UHD Panoramic Video Coding for Multi-Camera and Multi-Processor Acquisition Systems

Joao Duarte

Instituto de Telecomunicações
Leiria, Portugal
Email: joaoerduarte@gmail.com

Joao Carreira and Pedro Assuncao

Instituto de Telecomunicações
Politécnico de Leiria / ESTG
Leiria, Portugal
Email: {jcarreira, amado}@co.it.pt

***Abstract*—This paper addresses the problem of encoding Ultra-High Definition (UHD) panoramic video in multi-camera and multi-processor systems, where a wide Field of View (FoV) is captured by multiple cameras, each one corresponding to a small FoV. The UHD panoramic images are then formed by stitching high-definition images captured by the individual cameras, covering a wide FoV around the acquisition viewpoint. A simulation study is carried out to evaluate the rate-distortion-complexity performance of multi-encoder systems using independent processors for encoding sub-images with small FoV, as part of a wider panoramic UHD resolution. This study uses an image partitioning scheme to distribute the UHD images with wide FoV over various processors and evaluates the rate-distortion-complexity performance of HEVC encoding in comparison with classic encoding of a single frame per processor. The results show that the rate-distortion performance of multi-processor systems is quite similar to single processor ones, which allows to distribute the huge computational requirements of HEVC encoding across several low-cost processors.**

Keywords–Panoramic video Coding; Multi-processor encoding systems; Rate-distortion-complexity.

I. INTRODUCTION

The increasing use of Ultra-High Definition (UHD) video, such as 4k and 8k resolutions, drives significant research efforts to develop efficient coding systems, not only to improve the rate-distortion performance, but also to cope with the demand of huge computational resources [1]. The latest encoding standard, High-Efficiency Video Codec (HEVC), also known as H.265, is currently the most adequate to encode UHD video [2]. However, the better compression efficiency in comparison with its predecessor H.264/AVC is achieved at the cost of much higher encoding time due to the heavy computational requirements. This poses limitations on the acquisition and coding of UHD video using low-cost equipment with reduced computational resources. In the case of UHD panoramic video, the technical challenges arise from the huge amount of data required to represent visual information with wide Field of View (FoV). Thus, acquisition, processing and coding of UHD panoramic video has been driving research efforts [3][4]. However, despite the existence of fast hardware to deal with the high resolutions of UHD panoramic video, the lack of low-complexity acquisition and encoding systems still limit the development of new applications for non-professional consumers. One of such low-complexity systems is described in [5], where nine low-power processing units are used to capture a panoramic image.

Current acquisition and encoding systems for UHD panoramic video can be found in both consumer and professional markets with quite different characteristics. On the one hand, cheap systems available for consumers using either one or two cameras with ultra wide-angle fisheye lenses, suffer from limited resolutions and optical distortions. On the other hand, professional equipment include several high quality cameras, each one capturing high resolution images with limited FoV. To use the best characteristics of both type of systems requires multiple cheap cameras and processors for capturing and encoding the whole FoV in a distributed manner by using low-cost processors.

This paper presents a contribution for the development of such systems by investigating the rate-distortion-complexity performance of a multi-processing system to encode UHD video using multiple independent processors to encode part of the panoramic visual data, i.e., a narrow FoV. A simulation study is carried out to evaluate whether the encoding performance achieved by distributing narrow FoVs across several encoders/processors is comparable with a single high-end encoder with only one processor for the full-FoV UHD panoramic images.

This paper is organized as follows. Next section presents an overview of related work. Section III describes the simulation study procedure and Section IV presents the results along with their discussion. Finally, Section V concludes the paper.

II. RELATED WORK

Recent improvements in the coding efficiency of predictive algorithms, when compared to the previous standards, are mostly achieved at the expense of a great increase in computational complexity [1]. For instance, the HEVC is very efficient at compressing video data, but it requires significantly more processing power when compared to the previous standards [6], making it difficult to deal with very high resolution video, such as UHD panoramic video. A possible approach to deal with such high computational power requirements is to use parallel processing, where the input video data is partitioned and independently encoded by multiple processors. There are different methods that are able to accomplish this goal, as explained below.

At high-levels of the video data hierarchy, one can use parallel processing on the basis of Groups of Pictures (GOP). In this approach, the input video is divided in temporal segments, each one assigned to a different GOP independently encoded by a different processor [7]. Although this

 ISBN: 978-1-61208-736-8

is simple to implement, the overall latency of such coding process is non-negligible and does not allow to achieve real time communications [8]. Memory limitation may also pose problems because typical caches have insufficient storage space for multiple frames. To achieve a more fine control, parallelism can be defined at the frame-level, where several frames, in the same GOP, are encoded at the same time. However, such an approach imposes constraints to the temporal dependency between frames, which significantly reduces the motion estimation efficiency. Moreover, synchronisation between processing threads is required to guarantee that all prerequisites for motion estimation are encoded.

Alternatively, each frame can be divided into several slices to be processed in parallel. As each slice is independent from each other, it is straightforward to process multiple slices in parallel, without inter-process communication, except for motion compensation prediction. Although this is a simple approach, it incurs in substantial coding overhead due to the higher number of slice headers, and reduction of causal neighbours for prediction, due to lack of predictions across slice boundaries. In order to reduce the slice overhead, one can use tile partitioning. Then, each tile can be processed independently as defined in the HEVC standard [9]. Although the use of tiles is similar to slices, tiles are able to achieve efficiency frame partitioning for parallel processing with lower overhead [10].

Finally, each row of Coding-Tree Units (CTU) can be processed independently using the Wavefront Parallel Processing (WPP) mechanism proposed in the HEVC standard [11]. Contrary to slice and tile boundaries, no dependence is broken at each row boundary so the rate-distortion penalty is small when compared to other methods, as the context of the arithmetic coder is propagated between rows. However, to maintain the context, a delay of one CTU has to be introduced in each row. In this approach, the number of threads does not affect the coding efficiency, but the requirements of inter-process communication substantially increases.

Even though HEVC already has some parallel processing mechanisms to deal with the problem of high computational complexity, as mentioned before, they are often not enough, specially when using hardware with quite limited resources. Moreover, the techniques introduced in HEVC for parallel processing dependent on inter-processor communication and are not suitable for independent processing by different cores or processing units.

The idea of using multi-processor units was also investigated for system-on-chip [12] and in the case of multi-view video coding [13]. However, the relative performance of image data splitting into sub-images corresponding to a narrow FoV of a panoramic video remains mostly unknown. Thus, this study addresses the impact on rate-distortion-complexity performance of using multiple independent encoding processors, each one covering a limited FoV captured by independent cameras. This study fo Specifically, this work studies multiple schemes with FoVs of different sizes under different coding parameters and compares the impact on the coding efficiency. By evaluating how the coding performance varies with the video signal characteristics (e.g., spatial and temporal complexity), the amount of processing units and the size of each FoV, one can easily design efficient video acquisition and encoding systems based on multi-camera and multi-processor architectures.

TABLE I. TEST SEQUENCES USED IN THE EXPERIMENTS.

Sequence	SI	TI	Description
Beauty	10.6	8.35	Very high spatial details in some regions (hair) and flat background
Bosphorus	13.4	3.75	Boat shipping at low motion with moderate complex background
HoneyBee	8.24	2.54	High spatial detail, with one low motion object
Jockey	11.5	16.2	High motion with one horse rider
ReadySteadyGo	18.0	19.0	Very high motion with several horse riders

III. SIMULATION STUDY PROCEDURE

In the simulation study, the open source implementation of the HEVC encoder x265 was used [14].

The goal was to evaluate the rate-distortion-complexity of a multi-processor system, where each processor runs an independent encoder for sub-images of the full-FoV UHD resolution, i.e., partial FoV corresponding to a vertical stripe of the original image. The performance is evaluated in comparison with a conventional system using a single processor running only one encoder for the full-FoV UHD resolution. To make a fair comparison, the same video sequence is encoded in both systems. The small FoV sub-images captured by independent cameras are simulated by splitting the original images into multiple vertical stripes of equal size.

The five UHD video sequences presented in Table I were used in the experiments. These test sequences have 4k spatial resolution, i.e., 3840×2160 pixels, and were selected as they are commonly used for UHD HEVC evaluation tests and are public available [15]. As shown in Table I, the test sequences have different types of motion and texture complexity, demonstrated by the measures of spatial information (SI) and temporal information (TI), which follow the definitions given in [16]. These high resolution video sequences are used to simulate a wide FoV. Table II defines the six different sub-image splitting modes used in this study along with the corresponding spatial resolution of the resulting FoV. Figure 1 shows an example for the partition into six narrow FoVs.

After splitting the UHD images, each video sequence corresponding to either a full or limited FoV were encoded using five different native presets of the x265 encoder: 0-ultrafast, 3-fast, 5-medium, 7-very slow and 9-placebo, which have direct impact on rate-distortion and encoding time. These presets define various control variables within the encoding process such as maximum and minimum coding unit (CU) size, maximum consecutive B-frames, number of frames for lookahead

TABLE II. FOV PARTITIONS AND THEIR SPATIAL RESOLUTION.

Number of partitions	Spatial resolution of the FoV
One	3840×2160
Six	640×2160
Eight	480×2160
Ten	384×2160
Twelve	320×2160
Fifteen	256×2160

 ISBN: 978-1-61208-736-8

Figure 1. Example of a full-FoV UHD video frame (top) and its corresponding partitioning into 6 FoVs (bottom).

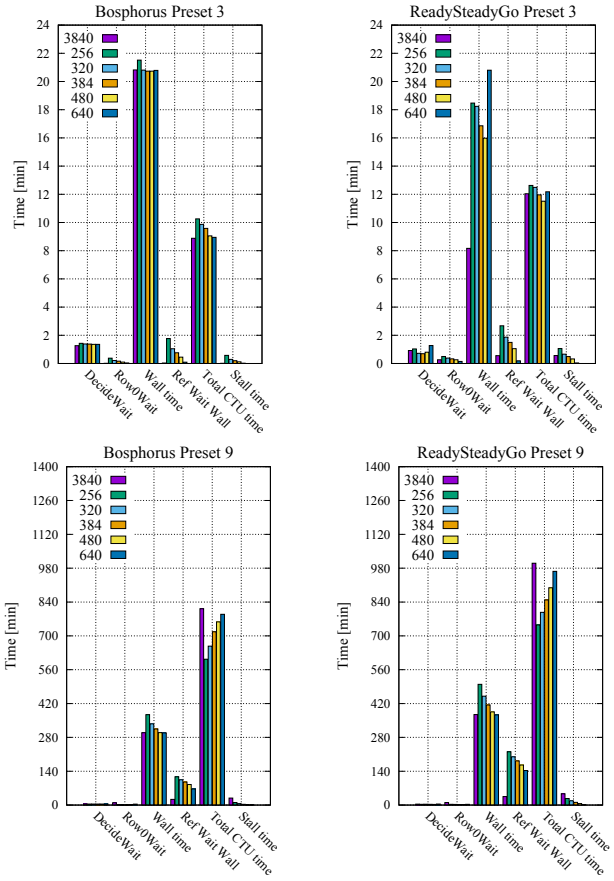

Figure 2. Encoding times for the for CRF 21 and different preset and FoV splitting.

slice-type decision, motion search algorithm, motion range and merge mode configuration [17]. For each preset, the Constant Rate Factor (CRF), which is used to control the Quantisation Parameter (QP) was also configured to the following values: 11, 16, 21 and 26 (lower CRF results in higher quality). Finally, each reduced FoV sequence is encoded 20 times to obtain valid average results for encoding times. Considering all possible encoding configurations and sequences, a total of 1200 encoding runs were performed in this simulation study.

From the output produced by the x265 software, the following time-related variables were extracted for each condition:

- *DecideWait*: time that the encoder waits since the previous frame was retrieved by the API thread, before a new frame is given for encoding. This is the latency introduced by slice-type decisions (lookahead).
- *Row0Wait*: time that the encoder has to wait since it receives a frame to encode until its first row of CTUs is allowed to start compression. This is the latency introduced by reference frames being reconstructed and making filtered rows available.
- *Wall time*: difference between when the first CTU is ready to be compressed and the entire frame is output to the coded stream.
- *Ref Wait Wall*:difference between when the first and the last reference row become available.
- *Total CTU time*: the total time spent by working threads in compression and filtering operations of the CTUs of a given frame.
- *Stall Time*: the total time spent with zero working threads, i.e, no compression operation was performed.

To evaluate the quality of the encoded frames and sub-frames (smaller FoV) the Peak Signal-to-Noise Ratio (PSNR) is used, both at the frame and video-level. The coded frame size was also taken into consideration, as shown in the next section.

IV. Results and Analysis

In this section, the results of the simulation study are presented and discussed in detail. This analysis is organised in three parts: (i) analysis of the encoding times, (ii) analysis of compression results, i.e., coded frame size, versus encoding times and (iii) overall performance evaluation using rate-distortion results.

A. Evaluation of encoding time

The objective of these experiments is to evaluate whether the processing time (i.e., computational complexity) required by single encoding of full-FoV UHD panoramic video is equal to overall multi-encoding time of several sub-video sequences, each one representing a smaller FoV of the same full-FoV UHD panoramic video. To this aim, the results obtained for the time-related variables described in the previous section are compared for different presets and CRFs

The first comparison is between the average encoding times of each frame. For the sub-video the encoding times of each FoV were added together in order to directly compare to the full-FoV video. Figure 2 contains the results of sequence

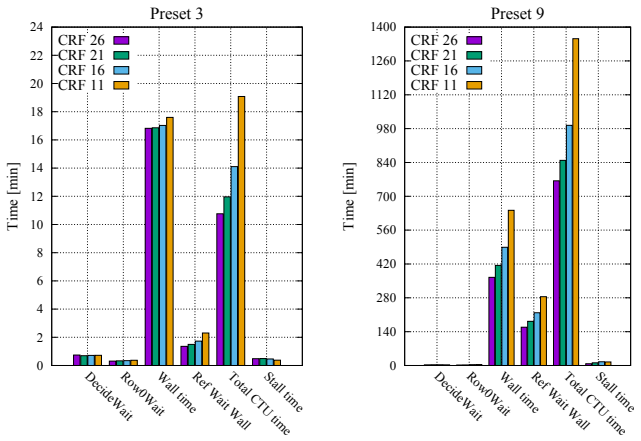

Figure 3. Encoding times for ReadySteadyGo sequence using 10 FoVs.

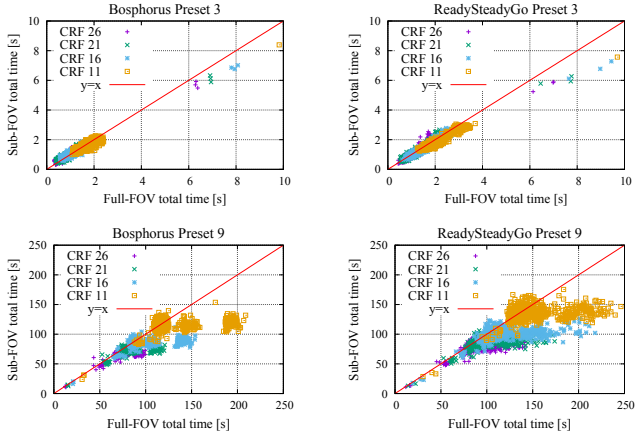

Figure 4. Total encoding time of each frame for full-FoV versus the limited sub-video encoding with 10 FoVs for the Preset 3 and 9.

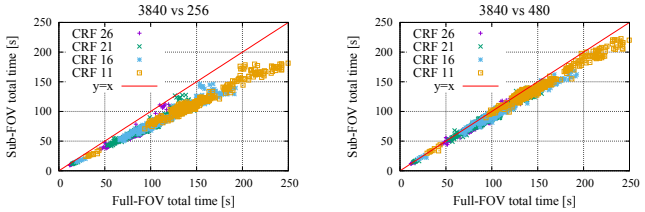

Figure 5. Total encoding time of each frame for full-FoV versus the narrow FoV encoding for the ReadySteadyGo sequence and the Preset 9.

ReadySteadyGo, which has a very high motion and Bosphorus, which has much lower motion (see TI in Table I), for the Presets 3 and 9 and CRF 21. Results reveal that the overall measured times add up to similar values of the single encoding of full-FoV. This can be seen by similar value of the *Wall time* which represents the time difference occurred during the frame encoding process.

Comparing results from both sequences in Figure 2, on the one hand the sequence with higher motion (i.e., ReadySteadyGo) shows higher variations in the encoding times, specially for encoding the whole frame with variations up to 700 minutes (50% increase). On the other hand, the results from Bosphorus are roughly similar with variations up to 7 minutes (25% increase). This indicates that in panoramic video with low motion content, the wide FoV can be divided into smaller FoV captured from several cameras and encoded across multiple processors with lower computational resources. In this case the overall processing requirements can be distributed across multiple processors without increasing the total encoding time. In the case of higher motion video, the encoding time variations show that the overall encoding time of multiple small FoV sequences is greater than encoding the same sequence with a single encoder for the full-FoV. This means that the encoding time of small FoV sequences cannot be estimated from the total encoding time of full-FoV sequences by simply dividing the full-FoV encoding time by the FoV splitting factor of the acquisition system. This also indicates that TI (Table I) can be a useful parameter to include in a processing time estimation model of split video.

The results of Figure 2 that in the Preset 9 (slower preset) the *Total CTU time* is higher than the *Wall time*. One should note that the former corresponds to the processing time and the latter to the elapsed time. Therefore, these results reveal that in slower presets the encoder takes more advantage of the parallel processing features, incurring in higher processing time in a short time period. Figure 3 shows the encoding times for different CRFs and fixed presets of 3 and 9. In this case a FoV width of 384 pixels was used, corresponding to a partitioning factor of 10. Results confirm the expected behaviour that the processing time increases with the increase in quality, however this is less noticeable for faster presets (e.g., Preset 3).

In order to show the relation between the encoding time of full-FoV against multiple smaller FoV, Figure 4 shows a scatter plot of all results from different sequences. The horizontal axis corresponds to the encoding time of full-FoV while the vertical axis represents multiple smaller FoVs using 10 partitions (FoV resolution: 384×2160 pixels). The sum of the FoV times is directly compared with the full-FoV video. Moreover, a diagonal line corresponding to $y = x$ is also represented to indicate the threshold from which the sub-FoV videos spend higher amount of encoding time than the reference case with full-FoV. These results reveal that quite linear correlation exists between the two encoding times, which is more noticeable for Preset 3 (faster than Preset 9). This indicates that the encoding times are not significantly affected by the video partitioning into sub-images. Moreover, it is also noticeable a faster preset can achieve a processing time reduction of approximately 25 times.

In the results of Figures 4 the points are most often bellow the diagonal line (i.e., $y = x$) revealing that the overall time produced by the sum of sub-image videos is slightly lower than the single image video. This trend can be observed in most tests using faster presets. However, when using slower presets the linear relationship is no longer observed. The reason is that the coding order of I, P and B slices for faster presets is somehow fixed and does not deviate much from a certain pattern, while in higher presets this predictable pattern does not exist, making this direct frame-by-frame comparison not fully valid.

To overcome such limitation, another set of tests were made with fixed encoding slice order. The results are shown in

 ISBN: 978-1-61208-736-8

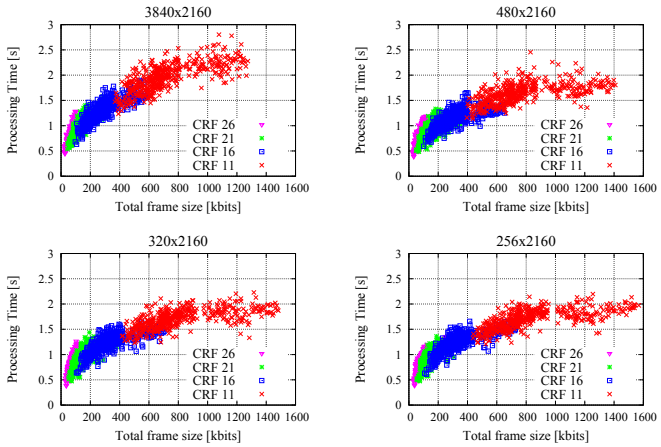

Figure 6. Relation between the processing time and the total frame size for different FoV partitioning sizes and Preset 3.

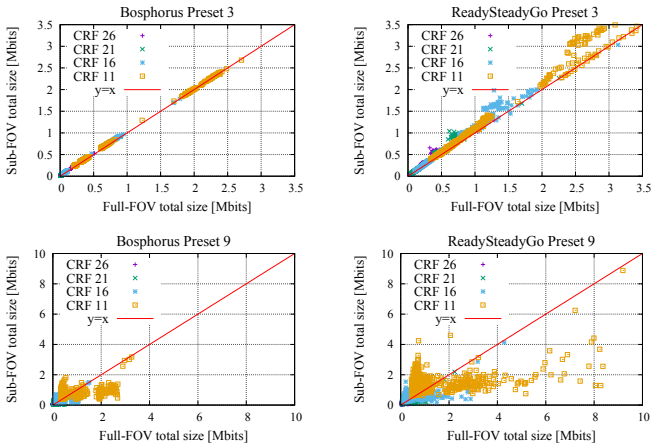

Figure 7. Total frame size for full-FoV versus the limited sub-video encoding with 10 FoVs for the Preset 3 and 9.

Figure 5, where one can observe that the linear correlation is now restored. There is also a pattern that can be seen in these results, indicating that for higher number of FoV partitions a lower total processing time is required to encode the full video. This pattern is also more noticeable in slower presets. This is due to the fact that each slice is independently encoded from the others, therefore, less processing time is consumed with prediction methods, resulting in smaller encoding times.

B. Coded frame size evaluation

1) Frame size versus encoding time: The relationship between the size of coded frames and corresponding encoding time was also evaluated. To this aim, tests were made in order to find whether there was a relation between the size of the compressed frames and their encoding times. Since the coded I-,P- and B-Slices are not consistent between them, only the results for B-Slices are analysed.

Figure 6 shows the relation between the encoding time and the size of the coded frame in the case of the ReadySteadyGo sequence. The results correspond to the Preset 3 and each CRF is illustrated with a different colour. These results correspond to both the full-FoV coding and sub-video coding (i.e., smaller FoV). Results reveal that video frames which require higher amount of coded bits normally take longer time to be processed. This is due to the fact that more coding modes are tested until an efficient rate-distortion trade-off is achieved. Moreover, these results confirm that decreasing the CRF (i.e., lowering the QP) leads to higher encoding time. The linear regression of the results shown in Figure 6, show a solid linear relation with the number of bits from each frame, with R^2 varying between 0.80 and 0.99 for all presets and CRFs combinations.

2) Coded frame size comparison: Moreover, the direct comparison of coded frame size between the full-FoV video and multiple FoV videos was analysed. As before, to compare with the full-FoV case the sum of the coded frame size of each sub-video was used. Figure 7 shows the comparison between the size of each full-FoV frame with the sum of the twelve 320×2160 FoV partitions, all encoded with Preset 3 and CRF 16. The diagonal line ($y = x$) is also shown. Results show that for a faster configuration, i.e., Preset 3, the overall

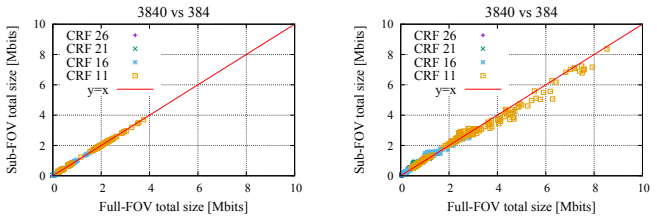

Figure 8. Total frame size for full-FoV versus the narrow FoV encoding for the ReadySteadyGo sequence and Preset 9 for different CRF.

size produced by the sum of all multiple FoV videos is slightly higher than the full-FoV video, as most points are above the diagonal line. However, in the case of Preset 9 there is not a clear relation between the total size of bitstreams, thus it is not possible to accurately determine which approach leads to smaller bitstream.

In regard to Figure 4, one can see that for Preset 9 there is not a linear relationship between the size of coded frames when using full and sub-image encoding. This is due the slice coding order selected by the encoder. Moreover, as shown in previous results, fixing the slice coding order results in a linear relation between bitstream size in both cases. This is shown in the results of Figure 8.

C. Coding efficiency evaluation

1) Rate-distortion analysis: The coding efficiency is evaluated by comparing the average PSNR and bitrate, for different FoV sizes. Figure 9 shows the quality obtained for different bitrates, by varying the CRF. Results in this figure reveal that for the same bitrate, increasing the number of FoV partitions results in lower average video quality, which is more noticeable for the sequence with higher motion (i.e., ReadySteadyGo sequence). However, for a sequence with lower motion the quality decreasing is less significant, revealing that this encoding approach does not have great impact in the overall performance. Moreover, one can notice that by decreasing the number of FoV partitions the coding efficiency increase, revealing that a trade-off between coding efficiency and number

 ISBN: 978-1-61208-736-8

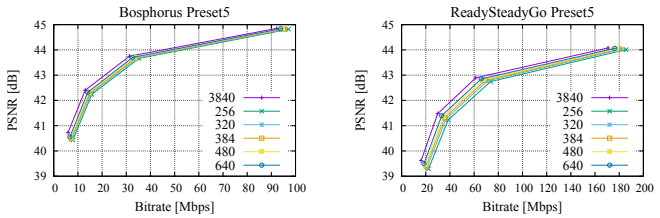

Figure 9. Rate-distortion results for the preset 5 and different FoV partition sizes – horizontal axis: bitrate (kbits/s); vertical axis: average PSNR (dB).

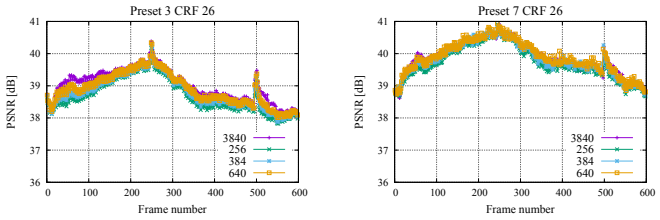

Figure 10. Average frame-by-frame quality for the ReadySeadyGo sequence and different FoV partition sizes.

of processing cores can be obtained for different application requirements.

2) Quality comparison: Finally, the overall quality obtained for the full-FoV and sub-image video sequences is evaluated using the PSNR metric. Figure 10 shows the PSNR results for each frame using different FoV partitions with presets 3 and 7. The results show that for the same preset and CRF the overall quality does not significantly change between partitions. Although there is not a clear best option when comparing the full-FoV with the average of all sub-videos. Such overall behaviour is similar for all the tests carried out in these simulations.

V. Conclusion

In this work, the performance of encoding UHD panoramic video as several sub-sequences of smaller FoVs was evaluated for multi-encoding systems with multiple independent processors. The simulation study was based on splitting the full-FoV UHD panoramic scene into various smaller FoVs and encode each of them in a single encoding processor. The rate-distortion performance, as well as the computational complexity was evaluated in comparison with conventional coding of a single frame with full-FoV per processor. The results show that the rate-distortion performance of multi-processor systems is quite similar to single processor ones, which allows to distribute the huge computational requirements of HEVC encoding across several low-cost processors. Therefore, this simulation study provides relevant insights on future research directions and allow efficient development of UHD panoramic video acquisition and coding systems using multiple cameras and processors with reduced computational resources. The results are particularly useful in the design of wide FoV multi-camera rigs, such as those used to capture 360-degree video. Overall the paper demonstrates that the smaller FoV captured by each camera can be independently encoded using low-cost processors and then sent to central unit for further processing, without incurring additional loss of performance in comparison with a single encoder system.

Acknowledgements

This work was funded by Instituto de Telecomunicações through Fundação para a Ciencia e Tecnologia, Portugal through the projects: ACode180Video UID/EEA/50008/2019 and ARoundVision PTDC/EEI-COM/30652/2017 (SAICT-45-2017-POCI-01-0145-FEDER-030652)

References

[1] G. Correa, P. Assuncao, L. Agostini, and L. S. da Cruz, "Performance and computational complexity assessment of high-efficiency video encoders," IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, Dec. 2012, pp. 1899–1909.

[2] ISO/IEC JTC1 ITU-T, "High efficiency video coding, ITU-T rec. H.265 and ISO/IEC 23008-2," ITU-T/ISO, Standard, Feb. 2018.

[3] R. Skupin, Y. Sanchez, Y. . Wang, M. M. Hannuksela, J. Boyce, and M. Wien, "Standardization status of 360 degree video coding and delivery," in IEEE Visual Communications and Image Processing (VCIP), Dec. 2017, pp. 1–4.

[4] A. Mazumdar, T. Moreau, S. Kim, M. Cowan, A. Alaghi, L. Ceze, M. Oskin, and V. Sathe, "Exploring computation-communication tradeoffs in camera systems," in IEEE International Symposium on Workload Characterization (IISWC), Oct. 2017, pp. 177–186.

[5] J. Duarte and P. Assuncao, "Low-complexity acquisition of 180-degree panoramic video using early data reduction," in IEEE International Conference on Smart Technologies (EUROCON 2019), Jul. 2019, pp. 1–4.

[6] C. Herglotz, D. Springer, and A. Kaup, "Modeling the energy consumption of HEVC P- and B-frame decoding," in IEEE International Conference on Image Processing (ICIP), Oct. 2014, pp. 3661–3665.

[7] A. Gürhanlı, C. Chung, P. Chen, and S.-H. Hung, "Coarse grain parallelization of H.264 video decoder and memory bottleneck in multi-core architectures," International Journal of Computer Theory and Engineering, vol. 3, no. 12, Jan. 2011, pp. 375–381.

[8] H. Migallón, J. Hernández-Losada, G. Cebrián-Márquez, P. Piñol, J. Martínez, O. López-Granado, and M. Malumbres, "Synchronous and asynchronous HEVC parallel encoder versions based on a GOP approach," Advances in Engineering Software, vol. 101, 2016, pp. 37–49.

[9] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou, "An overview of tiles in HEVC," IEEE Journal of Selected Topics in Signal Processing, vol. 7, no. 6, Dec. 2013, pp. 969–977.

[10] R. Rodrguez-Snchez and E. S. Quintana-Ort, "Tiles-and WPP-based HEVC decoding on asymmetric multi-core processors," in IEEE Third International Conference on Multimedia Big Data (BigMM), Apr. 2017, pp. 299–302.

[11] K. Chen, J. Sun, Y. Duan, and Z. Guo, "A novel wavefront-based high parallel solution for HEVC encoding," IEEE Transactions on Circuits and Systems for Video Technology, vol. 26, no. 1, Jan. 2016, pp. 181–194.

[12] H. K. Zrida, A. Jemai, A. C. Ammari, and M. Abid, "High level H.264/AVC video encoder parallelization for multiprocessor implementation," in Design, Automation Test in Europe Conference Exhibition, Apr. 2009, pp. 940–945.

[13] C. G. Gurler, A. Aksay, G. B. Akar, and A. M. Tekalp, "Multithreaded architectures and benchmark tests for real-time multi-view video decoding," in IEEE International Conference on Multimedia and Expo (ICME), Jun. 2009, pp. 237–240.

[14] x265 HEVC Encoder / H.265 Video Codec. [Online]. Available: http://x265.org (Retrieved: July 17, 2019)

[15] Ultra Video Group test sequences. [Online]. Available: http://ultravideo.cs.tut.fi/ (Retrieved: July 17, 2019)

[16] ITU-T, "Recommendation P.910, Subjective video quality assessment methods for multimedia applications."

[17] x265 HEVC encoder preset. [Online]. Available: https://x265.readthedocs.io/en/default/presets.html (Retrieved: July 17, 2019)

 ISBN: 978-1-61208-736-8