EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

THU
Technische
Hochschule
Ulm

The Effect of Image Coding on CMS Image Quality Parameters Using Embedded Video Coding of an MPSoC FPGA

Jannik Mehrke, Anestis Terzis, Oliver Bringmann

***Abstract*—One of the main sensors used to enable advanced driver assistance systems and autonomous driving are cameras. The image signal processing, required for this, is very demanding in terms of computing power and real-time requirements. The overall project validates the outsourcing of the very demanding image processing algorithm into an FPGA cloud. Therefore, a tradeoff between image quality of the compressed signal and real-time process is analyzed and discussed. When using more compression to the image, transmission of the data into the cloud is faster, but the quality of the image for the object detection is worse. This paper shows how the effect of image coding on image video parameter can be measured and validates the results.**

***Index Terms*—Camera Monitor System, Cloud, MPSoC FPGA, Hybrid Image Processing, Image Compression**

I. Introduction

Cloud-based data processing has become more important within the last decade. Scalability, distributivity, interoperability, modularity are some of the main advantages of cloud computing [1]. The consideration of new cloud-based architectures is a promising approach because of the increasing processing requirements for advanced automotive use cases [2]. A novel hybrid cloud-based processing platform, based on FPGAs, is designed and exemplary implemented within this work. The main purpose of the overall project is to generate a platform to enable investigations of the effects of different parameters on real-time cloud-based processing architectures. The chosen example architecture is the implementation of a Camera-Monitor-System (CMS) as Mirror replacement in vehicles as introduced in [3]. The normative framework for CMS as mirror replacement is defined in [4].

This paper focuses on the real-time image compression of the data that are transmitted to a cloud and on the effects of the compression on the image quality parameters. The purpose is to make the effect of image compression on image video parameter measurable. The overall system generates image data in a vehicle and transmits these data to a cloud system for object detection or scene interpretation. Because of the size of the generated raw-image data and the available bandwidth for the sending during driving, compression of the raw-data is indispensable. Therefore, two state of the art real-time embedded image coding algorithms are evaluated according to their processing time, compression ratio and effects on the image quality parameters with the introduced platform.

The effects on the image quality parameters are measured with a modified object detection algorithm. This algorithm is filled with generated videos. These videos are simulated night drives in the laboratory. The night scenario is chosen because of the low information content of the images . Vehicles are seen as point light sources (PLS) in night scenarios. The predominant part of the images is dark, so it does not include any information. Only some areas inside the images include information other than "darkness". Depending on the configuration of the image compression algorithms, they can plane those non-dark areas for example because of their minor size. With the planing of those areas, the information inside the image gets lost, which has major impacts on the object detection. For the validation of the effects, a night algorithm is used. This night algorithm detects point light sources in dark images. For the validation of the effects on image quality parameter, the results of detected vehicles before and after compression are compared. The results show major bias for the different compression algorithms with varying configurations.

This paper is organized as follows, the next section introduces the implemented and modified night algorithm. Section 3 focuses on the real-time image compression and the configuration possibilities. Section 4 defines the source material for the validation before section 5 shows and discusses the results of the measurements. At the end a conclusion is drawn.

II. Implementation of the Night Algorithm

The implemented MATLAB Night Algorithm is based on a "Haar algorithm" [5]. The Night algorithm uses Haar feature detection to detect point light sources (PLS) in the first stage. The second stage searches PLS with the same or close radius within a defined area around the source. When there is another PLS

Jannik Mehrke, Jannik.Mehrke@thu.de, Anestis Terzis, Anestis.Terzis@thu.de, both at University of Applied Science Ulm. Oliver Bringmann, Oliver.Bringmann@uni-tuebingen.de, Eberhard-Karls-University Tuebingen.

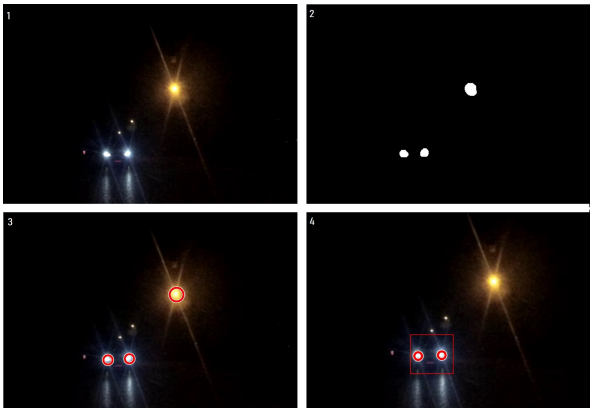

Figure 1. Steps of the Night Algorithm.

that matches the origin, those two PLS are merged to one detected vehicle. At the end, frames with detected vehicles and the number of vehicles are counted for the validation of the results. The PLS within this context all comply with the normative framework of the UN Regulation Number 46 [6].

In detail, the Night Algorithm works as follows: The binarization of the image is the first step to achieve a black and white image. Therefore, an offset value is defined to decide whether one pixel is black or white. Then a Haar-based approach is used for the edge detection as described in [5]. This approach detects transitions between black and white in the image. Once all the edges are detected, the MATLAB function "imfindcircles" [7] is used to detect circles along the found edges. This function provides their diameter and center position when a circle is detected. It searches circles in images whose radii are approximately equal to a radius within a defined range. The last step compares all found objects with each other. When two objects within a defined range have the same or similar radius, these two are merged to one vehicle. Figure 1 shows the individual steps of the night algorithm. The first image shows the raw data. The result of the binarization is visualized in the second image. In the third image, all detected circles are highlighted. This stage detects all lights in the image. This is the reason why the street light is also detected as a point light source. In the last picture, the final stage of the algorithm is visualized. Both point light sources of the vehicle are connected and highlighted.

The implemented Night Algorithm is not a state of the art benchmark algorithm. This algorithm was implemented to make the effect of compression measurable and for the validation of the concepts based on criteria of the UN Regulation Nr 46. This algorithm analyzes the generated video files offline after their generation. It uses elementary detection methods and is only suitable for the detection of normative point light sources. Within real driving scenarios there are various situations and constellations within the image for which this algorithm is not suitable.

Table I
RESOURCE UTILIZATION OF THE SENDER SIDE DESIGN.

Resource	Utilization	Available	Utilization [%]
LUT	58883	230400	25.56
LUTRAM	4774	101760	4.69
FF	81864	460800	17.77
BRAM	69	312	22.12
URAM	44	96	45.83
DSP	97	1728	5.61
IO	22	360	6.11
GT	3	20	15.00
BUFG	26	544	4.78
MMCM	3	8	37.50

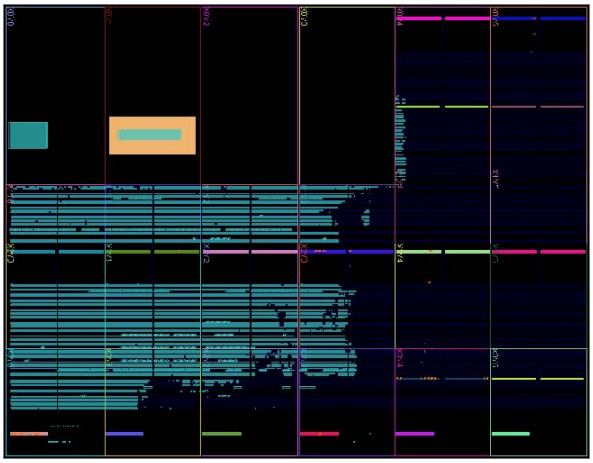

Figure 2. Resource usage on the chip of the sender side component.

III. IMAGE COMPRESSION

For the implementation of the introduced system, a Xilinx Zynq UltraScale+ MPSoC ZCU106 FPGA board [8] is used. The board provides many FPGA resources. The resource utilization is listed in Table I. Graphical representations of the resource utilization and the routing on the FPGA are provided in Figure 2 and Figure 3. The used SoC component is a state of the art FPGA-based SoC for modern ADAS and streaming applications. One main advantage of this chip is that it comes with a hardware implemented Video Codec Unit (VCU) [9] within the SoC. This VCU can be used to hardware-implement a H.264 or a H.265 Video Codec. Both are state of the art video codecs that are used at real-time streaming platforms. These two codecs come with a bunch of configuration options. The configuration features of primary focus are shortly introduced within the following sections. Another advantage of the FPGA-based system design is the energy consumption of the complete system. FPGA chips are more power efficient than CPU or GPU based chips according to a study of Inaccel [10].

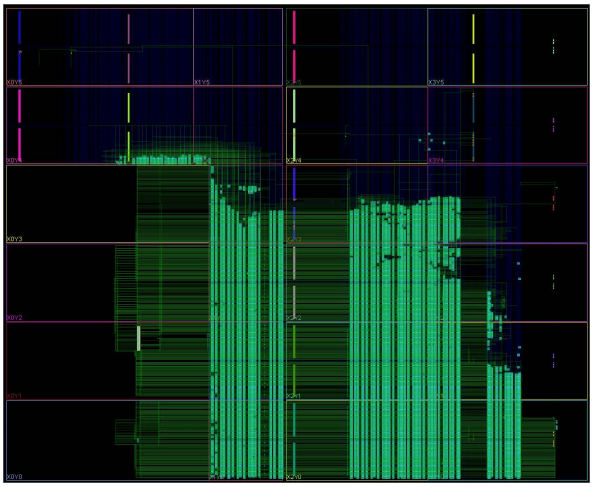

Figure 3. Routing on the chip of the sender side component.

The overall power consumption of the sending side design at full power is $6.195W$.

The relevant streaming pipeline for this paper consists of mainly three parts:

- the HDMI input
- the image pre-processing
- and the image compression and transmitting over Ethernet.

All parts are controlled or processed by the FPGA. The HDMI input provides the raw image data to the FPGA. The FPGA itself then performs the pre-processing. Within the pre-processing the image data can be scaled. The scaling includes a horizontal and vertical scaling of the image. Therefore, two factors are defined as scaling factors from the original size to a smaller size. Upscaling of the image is not implemented within this project, only downscaling is possible. In addition a frame rate splitter is implemented. With the splitter it is also possible to define a lower frame rate for the transferred data. The pre-processing block can be used as an additional data reduction block for further applications. With this block the image data can be pre-processed to reduce the amount of transported data. It is implemented as a black box that is running on the FPGA. The user sets the three parameters within the streaming pipeline to configure this block. By default all three parameters are set to 1. This causes that the raw image is transferred to the image compression. The last part is the compression and sending part. These are two hardware components that are only configured by the FPGA. The following sections describe the used configuration possibilities of the image compression in details.

A. Target bitrate

The variable target bitrate defines the allowed bit rate after the compression. The encoder will try to achieve the defined value to average throughout the video. The unit of target bit rate is $[\frac{kbit}{s}]$.

B. Number of Slices

The number of slices defines how many slices per frame are used with the H.265 encoder. The H.265 encoder is a so called differential encoder. It only sends differences of the actual frame to the previous frame. Therefore, it uses some kind of movement detection within the frame. This movement detection is done within slices. Each slice contains one or several full rows of the image in which differences are searched. The more slices are used, the more parallel operations can be done. The slices are spread over the frame as regularly as possible. A higher number of slices leads to a greater amount of required computational power. This is caused by a more parallel processing architecture. Each parallel process needs computational power to be processed as fast as possible. The number of slices that can be used on the SoC is limited by the processing power of the VCU. The hard-IP VCU component in the SoC is sufficient for parallel processing architectures until 16 slices. The calculation needs too many threads when using a higher number of slices which the processor can not process in parallel anymore. Then the system can not reach the real-time requirement because the overall processes take to long and each process is blocking the others by reserving and using the limited resources on the chip.

C. Latency Mode

The latency mode defines the used encoder latency mode, which defines the focus of the compression. There are two important modes for this system, the normal mode and the low latency mode. The focus of the normal mode is on quality while the focus of the low latency mode is on timing. Low latency mode can for example increase the number of slices for parallelism, which leads to less computational time but more computational power. It also provides subframe level output. This leads to the effect that the already processed parts of the images are forwarded while processing the other areas of the image. Common methods, like the normal mode, process the whole image before forwarding the results.

D. Configuration of the hardware components

For the running of the system, streaming pipelines need to be configured on the sending and receiving side. Therefore, videos are stored on the FPGA-based sending component. These videos are the basis for the compression and later for comparison of the results. The FPGA-based board is used to encode the videos and send them via UDP protocol over Ethernet to a computer. The receiving component decodes the UDP data and stores the decoded videos. On both devices the pipeline-based multimedia framework GStreamer [11] is installed and implemented to configure and start the streaming pipelines. Two different GStreamer

Table II
GENERATED VIDEOS FOR THE VALIDATION. LISTED ARE THE RESOLUTION (RES) [PX], THE FRAME RATE (FR) [FPS], THE VELOCITY OF THE VISUALIZED VEHICLES (VEL)[$\frac{m}{s}$], THE SIZE OF THE VIDEO FILES (SIZE) [MB] AND THE LENGTH OF THE VIDEO (LEN)[S].

Nr	Res[px]	FR[fps]	Vel [$\frac{m}{s}$]	Size[MB]	Len[s]
1	2560x1440	60	0,39	98,9	15
2	1920x1080	60	0,1545	95,6	35
3	1280x720	60	0,39	66,4	18
4	1280x720	60	0,1545	119,2	32
5	2704x1520	60	0,39	63,3	24
6	1024x576	15	N.A.	58,6	7

pipelines must be started on the different devices. Each pipeline configures the needed parameters for the different processing parts as described in the section III.

E. Precision and Recall

In the field of Object detection there are a variety of metrics available to compare different algorithm according to their accuracy. Two commonly used metrics are the Precision P and the Recall R. Algorithms sometimes detect objects without them really being in the image. The Precision represents those and provides a ratio. The Precision is defined as the number of correct detected objects divided by the total number of detected objects. This generates a number between 0 and 1. The higher the number, the better the algorithm. The second value is the Recall R. The Recall defines how many objects in the ground truth are detected correctly. Therefore, the Recall is defined as the number of correctly detected objects divided by the total number of objects in the ground truth. This also generates a number between 0 and 1. A higher number indicates a better result. Both variables are later used to evaluate the results of the detection. The number of detected objects in the raw images defines the ground truth. In case of a changed number after compression, an effect of the compression is measured.

IV. VIDEOS FOR THE INVESTIGATION OF THE ALGORITHM

For the measurement of the influences, different videos were produced containing different scenarios. For this paper, five laboratory videos and one real-driving video are used. The laboratory videos all show a model vehicle that drives with different constant velocities towards the camera. The camera is configured with different resolutions and fields of view for the recording. The real-driving video is a real-drive through a city at night. This is used to validate the theoretical results in real scenarios. The details of the videos are listed in the Table II.

Table III
BENCHMARK VALUES FOR THE DIFFERENT VIDEOS GENERATED WITH THE MATLAB NIGHT ALGORITHM.

Name	Frames with vehicles	Detected Vehicles
Video1	799	799
Video2	1348	1348
Video3	417	417
Video4	771	771
Video5	1278	1278
Video6	394	652

The different configurations for the cameras indicate different hardware components within the final mirror-replacement system. Because $Video6$ contains a real-driving scenario, the velocity of the pictured vehicles is unknown.

These videos are sufficient for the overall proof of concept. They represent a real driving scenario at night with different normative and reproducible configurations. The effects of different compression factors has major impacts on the quality of these videos and can be measured with the implemented night algorithm.

V. EFFECTS

This section presents the results of the investigation. While the coding time and compression ratio are directly measured, the effects of the compression on the image quality parameters are investigated with the night algorithm. For this investigation, the number of frames with detected vehicles in the uncompressed videos and the total number of detected vehicles in all frames are used as benchmark for the effects in the first step. The uncompressed videos showed the following results in the test (see Table III).

In the first step only the numbers of detected objects are compared. In the second step, the quality of the results is measured with the Precision and Recall functions. The results for all videos are nearly the same, therefore in the following sections only the results for $Video1$ are presented.

A. Compression ratio

The compression ratio of image coding is defined as the relative size of data representation produced by a data compression algorithm. To calculate the compression ratio for the different algorithms, the sizes of the videos before and after coding are compared. These rates are calculated for both H.264 and H.265 with different target bitrates. The higher the target bitrate, the less compression is needed to achieve this value. The results can be seen in Table IV.

These results are generated with the same configurations for both algorithms. The target bitrate values are configured from $50000\frac{kbit}{s}$ down to $1\frac{kbit}{s}$. These limits are chosen to have a wide spectrum of compression ratios from nearly uncompressed to extremely

Table IV
VIDEO SIZE FOR DIFFERENT TARGET BITRATES AFTER H.264 AND H.265 ENCODING.

Name	Size H.264 enc. [MB]	Size H.265 enc. [MB]
Raw	98,9	98,9
TB 50000	76,9	76,9
TB 40000	61,5	61,5
TB 30000	64,2	64,1
TB 20000	31,1	30,8
TB 10000	15,800	15,600
TB 7500	12,000	11,700
TB 5000	8,060	7,980
TB 4500	7,270	7,000
TB 4000	6,510	6,230
TB 3500	5,660	5,450
TB 3000	4,840	4,630
TB 2500	4,040	3,860
TB 2000	3,170	3,100
TB 1500	2,390	2,350
TB 1000	1,610	1,590
TB 750	1,230	1,220
TB 500	0,850	0,864
TB 250	0,520	0,483
TB 100	0,517	0,363
TB 50	0,478	0,369
TB 1	0,505	0,384

compressed. While $50000\frac{kbit}{s}$ is achievable for the Video Codec, $1\frac{kbit}{s}$ could not be achieved. The Video Codec adjusted itself to a value close to the configured one. That is the reason why the results for $1\frac{kbit}{s}$ and $50\frac{kbit}{s}$ are incorrect and not representable results. The first correct result is at $100\frac{kbit}{s}$. This investigation shows that the H.265 algorithm is slightly better than the H.264 algorithm according to the achieved data size after compression. This directly causes a faster transmission of the data to the cloud. The results are as expected at the beginning of this investigation. The next step is to discuss the effects of the target bitrate on the object detection.

B. Effects of target bitrate

The night algorithm can directly measures the effects of the target bitrate. Therefore, the video-files with different target bitrate configurations are loaded into the Night algorithm. In first instance the algorithm only counts the frames with detected vehicles. The next step is to have a closer look at the results and calculate their Precision and Recall. The results of the frames with detected vehicles can be seen in Figure 4. The results for the Precision for different target bitrates as shown in Figure 5.

The results show that right from the beginning the H.265 algorithm is better than the H.264 algorithm. While decreasing the target bitrate, the quality of the video also decreases. At some point, the algorithms can not detect the point light sources correctly any more. Then in some frames, more than one PLS is detected at the real position (false-positive detection) or sometimes

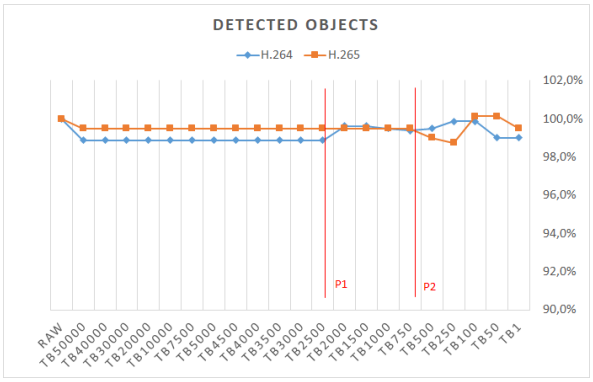

Figure 4. Percentage of frames with detected vehicles compared to the raw video for different target bitrates.

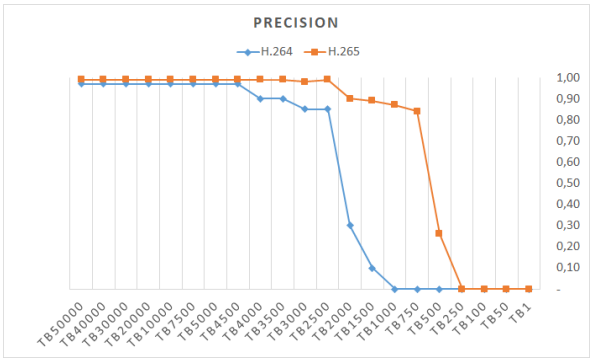

Figure 5. Precision for the codecs for different target bitrates.

none is detected (false-negative detection). This leads to an incorrect result and causes a bad Precision and Recall. Within these graphs two lines are drawn. These two lines indicate the points where Precision and Recall got bad for the compression algorithm. At point $P1$ the H.264 algorithm leads to bad results. At this point the number of false-positive and false-negative detections increases drastically. With a target bitrate of $2500\frac{kbit}{s}$ and lower the H.264 algorithm is not sufficient for the task anymore. The H.265 algorithm can achieve significant better results until point $P2$. With an target bitrate lower or equal to $750\frac{kbit}{s}$, the H.265 algorithm is not sufficient anymore. This result is also as expected, while H.264 sends the whole frame, H.265 only sends the differences between the last frame and the current frame. Therefore, the amount of data to send is reduced and so it was expected to achieve a higher compression without quality loss. At the part after point $P2$ the lines begin to vary. This is due to the fact that the quality of the image is too bad such that many objects are detected although there is only one PLS. The Precision and Recall at this stage become bad.

C. Effects of number of slices

The effects of the number of slices is measured the same way as the effects of the target bitrate. The

Table V
AVERAGE TIMING FOR THE H.264 AND THE H.265 ALGORITHM FOR DIFFERENT PROFILES.

Name	Encoding time [s]	Decoding time [s]
H.265	7.3940	9.9288
H.264	7.2215	10.1485

number of slices are variate for a constant target bitrate. The major effect is seen in the needed bandwidth for the transportation. The number of slices has neither significant impact on the quality of the image nor on the time needed to encode or decode the image. The major impact is on the transportation. Each slice calculates its area and sends the results when finished. When one slice is used, the whole image is processed by this slice and at the end the whole image is send. When 12 slices are used, then each slice is processing its area and sends its results when finished. This generates 12 packets of frames that are sent compared to only one packet. For the transportation this means that less bandwidth at the same time is needed because of the separation of the packet caused by different processing times of the slices. As mentioned before, the overall time is nearly the same in all configurations because this depends mainly on the target bitrate.

D. Effects on timing

The results of the timing are presented in Table V. This table shows the average timing over all profiles for the H.264 and H.265 coding. It shows that in average the H.265 encoding takes more time than the H.264 encoding. This is due to the fact of the higher complexity of the H.265 encoding. As mentioned before, H.265 detects the differences within two frames and only encodes those parts while H.264 is encoding the whole image with the same scheme. For the decoding, the H.265 algorithm is much faster than the H.264 decoding. This is due to the low data size that is needed to be decoded. This validation does not include the transmission of data to the cloud. As mentioned in section Table IV the size of the H.265 video is smaller than the H.264 encoded videos. In total the H.265 algorithm leads to a much faster transmission of the data although the computational complexity of the encoding is much higher.

E. Results

The investigations shows that the H.265 coding is more sufficient for real-time image compression than the H.264 coding. The faster process- and transportation time and the higher achievable compression cause this. The results are as expected, the new part within this paper is that now it is possible to measure the effect of the compression. From now on it is possible to define the lower boundary of the data transmission in the overall system concept. It was known that a higher compression generates a lower quality of the image but with lower coding- and transportation-latency. As mentioned in the beginning of this paper, the overall project investigates a real-time cloud processing architecture. Depending on the quality of the connection between the vehicle and the cloud, the image compression can now be adjusted. For low-bandwidth connections, the target bitrate can be reduced to guarantee a working system. When the bandwidth is too low such that a target bitrate lower than $750\frac{kbit}{s}$ is needed for the H.265 algorithm, the system must shut itself down due to bad results caused by bad quality of the images in night scenarios.

VI. CONCLUSION

In this paper, the effects of image compression on image quality parameter are measured and discussed. Therefore, an object detection algorithm for normative point light sources (PLS) in night scenarios was implemented. This algorithm makes the effect of image compression for a specific CMS application measurable by counting detected vehicles and calculating the Precision and Recall. Several videos are generated with different configurations for different codecs. These videos are analyzed with the implemented algorithm and the results are compared. The evaluation platform is implemented on an FPGA-based SoC board. The effect on the image quality mainly depends on the compression ratio of the used video codec. A lower target bitrate leads to a higher compression but reduces the quality of the image. This reduction can be seen in the results by fewer detected vehicles and a worse Precision and Recall. As expected at the beginning, the H.265 video codec is more sufficient for the real-time image compression than the H.264 algorithm. The main advantage of this research is that now it is possible to find a minimum configuration for the compression with different codecs. When for example using a H.265 codec at night scenarios, it is possible to reduce the target bitrate until $750\frac{kbit}{s}$ without major impact on the Precision of the system. In the overall system, the configuration of the video codec is adjustable depending on the available cloud connection parameters. For example with more bandwidth available for the transmission of data into a cloud, a lower compression is used to achieve better detection results. The research in this paper generated a lower boundary for the transmission in night scenarios. When the available bandwidth of the cloud connection is below $750\frac{kbit}{s}$ for the H.265 algorithm, it is not sufficient to use the object detection in the cloud. In this case the system must be shut down until a "better" connection is available. Otherwise the amount of false-positive or false-negative results increases caused by the low image quality. In a future work, this investigation is done for day scenarios.

REFERENCES

[1] C. Kachris and D. Soudris, "A survey on reconfigurable accelerators for cloud computing," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2016, pp. 1–10.

[2] A. Iordache, G. Pierre, P. Sanders, J. G. D. F. Coutinho, and M. Stillwell, "High performance in the cloud with fpga groups," in *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, Dec 2016, pp. 1–10.

[3] A. Terzis, Ed., *Handbook of Camera Monitor Systems - The Automotive Mirror-Replacement Technology based on ISO 16505*. Springer International Publishing, 2016, vol. 1, no. 10.1007/978-3-319-29611-1.

[4] "Road vehicles — ergonomic and performance aspects of camera monitor systems — requirements and test procedures," *International Organization for Standardization*, 2019.

[5] D. Peleshko and K. Soroka, "Research of usage of haar-like features and adaboost algorithm in viola-jones method of object detection," in *2013 12th International Conference on the Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*, 2013, pp. 284–286.

[6] "Regulation No 46 of the Economic Commission for Europe of the United Nations (UNECE) - Uniform provisions concerning the approval of devices for indirect vision and of motor vehicles with regard to the installation of these devices," Addendum 45, Revision 6, 2016.

[7] MATLAB. (2020, Sep.) imfindcircles describtion. [Online]. Available: https://de.mathworks.com/help/images/ref/imfindcircles.html

[8] Xilinx, *ZCU106 Evaluation Board*, 2019.

[9] X. , *H.264/H.265 Video CodecUnit v1.2 - LogiCORE IP Product Guide*, 2020.

[10] Inaccel, "www.inaccel.com," *Inaccel*, February 2022.

[11] GStremer, *GStreamer - API reference guide*, 2020.

Jannik Mehrke studied electrical engineering and Information technology at the Ulm University of Applied Science. During his Bachelor study, he specialized on automotive electronics. Later he finished his Master's degree at the Technical University of Munich where he also studied in the field of electrical engineering with specialization on automation technologies. Prior to this he worked two years in an engineering office in the field of automated testing of vehicles. Since 2018, Jannik Mehrke works as a researcher at the Ulm Technical University of applied science in the field of embedded systems. His research focus on cloud-based hybrid image processing architectures on SoC platform.

Anestis Terzis is professor for digital systems design and the head of the Institute of Communication Technology at Ulm University of Applied Sciences in Germany. Prior to this, he was with Daimler AG for ten years and worked in the Group Research and Mercedes-Benz Cars Development, with a main focus on future advanced driver assistance systems. Anestis Terzis received his diploma in Communications Engineering from Ulm University of Applied Sciences and his doctoral degree in Electrical Engineering from the University of Erlangen-Nuremberg. He is an expert member of the standardization and regulation committees (ISO, SAE, IEEE) in the field of camera monitor systems.

Oliver Bringmann received the Diploma (M.S.) degree in computer science from the University of Karlsruhe, Karlsruhe, Germany, and the Doctoral (Ph.D.) degree in computer science from the University of Tübingen, Germany, in 2001. He was with the FZI Research Center for Information Technology, Karlsruhe, in various positions as a Department and Division Manager and a Member of the management board, until 2012. He has been a Professor and the Director of the Chair for Embedded Systems with the University of Tübingen since 2012, where he is also serving as the Vice Head of the Department of Computer Science since 2014. He has authored and coauthored of more than 220 publications in the area of electronic design automation, embedded system design, and SoC architectures for automotive electronics and edge devices. His current research interests include electronic design automation, embedded system design, timing and power analysis of embedded software, embedded AI architectures, hardware-enhanced security, and robust perception. Oliver Bringmann is an IEEE member.