CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Performance Analysis of Large-scale Real-time Video Stream Configurations

Bachelor of Science Thesis in Software Engineering and Management

ERIK LAURIN
JOACIM EBERLÉN

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Supervisor: Christian Berger
Examiner: Richard Berntsson Svensson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Performance Analysis of Large-scale Real-time Video Stream Configurations

Erik Laurin
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
erikgustavlaurin@gmail.com

Joacim Eberlén
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
jeberlen@gmail.se

***Abstract*—The rapid technology development has resulted in sensors able to deliver very high-quality output. The high quality calls for efficient compression algorithms to handle the vast amount of data produced. This study aims to find a compression algorithm that delivers video streams of the highest possible quality given the constraints real-time processing using the User Datagram Protocol (UDP). This paper describes the experimental approach created to find such compression algorithm. Machine learning in the form of Bayesian optimization was applied to evaluate and hence deduce the optimal encoder parameters for each encoder and resolution in scope.**

WebM Project's VP9 implementation proved to be the most optimal encoder in scope for all resolutions evaluated in the experiment but the highest (QXGA - 2048x1536). For video streams in QXGA, VP9 hardware accelerated by Intel's QuickSync was found to perform best.

***Index Terms*—Video coding, Encoding, Codecs, H.264, VP9, QSV, Intel QuickSync, Real-time encoding, SSIM, Autonomous driving, Bayesian Optimization**

I. Introduction

A. Background

Sensors are becoming increasingly advanced, producing more data, due to higher quality outputs. As the output quality is improved, systems relying on the sensors' outputs are able to make more precise calculations. To take advantage of the increased sensor quality, the systems must also be able to cope with the enormous amount of data produced. Vehicles equipped with sensing devices for autonomous driving are examples of such systems with a data production of up to 1,000 megabytes per second [1].

Lossy video compression is an area where much research and development are invested in. This is primarily because of video content that is typically streamed over networks. Video streaming is expected to continue growing over the coming years [2]. Lossy video compression algorithms are typically optimized for content used for entertainment purposes such as web-based or TV-boxes-video streaming services. However, there are other areas where some degree of quality reduction can be tolerated, in exchange for less amount of data. One case is within the automotive domain where disk space and bandwidth are finite and expensive. The requirements and constraints differ between the entertainment and the automotive domains. Therefore the most optimal lossy video compression algorithm strategy cannot simply be transferred. A compression strategy streamlined for the automotive domain is highly desirable, especially in the light of the upcoming 5G communication standard that is said to become an essential resource for autonomous driving.

Numerous codecs exist on the market. H.264 is an established codec found as hardware implementation in modern smartphones and computer chips. H.264 is still a dominating codec in the web world. However, newer implementations found in competing H.265 and VP9 bring newer technologies, which pave the way for better performance.

B. Problem Domain & Motivation

The sensing device set-up in a modern car can produce enormous amounts of data. The data translates to major bandwidth and, depending on application, disk space consumption. In the automotive domain, self-driving algorithms and Advanced Driver Assistance Systems (ADAS) often rely on camera sensors as part of their input. Hence, self-driving algorithms and ADAS are likely to benefit from increased video quality [3]. The ongoing development of both safe algorithms in general, but especially AI systems, requires a lot of data for training, testing, and validating. High-performance sensor output, such as a more detailed camera feed, is therefore highly relevant.

A consensus of which lossy video compression algorithm is most useful, in terms of specific application, such as web-based video-streaming for entertainment, can be deduced from existing research. However, obtaining a broader understanding of which algorithm fits an entire domain most optimally, is harder. This is especially true for the automotive domain as the requirements are different to the entertainment domain where most existing research has been done. While quality and size are important in both domains, encoding time is of high significance for the automotive domain.

C. Research Goal & Research Questions

This paper strives towards elucidating the performance of lossy video compression algorithms for the use with self-driving vehicle algorithms such as object- or lane-detection. We set out to deduce the most suitable video compression

Fig. 1: Revere's Volvo XC90 equipment with a stereo configuration of high-performance Basler cameras

algorithm for a set of different pre-recorded lossless driving scenario video sequences (our datasets). The algorithms were validated on datasets from Revere as well as on the acknowledged KITTI dataset from the Karlsruhe Institute of Technology for assessing our results' transferability [4].

Different architectural approaches exist for creating sophisticated systems that support the enormous amount of data an autonomous car produces and distributes. A Volvo XC90 from Revere's test vehicle fleet, equipped with a stereo configuration of high-performance Basler cameras capable of capturing video frames losslessly, was used for data collection to create our dataset (see Fig. 1) [5]. The test vehicle was using UDP in its communication middleware used for relaying messages for the microservices that constitute the vehicleâĂŹs autonomous software system. The UDP has a packet limitation and thus the test vehicle had an inherited limitation of that each frame could not exceed the protocol specific 65507 bytes.

In addition, the encoding time could at most take 40 microseconds. 40 microseconds corresponds to a frame rate of 25 frames per second (FPS). This constraint was set to ensure a smooth motion for human perception of video.

Objective video quality can be assessed in numerous ways. From simpler Video Quality assessment Metrics (VQM) such as Mean-Square-Error (MSE) to more sophisticated metrics such as Structural Similarity (SSIM). The mentioned VQMs are of full reference type meaning they are used when both the uncompressed and the compressed data are available to calculate the quality difference.

SSIM provides a good trade-off between complexity and accuracy and has consequently become a broadly recognized VQM by both academic researchers and industrial practitioners [6], [7]. Therefore, SSIM was used to assess the objective quality of the compressed video sequences in the study.

RQ-1 Which video compression algorithm performs the best in terms of SSIM given typical driving situations for inter-urban and urban driving situations in daylight with good weather while aiming at a maximum frame size of not more than 65507 bytes to allow for network broadcast at a frame rate of not less than 25FPS?

RQ-2 How transferable are the resulting parameters for the various video compression algorithms when they are compared to a different video dataset, for instance from KITTI?

D. Contributions

The experiment proved the value of using different encoders and encoder configurations dependant on video stream and encoding machine. Our results shows that, out of the encoders in scope (see List 2, 3), VP9 provides a higher SSIM value on lower resolutions, but is incapable to encode the highest resolution (QXGA) unless hardware accelerated. Combined with the results, the authors also provide graphs comparing the encoder performance based on resolution, encoder and relative SSIM results on a H.264 baseline per dataset.

The study's transferability was evaluated by comparing the results from the datasets collected by Revere Laboratory to the KITTI dataset. As can be seen in section IV-B, the best compression algorithm was the same regardless of which dataset the video feed originated from.

In parallel with knowledge acquisition, a software artifact was developed using modern Software Engineering practices. The software artifact developed, called Coordinator Script (CS), can run on Unix-like platforms. New datasets can be added by simply including them in the 'datasets' folder and thanks to CS's high modularity, new encoders can easily be added to extend the experiment's scope [8].

E. Scope

The goal of our work was to analyze which compression strategy, in terms of codec, type of acceleration as well as codec configuration, provides the best objective quality results, while meeting certain constraints relevant to the field of autonomous driving. The work was solely geared towards fields where quality is important, but encoding time and size are of primary concern. Consequently, our scope and thus the applicability of this study is systems where the compression must be close to 'real-time' and the resulting size from the compression limited.

F. Structure of the Article

The paper is structured as follows: Section II provides a brief literature review of published related work. Section III outlines our methodology to collect, process and evaluate the data obtained in the study. Section IV presents the results obtained in the study. Section V discusses the results of our study and how it contributes to research. In Section V-D, we discuss how the threat to validity may compromise the study. Our work is concluded in Section VI where our conclusion and ideas about future works are presented.

II. Related Works

To understand the ins-and-outs of video encoding, the authors gathered knowledge through their thesis supervisor,

shorter articles and forum posts. Using the basic understanding of video codecs and VQMs, research papers were found, mostly through Google Scholar and earlier research studies conducted in the Chalmers Revere Laboratory.

A. Quality Metric Choice

Multiple VQMs are freely available. We previously discussed two widely used VQMs namely PSNR and SSIM [9]. According to Dosselmann and Yang, SSIM produces much more reliable result for lossy images than PSNR is able to do. Dosselmann and Yang made a comparison between SSIM and Mean-Squared Error (MSE). The formula for PSNR is directly correlated to the MSE value of an image [10]. Therefore, we have a strong argument for choosing SSIM over PSNR as our video quality metric. The aforementioned paper compares the two metrics statistically and using algebraic functions [10]. While this applies to the human visual system, and this study was realized in the field of autonomous driving using machine vision, there were reasons to evaluate machine vision VQMs further.

No standardized VQM for machine vision was not found by the authors. However, other special case VQMs exist, for example SC-VQM [11]. This VQM was developed to better predict and evaluate image quality for virtual and augmented reality domains. According to Haccius and Herfet, their VQM was better than conventional VQMs at evaluating the quality in video streams containing virtual and augmented reality [11]. Our focus is not to develop a new VQM and no standard VQM for the machine vision domain exists, as of January 2019. Therefore, we chose SSIM as our VQM.

B. Optimization Algorithm

An encoder can often be configured by numerous parameters to obtain a certain behavior. To find the best encoding configuration candidates, the authors had to solve a large optimization problem. We discussed which path to take: either brute forcing/manually testing every combination, or to use a machine learning optimization algorithm. The machine learning approach had to be chosen due to the vast number of parameters that needed to be optimized and the limited time frame given for this thesis.

Bayesian optimization is an optimization technique created to solve problems that are time-consuming and contain a large parameter space. The parameter space is every possible parameter and the parameters' individual ranges [12]. Bayesian optimization is composed of two main components. An objective function is what the algorithm optimizes by altering its dependent parameters. The algorithm begins by choosing multiple points in the parameter-space. Often these points are chosen completely at random. When the objective function has been evaluated with the initial points the algorithm tweaks the parameters for the next N iterations. In every iteration, the function's metric is evaluated and the parameters are changed to further improve the metric [12]. In our case, the parameter-space is quite large so this optimization algorithm was the one we found suitable for our problem.

C. Codec Performance

Encoder comparison is also a critical field to review. According to D. Grois et. al., the encoder with the highest coding efficiency is H.265 (HEVC), which outperforms the VP9 encoder by 79.4% on average. The VP9 encoder was also outperformed by H.264, with regards to coding efficiency. D. Grois et. al. attempted to configure the encoders in as similar fashion as possible. The encoders were then asked to encode four video streams of high resolution [13]. However, this study was not concerned with real-time encoding but the general video quality. The results were measured using Bjøntegaard-Delta Bit Rate (BD-BR). BD-BR yields a negative value if the bit rate is reduced and a positive if increased [13].

Similar results were found in a study of different amount of movement captured using Youtube, Twitch, Vimeo and Daily Motions recommended bit rates. The results state the H.265 (HEVC) encoder outperformed both VP9 and H.264, in terms of compression efficiency, in every case of the paper [14]. An issue found in 'FFmpeg based Coding Efficiency Comparison of H.264/AVC, H.265/HEVC and VP9 Video Coding Standards for Video Hosting Websites' was the recommended bit rates. The recommended bit rate is stated as 8000 kb/s for FHD resolution, while the actual recommendations are lower from the streaming websites. The recommended bit rates are:

- Youtube has a recommendation of 3000-6000 kb/s [15],
- Twitch has a recommendation of 3500-5000 kb/s [16],
- Vimeo has a recommendation of 10 000-20 000 kb/s [17], and
- Daily motion recommends 4000 kb/s [18].

So the claim of using the recommended bit rates of the major streaming websites is flawed which compromises the paper's general validity [14]. However, this indicated that the H.265 encoder might prove to outperform the codecs in our scope on a higher bit rate than plausible for our constraints. When comparing H.264 and VP9, J. Pavlič and J. Burkeljca show that during a video stream with no movement H.264 performed 18.03% better, but for all other cases VP9 was the superior [14].

D. Hardware and Software Encoding

Video encoding can be done either using a software encoder or a hardware accelerated encoder. A software encoder utilizes the regular CPU cores of the machine. Hardware accelerated encoder utilizes dedicated cores, either on the CPU, such as Intel's QuickSync, or on the GPU, such as Nvidia's NVENC. As our scope includes both software and hardware encoders, a comparison of the encoding types proved useful. In 'Software and Hardware HEVC Encoding' J. Kufa and T. Kratochvil present results showing that software accelerated x265 was unable to encode a video stream in FHD resolution in real-time. However, the hardware accelerated QSV-H.265 did manage the aforementioned, even with slightly better PSNR than what x265 managed in not real-time [19]. This proves the value of hardware encoding on higher resolutions. Software encoding provides a better quality but with a 20% longer encoding duration than QSV when using the same base codec [19].

III. Methodology

A controlled experiment approach was designed and adopted to answer the research questions. The experimental unit used was the main computer in Revere's truck named GDL Truck (see List 1. The aforementioned machine was the computational unit that ran the experimental object, the artifact, the so-called Coordinator Script. The dependent variable, SSIM, was calculated by the script. The constraints frame size and encoding time, together with the experimental unit and object, were all held constant to ensure that these controlled variable would not interfere, when assessing the relation between the dependent and independent variable.

List 1. Experimental Unit Specifications

- Hardware
  - CPU: Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz
  - GPU: NVIDIA Corporation GP107 [GeForce GTX 1050 Ti]
  - RAM: 16227828 kB
  - Motherboard: Supermicro X11SCZ-F
  - Storage:
    - 2x Western Digital Ultrastar DC HC510 8TB
    - 1x Crucial P1 500GB
- Software
  - OS: ArchLinux
  - Kernel: Linux voyager-apollo-x8664-1 5.0.7-rt5-1-rt #1 SMP PREEMPT RT Mon May 6 14:12:28 CEST 2019 x86_64 GNU/Linux
  - Docker: 18.09.5-ce, build e8ff056dbc
  - Drivers:
    - Graphics: NVIDIA 418.56
  - Python dependencies:
    - python: 3.6
    - docker: 3.7.2
    - docker-pycreds: 0.4.0
    - matplotlib: 3.0.3
    - numpy: 1.16.3
    - scikit-optimize: 0.5.2
    - scipy: 1.2.1

Running CS on a single machine solidifies our environment. By using strict version control of each software component used, the environment was considered controlled. The software developed was not included as an end result of the research so design-science with iterations would be redundant. The results were based on a single run of a dataset using the software developed. Therefore multiple evaluations of the artifact provides little to no value.

To answer the two research questions, SSIM, encoding time and frame size had to be obtained for the datasets and compression algorithms in scope. CS was developed to handle video frames, compress them, and calculate the aforementioned values. The artifact relied on two different microservices to operate; the frame-feed-evaluator (FFE), to interact with the OpenDLV software ecosystem and the encoder microservices [20]–[23], to compress the video frames [24]. The FFE made the frames available for the encoder microservices which compressed the frames based on a particular configuration. The compressed frames were then evaluated in the FFE, where the SSIM, encoding time and the frame size were calculated. CS automated this process by systematically evaluate the different codecs, type of encoding and resolutions we wanted to investigate. This was done in three layers:

1) Try all available codecs (see List 2) both hardware and software accelerated (see List 3),
2) In all resolutions (see List 4), and
3) Continuously try to improve the encoding performance (highest SSIM given the encoding time and size constraint) by altering the encoder configurations.

Due to the limited time frame for this bachelor's thesis, we were provided the FFE and the encoder microservices from the Revere laboratory. Hence our focus lied on developing a methodology and an artifact that would collect data to provide insight into our problem domain. Consequently, our choice of encoders and acceleration types were therefore limited to what had been developed beforehand at Revere. We chose to use all encoders and acceleration types that we had at our disposal (see List 2 and 3) However, to make the encoders suit our research, we modified them to give us more control over their configurations (see Appendix A for which parameters we chose to open up for evaluation).

The same goes for the datasets we chose to use in the research. The datasets 'COPPLAR route' and 'AstaZero Rural Road' were simply selected, because they were the datasets we had access to that fulfilled our requirements of being lossless and in high resolution (QXGA - 2048 x 1536). The KITTI dataset was chosen because its relatively high resolution (KITTI - 1392 x 512) but also as it contained sequences similar to the COPPLAR route dataset.

Evaluating various resolutions rather than focus solely on one led to broader results and thus wider application of those results. The resolutions that were used in the experiment can be seen in List 4. The resolutions were chosen not only because of their commonplace, but also since they provide a nicely spaced diverse range.

In the software accelerated encoders, the number of threads were set to 4. This was done to control that each encoder worked on the same premises and thus obtain comparable results. As for the hardware accelerated encoders, Intel QuickSync, the number of threads could not be explicitly set as the implementation did not allow such setting.

List 2. List of codecs

- H.264
- VP9

List 3. List of acceleration types

- Software encoding
- Hardware encoding

– Intel QuickSync

List 4. List of resolutions

- VGA - 640 x 480
- SVGA - 800 x 600
- XGA - 1024 x 768
- WXGA - 1280 x 720
- KITTI - 1392 x 512
- FHD - 1920 x 1080
- QXGA - 2048 x 1536

List 5. List of video datasets

- COPPLAR route, daytime and sunny - urban
- AstaZero Rural Road, daytime and sunny - inter-urban

Note that because the encoder microservices use the following implementations we use the naming H.264 to refer to Cisco's openh264 v1.8.0 implementation [25], VP9 to refer to WebM Project's libvpx v1.8.0 [26] as well as QSV-H264 and QSV-VP9 to refer to Intel's libyami 1.3.1 [27].

A. Research Question 1

To attempt to answer research question 1, SSIM, encoding time and frame size had to be obtained for all the compressed video sequences to compare the different compression algorithms.

The configuration of the compression algorithms has direct causation to the compression result, both in terms of quality, encoding time and frame size. Thus the best configuration for every algorithm, resolution and video sequence was attempted to be used to obtain results allowing a fair comparison of the different compression algorithms. The CS was developed, and utilized for this purpose. To attempt to find the optimal configuration for each of the algorithm, resolution and video sequence, given the encoding time and frame size constraints, machine learning was applied. The form of machine learning used was Bayesian optimization based on Gaussian process regression through the Python library Scikit-Optimize [28]. For each optimization iteration, the algorithm automatically changed the configuration of the encoder to maximize the SSIM while not violating the encoding time nor the frame size. Each encoder optimization was initialized with a default configuration that had been showed to historically give a mediocre SSIM to increase the probability of faster convergence than initializing the optimization with random configurations. In addition to the initialization above, 12 different configurations were randomly generated. Based on these 13 configurations, the machine learning algorithm attempted 80 times to improve the configuration. In each iteration, the encoding configuration was changed by the optimization algorithm on its effort to improve the SSIM. The optimization algorithm took the resulting SSIM and attempted to distinguish how the tuning of the encoding configuration affected the VQM to continuously try to improve the SSIM. The mentioned optimization did not only reduce the resources required to find the best candidate configurations of the compression algorithms but it also limited the human element from the optimizing process.

The artifact was then moved to the experimental unit (see List 1) where the experiment was executed. The above was done on all the different datasets (see List 5) to get the different compression algorithms performance on all driving scenarios. The artifact generated data in the form of reports containing SSIM, encoder time and frame size for each encoder, resolution and dataset as well as graphs illustrating the obtained data.

Fig. 2 shows a high-level flow chart of the CS.

B. Research Question 2

To answer our second research question, we executed CS again but with frames from the KITTI dataset solely in the KITTI resolution (see list 4). Consequently, we obtained the same result from the artifact as for research question 1, but for the KITTI dataset and resolution. The entire KITTI dataset was not used but a subset from the 'CITY' category in the dataset corresponding to one of our datasets, namely COPPLAR route [29]. The aforementioned KITTI CITY subset was chosen as it was the most similar in terms of weather, time of the day, driving environment and traffic to the COPPLAR route dataset as an attempt to control these variables to the degree possible. The resulting best suitable compression algorithm and its configuration could hereby be compared between the KITTI subset and COPPLAR route to see how transferable the results thus the encoding strategies and their configurations were.

C. Data collection

Using Bayhesian optimization with Gaussian regression the authors tried finding the global minimum, the maximum SSIM, for each encoding configuration, resolution and dataset. If necessary, the SC handles the build process of the Dockerized encoder microservices concurrently (see Appendix C). In every iteration of the optimization, the FFE and one encoder microservice is combined to evaluate the current dataset and resolution. The main method of the coordinator iterates through every encoder, also called codec module in scope. The codec modules are composed out of functions directly correlated to our chosen encoders (2). Each codec module includes a parameter space, which defines the range and name of each parameter, a default configuration, and the objective function. The objective function included in the codec modules handle the running of Docker containers and evaluates the results of these.

During the evaluation of all the encoders, CS creates reports, in the form of .csv files. The reports contain SSIM, frame size and and encoding time. When all the resolutions have been evaluated for one encoder, a subsequent script is run which generates a box and whiskers plot displaying the results of

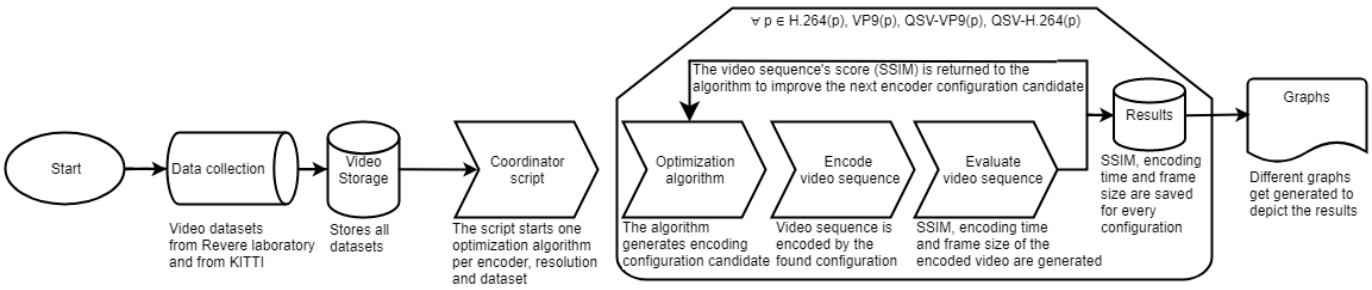

Fig. 2: Flow chart - Coordinator Script

the best candidate configuration for each resolution (4) found during the evaluation of that encoder. The graph depicts one compression strategy with its best candidate configuration and the resulting average SSIM, frame size, and encoding time for respective resolution on the video sequenced used. The test run evaluated the first 900 frames of all datasets (see List 5).

D. Datasets

During the course of the thesis multiple datasets were used. Two sets collected by the Revere laboratory, the COPPLAR route and the AstaZero Rural Road at the AstaZero proving grounds. In addition, the KITTI dataset, collected in Karlsruhe by the KITTI team was also used. The AstaZero Rural Road dataset was collected to provide a set not governed by the European GDPR statutes. This dataset was the only one that the research team could examine manually.

The AstaZero dataset was collected using the same camera, in the same car as the COPPLAR route dataset. The collection of video streams were done through Dockerized microservices, developed by Revere. The microservices used to collect the sets were:

1) opendlv-video-x264-recorder, and
2) opendlv-device-camera-pylon [30], [31]

The KITTI dataset was simply downloaded from their official website [4].

IV. Results

CS generated three kinds of graphs together with reports containing both encoder configurations and data for each evaluated frame.

One ‘encoder graph‘ was generated for every encoder in every dataset (see Fig. 5). Note that the encoder graph was only generated if the encoder managed to produce a valid configuration for the dataset. The graph shows how the encoder’s best candidate configuration performed in terms of SSIM, frame size and encoding time for every resolution it managed to encode. At the bottom of the graph the number with the prefix ‘C‘ indicates which encoder configuration was used to obtain those values. The aforementioned graph and all other encoder graphs can be found in full-size in Appendix B-C.

To easier get an overview of how the encoder performed, one ‘joint graph‘ was generated for every resolution in the datasets (see Fig. 6). The joint graph shows the SSIM that each encoder obtained for the specific resolution. The x-tick labels show which encoder corresponds to the bar plot and which encoder configuration was used (the best candidate configuration). The aforementioned graph and all other joint graphs can be found in full-size in Appendix B-B.

One ‘comparison graph‘ was generated per dataset (see Fig. 3 and 4) The graph shows how the different encoders compare in terms of SSIM percentage change in comparison to the baseline encoder, H.264. H.264 was chosen as the baseline encoder because it managed to produce a valid configuration that met the constraints on all resolutions. The aforementioned graphs can be found in full-size in Appendix B-A.

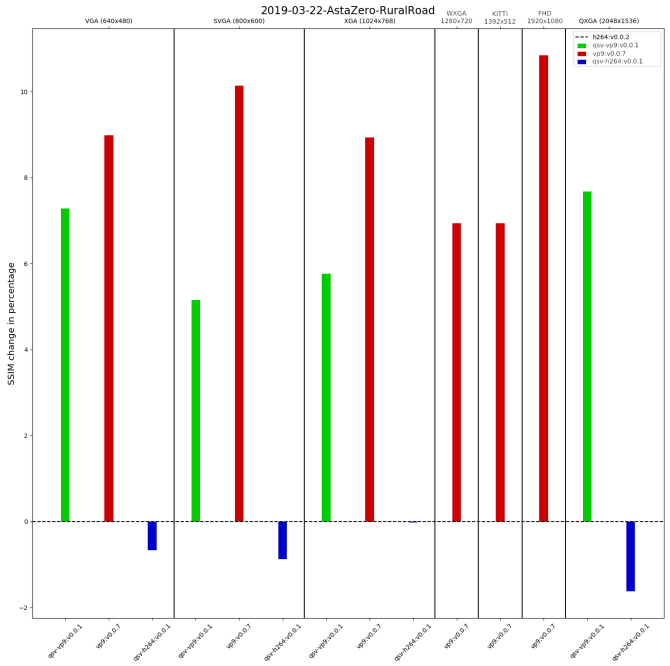

Fig. 3: Comparison graph showing the SSIM change in percentage for the encoders in comparison to the baseline encoder - H.264 - for dataset AstaZero Rural Road

A. Research question 1

Which video compression algorithm performs the best in terms of SSIM given typical driving situations for inter-urban and urban driving situations in daylight with good weather

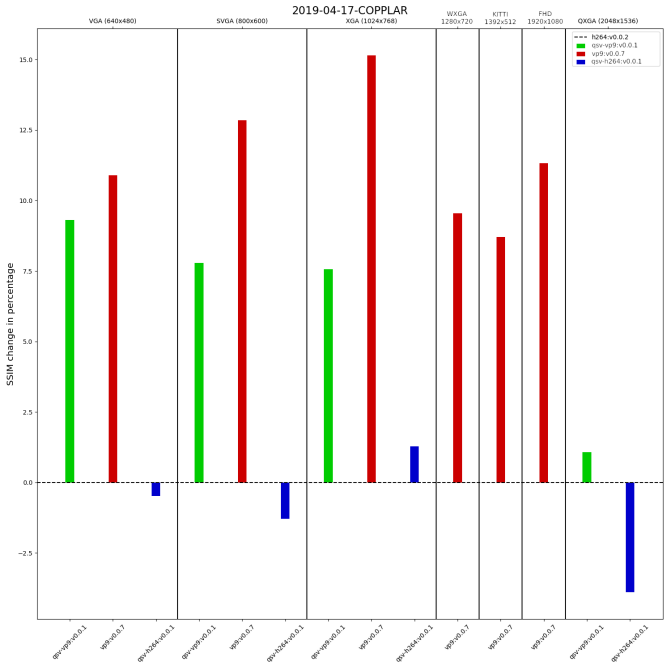

Fig. 4: Comparison graph showing the SSIM change in percentage for the encoders in comparison to the baseline encoder - H.264 - for dataset COPPLAR route

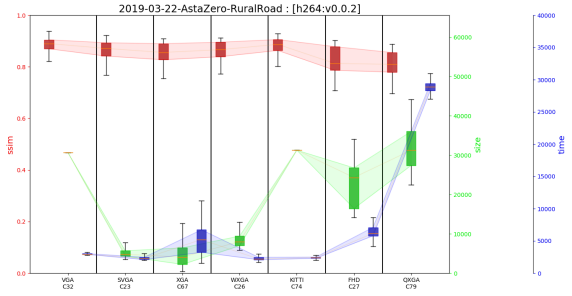

Fig. 5: Encoder graph - depicting SSIM, frame size and encoding time for the H.264 encoder across all resolution in dataset AstaZero Rural Road

while aiming at a maximum frame size of not more than 65507 bytes to allow for network broadcast at a frame rate of not less than 25FPS?

Both comparison graphs for the respective datasets (COPPLAR Route - urban and AstaZero Rural Road - inter-urban) depicted a similar trend (see Fig. 3 and 4 as well as Fig. 10 and 11 in Appendix B-A for full-size). The trend showed that VP9 produced the highest SSIM for all resolutions but the highest, QXGA (2048x1536), in respective driving situation. For QXGA, the VP9 encoder was unable to encode the datasets without violating the constraints. Instead, for this very resolution, the hardware accelerated version of VP9 using Intel QuickSync produced the highest SSIM in both driving situations.

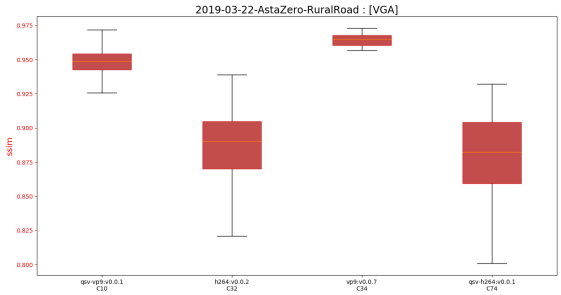

Fig. 6: Joint graph - depicting SSIM for VGA resolution in terms of encoder in dataset AstaZero Rural Road

	Dataset	
Parameter	*KITTI*	*COPPLAR route*
GOP	59	92
rc_dropframe_thresh	19	66
rc_resize_allowed	0	1
rc_resize_up_thresh	61	28
rc_resize_down_thresh	22	42
rc_undershoot_pct	43	76
rc_overshoot_pct	43	84
rc_min_quantizer	1	20
rc_end_usage	1	1
rc_buf_sz	4160	1578
rc_buf_initial_sz	2969	2482
rc_buf_optimal_sz	4225	1222
rc_target_bitrate	4872672	2255460
kf_mode	0	1
kf_min_dist	0	0
kf_max_dist	171	94
VP8E_SET_CPUUSED	9	6

TABLE I: Best candidate encoder configurations for KITTI resolution for the urban datasets

The different best encoder configurations candidates used in respective dataset and resolution were recorded (see Appendix B-D).

B. Research question 2

How transferable are the resulting parameters for the various video compression algorithms when they are compared to a different video dataset, for instance from KITTI?

In the KITTI dataset, VP9 managed to obtain a SSIM over 40 % higher than the baseline encoder H.264 (see Fig. 7 and 8 as well as Fig. 12 and 27 in Appendix B). For the KITTI resolution, only the software accelerated codecs managed to produce results that did not violate the constraints. Consequently, for the KITTI dataset, VP9 encoder performed best in terms of SSIM.

For the KITTI resolution in the other corresponding urban dataset, the COPPLAR route, VP9 also performed superior to the other encoders in terms of SSIM (see Fig. 9 as well as 24 in Appendix B-B for full-size). Therefore, we can establish that the compression algorithm that has yielded the best result, regardless of dataset, was VP9.

The encoder configuration for VP9 differed between the different datasets to the extent that no trend clear trend could

be deduced. TABLE I) display the encoder configurations side by side for an easy comparison.

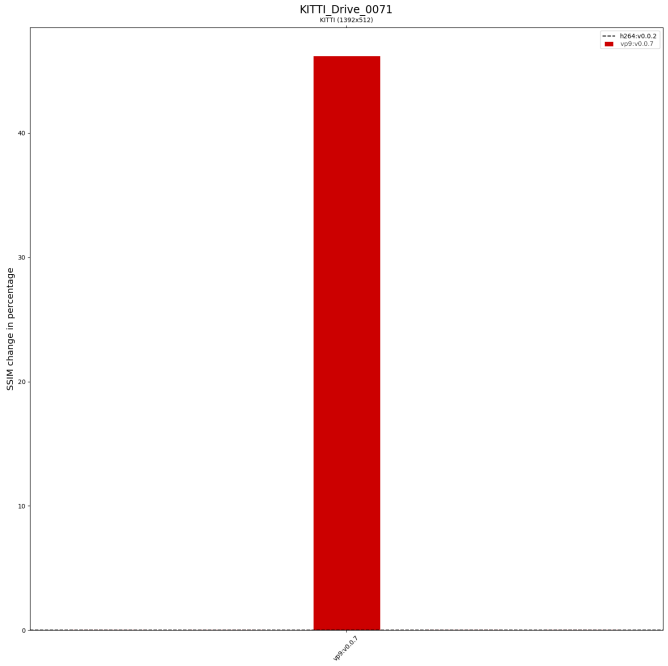

Fig. 7: Comparison graph - KITTI

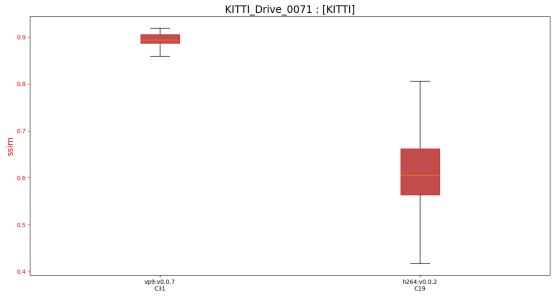

Fig. 8: Joint graph - depicting SSIM for KITTI resolution in terms of encoder in dataset KITTI

V. Analysis & Discussion

A. Graphs

1) Encoder graphs: Each encoder graph display respective encoder's performance, in terms of SSIM, frame size and encoder time, when using the best candidate configuration for each resolution. The two datasets' H.264 encoder graphs were not very similar. By visualizing the data produced by this encoder in the encoder graphs, one can soon see that H.264, while yielding acceptable results, fluctuates more than other encoders in terms of frame size, encoding time and SSIM (see Fig. 28, 32 and 36).

For QSV-H264, the hardware accelerated H.264 codec, a more stable trend between the two datasets was observed. This

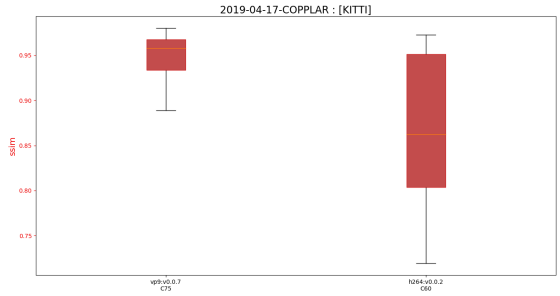

Fig. 9: Joint graph - depicting SSIM for KITTI resolution in terms of encoder in dataset COPPLAR route

trends entails a better transferability between datasets when using hardware acceleration for the H.264 encoder (see Fig. 29 and Fig. 33).

The different encoder graphs for software VP9 also shows similar fluctuation to the software H.264 encoder. For the VGA resolution, the SSIM result was the best of all encoders. If the size was taken into consideration, it does not compare well to its hardware accelerated counterpart (QSV-VP9). In this thesis work however, size was a constraint and not considered a metric. The VP9 software encoder, while fluctuating more on size and time, provides the best results in terms of SSIM for every resolution except QXGA (see Fig. 30, Fig. 37 and Fig. 34). The performance results of VP9 was in accordance with the encoder comparison by D. Grois et. al. [13] discussed in the related works section. The results from this paper also states that VP9 has a higher BD-BR value.

The encoder graphs for QSV-VP9 were similar between the different datasets with a small difference in SSIM (see Fig. 31 and Fig. 35). The datasets includes different frames but yield similar values in SSIM, size and time.

While software encoders might yield a better result, for most resolutions, we can conclude that in terms of size and time, a hardware accelerated QSV encoder provides more stability at nearly the same SSIM score, and this can factor into the future work in this field.

2) Joint graphs: In the joint graphs we were looking into the resulting SSIM for each encoder's best configuration candidate for a specific resolution. The KITTI dataset was excluded for every resolution but the KITTI resolution. For VGA, SVGA, and XGA, every encoder could find a best configuration candidate. For WXGA, KITTI and FHD only the software encoders found a configuration that met the constraints (See the Threats to Validity section V-D). For the highest resolution H.264, QSV-H264 and QSV-VP9 found configurations that met the set constraints B-B.

3) Comparison graphs: The comparison graphs states that, as previously mentioned, VP9 as the encoder yielding the highest SSIM. We can also see that for most resolutions, QSV-H264 returned a worse SSIM in comparison with the H.264 baseline. In conjunction with the results discussed in the Encoder graphs, section V-A1, the QSV-VP9 had worse

Frame	Encoding time in μs
1555491996050879.png	49827
1555491996157157.png	8774

TABLE II: One of the Results from this study, VP9-C24 on the QXGA Resolution (COPPLAR)

SSIM in comparison to its software counterpart, and failed on 3 resolutions (WXGA, KITTI and FHD).

The baseline was chosen to be the H.264 encoder, due to early tests which showed the authors that the encoder had the possibility to find a valid configuration thus generating a result on every resolution. As the results show, software H.264 was generally a valid choice of encoder given our constraints, but was outperformed by the more modern encoder in our scope. The authors decided that a resolution without H.264 as a base line would be redundant because of the other graphs in the comparison graphs B-A

The findings of this study agree with earlier research, both D.Grois et. al. as well as J. Pavlic and J. Burkeljca found that the VP9 encoder was the best choice for their individual purposes, without H.265 considered [13], [14]. The fact that the VP9 encoder could not handle the QXGA resolution in our study, but could in theirs, was most likely due to our encoding time and frame size constraints. This also proves the value of investigation of the H265 encoder.

B. Research Question 1

On the highest resolution, the QSV-VP9 encoder was found to be the most optimal choice. The software VP9 encoder was unable to encode the video streams without violating the constraints. The authors manually looked into the reasons behind this fact. Table II illustrates the encoding time for two frames on one specific configuration in QXGA. This configuration resulted in a frame with the encoding time of 49827 ms, which was approximately 25% above the maximum encoding time allowed. Earlier research suggest that hardware accelerated encoders can code around 20% faster than software encoders and thus can explain why the hardware accelerated VP9 managed to encode the video given the constraints [19].

The best configuration candidates differed between the two datasets (see Appendix B-D2 and B-D3). As the global minimums were unknown, we cannot deduce that the best configuration candidates for respective datasets differed because the global minimums were found. On the other hand, neither can we deduce that the global minimums were *not* found and hence the best configuration candidate differentiated.

For instance, as the two datasets were likely to contain a difference in motion between frames. The compression algorithms used by the encoders will optimize differently (as the change delta between frames were different). When a video has no movement, the encoder only has to place one I-frame at the beginning and still be fully optimal. This would explain the differences in parameters that depend on I-frame handling in the encoder.

Another reason for the difference in configuration can be the optimization algorithm used in CS. The algorithm might converge in a local minimum and hence there was no guarantee that the best configuration candidate was the actual optimal point in the parameter space.

C. Research Question 2

When collecting data from the KITTI dataset the VP9 encoder obtained a SSIM more than 40% higher than the H.264 encoder. The reason for this might be that the optimization algorithm simply converged in a local minimum for the H.264 encoder. It may also be an effect of the KITTI dataset that was of lower resolution and lossy in comparison to the lossless COPPLAR Route dataset which simply could benefit VP9. However, as VP9 was proved to perform considerably better than H.264 on scenes with a lot of movement [14], the result was not very surprising as KITTI was recorded with half the FPS than the COPPLAR Route dataset resulting in more movement.

VP9 proved to be the best encoder in terms of SSIM in respective dataset evaluated as shown in section IV-B. However, as can be observed in TABLE I in section IV-B, the best encoding configurations candidates differed greatly between the datasets and no clear trend could be deduced. Consequently, it suggests that the encoder, VP9, is transferable as the best compression algorithm while its configuration is not and should be tweaked for the application and platform to obtain the best possible results.

D. Threats to Validity

1) Internal threats: The experiment relies on multiple encoder microservices. These encoders are developed and maintained by the Revere Laboratory and extended by us. As each microservice had to be implemented and tested by an external force in order to be accessible to us, the number of encoders in the experiment was directly dependant on which encoders Revere offered. Consequently, many other compression algorithms than the ones evaluated in the experiment exist. Therefore, we cannot conclude that our research deduces the best encoder given our constraints, but rather has the potential to show the most suitable encoder of the ones evaluated. A broader set of encoders including, for instance, the codec H.265 (HEVC) and the hardware acceleration NVENC, would have the potential to give better coverage and potentially better encoding strategies. Therefore, the CS was designed in a modular fashion to encourage further investigation of more encoders at a later date.

The initial plan was to have larger set of datasets in the experiment. The datasets used for research question 1 had to be lossless and of QXGA (2048 x 1536) resolution to keep consistency, comparability and hence to reduce the potential introduction of more threats. Unfortunately availability of datasets, that met the constraints, was limited. We cannot claim that the result of this paper applies to all video compression applications within the automotive domain. Yet, our result for research question 2 do indicate that our study does provide a general consensus of which encoding strategy yields good performance. As stated previously, the CS was constructed in

such manner that adding additional datasets, to increase the experiment's scope, is a simple task.

Investigating the second research question, assessing the transferability of our results, required another dataset collected independently from datasets from the Revere laboratory. Therefore, open datasets available online were researched. However, this posed a challenge as most public free datasets were not of the high resolution we desired. For instance, the nuScenes dataset published in the beginning of 2019 was of surprisingly low resolution and therefore deemed unfit for our application. Instead, the acknowledged KITTI dataset from 2011 was used. It did not possess such high resolution as datasets gathered with the Basler cameras from the Revere laboratory, but it was still of high resolution (KITTI resolution - 1392 x 512). Accurately mapping one dataset from Revere to a video sequence in KITTI posed a threat as this was done by simply choosing the two sequences that looked the most similar to one another subjectively. Yet, fundamental differences of the videos exist such as FPS differences (KITTI: 10, COPPLAR route: 20). The KITTI dataset was included to assess the transferability of our results. However, as fundamental differences do exist between the datasets, the conclusion on the results being transferable or not may have been done on false premises as a result of the datasets being potentially incomparable.

As of 2019, very limited research has been done on video quality metrics for machine vision and an industry standard is yet to be established. As developing a machine vision metric was outside the scope of this thesis, SSIM was adopted as it was regarded as a major and acknowledged VQM. However, the perception of video quality may differ between human and machine vision and hence may affect the result.

Our results shows a single best configuration candidate for each encoder, resolution and dataset. However, during the optimization process we could observe multiple configurations yielding the same SSIM result, especially on lower resolutions. On the higher resolutions - KITTI, FHD, and QXGA - the results produced seemed to contain more similar configuration parameters. From this, we can see that there were in fact multiple best configurations, which might be seen as a threat to the testing validity. We used the earliest encounter of the same SSIM, if was the highest obtained, as the end result. Another method to validate these configurations could be to compare the frequency of each of the optimal configurations but also to include frame size and encoding time in the score for the configuration and not solely the SSIM.

Each encoder configuration could be tailored by a set parameters. Which parameters to be altered in the encoders were decided through literature review (see Appendix A). As it might pose as a threat to not include a parameter, which potentially could be a crucial parameter, as many parameters as possible were included as long as they did not render significant instability or had literature arguing against the usage.

Initially, in the software artifact, the optimization algorithm always got the inverted SSIM value (range from 0 to 1) returned after each iteration. Every time any constraints were violated the script simply returned the worst possible value (1, the worst inverted SSIM). A time violation of 1 percent returned the same value to the optimization algorithm as a time violation of n percent. Consequently, the optimization algorithm was unable to differentiate an encoder configuration that slightly violated the constraints to one that violated them radically. Therefore, it was unable to make the next decision well informed. The solution for this was to define a sane max violation of 150 percent. This value was derived by analyzing the behavior of the optimization algorithm and discussing the issue with the Revere Laboratory. In theory, the larger violation range the better, as the algorithm easier could approximate how illegal a configuration was. However, a larger violation range also meant a longer run time and thus 150 percent was agreed as a fair compromise. With this change, a time violation of n percent would evaluate to the worst SSIM plus the violation, $1 + n$. This solution gave the algorithm the possibility to easier prevent itself from converging in a local minimum. Yet, a larger violation range would reduce this threat even further but at the cost of even longer run time.

Using an optimization algorithm rather than brute-force to find the optimal configuration for each encoder was necessary due to the time frame of the thesis work. Although this did pose a threat as we cannot know for certain that the machine learning algorithm actually found the global minimum and did not converge in a local minimum. To reduce this threat, and considering our limited knowledge in machine learning, the authors sought information and guidance on the matter in the Revere Laboratory as well as they chose to adapt a fully working library using an acknowledged optimization strategy for the encoder configuration optimization [28].

The larger the dataset, the longer run time for CS. However, the optimization algorithm in CS could be slightly tweaked to execute fewer optimization iterations and hence have a shorter run time. Yet, decreasing the number of iterations would have lead to a greater risk of converging in a local minimum. Due to machine availability and time constraint, both the number of frames and the optimization iterations had to be limited. A convergence graph for each optimization was therefore generated to see around which iteration the optimization algorithm generally plateaued on improving SSIM. A value higher than this was used as the number of iterations for the optimization algorithm (80 iterations). However, as this value was obtained on another platform to where the artifact was finally ran, it may be too small than what was desired. In addition, the time constraint and machine availability made us to have to use a subset of our datasets. Using subsets of the datasets has the potential to reduce the frames' diversity. If unlucky, the frames used could reflect a very monotonous sequence in the video (for instance where the vehicle was at a standstill). The team therefore chose to start the subset (900 frames) at the 100th frame in each dataset to ensure that the vehicle was on the move at least in the start of the subset. Thoughts on picking frames in a specific interval was had. However, the encoders would compress the synthetic subsets poorly as I-frames would

constantly have to be added to encode correctly. Consequently, the synthetically created subset containing nonconsecutive frames would have a bad resemblance with real-time streams and thus generate results not relevant to us. Another way of covering a larger set of potential configurations would be to take subsets of the datasets, approximately a tenth in size to the test sets, and use them as validation sets. A validation run would be ran in addition to the rest run with the optimization algorithm tuned to a greater number of optimization iterations and higher mutation rate. By initializing the test run with the best candidate configurations obtained for the different encoders from the validation run, the chances of the test run to converging in a local minimum would be reduced.

When experimenting on a technology which is self-optimizing, encoders, we cannot fully assert that the difference between best configuration candidates was a result of different datasets. As the encoder was constantly adjusting how to handle upcoming frames, the authors were faced with a risk of not being able to adhere the results to a specific cause. The risk was analyzed through reading about different amounts of difference between frames, namely the paper by J. Pavlic and J. Burkeljca [14]. This risk was still present in the study, but acknowledged, as the encoders may obtain a best configuration candidate based on the amount of movement in the video feed rather than their own configuration.

Some resolutions proved to be more difficult to encode for encoders using Intel QuickSync (QSV) than others. The resolutions VGA, SVGA, XGA and QXGA, were encoded without any problems. The other resolutions; WXGA, FHD and KITTI, were also encoded but with a considerably longer encoding time (at least 20 times longer). What differentiated these resolutions, other than the number of pixels, were the aspect ratios. The resolutions that were encoded with normal encoding time was of aspect ratio 4:3 while the resolutions with abnormally long encoding time had either an aspect ratio of 16:9 or 2.76:1 (11:4). The long encoding time for the two latter aspect ratios made the hardware accelerated encoders useless as the resulting encoding time well exceeded the constraint. An explanation for the aforementioned phenomenon has tried to be found, but unfortunately without any luck. Consequently, the conclusion validity for both research questions might have gotten compromised as the QSV encoders could somehow have been made incompatible to encode such resolutions by the authors. Yet, it might also be a limitation of the hardware acceleration. However, this was not very likely (at least for WXGA and FHD) as there are numerous reports on the internet, including from Intel themselves, of successfully encoded WXGA and FHD video streams. Unfortunately, this limitation could not be explained and other hardware accelerated encoders could not be substituted as no other was available at the time of the experiment.

2) External threats: To ensure transferability thus reduce the threat to external validity, care was put into having a set of diverse datasets. This was be done by using video sequences from another vehicle than ours in another geographic area but with similar content. The risk of our result only being applicable for our car, our camera, and our location, was by the aforementioned therefore reduced.

VI. Conclusion and Future Work

The research field of autonomous driving is of global interest, and most sensing device setups include camera sensors. Based on this paper, other researchers and organizations can choose the correct encoder for their video feeds.

Video feeds differ in resolution, in difference between frames and in quality. The authors of this study used the standard resolutions - used in many cameras - to make the results as re-usable as possible, with the forethought of giving other researchers the possibility to easier select an encoder. As a future research topic, the amount of difference between frames could be in scope.

Our study was conducted using two different codecs both accelerated by software and hardware. The next step of much interest would be to include NVIDIA's NVENC encoding and additional codecs using various implementations. As previously mentioned in section II, the H.265 codec is likely to outperform both VP9 and H.264 in terms of bit rate savings and bit rate overhead [14]. Therefore, it would be of high interest to examine if this is true in our context, as a replication study.

To conclude, our study proves the VP9 encoder to be the most optimal encoder on all resolutions expect in QXGA resolution. When working with video feeds in real-time and in high resolutions (QXGA and above) hardware acceleration (QSV) is recommended. In addition, as the best encoding configurations candidates differed greatly between the urban datasets and no clear trend could be deduced. Consequently, it suggests that the encoder, VP9, is transferable as the best compression algorithm while its configuration is not and should be tweaked to perform optimally.

VII. Acknowledgements

This study would not have been possible without the support of the Revere Laboratory team, Christian Berger, Arpit Karsolia, Ola Benderius, and anyone else in the Revere Laboratory involved. Christian developed the encoder microservices that the authors have extended with additional parameters as well as the FFE, the fundamentals of our experimental instrument. In addition, Christian was our academic supervisor providing invaluable expertise, guidance but also the experimental unit. Arpit gave input on how to presenting the data collected and additional pointers on research methodology. Ola supported us in the knowledge acquisition surrounding machine learning and optimization. The Revere Laboratory as such also provided the platform for the experiment, expert knowledge on general research, and workspace for the authors.

Additionally, we would like to express our gratitude to Justinas Stirbys, who proof-read the paper and provided general writing advice.

References

[1] F. Giaimo and C. Berger, "Design criteria to architect continuous experimentation for self-driving vehicles," in *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2017, pp. 203–210.

[2] V. Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022," *White Paper*, 2018.

[3] C. Yan, W. Xu, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," *DEF CON*, vol. 24, 2016.

[4] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[5] "Research | safer - vehicle and traffic safety centre at chalmers," reVeRe - Research Vehicle Resource at Chalmers. [Online]. Available: https://www.saferresearch.com/research#revere

[6] T. Zhao, K. Zeng, A. Rehman, and Z. Wang, "On the use of ssim in hevc," in *2013 Asilomar Conference on Signals, Systems and Computers*. IEEE, 2013, pp. 1107–1111.

[7] S. Wang, A. Rehman, Z. Wang, S. Ma, and W. Gao, "Ssim-motivated rate-distortion optimization for video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 4, pp. 516–529, 2012.

[8] E. L. Christian Berger, Joacim Eberlen, "se-research/video-codec-performance-for-autonomous-driving," https://github.com/se-research/video-codec-performance-for-autonomous-driving/releases/tag/v2.1, 2019, experimental setup to systematically study the performance of video codecs for autonomous driving. commit: e6c775317612df29ef996bcc7098c06a56e11ae9.

[9] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th International Conference on Pattern Recognition*, Aug 2010, pp. 2366–2369.

[10] R. Dosselmann and X. D. Yang, "A comprehensive assessment of the structural similarity index," *Signal, Image and Video Processing*, vol. 5, no. 1, pp. 81–91, Mar 2011. [Online]. Available: https://doi.org/10.1007/s11760-009-0144-1

[11] C. Haccius and T. Herfet, "Computer vision performance and image quality metrics - a reciprocal relation," in *Computer Vision Performance and Image Quality Metrics - A Reciprocal Relation*, 01 2017, pp. 27–37.

[12] P. I. Frazier, "A tutorial on bayesian optimization," 07 2018, a simple tutorial on Bayesian Optimization.

[13] D. Grois, D. Marpe, A. Mulayoff, B. Itzhaky, and O. Hadar, "Performance comparison of h.265/mpeg-hevc, vp9, and h.264/mpeg-avc encoders," in *2013 Picture Coding Symposium (PCS)*, Dec 2013, pp. 394–397.

[14] J. Pavlic and J. Burkeljca, "Ffmpeg based coding efficiency comparison of h.264/avc, h.265/hevc and vp9 video coding standards for video hosting websites," *International Journal of Computer Applications*, vol. 182, pp. 1–8, 01 2019.

[15] "Live encoder settings, bitrates, and resolutions," https://support.google.com/youtube/answer/2853702?hl=en, accessed: 2019-05-23.

[16] "Broadcasting guidelines," https://stream.twitch.tv/encoding/, accessed: 2019-05-23.

[17] "Encoder guides," https://help.vimeo.com/hc/en-us/articles/115012811208-Encoder-guides, accessed: 2019-05-23.

[18] "Encoding parameters," https://faq.dailymotion.com/hc/en-us/articles/203655666-Encoding-parameters, accessed: 2019-05-23.

[19] J. Kufa and T. Kratochvil, "Software and hardware hevc encoding," in *2017 International Conference on Systems, Signals and Image Processing (IWSSIP)*, May 2017, pp. 1–5.

[20] C. Berger, "Release version v0.0.2," https://github.com/chalmers-revere/opendlv-video-h264-encoder/releases/tag/v0.0.2, 2019, openDLV Microservice to convert an image in shared memory to an h264 frame using openh264. commit: 87e8f0e4a1317597656fc4e9c3ccd48ea1f65be6.

[21] ——, "Release version v0.0.8," https://github.com/chalmers-revere/opendlv-video-vpx-encoder/releases/tag/v0.0.8, 2019, openDLV Microservice to convert an image in shared memory to a VPX frame (VP8 or VP9). commit: 86b5f627b4a42ffe5ee96931934f943d5cf4cab3.

[22] ——, "Release version v0.0.2," https://github.com/chrberger/video-qsv-h264-encoder/releases/tag/v0.0.2, 2019, intel QuickSync hardware-accelerated video encoding for h264. commit: 8b9665d05e8e46f6858c29dd682d9cf56b2edef7.

[23] ——, "Release version v0.0.1," https://github.com/chrberger/video-qsv-vp9-encoder/releases/tag/v0.0.1, 2019, intel QuickSync hardware-accelerated video encoding for VP9. commit: 30bcab334109051a8faedaaf7d7b49a9821bd3f3.

[24] ——, "frame-feed-evaluator," https://github.com/chrberger/frame-feed-evaluator, 2019, repository to systematically evaluate video compression codecs. commit: 5c6424181d6ec7e28e221cc72045a57cdd300149.

[25] https://github.com/cisco/openh264/blob/v1.8.0/CONTRIBUTORS, "Release version 1.8.0 - cisco/openh264," https://github.com/cisco/openh264/releases/tag/v1.8.0, 2019, repository for the h264 codec implemented as openh264. commit: 6fe15a6b82d492bebe388c55b7ee5131208e7334.

[26] T. W. project, "Release version 1.8.0 - webmproject/libvpx," https://github.com/webmproject/libvpx/releases/tag/v1.8.0, 2019, repository for the VPX codec commit: b85ac11737430a7f600ac4efb643d4833afd7428.

[27] A. The WebM Project, Intel Corporation, "Release version 1.3.1 - intel/libyami," https://github.com/intel/libyami/releases/tag/1.3.1, 2019, repository for QSV hardware acceleration. commit: fb48083de91f837ddbf599dd4b5ad1eb1239e1cf.

[28] andreh7, N. Campos, M. Cherti, A. Fabisch, T. Fan, T. Head, M. Kumar, G. Louppe, K. Malone, nel215, M. Pak, I. Shcherbatyi, T. Smith, and Z. Vinicius, "Scikit-optimize," https://github.com/scikit-optimize/scikit-optimize, 2019, scikit-Optimize, or skopt, is a simple and efficient library to minimize (very) expensive and noisy black-box functions. It implements several methods for sequential model-based optimization. skopt aims to be accessible and easy to use in many contexts. commit: af5450a51599bbfa4846342188948c147ceba14c.

[29] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "2011_09_29_drive_0071 (unsynced+unrectified data, image_02)," http://www.cvlibs.net/datasets/kitti/raw_data.php?type=city, 2013.

[30] C. Berger, "chalmers-revere/opendlv-video-x264-recorder," https://github.com/chalmers-revere/opendlv-video-x264-recorder/releases/tag/v0.0.3, 2019, repository for an OpenDLV Microservice that converts an image in shared memory to a lossless h264 frame using libx264. commit: 203f7fc251da7ecf491e20bc7c022a504762b9a0.

[31] ——, "chalmers-revere/opendlv-device-camera-pylon," https://github.com/chalmers-revere/opendlv-device-camera-pylon/releases/tag/v0.0.1, 2019, repository for an OpenDLV Microservice that interfaces with GiGE cameras supported by the pylon library (eg., Basler cameras). commit: f74da40d25dfb9ed99c2c44acfdd62f39356dcfa.

[32] H. Schwarz, D. Marpe, and T. Wiegand, "Analysis of hierarchical b pictures and mctf." in *ICME*. Citeseer, 2006, pp. 1929–1932.

[33] H. Kalva, "The h. 264 video coding standard," *IEEE multimedia*, vol. 13, no. 4, pp. 86–90, 2006.

[34] K.-T. Fung, Y.-L. Chan, and W.-C. Siu, "New architecture for dynamic frame-skipping transcoder," *IEEE transactions on Image Processing*, vol. 11, no. 8, pp. 886–900, 2002.

[35] Intel, "libyami-utils/yamitranscode.1," https://github.com/intel/libyami-utils/blob/master/doc/yamitranscode.1#L27, 2019, repository to systematically evaluate video compression codecs. commit: 2164966849f011c4242663fefb5229a8b2be9e2b.

[36] Y. Su and M.-T. Sun, "Fast multiple reference frame motion estimation for h. 264/avc," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 447–452, 2006.

[37] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[38] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 614–619, 2003.

[39] G. R. M. J. Mirza and M. Javed, "In-loop deblocking filter for h. 264/avc video," in *Proceedings of the 5th WSEAS International Conference on Signal Processing, Madrid, Spain*, 2006, pp. 235–240.

[40] M. Rezaei, I. Bouazizi, V. K. M. Vadakital, and M. Gabbouj, "Optimal channel changing delay for mobile tv over dvb-h," in *2007 IEEE International Conference on Portable Information Devices*. IEEE, 2007, pp. 1–5.

[41] P. Ai, S. Chen, Z. Chu, P. Du, M. Ettl, A. Gal, X. Guang, L. Guo, Y. Guo, H. Huang, S. Huang, E. Hugg, C. Jennings, Z. Jia, D. Jin, J. Li, J. Li, K. Li, K. Li, M. Li, X. Li, B. Ling, A. Liu, W. Liu, V. Patil, E. Rescorla, A. Roach, S. Shan, S. Tao, M. StorsjÃű, B. Vibber, J. Wang, J. Wang, Z. Wang, H. Willems, G. J. Wolfe, K. Wu, G. Xu, J. Xu, G. Yang, L. Yao, J. Zhang, R. Zhang, V. Zhang, L. Zhu, J. Zhu, D. Zhang, H. Zhu, and

H. Shi, "Typesandstructures - cisco/openh264 wiki," https://github.com/cisco/openh264/wiki/TypesAndStructures, 2015, repository for Cisco openh264. commit: 38b50752583cfea02f2626229989858a7cf483d1.
[42] N. Ozbek and A. M. Tekalp, "H. 264 encoding of videos with large number of shot transitions using long-term reference pictures," in *2006 14th European Signal Processing Conference*. IEEE, 2006, pp. 1–4.
[43] H.-Y. Cheong, A. M. Tourapis, J. Llach, and J. Boyce, "Adaptive spatio-temporal filtering for video denoising," in *2004 International Conference on Image Processing, 2004. ICIP'04.*, vol. 2. IEEE, 2004, pp. 965–968.
[44] S. M. Kim, J. W. Byun, and C. S. Won, "A scene change detection in h. 264/avc compression domain," in *Pacific-Rim Conference on Multimedia*. Springer, 2005, pp. 1072–1082.
[45] "vpx_codec_enc_cfg struct reference," http://doxygen.db48x.net/mozilla/html/structvpx__codec__enc__cfg.html#a9abffd5b85a0babbe3073b763f3311e1, accessed: 2019-05-23.
[46] S. Winkler, M. Kunt, and C. J. van den Branden Lambrecht, *Vision and Video: Models and Applications*. Boston, MA: Springer US, 2001, pp. 201–229. [Online]. Available: https://doi.org/10.1007/978-1-4757-3411-9_10

APPENDIX A
CHOICES OF PARAMETERS

A. Intel QuickSync (QSV)

Two microservices used Intel's hardware acceleration QuickSync to encode VP9 and H.264. Therefore, the two microservices had many parameters in common.

As the experiment has an FPS of 25 as a precondition, the parameter *frameRateNum* was left to its default value and, as with all encoder in our scope, it was not included in the optimization process. The *intraPeriod* parameter was made available to control the group of pictures or the GOP structure to set the interval for new I-frames. As I-frames are the least compressible of the I, P and B-frames, controlling their frequency is important in a frame size point of view [32]. The parameter *ipPeriod* controls the distance between an I and a P frame. Again, as P-frames differ in size in regards to the other frames, controlling its frequency will affect the frame size and is therefore of interest to investigate. *bit rate* is another parameter that was chosen to be altered as it has a direct correlation to SSIM, frame size and encoding time if a rate control mode is selected that relies on this parameter [33]. *initQP* is the initial quantization parameter (QP) for a video sequence. If rate control that is using QP is used, this parameter will govern what the QP will be at the start of the video sequence. As QP is used to regulate an image's level of detail, and is directly connected to bit rate, it is interlinked with the constraints of this experiment and thus needed to be investigated [33]. The parameters *minQP* and *maxQP* sets the range of the QP. The parameter *disableFrameSkip* is used to toggle the frame skip function. Frame skip is used to ensure that the defined bit rate is met and therefore of interest to this experiment [34]. *diffQPIP* and *diffQPIB* are the differences in QP between adjacent I and P frames and I and B frames respectively [35]. As frame size is dependent on frame type, these two parameters were of interest in the experiment. The *numRefFrames* gives the encoder the possibility to select numerous previously decoded frames to use as a blueprint for the next frame and is thought to improve video coding performance [36]. The Intel QuickSync provides a parameter called *enableLowPower*. Power consumption is not in the scope of this research but rather speed and size and hence this parameter was set to false throughout the experiment. The parameter is also infamous for its instability on VP9 on higher resolutions with the current drivers at the time of the experiment. The *bitDepth* parameter governs how many colors that can be used in a pixel. Higher bit depth leads to higher frame size and was therefore selected to be investigated. *rcMode*, or rate control mode, decides which mode that will be used to control the number of bits used for each frame. Consequently, this parameter is of much interest to us since quality, encoding time and frame size is a direct effect of this parameter.

Unfortunately, the documentation of Intel QuickSync, in regards to the parameter set provided to configure the encoding, is very limited. Consequently, parameters where no information of their use and range were given, were ignored to not break the configuration. The following parameters were ignored: *level*, *VideoTemporalLayers*, *VideoTemporalLayerIDs* and *leastInputCount*.

1) H.264: In addition to the above parameters, some parameters were also codec specific. *enableCabac* toggles between the entropy codings CABAC and CALVC and *enableDct8x8* toggles the use of 8x8 transforms. Both parameters were chosen to be investigated as they are said to have high significance for the encoding efficiency [37]. The parameter *enableDeblockFilter* toggles the deblocking filter used in H.264. This parameter is very much in the scope of this research as it directly affects the encoding efficiency and time [38], [39]. The *idrInterval* specifies the Instantaneous Decoder Refresh (IDR) frame interval. As the IDR frames frequency affects the encoding efficiency and quality the parameter was chosen to be included in our parameter space [40].

Again, due to the poor documentation in Intel QuickSync, no information was given for the following parameters and they were therefore unusable in our scope as no ranges were given which caused crashes: *basicUnitSize*, *VUIFlag*, *SamplingAspectRatio* and *priorityId*.

2) VP9: For VP9 QSV-accelerated, only one parameter was available, namely *referenceMode*. The aforementioned parameter sets the reference mode for the encoder. Even though the documentation for it was scarce, we chose to include it in the parameter space as it was a boolean and thus could only be either true or false.

B. H.264

As previously mentioned the fps parameter - *fMaxFrameRate* - was set to a default of 25 fps, and not included in the optimization process. *iUsageType* dictated the 'encoder use' and as our scope was real-time video streams, we chose the corresponding setting, namely 'CAMERA_VIDEO_REAL_TIME' [41]. Similar to QSV, the intra period, parameterized as *uiIntraPeriod* in H.264, was altered to control the group of pictures (GOP). The GOP dictates the interval for new I-frames which differ in size in comparison to other frame types and hence

pose as an interesting parameter to us [32]. Similarly, *iTargetbit rate* was chosen to be controlled because of its direct correlation to SSIM, frame size and encoding time if a rate control mode is selected that relies on this parameter [33]. The *iNumRefFrame* parameter gives the encoder the possibility to select numerous previously decoded frames to use as a blueprint for the next frame and is thought to improve video coding performance [36]. Its two modes, 'AUTO_REF_PIC_COUNT' and 'I_NUM_REF_FRAME' were both chosen to be evaluated in the experiment. The parameter *iEntropyCodingModeFlag* dictates the entropy mode, either Context Adaptive Binary Arithmetic Coder (CABAC) or Context Adaptive Variable Length Coder (CAVLC) can be chosen. Due to its significant impact on encoding efficiency, it was chosen to be investigated [37]. The *bEnableFrameSkip* parameter governs the frame skip function which is used to ensure that the defined bit rate is met and is therefore of interest to this experiment [34]. The bit rate can be set to not exceed a certain value with the *iMaxbit rate* parameter. As bit rate is directly linked to SSIM, frame size and encoding time this parameter was of interest [33]. The quantization parameter (QP) is used to regulate an image's level of detail and is directly connected to bit rate. The QP can only be the range set by the two parameters *iMaxQp* and *iMinQp* and they are therefore of interest. *bEnableAdaptiveQuant* controls the function to adaptively control QP and thus has a correlation to bit rate which is of much interest to us. The parameter *bEnableLongTermReference* toggles the long term reference function. As this function has shown in experiential trials to reduce the frame size significantly, while keeping the quality intact. The function was therefore included in the experiment [42]. *iLoopFilterDisableIdc* controls if loop filtering should be activated. The filter has shown to improve video quality and affect frame size and is therefore of interest to be included in our study [39] [38]. H.264 denoising function is controlled by *bEnableDenoise* and since it has shown indication to improve both encoding efficiency and video quality, we chose to include it [43]. H.264's scene change detection function, toggled by *bEnableSceneChangeDetect*, has the potential to affect the computing overhead considerably and is therefore included in our experiment [44]. The rate control parameter, *iRCMode* is of significant interest as it controls how the number of bits that will be used for each frame. Consequently, the *iRCMode* parameter is included in the experiment since it has a direct impact on quality, encoding time and frame size. The parameter *iComplexityMode* can be set in three modes; low, medium and high complexity which the encoding speed and quality are influenced by [41]. Therefore, the parameter was included in our experiment.

A few parameters were also ignored. *bSimulcastAVC*, *bIsLosslessLink* and *iLTRRefNum* were ignored as they were not fully implemented in the current revision of H.264 [41].

C. VP9

Each encoder include multiple parameters, some more vital than others in regards to our constraints. The encoder includes an option to ensure, if possible, error resilience. The *g_error_resilient* parameter was not used in this experiment as the impact of enabling this mode was unclear, and not well defined in documentation [45].

In the VP9 encoder we are using a video control variable called *VP8E_SET_CPUUSED*, which is a range from 0 to 16 (Quality to Speed).

The *g_usage* parameter is unused in this project, as no value was added by redefining this variable from its default. The height and width of each frame was instantiated to have the same values as the current resolution 4.

The default for the profile parameter is used, due to datasets which are used in the experiment. The default "0" is defined as an 8 bit color depth and 4:2:0 chroma sub-sampling image [46]. The frames provided in the datasets are of this type, therefore the default value is the only applicable option.

To ensure continuous behavior across all encoders in scope the *g_threads* parameter was set to 4.

To decide which keyframe mode to use, automatic placement or a fixed interval, the parameter *kf_mode* is set to either 0 or 1. In libvpx terms, this is translated into *VPX_KF_AUTO* and *VPX_KF_DISABLED*. When disabled the placement of keyframes is governed by *kf_min_dist* and *kf_max_dist*. When these parameters holds the same value the placement is constant.

In the codec there is a possibility to define *rc_buf_initial_sz*, *rc_buf_optimal_sz* and *rc_buf_sz* which dictates how much buffering time the decoder should use for each frame. In the scope of this research the ranges, in mutual order: 0 - 4000, 0-5000, and 0-6000.

Another piece of functionality of the codec is to enable strategic frame drops to meet the requirements defined. The range is between 0-100 percent, when the target bit rate buffer falls below this percentage a frame is dropped.

VPx includes two rate control modes, Variable Bit Rate (VBR) and Constant Bit Rate (CBR). The parameter *rc_end_usage* governs this, the range on our space is 0 (CBR) or 1 (VBR). In CBR the bit rate cannot change to ensure that the frame conveys to the constraints set upon it, while VBR is lenient and will try to change the target bit rate to meet these constraints. The target bit rate (*rc_target_bit rate*) default is 800 000 and in our space it ranges from 100 000 to 5 000 000. The type of the target bit rate is kilobytes/second so the value from our space is always divided by 1000 in the encoder microservice.

To further control the bit rate two tolerance variables are in place, *rc_overshoot_pct* and *rc_undershoot_pct*. These define in percentage how much the bit rate can change in VBR. In the experiment space these range from 0-100 percent.

The quantizer parameters are the closest quality control variables available in libvpx, the *rc_max_quantizer* is, in our script, always the maximum of 52 - worst quality possible - while *rc_min_quantizer* varies between 0 and 52. The resulting frame will be somewhere in the range between these parameters.

On low data rates the possibility to create a lower resolution frame and up-scaled it to the predefined width and height of the encoder can create a higher quality frame with a smaller size. This is controlled by the boolean parameter *rc_resize_allowed* and the percentage based (0-100) parameters *rc_resize_down_thresh* and *rc_resize_up_thresh*.

The scope of the experiment states the the encoding have to be real-time. The team have defined a maximum encoding time as 40 ms. In the VP9 encoder, two-pass encoding and all interconnected parameters - passes and pass - have been skipped due to the impossibility of real-time encoding when active.

There is also a list of 2 pass specific parameters, which are all ignored:

- rc_2pass_vbr_bias_pct
- rc_2pass_vbr_maxsection_pct
- rc_2pass_vbr_minsection_pct
- rc_twopass_stats_in

In VP9, the presets available are best, good and real-time. In the scope of this experiment this will be constantly set to real-time.

As the experiment also states that our video-feeds are in 25 fps, therefore this parameter is not changed in the optimization process. The timebase of the encoder - *g_timebase* - is updated by the notifyAll call of the live camera interface. If the environment is not a live stream the default is 25 fps, a *g_timebase* value of *1 / 20*. The parameter lag-in-frames is ignored due to the statement in the official documentation of the codec [26]. The official documentation states that lag-in-frames is not appropriate when using the real-time preset.

APPENDIX B
DATA

A. Comparison graphs

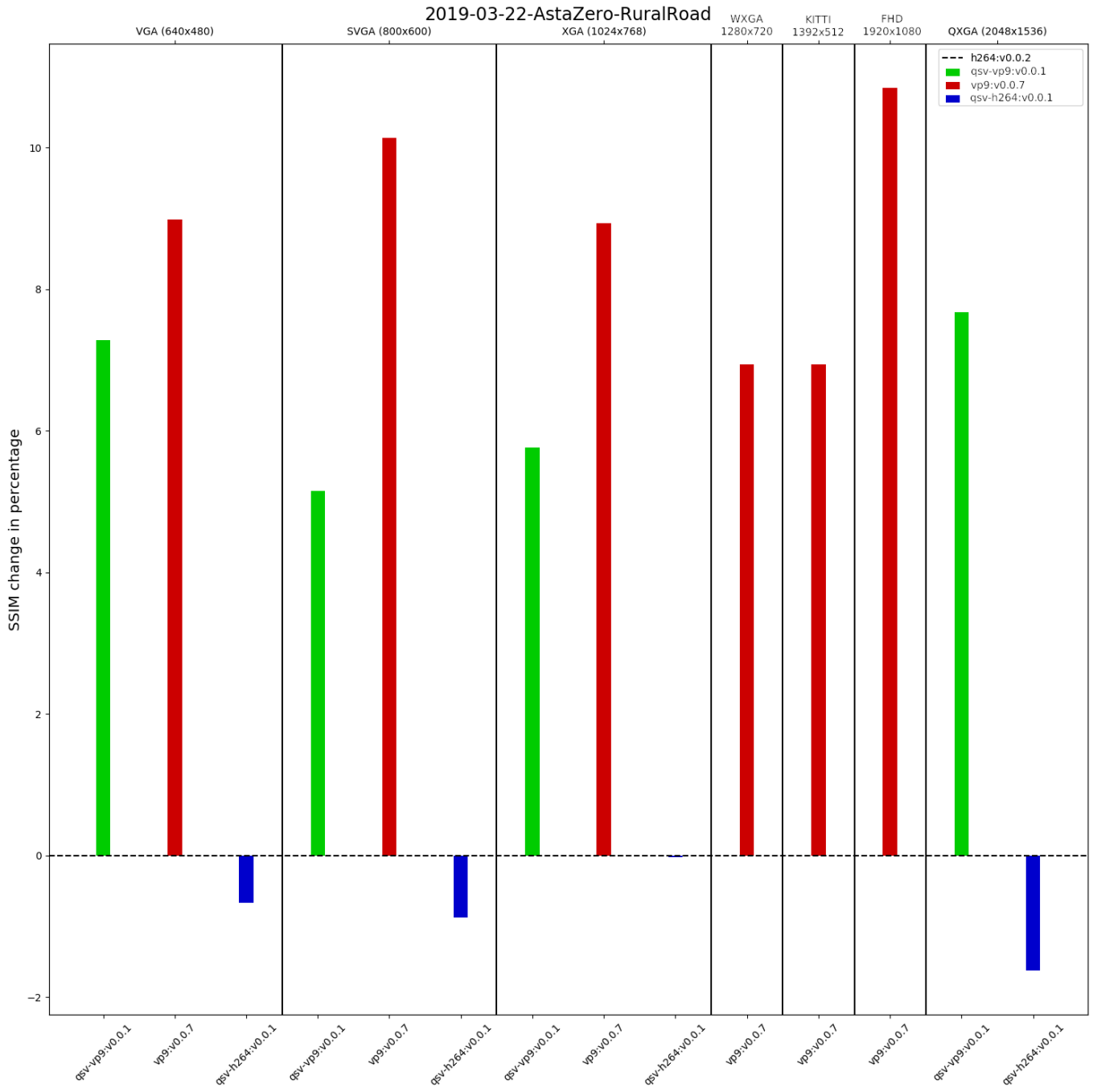

Fig. 10: Comparison graph of the AstaZero Rural Road dataset

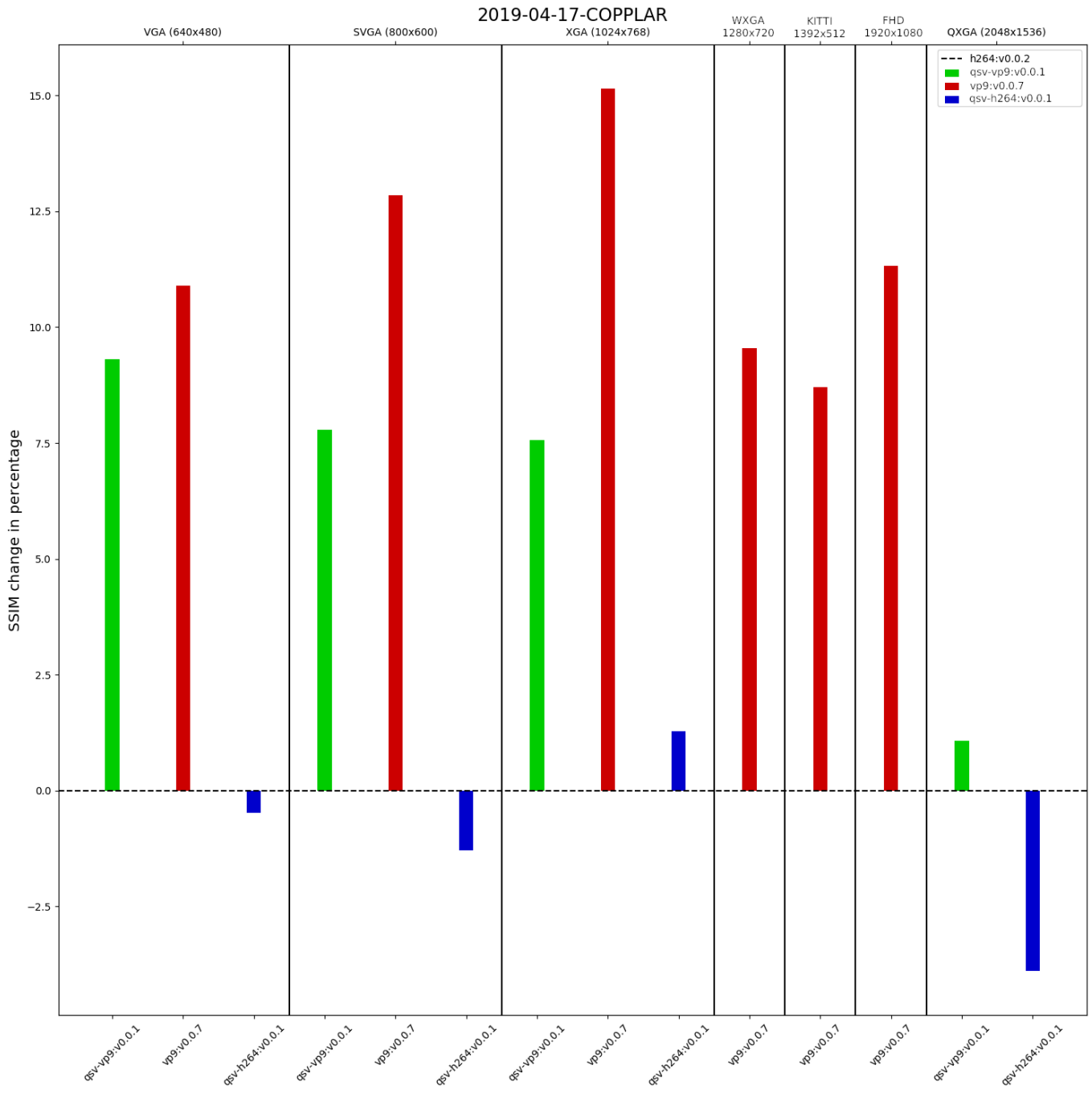

Fig. 11: Comparison graph of the COPPLAR dataset

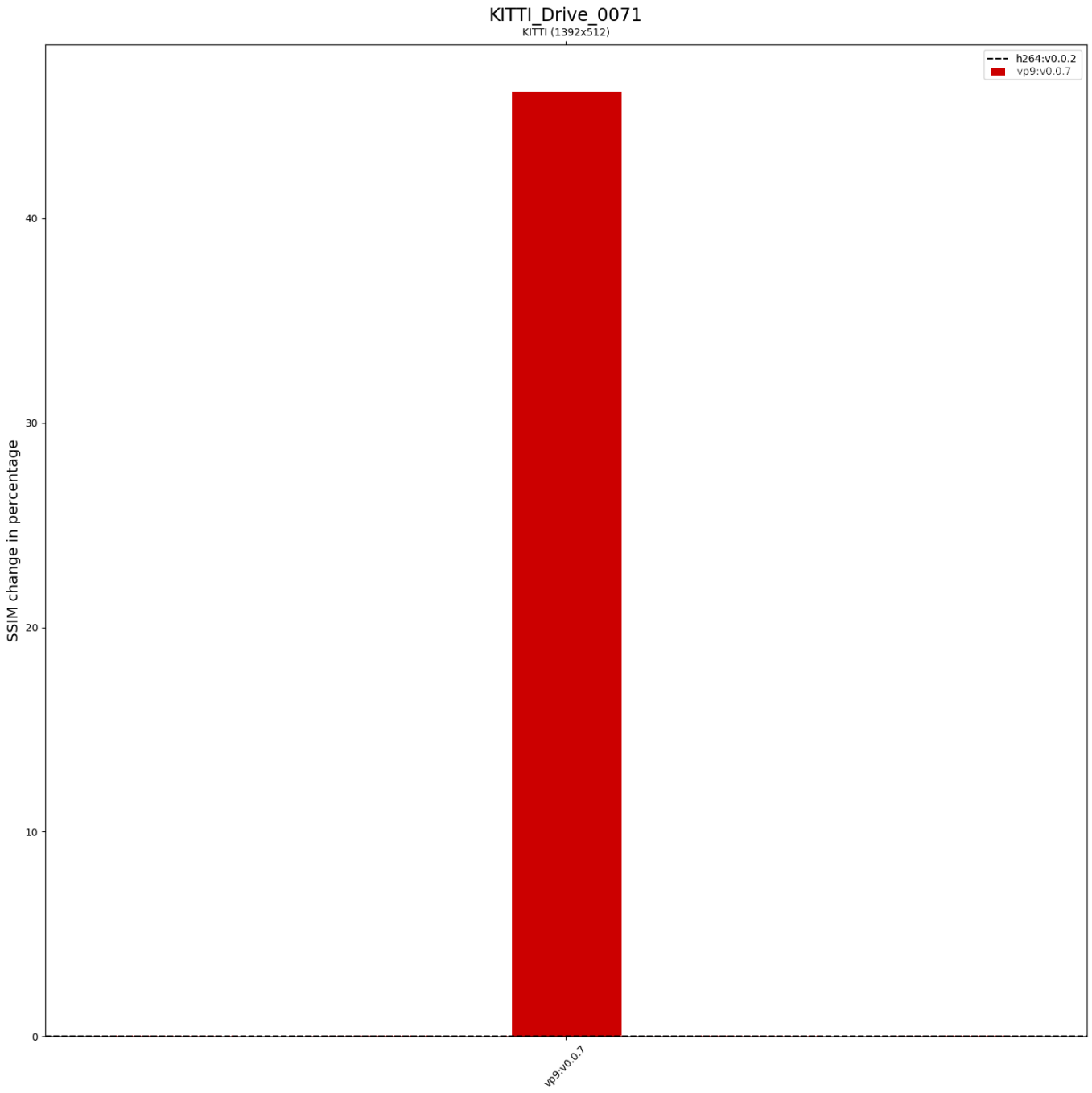

Fig. 12: Comparison graph of the KITTI dataset

B. Joint graphs

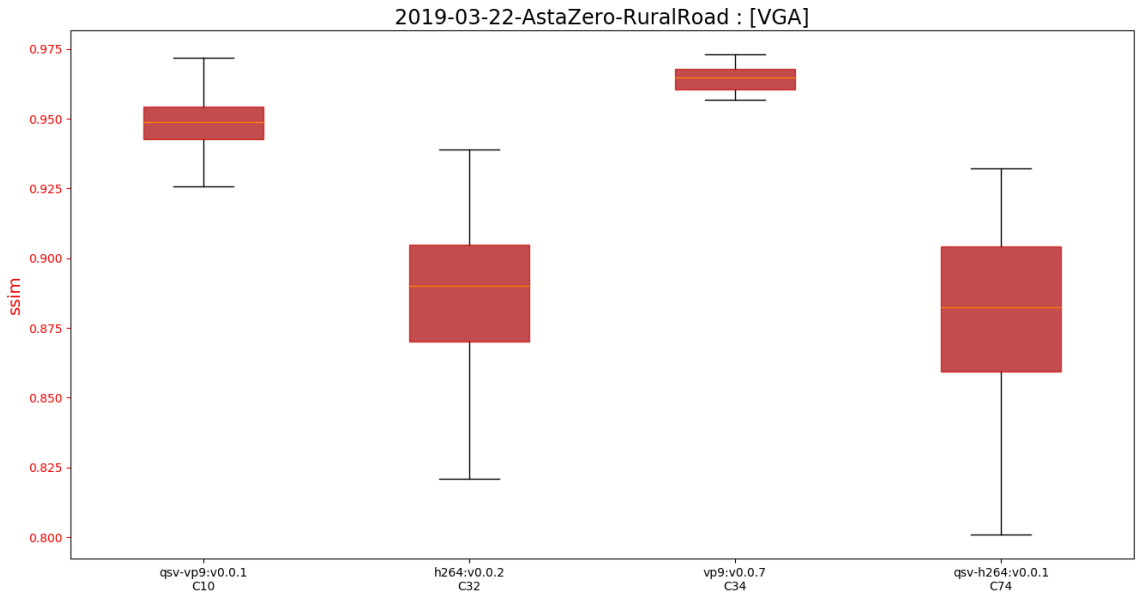

Fig. 13: SSIM comparison for VGA (AstaZero Rural Road)

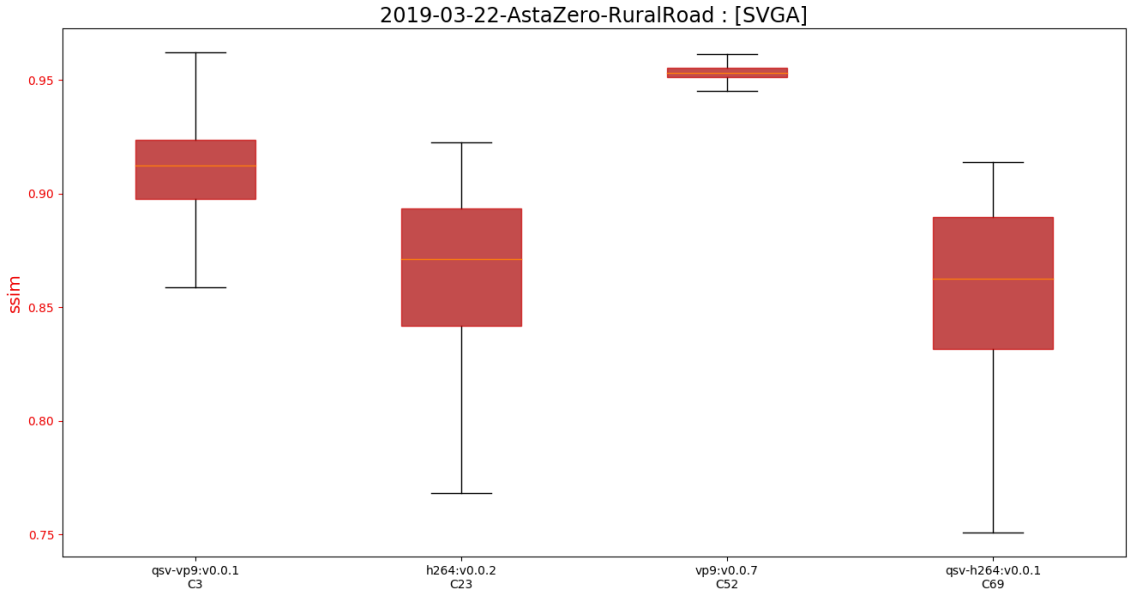

Fig. 14: SSIM comparison for SVGA (AstaZero Rural Road)

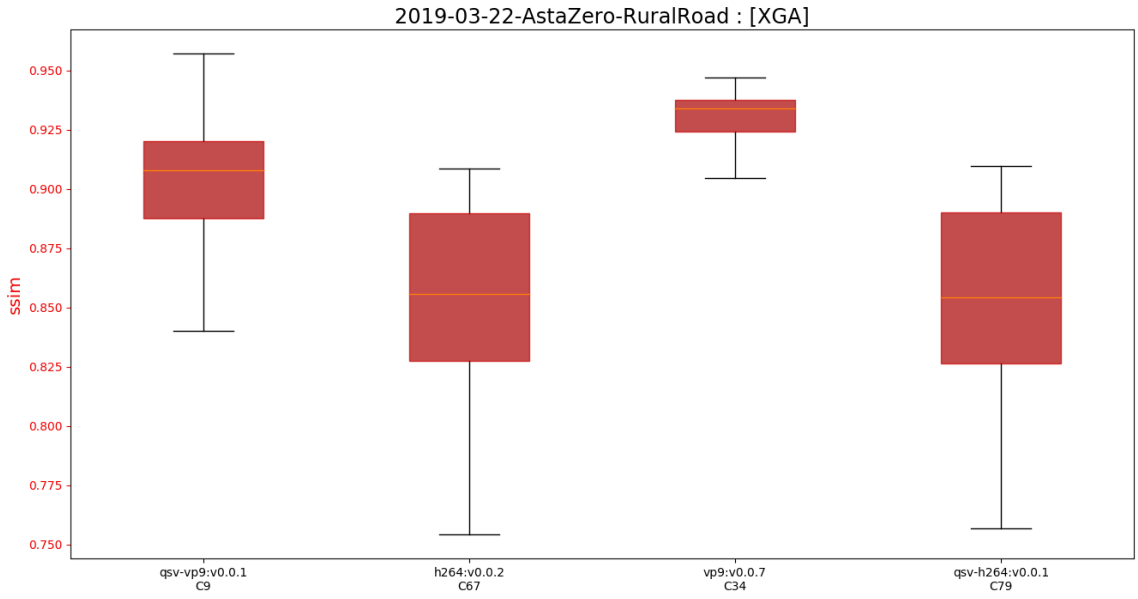

Fig. 15: SSIM comparison for XGA (AstaZero Rural Road)

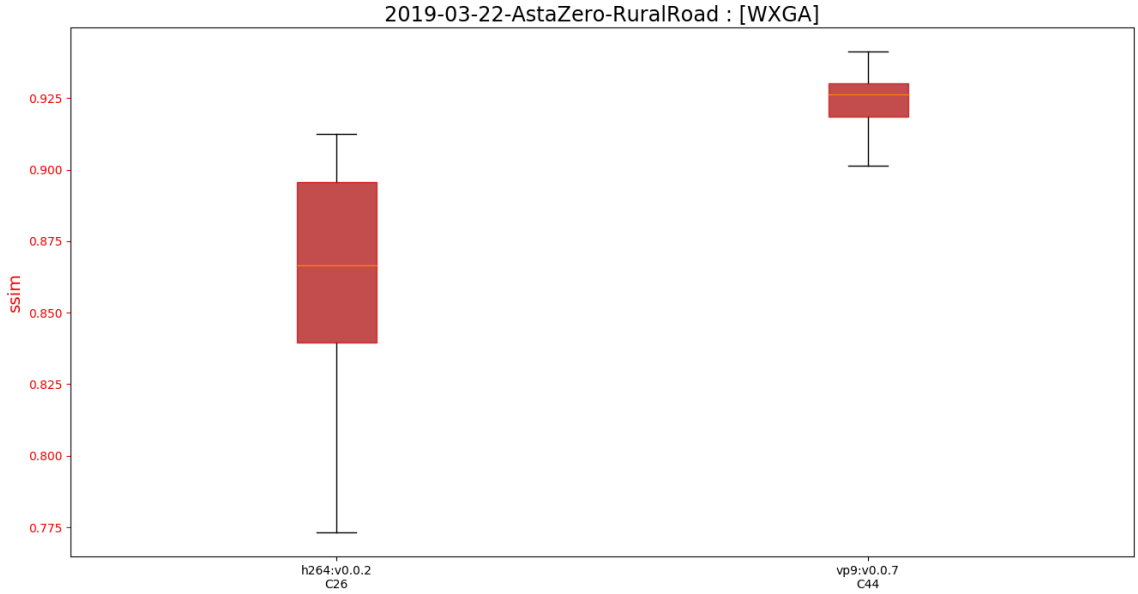

Fig. 16: SSIM comparison for WXGA (AstaZero Rural Road)

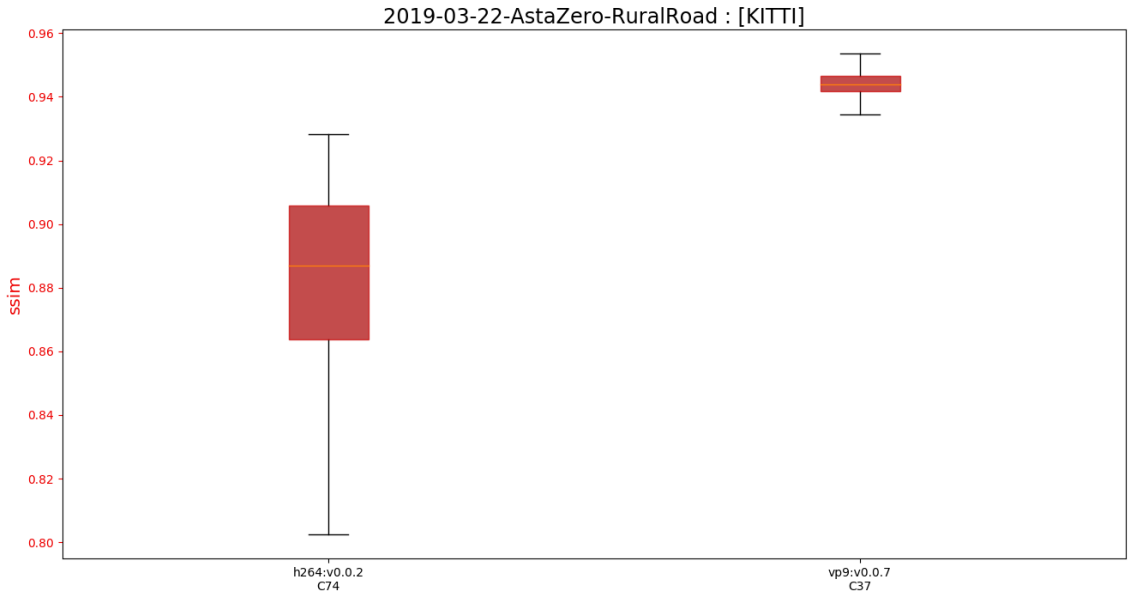

Fig. 17: SSIM comparison for the KITTI resolution (AstaZero Rural Road)

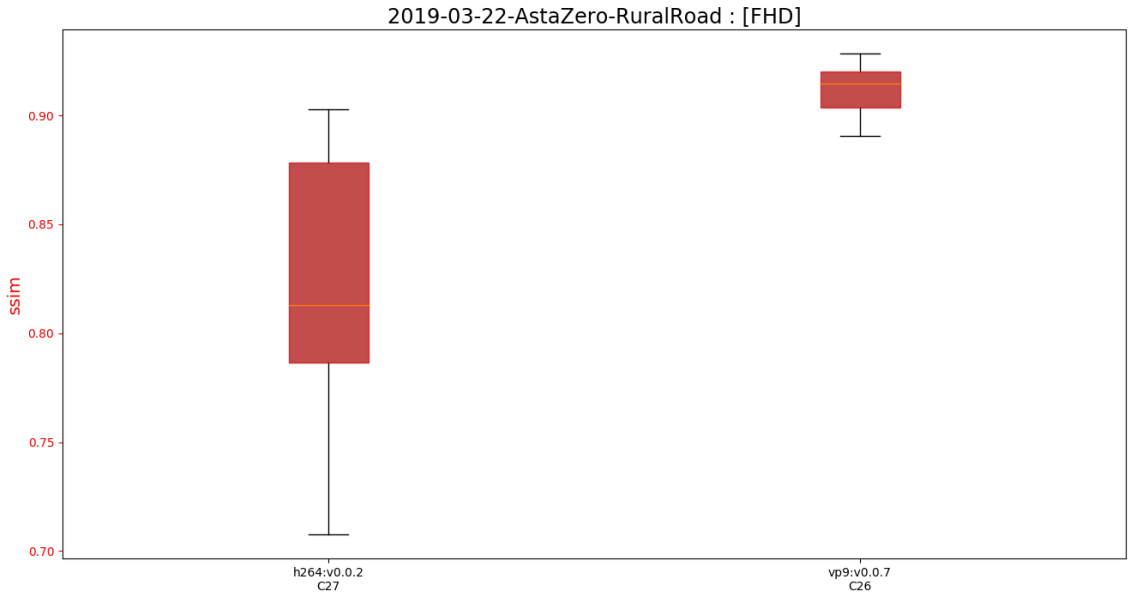

Fig. 18: SSIM comparison for FHD (AstaZero Rural Road)

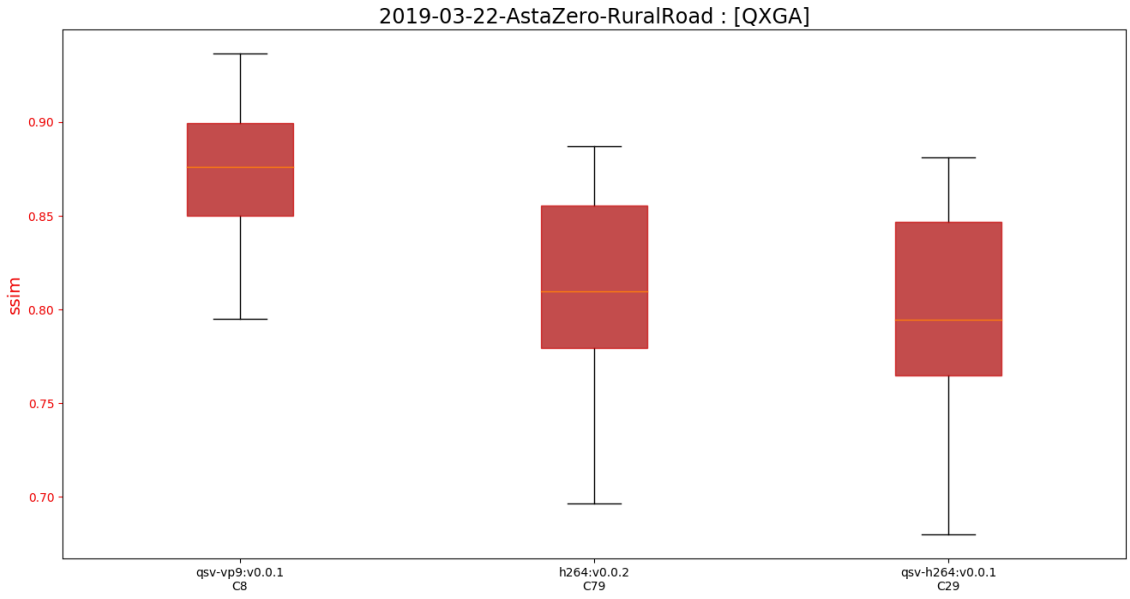

Fig. 19: SSIM comparison for QXGA (AstaZero Rural Road)

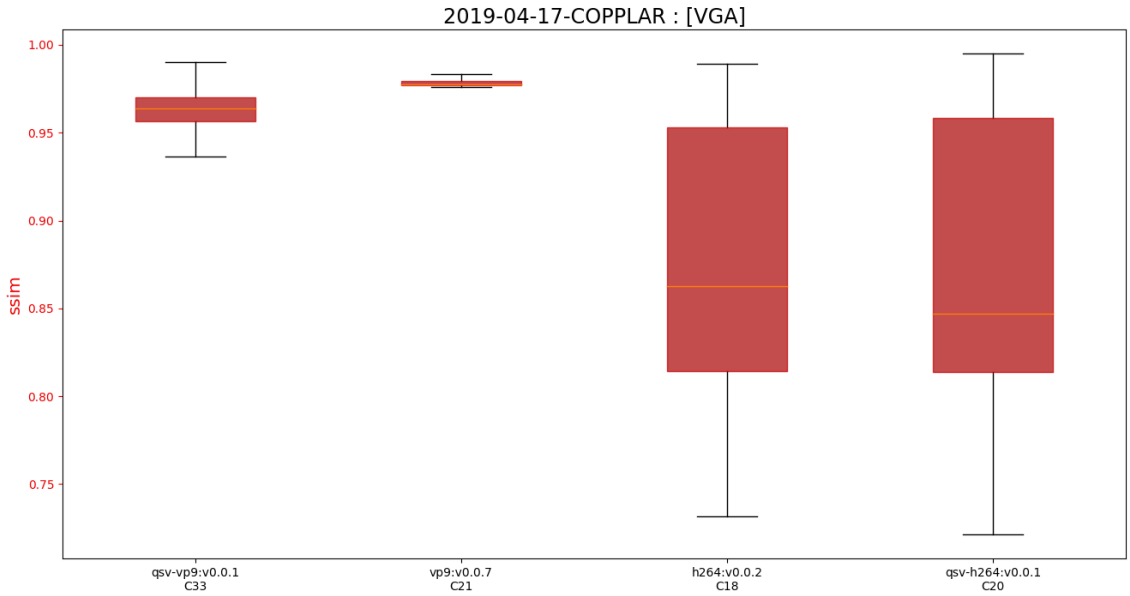

Fig. 20: SSIM comparison for VGA (COPPLAR)

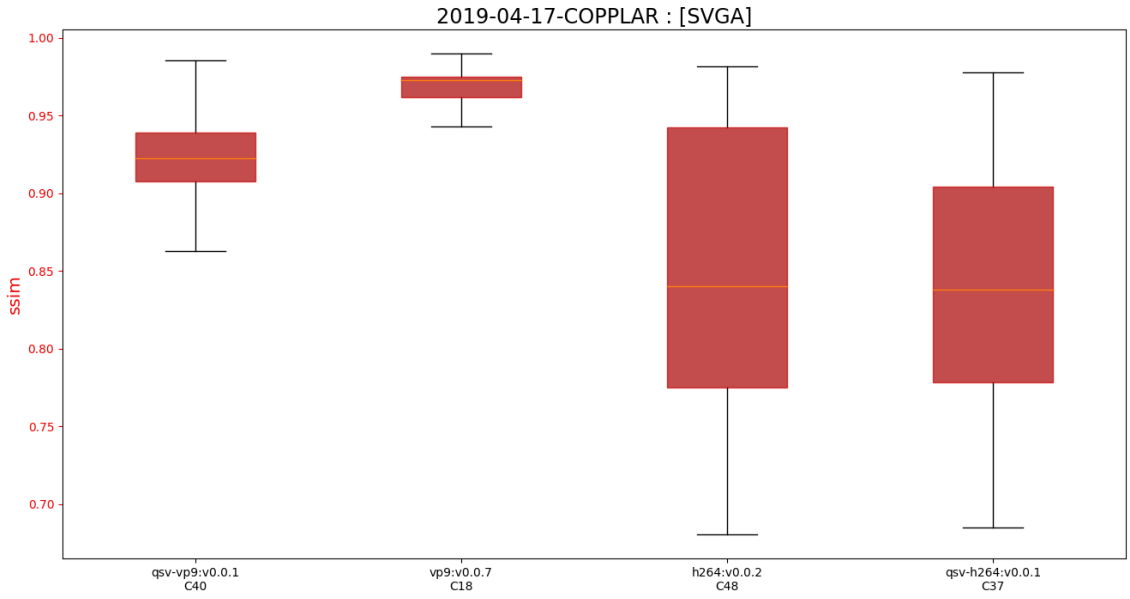

Fig. 21: SSIM comparison for SVGA (COPPLAR)

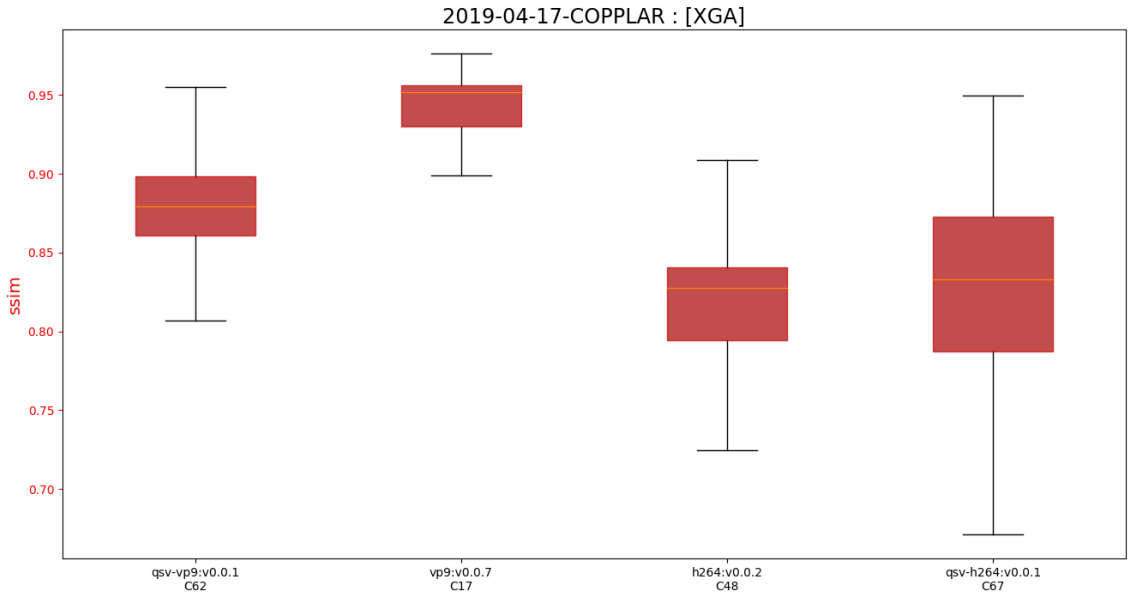

Fig. 22: SSIM comparison for XGA (COPPLAR)

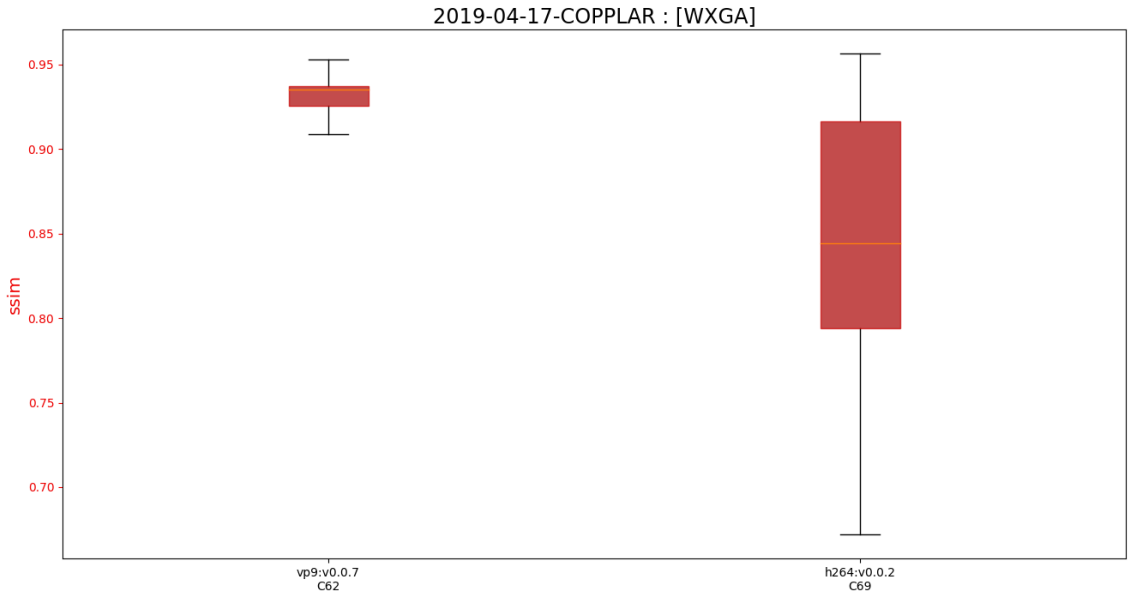

Fig. 23: SSIM comparison for WXGA (COPPLAR)

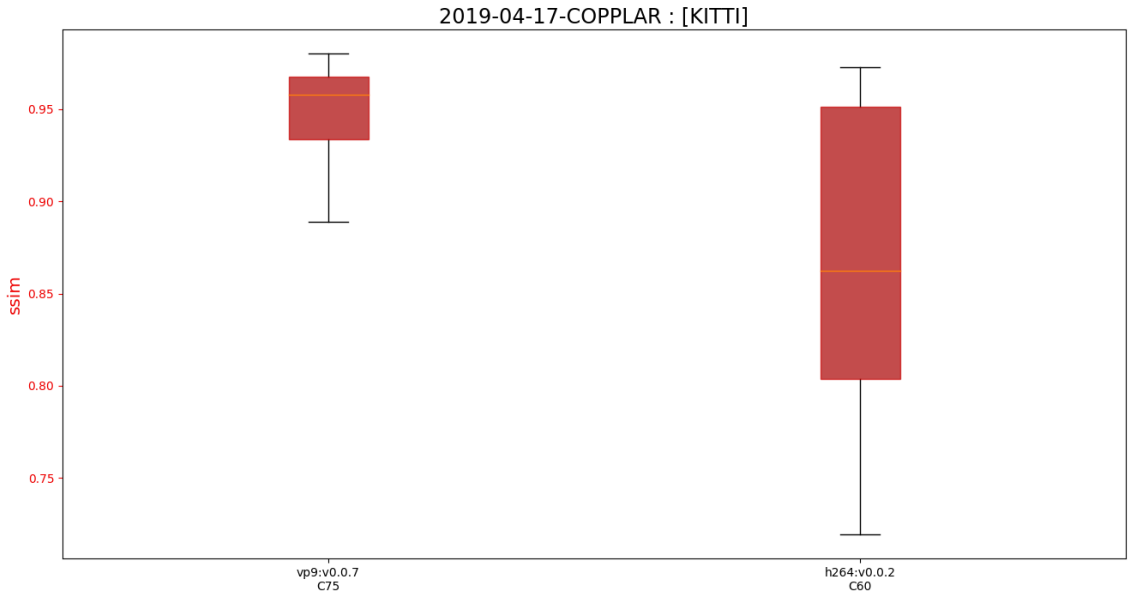

Fig. 24: SSIM comparison for the KITTI resolution (COPPLAR)

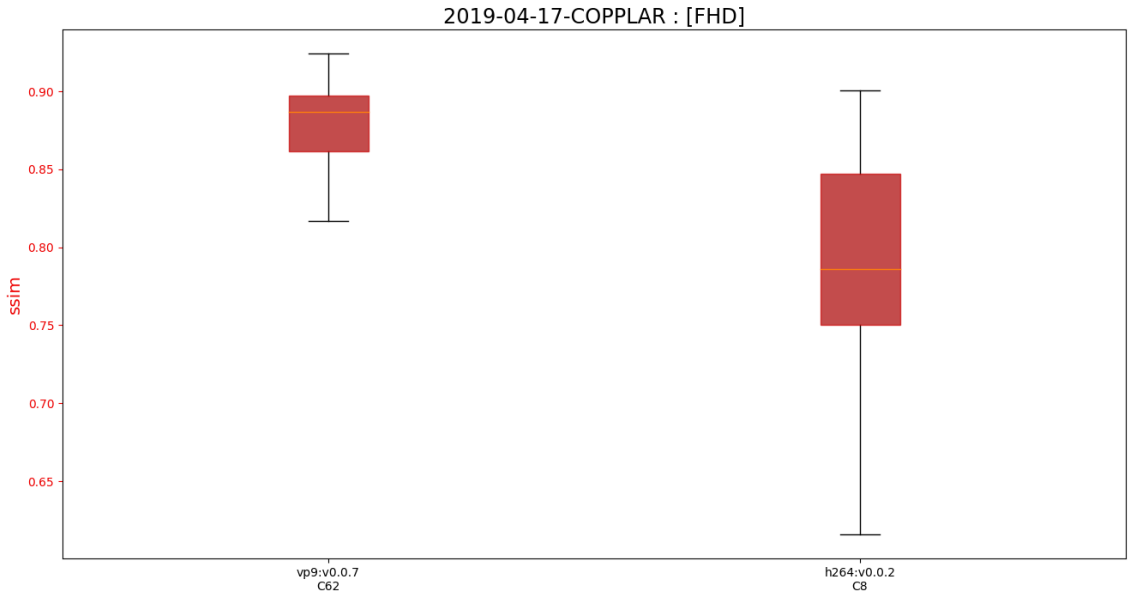

Fig. 25: SSIM comparison for FHD (COPPLAR)

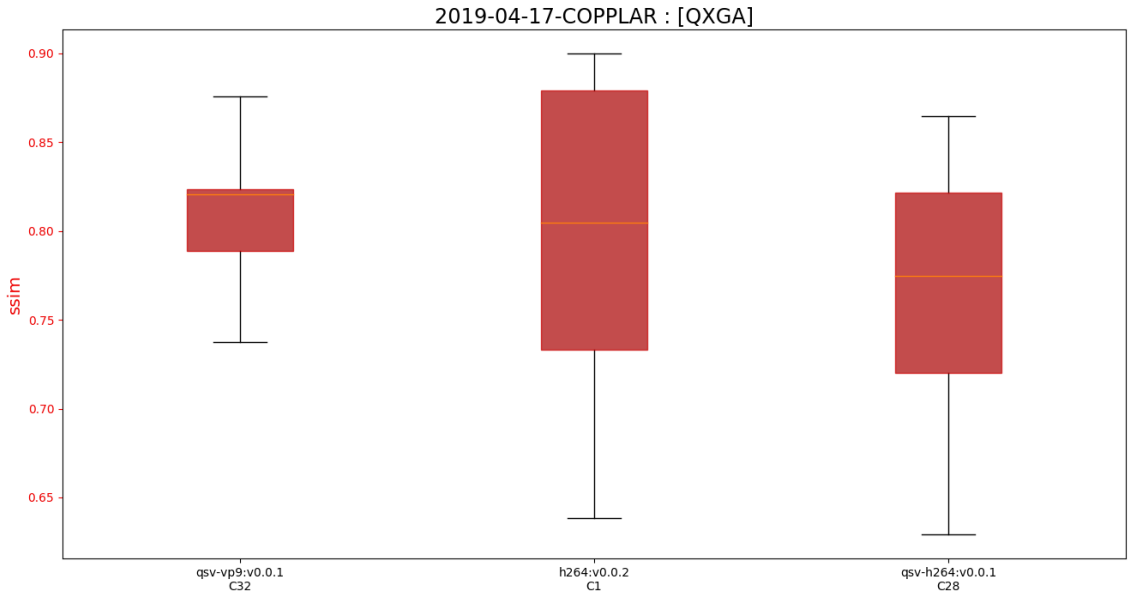

Fig. 26: SSIM comparison for QXGA (COPPLAR)

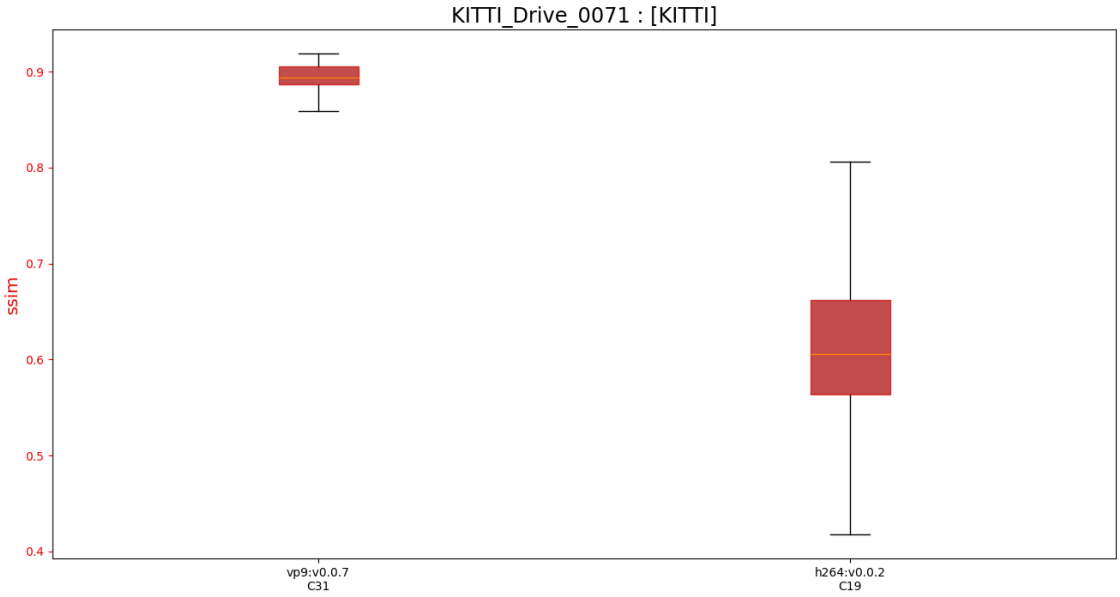

Fig. 27: SSIM comparison for the KITTI resolution (KITTI)

C. Encoder graphs

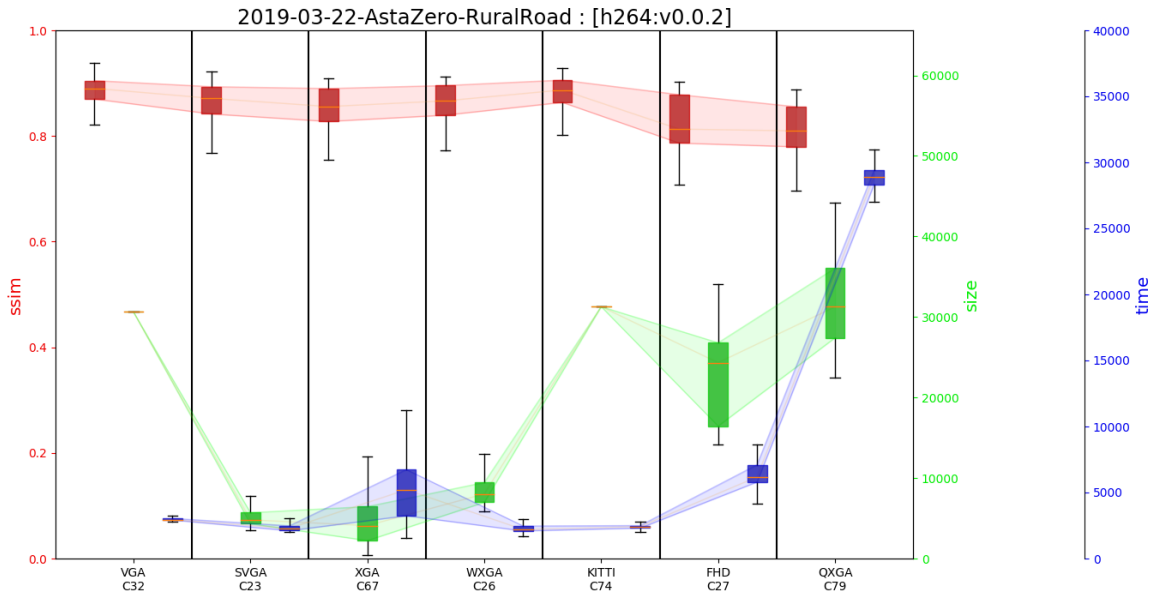

Fig. 28: Encoder graph for H264 (AstaZero Rural Road)

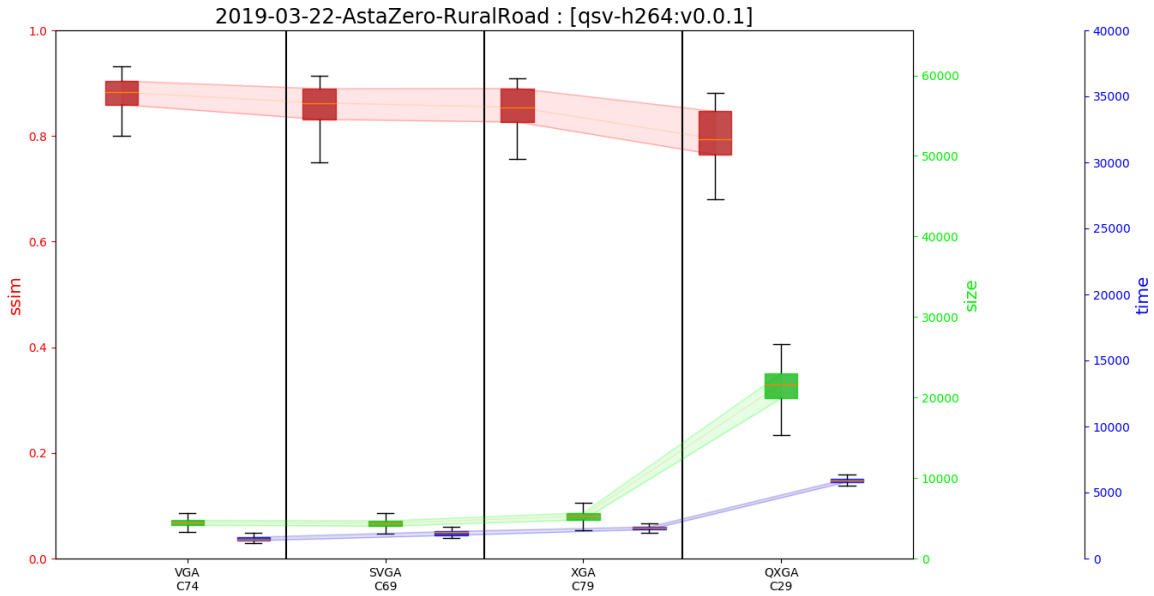

Fig. 29: Encoder graph for QSV-H264 (AstaZero Rural Road)

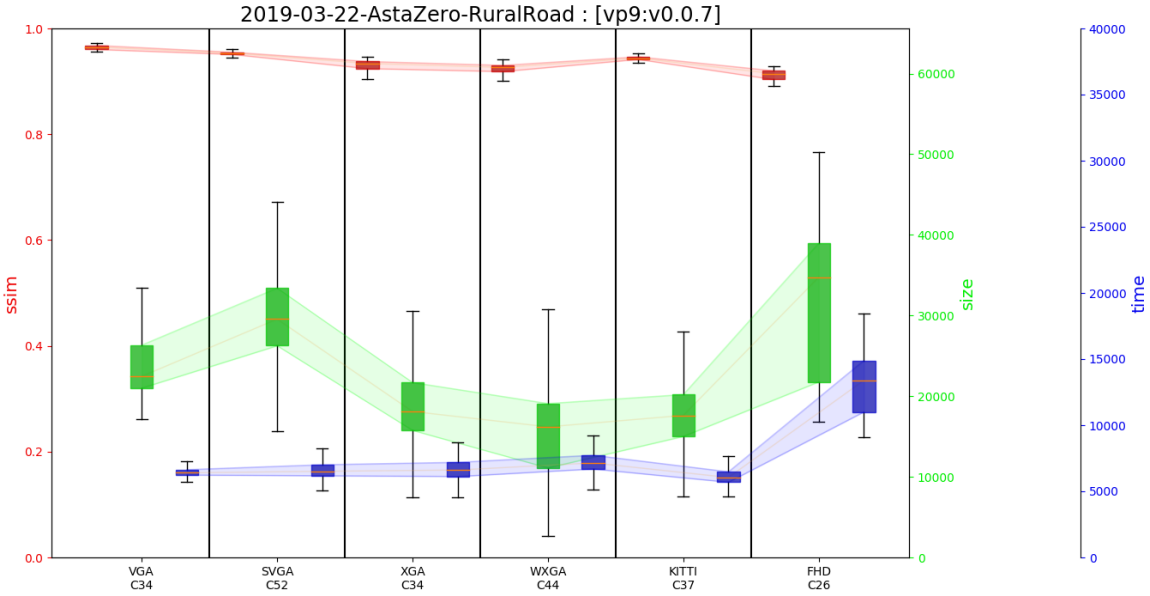

Fig. 30: Encoder graph for VP9 (AstaZero Rural Road)

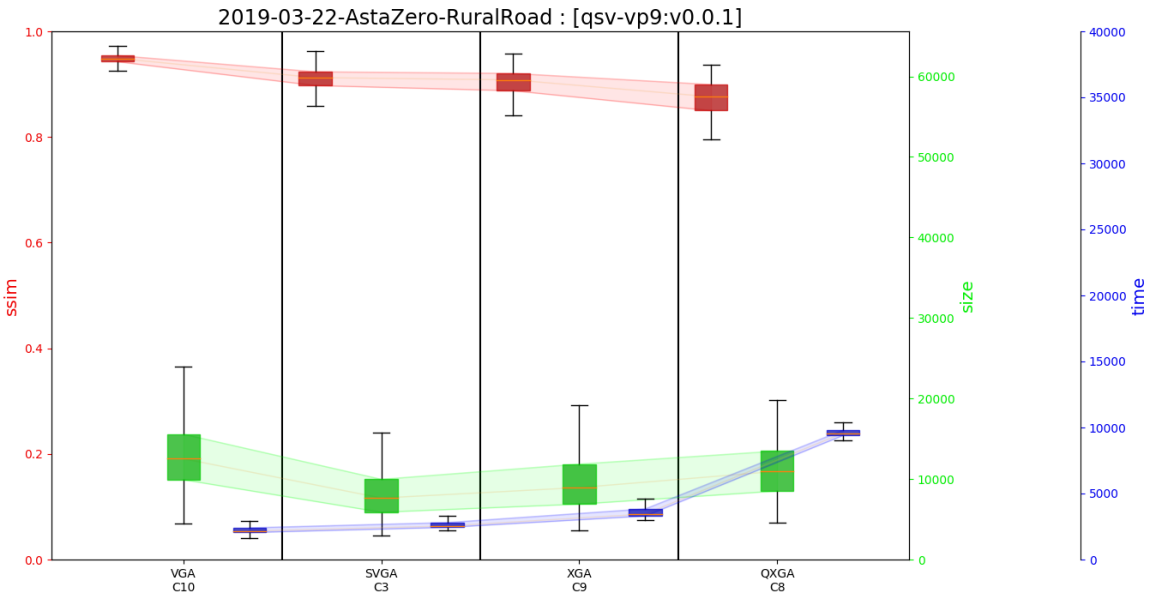

Fig. 31: Encoder graph for QSV-VP9 (AstaZero Rural Road)

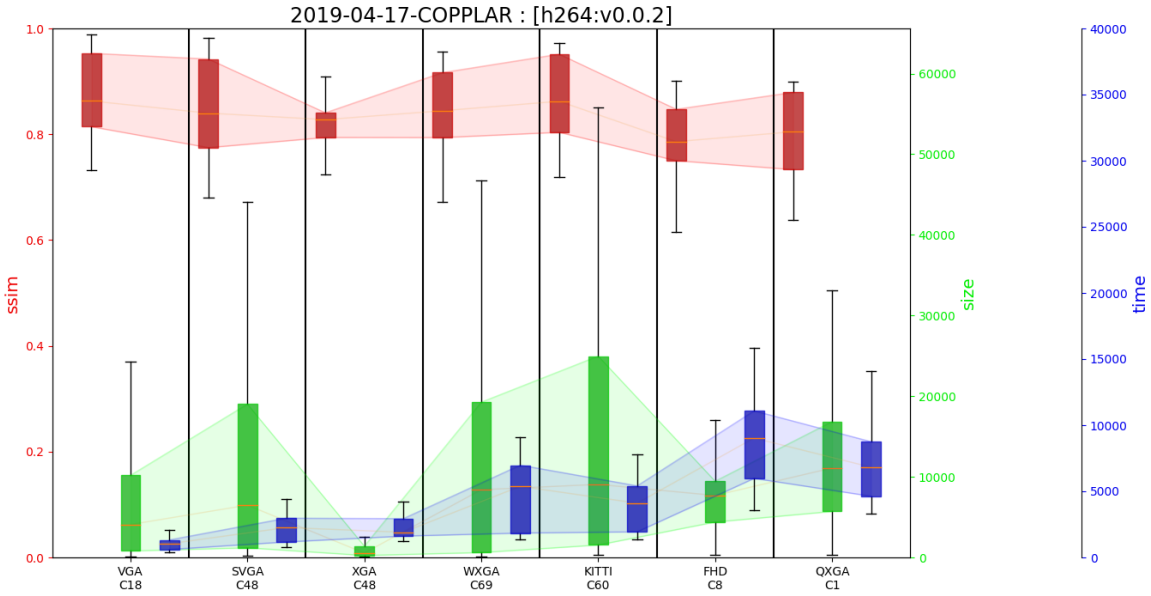

Fig. 32: Encoder graph for H264 (COPPLAR)

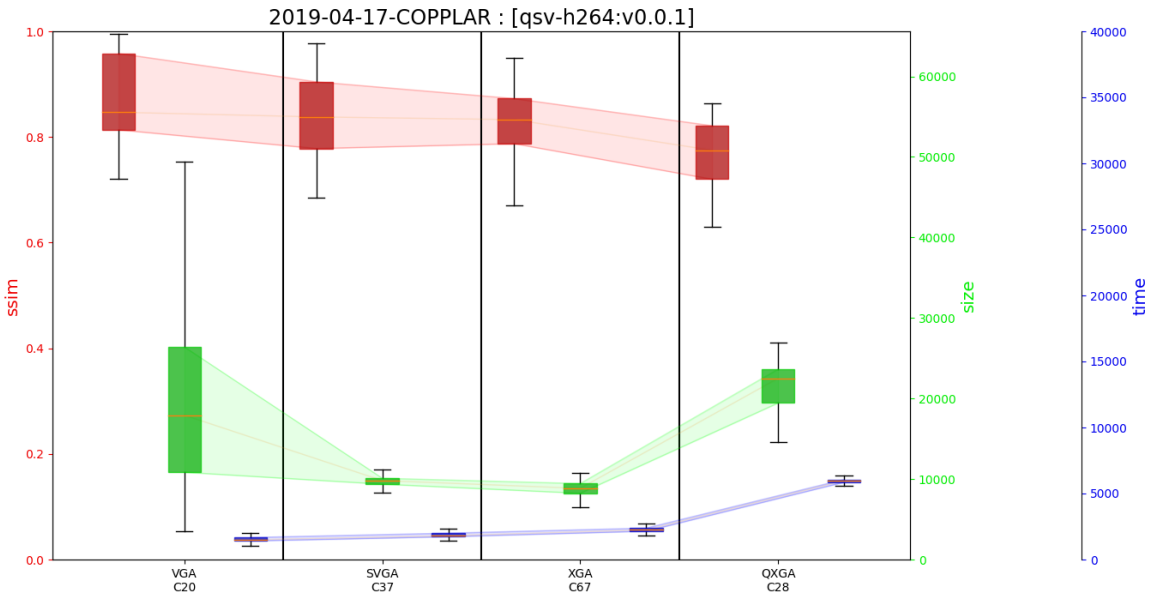

Fig. 33: Encoder graph for QSV-H264 (COPPLAR)

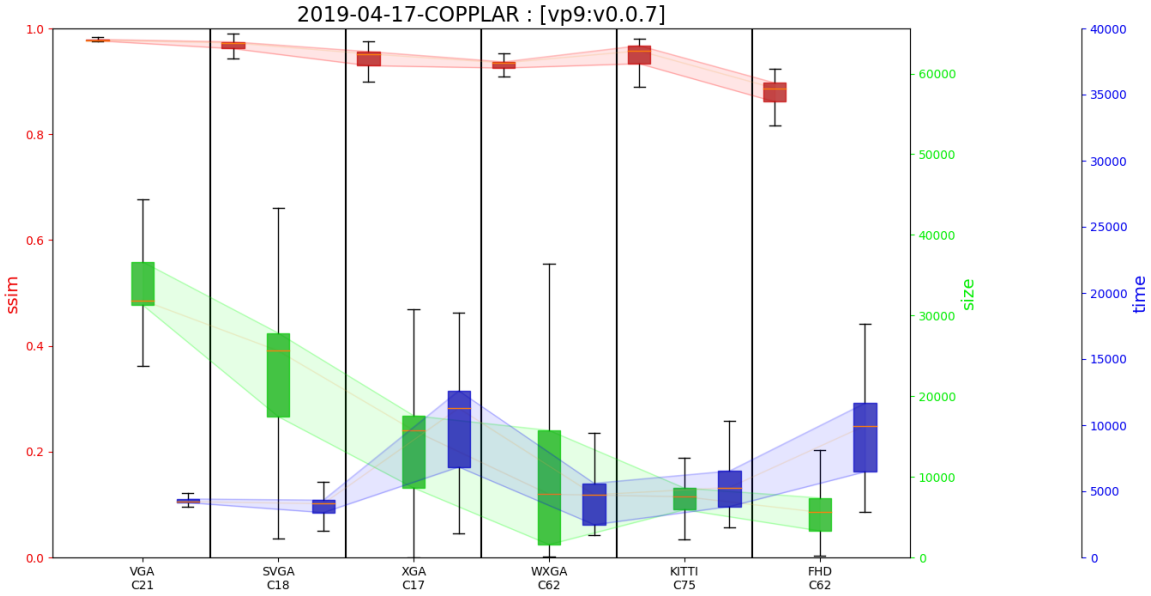

Fig. 34: Encoder graph for VP9 (COPPLAR)

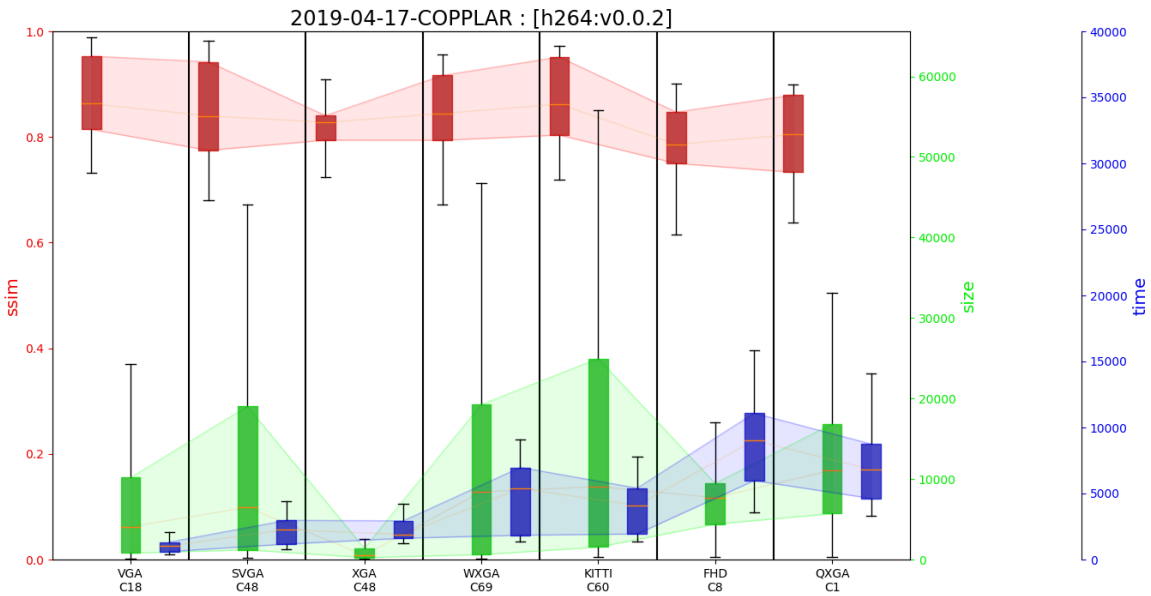

Fig. 35: Encoder graph for QSV-VP9 (COPPLAR)

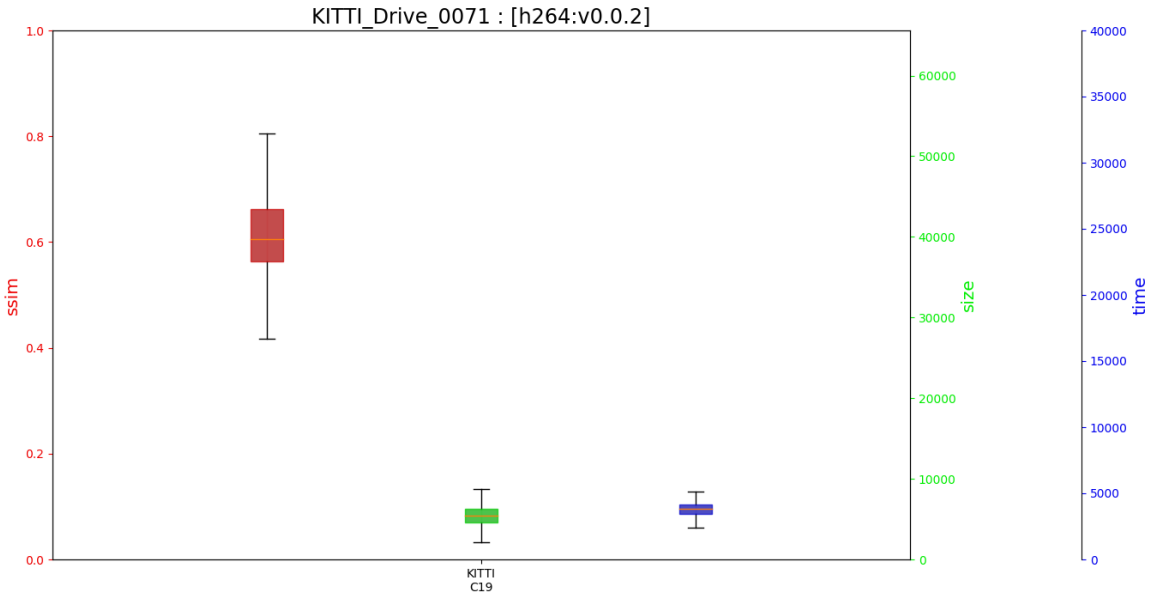

Fig. 36: Encoder graph for H264 (KITTI)

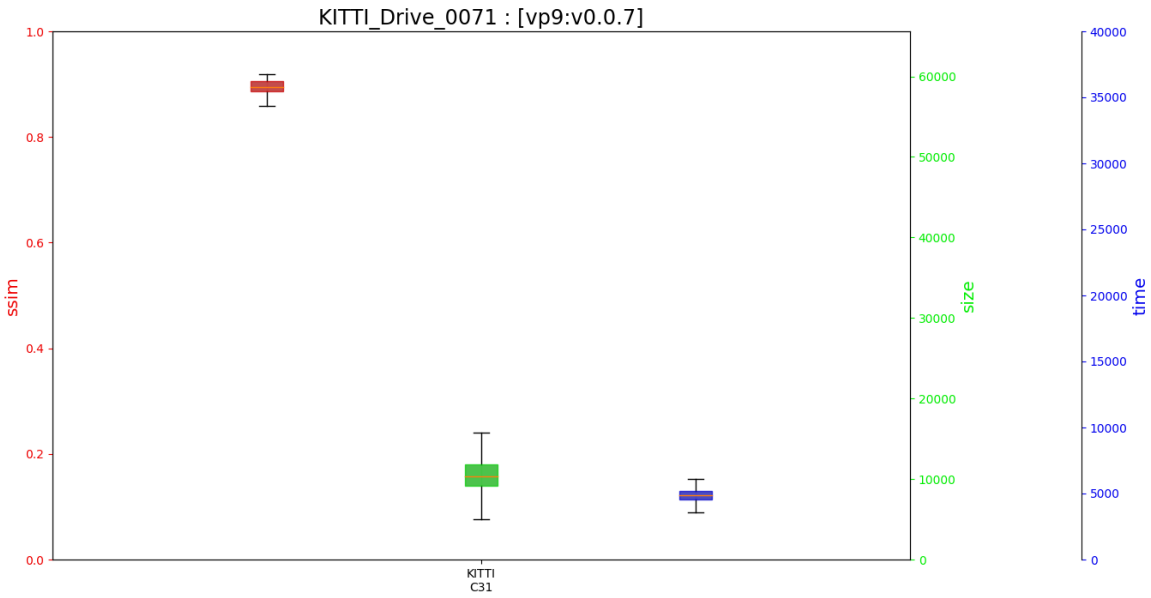

Fig. 37: Encoder graph for VP9 (KITTI)

D. Best candidate encoder configurations

1) KITTI:

List 1. VP9 encoder configuration KITTI resolution - C31

- gop: 59
- rc_dropframe_thresh: 19
- rc_resize_allowed: 0
- rc_resize_up_thresh: 28
- rc_resize_down_thresh: 61
- rc_undershoot_pct: 22
- rc_overshoot_pct: 43
- rc_min_quantizer: 43
- rc_end_usage: 1
- rc_buf_sz: 4160
- rc_buf_initial_sz: 2969
- rc_buf_optimal_sz: 4225
- rc_target_bitrate: 4872672
- kf_mode: 0
- kf_min_dist: 0
- kf_max_dist: 171
- VP8E_SET_CPUUSED: 9

2) AstaZero Rural Road:

List 2. VP9 encoder configuration VGA resolution - C34

- gop: 197
- rc_dropframe_thresh: 85
- rc_resize_allowed: 0
- rc_resize_up_thresh: 22
- rc_resize_down_thresh: 63
- rc_undershoot_pct: 51
- rc_overshoot_pct: 70
- rc_min_quantizer: 15
- rc_end_usage: 1
- rc_buf_sz: 2476
- rc_buf_initial_sz: 2719
- rc_buf_optimal_sz: 736
- rc_target_bitrate: 824902
- kf_mode: 0
- kf_min_dist: 0
- kf_max_dist: 0
- VP8E_SET_CPUUSED: 7

List 3. VP9 encoder configuration SVGA resolution - C52

- gop: 134
- rc_dropframe_thresh: 3
- rc_resize_allowed: 0
- rc_resize_up_thresh: 49
- rc_resize_down_thresh: 58
- rc_undershoot_pct: 94
- rc_overshoot_pct: 78
- rc_min_quantizer: 16
- rc_end_usage: 0
- rc_buf_sz: 3478
- rc_buf_initial_sz: 3250
- rc_buf_optimal_sz: 2794
- rc_target_bitrate: 4961811
- kf_mode: 1
- kf_min_dist: 1
- kf_max_dist: 63
- VP8E_SET_CPUUSED: 7

List 4. VP9 encoder configuration XGA resolution - C34

- gop: 41
- rc_dropframe_thresh: 45
- rc_resize_allowed: 0
- rc_resize_up_thresh: 39
- rc_resize_down_thresh: 19
- rc_undershoot_pct: 57
- rc_overshoot_pct: 75
- rc_min_quantizer: 24
- rc_end_usage: 1
- rc_buf_sz: 1158
- rc_buf_initial_sz: 2922
- rc_buf_optimal_sz: 3071
- rc_target_bitrate: 4710463
- kf_mode: 0
- kf_min_dist: 1
- kf_max_dist: 67
- VP8E_SET_CPUUSED: 8

List 5. VP9 encoder configuration WXGA resolution - C44

- gop: 215
- rc_dropframe_thresh: 23
- rc_resize_allowed: 0
- rc_resize_up_thresh: 27
- rc_resize_down_thresh: 73
- rc_undershoot_pct: 90
- rc_overshoot_pct: 64
- rc_min_quantizer: 28
- rc_end_usage: 0
- rc_buf_sz: 2299
- rc_buf_initial_sz: 1455
- rc_buf_optimal_sz: 3708
- rc_target_bitrate: 3641195
- kf_mode: 0
- kf_min_dist: 1
- kf_max_dist: 104
- VP8E_SET_CPUUSED: 7

List 6. VP9 encoder configuration KITTI resolution - C37

- gop: 210
- rc_dropframe_thresh: 18
- rc_resize_allowed: 1
- rc_resize_up_thresh: 3
- rc_resize_down_thresh: 34
- rc_undershoot_pct: 14

- rc_overshoot_pct: 60
- rc_min_quantizer: 23
- rc_end_usage: 0
- rc_buf_sz: 4678
- rc_buf_initial_sz: 1961
- rc_buf_optimal_sz: 4562
- rc_target_bitrate: 3269399
- kf_mode: 0
- kf_min_dist: 1
- kf_max_dist: 146
- VP8E_SET_CPUUSED: 7

List 7. VP9 encoder configuration FHD resolution - C26

- gop: 150
- rc_dropframe_thresh: 27
- rc_resize_allowed: 1
- rc_resize_up_thresh: 91
- rc_resize_down_thresh: 99
- rc_undershoot_pct: 0
- rc_overshoot_pct: 36
- rc_min_quantizer: 38
- rc_end_usage: 1
- rc_buf_sz: 2768
- rc_buf_initial_sz: 1792
- rc_buf_optimal_sz: 924
- rc_target_bitrate: 1373290
- kf_mode: 0
- kf_min_dist: 0
- kf_max_dist: 0
- VP8E_SET_CPUUSED: 8

List 8. QSV-VP9 encoder configuration QXGA resolution - C8

- intraPeriod: 127
- rcParams.bitRate: 3216
- ipPeriod: 17
- rcParams.initQP: 41
- rcParams.minQP: 3
- rcParams.maxQP: 4
- rcParams.disableFrameSkip: 1
- rcParams.diffQPIP: 48
- rcParams.diffQPIB: 28
- numRefFrames: 9
- rcMode: 1
- referenceMode: 0

3) COPPLAR:

List 9. VP9 encoder configuration VGA resolution - C21

- gop: 1
- rc_dropframe_thresh: 36
- rc_resize_allowed: 1
- rc_resize_up_thresh: 100
- rc_resize_down_thresh: 45
- rc_undershoot_pct: 39
- rc_overshoot_pct: 100
- rc_min_quantizer: 15
- rc_end_usage: 1
- rc_buf_sz: 6000
- rc_buf_initial_sz: 4000
- rc_buf_optimal_sz: 4673
- rc_target_bitrate: 3416808
- kf_mode: 1
- kf_min_dist: 0
- kf_max_dist: 178
- VP8E_SET_CPUUSED: 8

List 10. VP9 encoder configuration SVGA resolution - C18

- gop: 157
- rc_dropframe_thresh: 0
- rc_resize_allowed: 1
- rc_resize_up_thresh: 0
- rc_resize_down_thresh: 97
- rc_undershoot_pct: 99
- rc_overshoot_pct: 15
- rc_min_quantizer: 13
- rc_end_usage: 0
- rc_buf_sz: 6000
- rc_buf_initial_sz: 1202
- rc_buf_optimal_sz: 0
- rc_target_bitrate: 3935011
- kf_mode: 1
- kf_min_dist: 1
- kf_max_dist: 163
- VP8E_SET_CPUUSED: 9

List 11. VP9 encoder configuration XGA resolution - C17

- gop: 173
- rc_dropframe_thresh: 7
- rc_resize_allowed: 1
- rc_resize_up_thresh: 12
- rc_resize_down_thresh: 72
- rc_undershoot_pct: 98
- rc_overshoot_pct: 39
- rc_min_quantizer: 22
- rc_end_usage: 0
- rc_buf_sz: 3507
- rc_buf_initial_sz: 1355
- rc_buf_optimal_sz: 1770
- rc_target_bitrate: 2720561

- kf_mode: 1
- kf_min_dist: 0
- kf_max_dist: 165
- VP8E_SET_CPUUSED: 8

List 12. VP9 encoder configuration WXGA resolution - C62

- gop: 226
- rc_dropframe_thresh: 61
- rc_resize_allowed: 1
- rc_resize_up_thresh: 48
- rc_resize_down_thresh: 60
- rc_undershoot_pct: 5
- rc_overshoot_pct: 68
- rc_min_quantizer: 30
- rc_end_usage: 1
- rc_buf_sz: 431
- rc_buf_initial_sz: 146
- rc_buf_optimal_sz: 1060
- rc_target_bitrate: 4366560
- kf_mode: 1
- kf_min_dist: 0
- kf_max_dist: 140
- VP8E_SET_CPUUSED: 9

List 13. VP9 encoder configuration KITTI resolution - C75

- gop: 92
- rc_dropframe_thresh: 66
- rc_resize_allowed: 1
- rc_resize_up_thresh: 28
- rc_resize_down_thresh: 42
- rc_undershoot_pct: 76
- rc_overshoot_pct: 84
- rc_min_quantizer: 20
- rc_end_usage: 1
- rc_buf_sz: 1578
- rc_buf_initial_sz: 2482
- rc_buf_optimal_sz: 1222
- rc_target_bitrate: 2255460
- kf_mode: 1
- kf_min_dist: 0
- kf_max_dist: 94
- VP8E_SET_CPUUSED: 6

List 14. VP9 encoder configuration FHD resolution - C62

- gop: 90
- rc_dropframe_thresh: 48
- rc_resize_allowed: 1
- rc_resize_up_thresh: 94
- rc_resize_down_thresh: 50
- rc_undershoot_pct: 7
- rc_overshoot_pct: 61
- rc_min_quantizer: 33
- rc_end_usage: 1
- rc_buf_sz: 3488
- rc_buf_initial_sz: 304
- rc_buf_optimal_sz: 84
- rc_target_bitrate: 165729
- kf_mode: 1
- kf_min_dist: 0
- kf_max_dist: 12
- VP8E_SET_CPUUSED: 8

List 15. QSV-VP9 encoder configuration QXGA resolution - C32

- intraPeriod: 43
- rcParams.bitRate: 2034
- ipPeriod: 48
- rcParams.initQP: 0
- rcParams.minQP: 23
- rcParams.maxQP: 27
- rcParams.disableFrameSkip: 0
- rcParams.diffQPIP: 11
- rcParams.diffQPIB: 26
- numRefFrames: 16
- rcMode: 3
- referenceMode: 1

APPENDIX C
SOFTWARE ARCHITECTURE

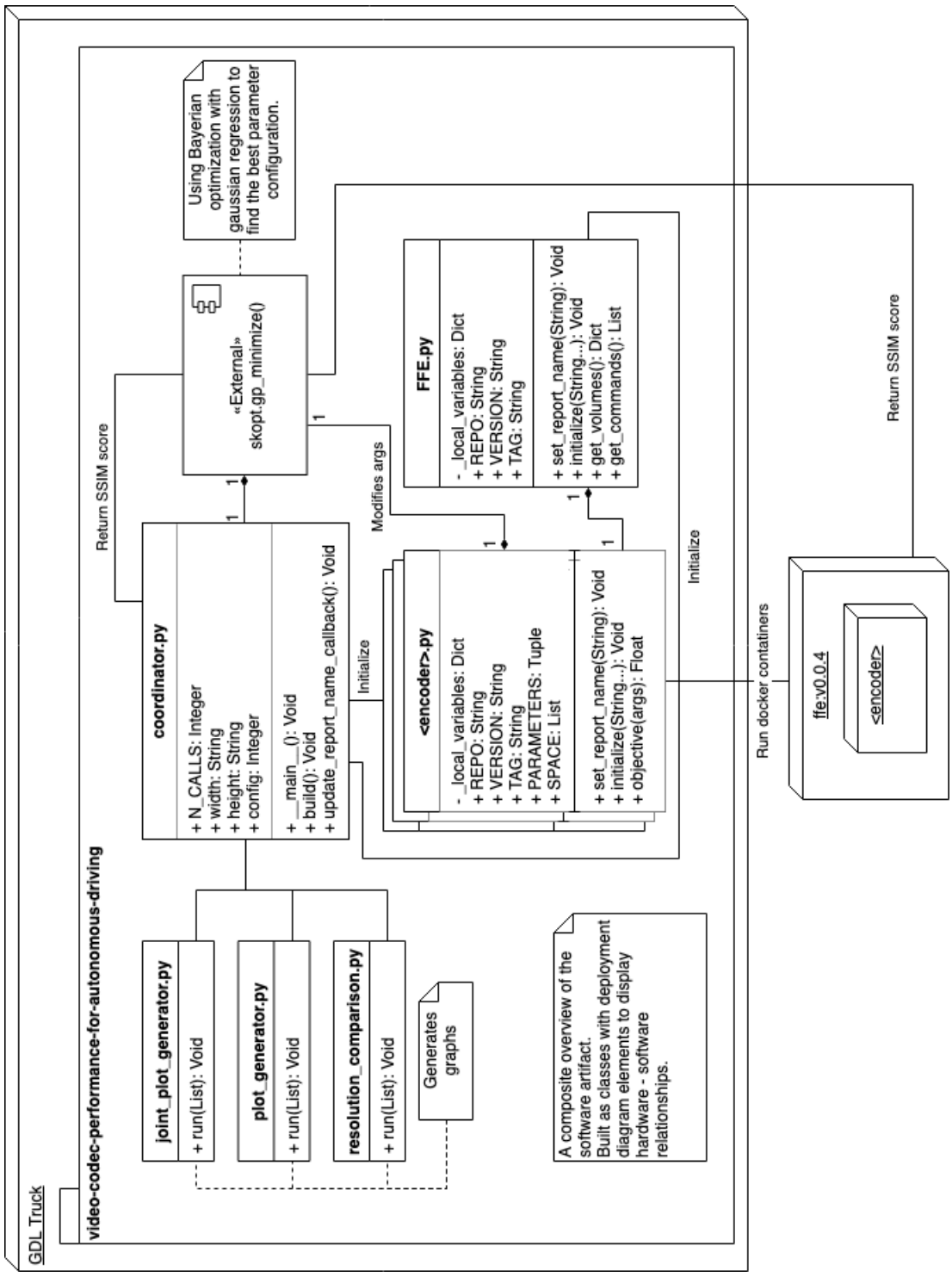

Fig. 38: Composite system architecture of CS