The Optimal Real-Time Video Encoding Scheme for Multi-Camera Panoramic System Using NVENC

Petr Kriz, Jiri Prinosil, Martin Buchta

***Abstract*—Today, modern, powerful, and relatively available computer technology allows the proposal of real-time camera systems working with several video streams. There is also a permanent tendency to increase image resolution, what can be useful for example in some image processing methods used in surveillance systems. Given this, an important part of a multi-camera real-time system proposal is optimal and efficient video compression. The modern encoding standards support 4K or 8K video content. One of these standards is H.264 or HEVC (High Efficiency Video Coding) and the popular available encoding technology using GPU (Graphics Processing Unit) acceleration is NVENC (Nvidia Encoding) provided by Nvidia. Many papers deal with the comparison of encoding standards in terms of quality or bitrate savings on testing video sequences [1], [2]. However, this paper is aimed primarily at a comparison of several practical implementations of NVENC technology using different presets of H.264 and HEVC standards. The experiments are mainly focused on testing the encoder's real-time ability to deal with single video input obtained from the industrial camera using different settings. To do that, some system performance parameters for the continuous monitoring during the NVENC encoding process were chosen. Results in tables I and II are compared, discussed and the best NVENC implementation with encoding scheme is chosen in the context of our briefly introduced novel panoramic system. Finally, the system is fully loaded by the 7 cameras, all streams are encoded using an optimal encoding scheme and a performance limit is established for our hardware configuration.**

***Index Terms*—NVENC, real-time encoding, H.264, HEVC, panoramic system, NVENC hardware performance analysis**

I. Introduction

Fast and quality video compression plays a crucial role in the implementation of real-time camera systems. In general, the requirements for video encoders are to provide the highest data rate in the shortest possible time while providing the best possible data compression with the best image quality achieved in the decoding process. Probably one of the most used standards in these days are H.264 (also known as MPEG-4 Part 10: Advanced Video Coding) and more efficient HEVC (also known as H.265 or MPEG-H part 2). Major improvements of HEVC over the H.264 are described in [3]. Authors in [4] shown that the HEVC is providing average bit rate saving up to 41 % compared to H.264.

Of course, there are other modern and recently finalized more effective standards such as versatile video coding VVC (H.266/MPEG-I part 3), essential video coding EVC (MPEG-5), or popular AV1 (AOMedia Video 1). Some papers are exploring these video codec's ability to process especially 4K/2160p content. Authors in [1] tested all these modern standards in terms of Bjntegaard-delta bitrate (BD-BR) peak-signal-to-noise ratio, bitrate savings, and encoding computational complexity. Considering the comparison of the two most popular standards H.264 and HEVC, work [2] provides results of testing to show that the HEVC design seems to be effective for low bit rates, high-resolution video content, and low-delay communication applications.

Concerning the implementation of fast encoding in real-time multi-camera systems, we were looking for available solid hardware-accelerated solutions. We can mention an encoding approach of Comprimato company using its unique proprietary software JPEG2000 Codec SDK using CUDA (Compute Unified Device Architecture) [5] to accelerate the encoding process. It uses jpeg2000 compression based on wavelet transform and theoretically, it should be able to deliver about 63 FPS of 4K video content (8 bit) using our GPU Quadro P2000. In contrast, NVIDIA provides an open-source software VideoCodec SDK using specialized hardware NVENC as part of some NVIDIA GPUs. This technology now implements the encoding standards H.264 and HEVC up to 8K video resolution.

One of the current similar multi-camera systems using NVENC encoding is the Bagadus architecture presented in [6] uses NVENC encoding. The Bagadus system works with five 1080p cameras, after the frame acquisition during the real-time stitching process the 4450×2000 px panoramic image is created and NVENC H.264 is used to encode personalized virtual view. The authors also show some experimental results from encoding 690 frames of the 1080p sequence (run on Intel Core i7-2600 and NVIDIA Quadro K2000 GPU) when their configuration and settings (H.264 – Low latency preset) achieves up to 133 FPS.

Another panoramic system [7] uses the X264 encoder module which receives raw planar YUV422 panoramic images (up to 4400×1800 px) from the stitcher module. Video encoding is performed using the libx264 library with ultrafast preset implemented on CPU (Central Processing Unit).

We decided to use NVENC technology in our system. The real performance of NVENC technology using specific hardware settings is the main objective of this work, which is structured as follows. Section II briefly describes the architecture of our multi-camera system, used hardware, the proposed

P. Kriz, J. Prinosil and M. Buchta are with the Faculty of Electrical Engineering and Communication, Brno University of Technology, Brno, Czech Republic (contacts: petr.kriz4@vutbr.cz, prinosil@feec.vutbr.cz). Manuscript received June 30, 2021

application, and the scenario for NVENC testing. Further, experimental results focused on computational efficiency are summarized in section III. Finally, the results are discussed in the conclusion section.

II. System settings and description

As depicted in the Fig. 1 our panoramic system consists of four units connected to the local network. The software running on the units is marked as blue (the code is written in C++ because of the fast algorithm implementation is required) or orange (the code is written in .NET) rectangle. The first unit we call the Encoding Server with the Multi-Camera Rig. This unit serves as a storage and system for the distribution of compressed video data which is acquired from industrial cameras. The Encoding Server provides on-demand particular source camera streams used by other units. The Video Analysis Server automatically processes the video content to detect potential events such as smoke or fire, and to track selected objects (person tracking). The central database containing settings of the panoramic system is running on the Control Server. The Control Workstation is intended for usage by the user. The main purpose of this unit is to take streams and find transformation matrices, compose and present a final virtual view called a virtual panoramic camera. The Control Workstation is described in the detail in our precedent work [8].

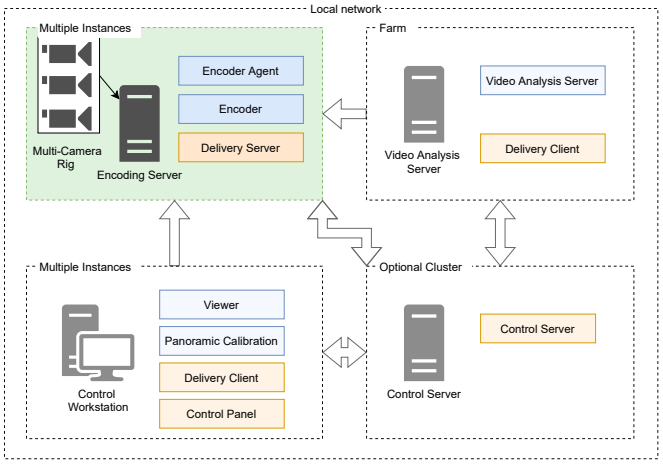

Fig. 1. Multiple camera panoramic system deployment.

A. Hardware for NVENC testing

According to Fig. 2 we can describe all main elements in our hardware chain together with the measured real baud rate. The Multi-Camera Rig consists of several industrial mvBlueFOX3 cameras with 3856×2764 px image resolution up to 7.3 FPS (Frames Per Second). In the "Highest speed" camera preset the transferred data achieves 77 MB/s per camera at maximal FPS. Cameras are connected to USB 3.0 four-port hub and then the data go to the Encoding Server using active optical cable for long-distance (up to 100 meters) connection. Experimentally, it was inspected that the maximal real throughput of 360 – 380 MB/s is possible to achieve with VIA USB 3.0 controller via PCIe x8. All tests run on the computer with the following configuration

- CPU AMD Ryzen Threadripper 1920X 12-Core 3.50 GHz,
- RAM 48 GB, SSD 250 GB,
- GPU Nvidia Quadro P2000,
- OS Windows 10 Home.

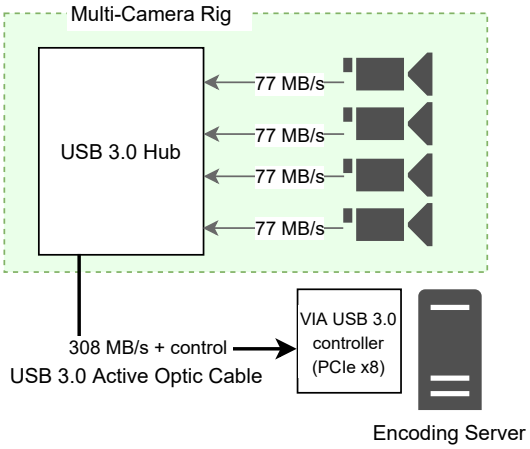

Fig. 2. Hardware connection with real baud rate.

B. Application for NVENC testing

The final implementation of NVENC encoding in the panoramic system is realized directly as a part of the Encoder code (see Fig. 1), but to make our experimental process easy to handle we proposed the control application with GUI (Graphical User Interface, see Fig. 2) for NVENC testing purpose only. The program is divided into several logical parts. As depicted in the Fig. 3 the main control GUI application is used to set key encoding parameters:

- Number of active cameras – choose the number of streams to be encoded,
- Cuda/DirectX/FFmpeg executable path – full path to the encoder (executable file) depends on NVENC implementation,
- API – selection of the concrete implementation of NVENC (Cuda, DirectX, FFmpeg),
- Codec – standard H.264 or HEVC,
- Preset – choice of the four codec presets (High quality, Low latency – High quality, High performance, Low latency – High performance),
- QP – setting of the encoding quality parameter (1 – 50),
- Segment length – length of encoded video segment in seconds,
- FPS – camera FPS in acquisition process,
- Write data to CSV file – saving all measured performance data to a CSV file to effectively evaluate results.

After the encoding setting is completed and the start button is pressed, the control application starts one of the three

processes (executable files) depends on selection. Cuda and DirectX executables are based on VideoCodec SDK 11 (released on Oct 14, 2020), inspired by public samples provided by Nvidia (AppEncCuda and AppEncD3D11 solutions). Except for Cuda and DirectX, the samples also show how to use NVENC with OpenGL. OpenGL interoperability is now supported on Linux machines only so this was excluded from testing since our system is based on Windows. In addition to C++ standard libraries, we used Matrix Vision SDK, WINAPI, and NVENC API to implement multiple-camera acquisition and video encoding. Furthermore, because the interaction between NVENC and FFmpeg is fully supported, the FFmpeg was installed (released on May 5, 2021) and its executable file was incorporated into the control application. Since the encoding process is in progress, all the system performance data related to the current encoding process are monitored in a separate thread of the application. Performance data is periodically acquired every 700 ms. When the encoding is finished, results are saved into a CSV file.

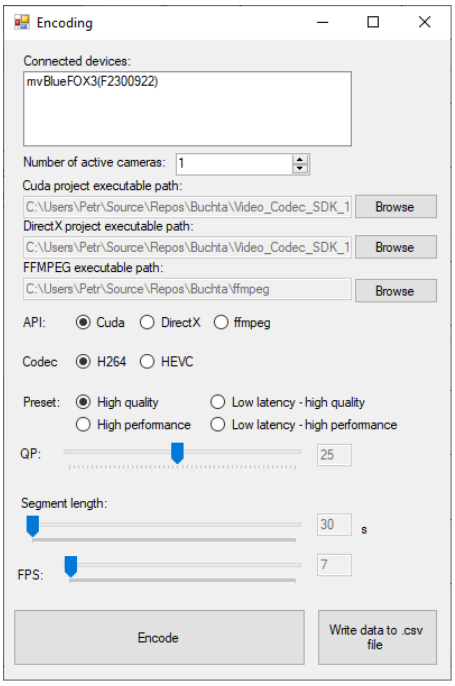

Fig. 3. Application for NVENC testing.

C. Testing scenario

The goal of NVENC testing is to find the most suitable encoding scheme with proper settings for our real-time implementation based on NVENC. There are two steps in this process. First, the optimal coding scheme is found based on a comparison of performance results. Second, the optimal coding scheme is set and the system is fully loaded by numbers of 4K cameras to reveal system performance limit. Selected system performance parameters for the monitoring are

- GPU memory [MB] – dedicated GPU memory used by the encoding process,
- GPU utilization [%] – usage of 3D/Cuda engine during the duration of the encoding process,
- CPU utilization [%] – usage of CPU during the duration of the encoding process,
- RAM [MB] – RAM memory used by the encoding process,
- Video Engine utilization [%] – usage of Video Encode Engine used by NVENC technology during the duration of the encoding process,
- Computational time [ms/frame] – the computational time of the algorithm consists of loading frame data to device memory and frame encoding (available for Cuda and DirectX NVENC implementation only),
- Encoded segment size [MB] – final video segment size to compare codec effectivity.

To ensure the most objective and repeatable testing, the scenario is as follows. In the beginning, only one camera is connected to the Encoding Server. The camera is placed in front of the PC monitor. In this PC the video called ”Rain in the forrest“ is played in a loop. It consists of a scene with static hut, moving leaves and trees, and heavy rain. This video content is captured by the camera, transferred, and encoded in real-time. During the encoding process in the first step of testing the position of the camera and lighting conditions are stable.

Because there are many possible combinations of encoder settings, in the first step of testing we set the following parameters as constant. QP was experimentally established and set to a value of 25. Length of each video segment is 30 s with 7 FPS. Instead of all four presets it was chosen only two of them.

1) *High quality* preset – VBR (Variable Bit Rate), 1x MultiPass, Tunning Info: High Quality, since version 10.0 the preset for HEVC and Pascal architecture is named as P6 (P5 for H.264).
2) *Low latency-high performance* preset – CBR (Constant Bit Rate), no MultiPass, Tunning Info: Low Latency, since version 10.0 the preset for HEVC and Pascal architecture is named as P4 (P2 for H.264).

Measurements are organized into two separate tables. In addition, we enclose also some qualitative results based on encoding standards and QP settings.

III. Results

All the performance measurements are organized and statistically evaluated in Tab. I and Tab. II.

A. Searching the optimal coding scheme

In the evaluation process to search for the optimal coding scheme for our system, we focus on some aspect of performance results. First, comparing the corresponding HEVC and H.264 formats, we can say that the HEVC standard is better or not significantly worse than H.264 (except for some categories such as Video Engine utilization or DirectX GPU utilization where the usage in HEVC can be significantly higher). But these observations may be also affected by a large

TABLE I
NVENC TESTING - H.264 CODEC RESULTS.

H.264		NVENC implementation					
		Cuda		DirectX		FFmpeg	
Encoding preset	Measurement	Average	Standard deviation	Average	Standard deviation	Average	Standard deviation
High quality	GPU memory [MB]	1057	0	**945**	0	1611	0
	GPU utilization [%]	3.4	0.8	3.4	1.1	**2.5**	0.8
	CPU utilization [%]	8.2	0.5	8.2	0.5	**4.9**	0.6
	RAM [MB]	**358.6**	21.4	434.4	15.7	1781	13
	Video Engine utilization [%]	**12**	17.7	17	21.8	17.5	22.1
	Computational time [ms]	9	0.8	**6.8**	1.7	-	-
	Encoded segment size [MB]	**49.7**	0.5	56	0.7	187.4	6.6
Low latency - HP	GPU memory [MB]	884	0	**754**	0	1144	0
	GPU utilization [%]	3	0.2	6.6	4.5	**2.6**	0.7
	CPU utilization [%]	8.2	0.5	8.1	0.6	**4.9**	0.5
	RAM [MB]	310.9	17	**307.2**	22.3	668.6	10.9
	Video Engine utilization [%]	**11.7**	3.3	14.9	6	17.5	4.8
	Computational time [ms]	8.9	0.4	**6.6**	1.4	-	-
	Encoded segment size [MB]	**106.3**	0.7	109.3	1.6	414.2	0.9

TABLE III
HEVC VS H.264 PSNR MEASUREMENTS DEPENDING ON THE QP.

QP	PSNR (HEVC) [dB]	PSNR (H.264) [dB]
1	**39.05**	37.12
22	**35.17**	33.72
50	**32.82**	32.80

standard deviation and due to the rapid increase and decrease in performance. For the next decision-making, we compare measurements in Tab. II only.

There are many ways how to find the best suitable coding scheme for particular system configuration. Because the Encoding Server is mainly limited by GPU Quadro P2000 (memory up to 5 GB) we focus on GPU memory utilization and overall performance balance. Multiple coded stream segments are transferred via network, so the smallest final segment size is another important factor. In this case, the output segment size for both presets in Cuda and DirectX is almost identical. As can be seen in Tab. III, it was also confirmed that standard HEVC achieves better results in compression quality compared to H.264. We can comment on another interesting observation in the results. Generally, the measured part of the DirectX algorithm (computational time) is always faster than in Cuda implementation (about 2 - 3 ms). DirectX also uses the lowest amount of GPU memory, but with higher GPU utilization. The Cuda implementation has the best overall performance balance.

FFmpeg beats other implementations in CPU utilization (about 3 % less usage) - this can be caused by better optimization than our implementations, or by the usage of DirectShow instead of Matrix Vision SDK to control the camera. On the other hand, the size of an encoded segment is with the same preset significantly higher. Because of the high GPU memory usage, segment size, and the lack of possibilities of customization, we do not use FFmpeg build for final implementation.

After results analysis we consider to use as most suitable coding scheme for our system

- NVENC implementation: DirectX; Standard: HEVC; Preset: Low latency - high performance; Advantages: the lowest amount of GPU memory usage, good encoded segment size

B. Reaching the system performance limit

According to Fig. 2, two modules of the Multi-Camera Rig were connected into the Encoding Server. So the maximal number of connected cameras (hardware limitation) can be 8 in one Encoding Server unit. In the control application, we set the encoding scheme chosen in the previous step with the same parameters as in the last testing. To achieve a maximal camera frame rate of 7 FPS we were able to connect 7 cameras to the Encoder Server. With this configuration, our system can encode about 49 frames (4K resolution) per second in a real-time acquisition process and consumes the following hardware capacity

- GPU memory [MB]: 1610 ± 1.8,
- GPU utilization [%]: 26.7 ± 19,
- CPU utilization [%]: $\mathbf{70.4} \pm \mathbf{22}$,
- RAM [MB]: 3467.8 ± 835.7,
- Video Engine utilization [%]: 28.2 ± 21.8.

As can be seen, after we ensure the low GPU memory usage by choosing DirectX implementation, the usage of CPU has a

TABLE II
NVENC TESTING - HEVC CODEC RESULTS.

HEVC		NVENC implementation					
		Cuda		DirectX		FFmpeg	
Encoding preset	Measurement	Average	Standard deviation	Average	Standard deviation	Average	Standard deviation
High quality	GPU memory [MB]	606	0	**476**	0	634	0
	GPU utilization [%]	3.1	0.5	6.3	3.8	**2.6**	0.9
	CPU utilization [%]	8.1	0.5	8.3	0.7	**5**	0.6
	RAM [MB]	353.3	21.7	**343.9**	18.6	631.1	13.7
	Video Engine utilization [%]	**25.8**	6.5	37	8	27.5	6.6
	Computational time [ms]	8.8	0.2	**6.7**	0.2	-	-
	Encoded segment size [MB]	**43**	0.2	44.3	0.6	274.9	7.8
Low latency - HP	GPU memory [MB]	607	0.3	**476**	0	632	0
	GPU utilization [%]	3.5	0.8	15.5	5	**2.6**	0.9
	CPU utilization [%]	8	0.4	8.1	0.3	**4.8**	0.4
	RAM [MB]	353.4	23.3	**341**	13.9	630	13
	Video Engine utilization [%]	**18.2**	4.7	27.1	6.9	22.9	5.8
	Computational time [ms]	9.4	1	**6**	0.4	-	-
	Encoded segment size [MB]	**44.3**	0.3	44.9	0.1	268.3	8.3

major impact on overall system performance. The enormous CPU usage is mainly caused by high parallelism. Each camera has two autonomous threads (one for data acquisition and one for the encoding process). Threads are not synchronized to each other to reduce the peaks in power performance. The cameras were connected gradually, and it was found that the CPU usage was increased with each camera by approximately 10 %. Furthermore, it was inspected that the simultaneous usage of 8 cameras is limited by a setting of 5 FPS to achieve a stable encoding process.

IV. CONCLUSION

In the previous sections, we introduced the proposed architecture of a multiple-camera panoramic system in the context of testing and using NVENC technology. The Control application for testing and three different implementations (Cuda, DirectX, and FFmpeg) were described. According to the chosen scenario, the performance measurements were acquired and visualized in Tab. I, II and III. The best average values in each row of the tables were highlighted, all the results were analyzed and arguments on how and why to choose the most suitable coding scheme based on DirectX (HEVC standard) were presented.

Finally, the system was fully loaded and tested with 7 industrial cameras. The current system configuration with chosen coding scheme was able to simultaneously acquire and encode 7×7 frames per second. Related to performance results, it is obvious that the system is limited mainly by the CPU power (70.4 ± 22 for AMD Ryzen Threadripper 1920X 12-Core 3.50 GHz).

The main benefit of this paper is providing a deep analysis of popular NVENC technology in real conditions. This analysis is focused mainly on hardware performance and using real-time high-resolution frame acquisition and encoding in the context of application in the multiple camera system.

Based on mentioned observations, future work will be focused mainly on overall system optimization to achieve better encoding results.

ACKNOWLEDGMENT

This work was supported by the Ministry of the Interior under Grant VI20172020105.

REFERENCES

[1] D. Grois et al., "Performance Comparison of Emerging EVC and VVC Video Coding Standards with HEVC and AV1, " in *SMPTE Motion Imaging Journal*, vol. 130, no. 4, pp. 1–12, May 2021, doi: 10.5594/JMI.2021.3065442.

[2] J. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards Including High Efficiency Video Coding (HEVC)," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669–1684, Dec. 2012, doi: 10.1109/TCSVT.2012.2221192.

[3] M. T. Pourazad, C. Doutre, M. Azimi and P. Nasiopoulos, "HEVC: The New Gold Standard for Video Compression: How Does HEVC Compare with H.264/AVC?," in *IEEE Consumer Electronics Magazine*, vol. 1, no. 3, pp. 36–46, July 2012, doi: 10.1109/MCE.2012.2192754.

[4] A. Banitalebi-Dehkordi, M. Azimi, M. T. Pourazad and P. Nasiopoulos, "Compression of high dynamic range video using the HEVC and H.264/AVC standards," in *10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, 2014, pp. 8–12, doi: 10.1109/QSHINE.2014.6928652.

[5] https://comprimato.com/

[6] H. K. Stensland, V. R. Gaddam, M. Tenne, E. Helgedagsrud, M. Nss, H. K. Alstad, A. Mortensen, R. Langseth, S. Ljdal, . Landsverk, C. Griwodz, P. Halvorsen, M. Stenhaug, D. Johansen, "Bagadus: An Integrated Real-time System for Soccer Analytics," in *ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP)*. January, 2014.

[7] R. Langseth, “Implementation of a Distributed Real-time Video Panorama Pipeline for Creating High Quality Virtual Views.”Masters Thesis, University of Oslo, Department of Informatics, 2014.
[8] P. Kriz, J. Prinosil, K. Riha and M. K. Dutta, “Proposed Methods for Real-Time Visualization of Panoramic Stadium Tribune Images in High Resolution, ” in *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2019, pp. 1–5, doi: 10.1109/ICUMT48472.2019.8970920.