertion: Enabling Low-Latency Random Access and Packet Loss Repair. Electronics **10**(6) (2021)
40. Van Wallendael, G., Staelens, N., Van Leuven, S., De Cock, J., Lambert, P., Demeester, P., Van de Walle, R.: Fast channel switching for single-loop scalable HEVC. In: 2014 IEEE International Conference on Image Processing (ICIP), pp. 5991–5995 (2014)
41. Van Wallendael, G., Van Lancker, W., De Cock, J., Lambert, P., Macq, J., Van de Walle, R.: Fast Channel Switching Based on SVC in IPTV Environments. IEEE Transactions on Broadcasting **58**(1), 57–65 (2012)
42. VideoLAN: x264, the best H.264/AVC encoder. URL https://www.videolan.org/developers/x264.html
43. VideoLAN: x265, the free H.265/HEVC encoder. URL https://www.videolan.org/developers/x265.html
44. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (2004)
45. Wiegand, T., Sullivan, G., Bjontegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 560–576 (2003)
46. Yang, C., Liu, Y.: On Achieving Short Channel Switching Delay and Playback Lag in IP-Based TV Systems. IEEE Transactions on Multimedia **17**(7), 1096–1106 (2015)

Hannes Mareen (Member, IEEE) obtained the M.Sc. degree in computer science engineering from Ghent University, Belgium, in 2017. From 2017 until 2021, he worked towards a Ph.D. as an SB fellow at IDLab, Ghent University – imec, with the financial support of the Research Foundation – Flanders (FWO). Since 2021, he is working in the same group as a postdoctoral researcher. In 2020, he was a Visiting Research Fellow with the School of Engineering and Information Technology, The University of New South Wales, Canberra, Australia, and in 2022, he was a Visiting Researcher at the University of Naples - Federico II, Naples, Italy.

His main areas of interest are multimedia security and forensics, as well as video coding and compression.



Martijn Courteaux received his M.Sc. degree in Computer Science Engineering from Ghent University, Belgium, in 2018. Since then, he is a researcher and Ph.D. candidate at IDLab, Ghent University – – imec, with the financial support of the Research Foundation — Flanders (FWO). His research currently focuses on the modeling and compression of light fields and light field videos, and is set in the context of statistics, signal processing and compression.

Pieter-Jan Speelmans obtained a M.Sc. degree in Engineering from KU Leuven, Belgium in 2011. Afterwards he has been working as an innovator within the contexts of media streaming and media playback. His current focus areas are on scalable media delivery with a high user experience towards large audiences and its practical applications within the industry.

Peter Lambert is a full-time Associate Professor at IDlab, Ghent University – imec (Belgium) since 2013. He received his Master's degree in Science (Mathematics) and in Applied Informatics from Ghent University in 2001 and 2002, respectively, and he obtained the Ph.D. degree in Computer Science in 2007 at the same university. In 2009, he became a Technology Developer at Ghent University, which he combined with a part-time Assistant Professorship at IDLab since 2010. His research interests include (mobile) multimedia applications, multimedia coding and adaptation technologies, and 3D graphics.

Glenn Van Wallendael obtained the M.Sc. degree in Computer Science Engineering from Ghent University, Belgium in 2008. Afterwards, he obtained the Ph.D. at IDLab, Ghent University, with the financial support of the Research Foundation - Flanders (FWO). Since 2019, he works as aa Assistant Professor for both Ghent University and imec on topics such as the efficient representation and compression of visual information, including 360 degree video, light field, virtual reality and the different operations on these modalities such as (scalable) compression, transcoding, encryption, watermarking, personalized delivery, and quality estimation.