ENCODING OPTIMIZATIONS FOR VIDEO GAME LIVE STREAMING

J. Le Tanou and N. Francisco

MediaKind, France and UK

ABSTRACT

In recent years, live video-game-centric streaming platforms have experienced dramatic growth, driven by the explosion of the esports market. Besides, live-event rightsholders and broadcasters are successfully 'gamifying' their video offerings. Given the rise of video game live streaming, we address in this paper the questions of how to optimize and adapt the encoding strategies for efficient game content coding. We first characterize game content, focusing on how its signal characteristics differ from 'natural' content. We then share insights on selected encoding strategies targeting improvements in both compression efficiency and density/encoder run-time for signals with such characteristics. Additionally, we examine the relevance of Screen Content Coding (SCC) tools, as adopted in HEVC SCC and VVC standards, in a video game coding context. Finally, we conclude the paper by highlighting the benefits of using game-content aware technology with better compression efficiency under real-time constraints.

INTRODUCTION

The proliferation of IPTV and OTT media delivery technologies have helped establish video game live streaming as a major market segment. In recent years, video-game-centric streaming platforms such as Twitch, YouTube, or Facebook Gaming, have experienced a dramatic growth (101%, 65%, and 238% respectively in 2020). Overall, the related global esports market – the competitive and professional element of the gaming world – generated more than $1 billion in revenues in 2020 and is forecast to hit $1.6 billion by 2023 [1]. In addition, we should expect to see live-event rightsholders and broadcasters 'gamify' their video offerings with a greater degree of social interaction and mixed-media experiences. For example, the NBA has successfully trialed replacing live action with video game simulations [2]. Likewise, during lockdown SRO Motorsports successfully switched real races to esports, even holding them during their originally scheduled date and time. The unique Fanatec Esports GT Pro Series, initiated in 2021, where real racing drivers compete in a virtual environment for points towards their real-world championship will survive past the lockdown period and return for 2022 streamed live across YouTube, Twitch and Facebook [3]. Given the rise of video game live streaming, questions on how to optimize and adapt the encoding strategies to game content open an important field of research.

In this paper, we discuss some key elements to efficiently encode video game content in real-time and how this can lead to 'game-content aware' encoding solutions. We start by characterizing game content, focusing on how its signal characteristics differ from 'natural' content and may require specific encoder optimizations. We then share insights about some specific encoding strategies (e.g., rate control, adaptive quantization, in-loop filtering, motion

estimation, etc.), targeting improvements in both compression efficiency and density/encoder run-time for signals with such characteristics. Additionally, and since video compression standards, such as HEVC or VVC, include coding tools specifically designed to compress screen content, we examine the relevance of those tools in a video game coding context. Finally, we show the benefits of using game-content aware technology with better compression efficiency under real-time constraints.

CHARACTERIZATION OF VIDEO GAME CONTENT

Video games generally refer to interactive games that run on electronic media platforms. Popular mainstream games may take the form of computer games, console games, mobile games, handheld games, VR games, cloud games, among others.

Game content signal characteristics significantly differ from ‘natural’ video content, which are commonly compressed and delivered over various networks (i.e. Live broadcast, broadband/IPTV/OTT).

Gaming videos are computer generated (i.e. using a rendering engine) while ‘natural’ videos are captured from optical/electronical cameras/sensors. As such, video game content is assumed to be mostly characterized by presenting plain and smooth color gradient areas, sharp and well-defined objects (e.g. with text and graphics), no spatial, temporal, or cross-component noises, no focus or motion blur, and low average motion complexity over time, but with possible high complexity motion bursts and with typically non-translational motions (e.g. first-person shooters games).

Nevertheless, gaming videos streamed over the internet cover a wide range of games, varying largely in their encoding complexity. Games fall into various genres, including role-playing games, adventure games, action games, first-person shooters, real-time strategy games, fighting games, board games, massive multiplayer online role-playing games, and others. For this work we built a representative test set borrowed from the ‘GamingVideoSET’ database [4] that contains 24 gaming contents recorded from 12 different games as shown in Figure 1. Each video is 30s length in 1920x1080 / 30fps / 8-bit, 4:2:0, YUV format.

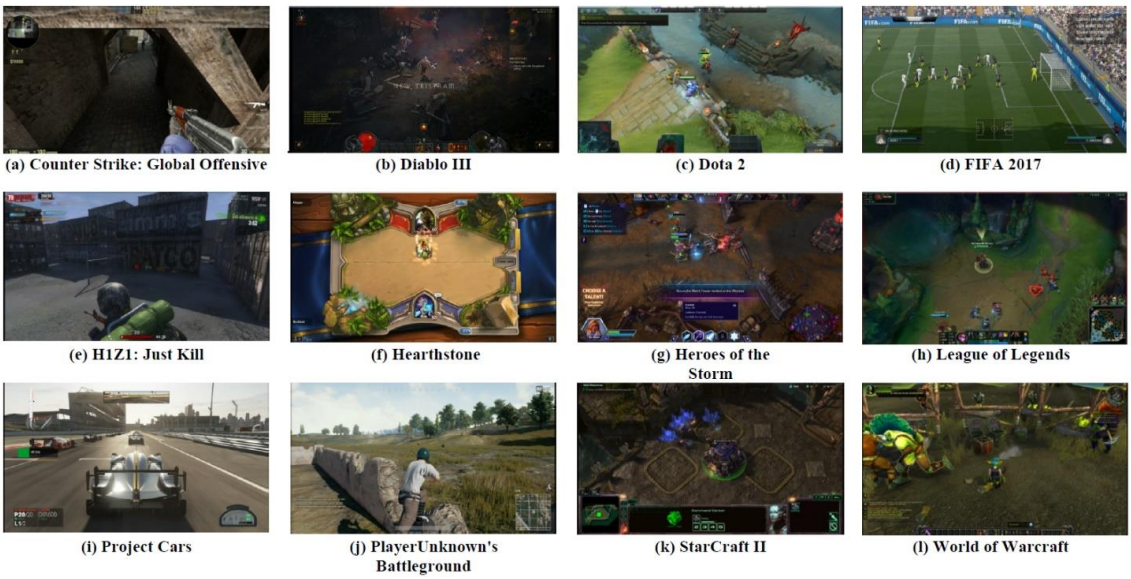

Figure 1 - Screenshots of the gaming videos test set

Complementary to the gaming video test set, we defined a second test set representative of 'natural' videos, to establish a comparison point. It consists of 5 full-HD sequences borrowed from the *JVET - Common Test Conditions (CTCs),* namely *Class B* of the JVET CTCs, and shown Figure 2.

Figure 2 – Screenshots of the natural videos test set

For the two test sets we plotted and compared their respective coding complexities. The ITU-T Rec. P.910 commonly defines Spatial Information (SI) and Temporal Information (TI) values to approximate a measure of complexity. In this work, we introduce and define an analogous but slightly different content complexity measurement, using four metrics, that can be easily estimated by a Look-ahead of an encoder, and better reflect the coding complexity of the input content. Each of the four metrics are computed on a 16x16 block-basis over each individual sequence frame:

Spatial or Intra coding complexity

The spatial or intra coding complexity, C_{intra}, of a sequence is defined as the average over the sequence of the 16x16-block transformed residual energy after intra prediction from source sample neighboring. H264 intra prediction modes are used for sample prediction. The Hadamard transform is used as transform type.

Temporal or Inter coding complexity

The temporal or inter coding complexity, C_{inter}, of a sequence is defined as the average over the sequence of the 16x16-block transformed residual energy after motion estimation (ME) and compensation (MC). A hierarchical motion estimation algorithm is used as ME.

Sequence coding complexity

The sequence coding complexity, C_{seq}, is defined as the average over the sequence of the minimum between Intra and Inter transformed residual energies for each block. This metric gives an estimate of how complex a given signal is to predict and compress, as well as how much residual signal information needs to be coded (i.e. residual signal coding cost)

Motion coding complexity

The motion coding complexity, C_{motion}, is defined as the average over the sequence of 16x16-block motion vector difference with a motion predictor. The motion vector predictor for each block is defined as the median value of the 3 neighboring (left, top-left and top) MVs if available. This last metric gives an order of the variability of the motion, as well as how much residual motion information needs to be coded (i.e. motion coding cost)

The complexity metrics were computed for each sequence and test set, with resulting values depicted Figure 3.

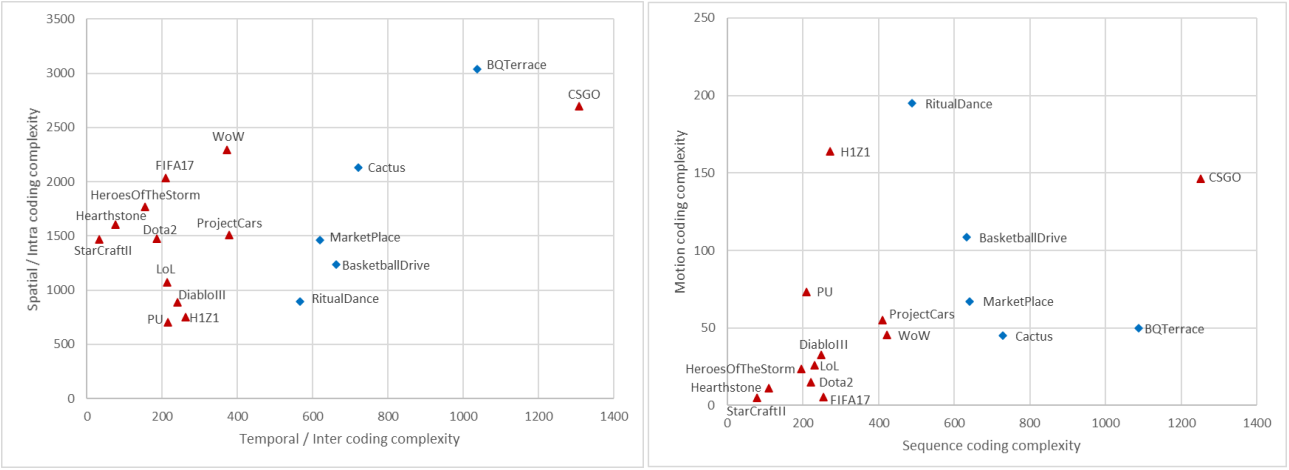

Figure 3 – Coding complexities for the game (red) and natural (blue) test sets

From the analysis of the complexity results, we can first observe on the left plot, that the temporal or inter coding complexity is clearly lower for the gaming video test set. There is one exception for the *'Counter Strike: Global Offsensive' (CSGO)* which present strong and complex motion (i.e. non-translational) as a first-person shooting game; motion which is not well captured by a block-based translational motion estimation/compensation model. Surprisingly, the spatial or intra coding complexity for the game test set is in average on par with the complexity of the natural set. A closer analysis of the natural test content, i.e. '*MarketPlace'*, '*BasketballDrive'* and '*RitualDance',* shows that those sequences present a lot of focus and motion blurs, smoothing a large proportion of the highly texture areas. It may explain their relatively low intra-complexity scores. The right plot shows the overall sequence coding complexity score against the motion coding complexity. With exception of the '*CSGO'* sequence, the video game test set is predictable with significantly more signal information redundancy than the natural content set. In average, the motion information is low (i.e. homogeneous across the frame). The H1Z1 sequence shows a relatively higher motion cost (i.e. more motion variability across the frame) while being well compensated out of the prediction.

RELEVANT ENCODING OPTIMIZATIONS FOR VIDEO GAME CODING

From the previous assumptions and learnings on game content signal characteristics, we highlight in this section some relevant tracks for encoding optimizations targeting this specific type of content. There is no will to be exhaustive in the possible strategies and levers available from an encoder perspective for better compressing video games. As such, we discuss 4 selected strategies that are beneficial for game coding in terms of compression efficiency or coding complexity reduction, with application into MediaKind's HEVC SW optimized codec: UnCL-HEVC.

All the experimental results reported in this section rely on the same following test conditions: use of full UnCL-HEVC tool set, hierarchical B-frames, 1s Intra period with open-GOP, « Constant QP » mode comparison using 6 base QP points (17, 22, 27, 32, 37, 42), and compression efficiency measured in terms of bitrate saving for the same quality score (e.g. SSIM, MS-SSIM or PSNR) using the Bjöntegaard metric [5].

Rate control and adaptive quantization optimization

Bit budget repartition and the subsequent quantization control within a picture or a Group-Of-Pictures (GOP) is a fundamental optimization point for efficient video compression [6]. We extensively discussed in [6] the various ways for optimally trading bits between samples to code (i.e. block or coding unit (CU)) and how to adapt the quantization parameter from frame down to a block/CU within a GOP. Notably, we introduced a best-in-class Adaptive Quantization (AQP) algorithm: RDSTQ for Rate-Distortion-based Spatio-Temporal Quantization. The RDSTQ algorithm models the temporal distortion propagation from CU to CU within a GOP to estimate optimal QPs per CU, that minimize the total Distortion (or equivalently maximize the Video Quality (VQ)) for a given Rate constraint. It optionally includes a spatial psycho-visual weighting function accounting for Human Visual System (HVS) quality perception, and a "strength" parameter controlling the importance of the temporal distortion propagation and the dynamic of the output QPs. Interested readers can refer to [7] for a thorough description of the underlying theory, modelling and optimization problem resolution, as well as extensive data results and performance analysis. We've seen in the previous section that video game content tends to be characterized by:

1. plain and smooth color/luminance gradient areas mixed with sharp and well-defined objects (e.g. with text and graphics),
2. relatively well predicted motions, with several static or with low motion areas.

For addressing the observation point 1, it is considered to adapt the psycho-visual weighting function to best preserve main object edges/boundaries into RDSTQ model.

Briefly, the rate (*R*) – distortion (*D*) optimization problem solved by the RDSTQ is:

$$\left\{QP_{i_t}\right\}_{i_t}^{N_T} = ARGMIN\left(\sum_t \sum_{i_t} \psi_{i_t} D_{i_t}\right) \; subject\; to \sum_t \sum_{i_t} R_{i_t} = R_{Tot} \qquad (1)$$

where: $\left\{QP_{i_t}\right\}_{i_t}^{N_T}$ is the set of optimal QP for the GOP, i_t is the index of the CU/block i in the frame t of the GOP and ψ_{i_t} the psycho-visual weighting function applied for that block.

The default psycho-visual weighting used in RDSTQ model and published in [7], is based on local pixel variances of a block i_t. It models spatial masking in case of highly textured areas, and has a great correlation with SSIM or MS-SSIM quality metric. It is defined by the equation (2).

$$\forall i_t,\ \psi_{i_t} \approx \frac{1}{\sigma_{i_t}}\ with\ \sigma_{i_t} = \sqrt{(\sigma_Y^2 + \sigma_U^2 + \sigma_V^2)_{i_t}} \qquad (2)$$

As a generic improvement, and specifically relevant in the context of video game coding, it is proposed to adapt the default psycho-visual weighting based on local pixel variances and the modulus of the local gradient (*G*) (e.g. out of an edge detector). Such, the new psychovisual weighting, defined equation (3), models spatial masking in case of highly textured areas while preserving edges/boundaries of interest.

$$\forall i_t,\ \psi_{i_t} \approx \frac{1}{c(\|G_{i_t}\|)\times\sigma_{i_t}} \quad with\ c(\|G_{i_t}\|) = 1 \Big/ \left(1 + \left(\frac{\|G_{i_t}\|}{K}\right)^2\right) \qquad (3)$$

The benefits of the use of the new spatial psychovisual weighting function are evaluated (without loss of generality) for a spatial-only version of the RDSTQ model, i.e. RDSQ, dropping the temporal distortion propagation into the optimization problem (1).

Experimental results are given Table 1 for the two test sets based on 'natural' and 'game' content. It shows that the use of the spatial psycho-visual weighting function (3) with RDSQ in the context of game coding provides in average -2.72% bitrate-saving for the same MS-SSIM score against the default psycho-visual weighting. Bitrate saving can go up to -5.21% for some sequence. Bitrate savings for the 'natural' test set are more modest but still significant.

Table 1: bitrate-saving (%) of RDSQ model with (3) vs (2)

BD-Rate (%)	**SSIM**			**MS-SSIM**		
	average	best	worst	**average**	best	worst
All	**-1.27%**	-5.21%	0.55%	**-2.24%**	-7.11%	-0.31%
Natural	**-0.36%**	-0.76%	0.06%	**-1.08%**	-2.07%	-0.31%
Game	**-1.65%**	-5.21%	0.55%	**-2.72%**	-7.11%	-0.67%

We have highlighted one possible optimization to address characteristic 1 of video game content. If we now focused on the observation point 2, still in the context of AQP algorithm, it is considered to increase the strength (i.e. importance) of the temporal distortion propagation into the RDSTQ model. This way, the model will be pushing more bits to reference frames/areas and leveraging on the beneficial copy/paste mechanism provided by motion compensation. To validate such assumption, we've made varying the strength parameter of the RDSTQ model for the two test sets, 'game' vs 'natural', and plot the relative bitrate saving Figure 4. The results tend to show an "optimal" average strength value of 2.5 for game content vs 2.0 for the natural content confirming the initial assumption. In practice, a more sophisticated version of the RDSTQ model is implemented in MediaKind's UnCL-HEVC codec which includes an auto-strength estimation model based on input content characteristics, e.g. expressed as a function of the average spatial psychovisual weighting (ψ) and motion coding complexity (C_{motion}) over the GOP.

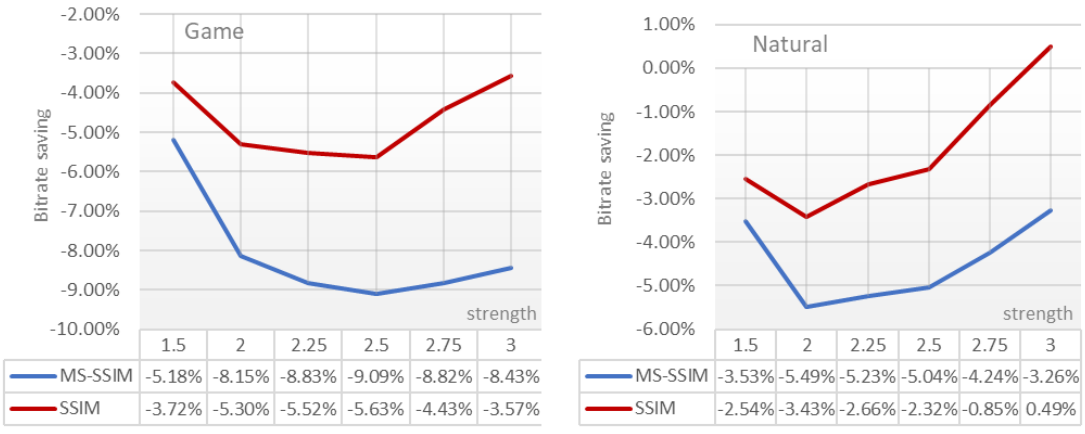

Figure 4 - Relative bitrate saving for various RDSTQ strength values vs a strength of 1.0

Adaptive and smoother in-loop filtering

Adaptive deblocking filter

H264/AVC, HEVC or VVC standards allow the use of in-loop adaptive deblocking filters to mitigate artifacts generated by discontinuities between transform edges after quantization. The filters use several local conditions and threshold parameters to determine if each edge is likely to be a natural edge, ideally to be left unfiltered, or the consequence of a transform discontinuity that should be filtered to mitigate blocking artifacts out of the quantization and reconstruction process. In addition to determine whether the deblocking filtering should be applied to the block boundary or not, the thresholding parameters control the use of normal or strong filter. Typically, in HEVC [8], two thresholding parameters β and t_C are specified (i.e. table known by the decoder) and dependent on the average QP value of two neighboring blocks with common block edge.

The deblocking parameters t_C and β provide adaptivity according to the QP and prediction type. However, different sequences or parts of the same sequence may have different characteristics. Conveniently, deblocking adjustment parameters can be sent in the slice header or picture parameters set (PPS) to control the amount of deblocking filtering applied. The corresponding parameters in HEVC are tc_offset_div2 and $beta\_offset\_div2$. These parameters specify the offsets (divided by two) that are added to the QP value before determining the β and t_C values.

We can then easily understand that deblocking parameter adjustment per frame is one key feature for encoding efficiency optimization, especially in the context of video game coding. Indeed, game content tends to have plain and smooth color/luminance gradient areas, very prone to blocking artifacts, where in-loop filtering is very important. It is also mixed with sharp and well-defined objects which should not be subject to strong filtering but tend to be easily identified by standard deblocking conditions.

For that purpose, we designed a frame-based deblocking parameter adjustment algorithm, able to determine offset values as specified by the standard. Coarsely, a negative offset reduces the filter impact, and a positive offset increases the filter effect.

The algorithm design considers the following aspects:

- the type of content where using negative offsets will be most effective, i.e. content that is quite easy to encode and is temporally consistent. Content with a lot of motion, and that is inconsistent, is likely to show blocking if we reduce the deblocking filter strength,
- Intra (I) and reference pictures are the most important frames for adapting the filter strength, as they form the basis of prediction for following frames,
- scene transitions should be treated with care. Blocking artefacts can be more visible following transitions. It is then sensible not decreasing the filter offsets too much during transitions.

Overall, the estimated frame offset values are defined as a function of the scene transition info, frame type, average motion compensated error and motion entropy for the frame. Performance of the proposed algorithm is summarized Table 2 for both test sets. Significant coding efficiency improvements can be observed for the two test sets, but average bitrate saving approximately doubles for video game content confirming the importance of such technique in the context of game encoding.

Table 2: bitrate saving (%) of proposed adjustment algorithm vs default deblocking filter

BD-Rate (%)	SSIM			MS-SSIM			PSNR		
	average	best	worst	**average**	best	worst	**average**	best	worst
All	**-2.90%**	-6.67%	0.16%	**-2.45%**	-6.02%	0.02%	**-0.55%**	-3.39%	0.32%
Natural	**-1.78%**	-4.59%	0.16%	**-1.59%**	-4.31%	0.02%	**-0.31%**	-0.92%	0.08%
Game	**-3.36%**	-6.67%	-0.38%	**-2.81%**	-6.02%	-0.42%	**-0.65%**	-3.39%	0.32%

Intra reference samples filtering

In HEVC or VVC, reference samples from spatial neighboring used for Intra prediction are conditionally filtered [9][10] to mitigate quantization error propagation, improving prediction and compression efficiency. In the context of HEVC standard version 1, only the use or not of strong reference sample smoothing (i.e. bi-linear interpolation filter) for 32x32 transform block size, can be controlled by the encoder and signaled at the sequence level (i.e. using SPS). It is designed to further reduce contouring artifacts caused by edges in the reference sample arrays, which can be more visible on 32x32 block size. The effect of disabling by default this strong intra reference sample smoothing is given Table 3. We can observe that on the two considered test set it seems beneficial disabling strong intra smoothing, particularly in the context of video game coding, with bitrate savings of up to -1.55%.

Table 3: bitrate saving (%) of disabling strong intra smoothing

BD-Rate (%)	SSIM			MS-SSIM			PSNR		
	average	best	worst	**average**	best	worst	**average**	best	worst
All	**-0.27%**	-0.94%	0.14%	**-0.34%**	-1.55%	0.10%	**-0.21%**	-0.93%	0.19%
Natural	**-0.19%**	-0.49%	0.04%	**-0.26%**	-0.35%	-0.11%	**-0.22%**	-0.32%	-0.07%
Game	**-0.31%**	-0.94%	0.14%	**-0.37%**	-1.55%	0.10%	**-0.20%**	-0.93%	0.19%

Use of 10-bit encoding accuracy with 8-bit signal input

10-bit internal bit-depth precision is optionally allowed for most of video standards/codecs (e.g. H264 High10, HEVC Main10, VVC Main, etc.) with the use of 4:2:0 chroma component sub-sampling format (as a subsampling format widely used for video streaming). It means that prediction, residual coding, and in-loop reconstruction are processed in 10-bit instead of 8-bit by default. Today, it is widely understood that even for 8-bit signal input there is a benefit in compression efficiency from processing signal with 10-bit accuracy. There are less truncation errors, especially in the motion compensation stage, increasing the efficiency of compression tools. It also helps to better mitigate quantization errors out of the compression process (in relation to noise shaping). For video game content, it visually improves the coding of plain areas with smooth colour or luminance gradient, reducing banding and/or blocking artifacts. Compression efficiency results for 10-bit encoding accuracy versus 8-bit is given Table 4. In the context of game coding, using 10-bit encode accuracy provides an average of -6.17% bitrate-saving for the same SSIM quality, and up to -15.43% and would be the de-facto recommended encoding format.

Table 4: bitrate saving (%) of using 10-bit encoding accuracy vs 8-bit

BD-Rate (%)	SSIM			MS-SSIM			PSNR		
	average	best	worst	**average**	best	worst	**average**	best	worst
All	**-5.11%**	-15.43%	-0.43%	**-1.91%**	-6.51%	0.56%	**-3.01%**	-12.07%	0.57%
Natural	**-2.56%**	-4.08%	-0.91%	**-0.40%**	-1.19%	0.31%	**-1.31%**	-2.51%	0.35%
Game	**-6.17%**	-15.43%	-0.43%	**-2.53%**	-6.51%	0.56%	**-3.72%**	-12.07%	0.57%

Adaptive motion vector resolution estimation and coding

Most video coding standards, including HEVC and VVC, make use of sub-pixel Motion Estimation (ME) with Motion Vectors (MV) at fractional precisions to achieve high compression ratios. Unfortunately, sub-pixel ME comes at very high computational costs due to the interpolation step and additional motion searches. There has been extensive research on reducing the complexity of subpel ME, with relevant works presented in [11], [12], [13], [14] and [15]. Most of the techniques formulate a subpel error surface with a mathematical model to directly determine the best sub-pixel MV cost and thus potentially reducing both search and interpolation complexities. Some other techniques such as [11][15] rely on global and local features, or coding statistics, of the content for conditionally skipping the subpel ME process on a block-basis. Those approaches that may be challenged in addressing a large variety of camera-captured video content, but could be particularly efficient in the context of computer-generated content such as video games.

In addition to ME computational complexity saving, HEVC – Screen Content Coding (SCC) extension or VVC can allow adaptive MV resolution coding and signaling at the slice and CU levels, respectively. It can provide further MV bit cost saving and compression efficiency as discussed in the next section.

SCREEN CONTENT RELATED TOOLS FOR VIDEO GAME CODING

HEVC through its SCC extension, and VVC have adopted coding tools to specifically compress Screen Content [16][17][18]. By screen content we refer to video containing a significant portion of rendered (moving or static) graphics, text, or animation rather than (or in addition to) camera-captured video scenes. Example applications include wireless displays, remote computer desktop access, and real-time screen sharing for videoconferencing. The motivation of this section is to assess the relevance and compression performances of the SCC-related tools in the context of video game coding, that has not been investigated in the literature. We first give a brief description of the main SCC tools available in both HEVC-SCC and VVC, then evaluate their respective coding efficiency based on their implementations into reference SW models. Given we focused on coding tools relevant for YUV 4:2:0 format only, the Adaptive Color Transform (ACT) tool, specifically designed for RGB 4:4:4 source coding is not covered here.

Coding tool overview

Intra Block Copy (IBC)

IBC is a block-based prediction technology whose mechanism is similar to inter-picture motion compensation. The essential difference lies in the fact that its reference samples are derived from inside the (reconstructed part of the) current picture. IBC was originally proposed during the standardization of H.264/AVC. It was later formally included in HEVC

SCC and then VVC (as well as in other recent standards such as AV1, EVC and AVS2). The IBC mode design in the HEVC SCC extension is implemented almost in the same way as the HEVC inter-picture motion compensation, using the same syntax structure and nearly the same decoding process. The current (partially) decoded picture before the in-loop filtering process (including deblocking and SAO) is also regarded as a reference picture, when the IBC mode is enabled for coding of the current picture. In this way, block-based motion compensation and block-based intra sample copy are unified. In HEVC SCC the reference sample region allowed for IBC includes every previously reconstructed Coding Tree Units (CTU) in raster-scan, with exception of the top-right regions of CTUs relatively to current CTU for parallel processing consideration (i.e. WPP). VVC has few differences in its IBC mode design. First, the reference range or region is constrained to a local area (i.e. samples from the left CTU and current CTU only) to mitigate HW memory bandwidth and implementation timing issues. As a second difference, IBC design is handling the new Dual Intra Tree Structure, where Luma and Chroma components are coded separately with a different tree structure. For such case, IBC mode is only allowed for Luma component. IBC is no longer considered as part of inter mode but an independent coding mode, having its own vector coding engine as compared with the motion vector coding schemes in VVC inter mode.

Palette Mode (PLT)

The motivation of palette mode coding for screen content comes from the observation that in local areas, computer generated content typically use a small number of colors to render the content. Thus, coding these small color sets directly can be more efficient than going through regular coding operations. The colors to represent a coding block are therefore referred to as color palette. Each sample in the block is converted into an index of one entry in the palette. A typical PLT mode consists of representing the color palette and coding the index map. A PLT coded block does not have any residues. A color palette can be either joint palette or separate palette. In the former case, a triplet—containing 1 luma value and two chroma values is used; for the later, the palette for luma is a single value and the one for two chroma components is a duplet. In HEVC SCC, the entries of palette for the current block (up to 64) are joint triplets and come from two sources: reusing the palette predictor (up to 128) and decoding from the bitstream. The palette mode in VVC is largely inherited from HEVC SCC with a few simplifications.

Transform Skip Mode (TSM)

Compared to the difference of the residue signal in camera captured contents, screen content residue signal tends to be sparse and of low magnitude. This characteristic may avoid the use of transforms for further decorrelating/compacting the residual signal. Therefore, for screen content, the option of skipping transform coding may provide good coding performance improvement as compared to always using transformed coefficient coding. In HEVC version 1, TSM is enabled only for 4x4 blocks. In HEVC SCC extension, the allowed TSM sizes were extended up to 32x32, (the maximum possible transform size). Besides, while the transform is skipped, the coding method of residue signals remains the same as the method designed for transform coefficients. In VVC, TSM is allowed and signaled for coding block sizes up to 32x32. The residue coding engine is modified to better fit the spatial residue distribution, in comparison to regular transform coefficient distribution, resulting in improved compression efficiency.

Block-based Differential Pulse-Code Modulation (BDPCM)

Intra prediction is by design usually less efficient in predicting sample more distant from the reference samples (i.e. top and left boundaries). Consequently, the residues of an intra predicted block may still possess directional patterns. To compensate for such inefficiency, further prediction is applied in VVC among the residue samples by using the BDPCM mode. With BDPCM mode, a flag is used for each block to choose the intra sample prediction and residue prediction from either the horizontal or vertical directions. Intra predicted residue samples are first quantized, with each quantized residue sample being further differentially predicted/coded from its neighbor along the horizontal or vertical direction. In BDPCM mode transform skipping is implicitly applied, and the same residue coding engine as used by TSM is applied. Finally, BDPCM can be turned on for luma and chroma components separately.

Adaptive Motion Vector Resolution (AMVR)

For camera-captured video, the movement of a real-world object is not necessarily exactly aligned to the sample positions in the camera's sensor. Motion compensation is therefore not limited to using integer sample positions (fractional motion compensation is used to improve compression efficiency). Computer-generated screen content video is however often generated with the knowledge of the sample positions, resulting in motion that is discrete or precisely aligned with sample positions in the picture. For this type of video, integer motion vectors may be sufficient to represent the motion. Bitrate savings can be achieved by not signaling the fractional portion of the motion vectors.

In HEVC-SCC, AMVR defines a slice-level flag to indicate that the current slice uses integer (full-pel) motion vectors for luma samples. If the flag is true, then the motion vector predictions, motion vector differences, and resulting motion vectors assume only integer values are allowed, savings the bits associated to the representation of the fractional values. In VVC, a CU-level AMVR scheme is introduced with finer-granularity in the MV resolution selection. In the nominal case, for each CU/block the MV can be coded and signaled in quarter-luma-sample, half-luma-sample, integer-luma-sample or four-luma-samples. In VVC, the AMVR scheme initially introduced for screen content has been generalized to camera-captured thanks to its local adaptivity.

It is important to note that computer-generated content such as video game, discrete or pixel-alignment motion precision will be very dependent of the rendering engine used (e.g. 2D vs 3D), and the subsequent techniques such as rasterization, fragment processing and shader operations, etc. as requested and performed by the GPU. Hence, as for camera-captured content, local adaptivity might be key for motion precision determination of game content.

Experimental results – compression efficiency

We evaluated coding performance of the previous SCC-related tools as implemented in HEVC-SCC and VVC reference models, using HM-16.21+SCM-8.8 [19] and VTM-16.0 [20], respectively. The following test conditions were used: All Intra (AI) and Random Access (RA) base configurations, constant QP mode comparison with 5 QP points (22, 27, 32, 37, 42), ~1s length encode (matching intra/key frame period).

For both HEVC and VVC evaluations, the references or anchors correspond to HM-16.21+SCM-8.8 and VTM-16.0 with all the SCC-specific tools turned on. Then, the coding efficiency impacts for all SCC tools off and each individual tool off against the anchors,

across the two test configurations and codecs, are summarized Table 5, Table 6 and Table 7. Data results show how much loss in compression efficiency we can get by disabling a given tool or tool set. Hence, a positive number reported in those tables stands for bitrate saving provided by the considered tool or tool set.

The analysis of the results Table 5 shows that SCC tools in the context of HEVC can provide significant improvements in coding efficiency of game content, with an average -6.5% (AI) and -3.5% (RA) bitrate savings for the same MS-SSIM scores and can go up to -14.6% (AI) and -9.6% (RA). As expected, coding efficiency improvements for natural content are more modest, in average -1.4% in AI and -0.7% in RA. In the context of VVC, SCC tools benefits for game coding are on average -3.7% (AI) and -1.76 (RA) BD-rate gain based on MS-SSIM, and up to -7.9% (AI) and 4.40% (RA). Note that in the case of VVC "all SCC-specific tools OFF" test configuration, the contribution of the AMVR is not included, i.e. it remained ON by default, as it is classified as a generic tool in the RA base configuration. However, its individual contribution for VVC is reported later on Table 7. From Table 6 and Table 7 we can conclude that IBC is the most efficient tool for coding video game whatever the codec HEVC-SCC or VVC, with average bitrate saving ranging from -3.5% to -4.5% in AI and from -1.5% and -2.4% in RA. Benefits from other tools such as PLT, TSM or BDPCM, without being negligible, are more modest. Regarding AMVR, the slice-based adaptive algorithm as implemented in HM-16.21+SCM-8.8 is apparently inefficient. In VVC, the finer AMVR design, based on CU-level adaptation and finer decision granularity, and as implemented in the VTM-16.0, provides -1.8% average bitrate saving whatever the content type, and up to -4.5% for game content.

Table 5: All SCC-specific tools OFF vs ON (anchor)

Codec / Config		**HEVC SCC**				**VVC**			
		AI		**RA**		**AI**		**RA**	
BD-Rate (%)		**Natural**	**Game**	**Natural**	**Game**	**Natural**	**Game**	**Natural**	**Game**
MS-SSIM	**average**	**1.39%**	**6.46%**	**0.71%**	**3.46%**	**0.88%**	**3.69%**	**0.36%**	**1.71%**
	min	-0.09%	0.77%	-0.82%	-0.38%	-0.24%	0.11%	-0.45%	-0.49%
	max	3.14%	14.64%	2.78%	9.60%	2.48%	7.93%	1.92%	4.40%
PSNR	**average**	**0.95%**	**7.19%**	**0.38%**	**3.74%**	**0.72%**	**3.54%**	**0.27%**	**1.45%**
	min	-0.20%	1.01%	-0.37%	0.64%	-0.16%	0.46%	-0.39%	0.07%
	max	2.18%	15.34%	1.17%	9.06%	1.88%	6.96%	1.51%	3.86%

Table 6: Individual HEVC SCC tool OFF vs all tools ON (anchor)

Configuration		**AI**			**RA**			
Avg BD-Rate (%)		**IBC**	**PLT**	**TSM**	**IBC**	**PLT**	**TSM**	**AMVR**
Game	**MS-SSIM**	4.64%	0.68%	0.17%	2.39%	0.51%	0.40%	0.00%
	PSNR	3.44%	2.27%	0.28%	1.65%	1.11%	0.59%	0.00%
Natural	**MS-SSIM**	1.74%	-0.03%	-0.10%	1.09%	-0.01%	-0.10%	0.00%
	PSNR	1.33%	-0.01%	-0.03%	0.68%	-0.05%	-0.01%	0.00%

Table 7: Individual VVC SCC tool OFF vs all tools ON (anchor)

Configuration		AI				RA				
Avg BD-Rate (%)		IBC	PLT	TSM	BDPCM	IBC	PLT	TSM	BDPCM	AMVR
Game	MS-SSIM	3.44%	0.09%	0.68%	0.14%	1.49%	0.11%	0.06%	0.07%	1.75%
	PSNR	2.72%	0.29%	1.56%	0.22%	1.04%	0.11%	1.55%	0.11%	1.41%
Natural	MS-SSIM	0.98%	-0.07%	-0.08%	-0.08%	0.42%	-0.11%	-0.30%	-0.12%	1.76%
	PSNR	0.80%	-0.05%	0.04%	-0.04%	0.30%	-0.06%	0.01%	-0.02%	1.43%

TOWARDS 'GAME-CONTENT AWARE' ENCODING TECHNOLOGY

In [21], we proposed an AI driven method to optimize the video encoder configuration according to the input signal characteristics, specifically designed for real-time encoding applications. This content aware encoding method monitors the CPU usage and seamlessly adjusts the encoder configurations according to the available resources. If the input content requires less computational resources, more encoding tools can be enabled to maximize compression efficiency. If the content complexity increases computational requirements, some encoding tools may be disabled or restricted to guarantee real-time constraints can be fulfilled.

This approach guarantees compression efficiency and computational resource usage are maximized but requires ranking each encoding tool and encoding parameter relative to its compression efficiency to computational complexity ratios. Since the impact of each tool is dependent on the input video characteristics, AI is used to estimate the impact and cost of each tool based on the input image characteristics on real-time, so that the encoding tool or parameter that provides the largest compression efficiency gains with a minimum increase in computational complexity for that specific content are prioritize if resources are underutilized, while the ones that provide the lowest ratios are prioritized to be disabled if required.

To validate this content aware approach for video game content, we compared it to a reference case where encoding tools are dynamically enabled in a fixed order, independent of the input content and defined to maximize the compression efficiency on natural content. Both configurations were run in the same computational resources and only standard encoding tools were included in this evaluation (the specific screen content tools mentioned in the previous sections are out of scope for this evaluation). It was observed that the content aware approach achieved compression gains of 7.4% in H.264 and 6.9% in HEVC (average BD-SSIM), by simply prioritizing encoding tools with a larger impact while encoding video game content.

Overall, it was observed that the motion estimation strategy has a very significant impact on compression efficiency for video game content, while the motion refinement itself has a lower impact than for natural content. Similarly, picture partitioning and mode decision strategy have a higher weight on the compression efficiency of video game content, with the compression efficiency being somewhat less dependent on the GOP sequencing and transition management.

CONCLUSION

Motivated by the growing traffic of live video game streaming, and related esport market, we investigate in this work the possible encoding optimizations and normative tool set to best compress and stream game content. We first identify the main distinguished signal characteristics of game content in comparison to natural content. From this knowledge we share some practical examples of relevant encoding strategies into an optimized HEVC encoder that provide significant improvements in compression efficiency or encode run-time/CPU cycle saving in the context of video game coding. The suggested optimization techniques are ranging from bitrate trading and adaptive quantization within a GOP, frame-adaptative deblocking and smoother in-loop reference sample filtering, the use of 10-bit encoding accuracy to adaptive motion precision estimation and fast subpel ME. All combined, they show potential for about 10% average BD-rate gain in compression efficiency. Additionally, we review normative SCC-related tools, as adopted in HEVC SCC extension and VVC, discuss their design, and assess their compression performance for game content based on reference SW models. This specific tool set provides substantial bitrate saving for the same MS-SSIM or PSNR score with the main interesting features being the IBC and the AMVR. Finally, leveraging on AI-based content-aware method, we report that compression efficiency can be improved by 7% in H.264 or HEVC optimized implementations by simply prioritizing encoding tools with a larger impact while encoding video game content. Overall, we highlight and demonstrate a strong basis of complementary techniques for significantly improving the coding of video game in practical video streaming solution.

REFERENCES

[1]. A. Pennington, 2021, The State of Live Streaming in 2021, Streaming Media online at: https://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=146325

[2]. C. Settimi, 2020, The NBA's Coronavirus Shutdown Led The Phoenix Suns Into A Virtual Arena, Forbes online at: https://www.forbes.com/sites/christinasettimi/2020/03/20/the-nbas-coronavirus-shutdown-led-the-phoenix-suns-into-a-virtual-arena-when-they-drew-3-million-fans-the-sports-world-took-note/?sh=5fd5b7ff74b8

[3]. SRO motorsports group, 2021, online at: https://www.sro-motorsports.com/news/53/sro-esports-launches-global-championships-for-sim-drivers-and-teams

[4]. N. Barman, S. Zadtootaghajy, S. Schmidty, M. G. Martini, S. Moller, 2018, GamingVideoSET: A Dataset for Gaming Video Streaming Applications, Proceedings of 2018 16th Annual Workshop on Network and Systems Support for Games (NetGames), June 2018.

[5]. G. Bjøntegaard, 2001, Calculation of average PSNR differences between RD-curves, Technical Report, VCEG-M33, ITU-T SG16/Q6, 2001,

[6]. M. Ropert, J. Le Tanou, M. Blestel, 2021, Mastering Quantization is key for Video Compression, Proceedings of 2021 International Broadcasting Convention (IBC), December 2021.

[7]. Bichon, M. Le Tanou, J. Ropert, M. Hamidouche, W. and Morin, L. 2019. Optimal Adaptive Quantization based on Temporal Distortion Propagation model for HEVC, IEEE

Transactions on Image Processing (TIP). Vol. 28, Issue 11, pp. 5419 to 5434, November 2019.

[8]. A. Norkin et al. 2012, HEVC Deblocking Filter, IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), Vol. 22, No. 12, December 2012

[9]. J. Lainema et al. 2012, Intra Coding of the HEVC Standard, IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), Volume: 22, Issue: 12, Dec. 2012.

[10]. J. Pfaff et al. 2021, Intra Prediction and Mode Coding in VVC, IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), Volume: 31, Issue: 10, Oct. 2021.

[11]. S.G. Blasi, I. Zupancic, E. Izquierdo, Adaptive precision motion estimation for HEVC coding, in: Proc. 31th Picture Coding Symposium, PCS, Cairns, Australia, 2015, pp. 144–148.

[12]. W. Dai, O.C. Au, W. Zhu, W. Hu, P. Wan, J. Li, A robust interpolation-free approach for sub-pixel accuracy motion estimation, in: Proc. IEEE International Conference on Image Processing, ICIP, Melbourne, Australia, 2013, pp. 1767–1771.

[13]. Y. Li, Z. Liu, X. Ji, D. Wang, HEVC Fast FME algorithm using IME RD-costs based error surface fitting scheme, in: Proc. Visual Communications and Image Processing, VCIP, Chengdu, China, 2016.

[14]. Q. Zhang, Y. Dai and C.-C. Kuo, "Fast sub-pel motion vector prediction via block classification", Image Processing (ICIP) 2009 16th IEEE International Conference on, pp. 1569-1572, Nov 2009.

[15]. Sang-hyoPark, 2019, A sub-pixel motion estimation skipping method for fast HEVC encoding, ICT Express, Volume 5, Issue 2, June 2019, Pages 136-140

[16]. J. Xu, R. Joshi; R. A. Cohen, 2015, Overview of the Emerging HEVC Screen Content Coding Extension, IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), Volume: 26, Issue: 1, Jan. 2016.

[17]. T. Nguyen et al., 2021, Overview of the Screen Content Support in VVC: Applications, Coding Tools, and Performance, IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), Volume: 31, Issue: 10, Oct. 2021.

[18]. X. Xu, S. Liu, 2022, Overview of Screen Content Coding in Recently Developed Video Coding Standards, IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), Volume: 32, Issue: 2, Feb. 2022.

[19]. HM+SCM reference software for HEVC SCC extension, tag HM-16.21+SCM-8.8, online at https://vcgit.hhi.fraunhofer.de/jvet/HM/-/tree/HM-16.21+SCM-8.8

[20]. VTM reference software for VVC, tag VTM-16, online at https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/VTM-16.0

[21]. N. Francisco, J. Le Tanou. 2022. Optimizing real-time video encoders with ML, in Proceedings of the 1st ACM Mile-High Video Conference (MHV '22), March 2022.