TUM School of Computation, Information and Technology
Technische Universität München

Adaptive Live Video Streaming for Teleoperated Driving

Markus Hofbauer, M.Sc.

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzende(r):
apl. Prof. Dr.-Ing. Walter Stechele

Prüfer der Dissertation:
1. Prof. Dr.-Ing. Eckehard Steinbach
2. Prof. Dr.-Ing. Markus Lienkamp

Die Dissertation wurde am 31.03.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 03.11.2022 angenommen.

Abstract

Failures of autonomous vehicles are inevitable. One possible solution to cope with foreseeable failures is teleoperation. In teleoperated driving, a human operator controls the vehicle from a distance. The remote operator perceives the traffic situation via video streams from one or more cameras deployed in the vehicle. The control commands of the operator are transmitted to the vehicle. To resolve complex traffic situations and ensure safety, the operator requires a reliable low delay video transmission. This is particularly relevant for mobile networks with time-variant and limited transmission rates. To provide the remote operator with the best possible situation awareness, while matching the available transmission resources of the network, an adaptation scheme for the individual video streams is required.

In this thesis, we investigate adaptive video streaming for teleoperated driving. We consider the entire pipeline of a general teleoperated driving system and we propose four main additions: an in-lab teledriving system and streaming pipeline, a driver situation awareness assessment system, a traffic-aware multi-view adaptation scheme, and a preprocessor rate control approach designed for the resource-constrained encoding hardware of autonomous vehicles.

To evaluate all proposed methods in a controllable network and driving environment, we design a teleoperation framework that extends existing driving simulators by vehicle remote control. This framework includes a customizable user interface as well as a low delay video streaming pipeline. The streaming pipeline provides an adaptation interface for controlling the frame rate, frame size, bitrate, and quality of the video stream.

While a framework with low delay video streaming can provide the driver with the necessary visual information to understand the current situation, there is no guarantee that the driver actually recognizes all relevant elements correctly. As the second main contribution, we therefore propose a method for assessing the driver situation awareness in realtime. The driver's current situation awareness is measured using eye tracking and compared to the optimal situation awareness. The optimal situation awareness is estimated by a state-of-the-art region of interest prediction network, which we extended for multiple camera views. The proposed driver awareness model enables us to accurately measure the driver situation awareness, which has been validated in a user study for eight driving scenarios.

Next, we propose a traffic-aware multi-view adaptation scheme to provide the operator with the best possible situation awareness for the current traffic situation. The adaptation scheme first estimates the importance of each camera view based on the vehicle's realtime movement in traffic. Then, the optimal combination of frame rate, frame size, and target rate/quality is estimated using a quality-of-experience-driven multi-dimensional adaptation scheme to control the encoder of each individual video stream. Evaluated for three representative driving scenarios with a six camera setup, the proposed adaptation scheme increases the average video quality per camera by 5 % VMAF score compared to a uniform adaptation.

Finally, the hardware of autonomous vehicles is limited by size and cost. The vehicle is therefore often restricted to a single hardware encoder. Encoding all camera views at the same time with a single encoder requires the combination of the individual frames into a single superframe. While this is a possible solution for streaming multiple camera views, it prevents the adaptation of the individual camera views. In this thesis, we propose a preprocessing concept that allows for individual rate/quality adaptation while using a single encoder. The preprocessing filters control the frame rate, frame size, and target rate/quality of the individual frames before combining them into a single superframe. For three representative driving scenarios, the proposed approach achieves the same video quality for the most important views as when using multiple encoders, while it causes only

1.8 % quality reduction on the remaining views. In comparison, an approach using a single encoder without preprocessing causes quality degradations of 4.5 % for the remaining views and even 5.6 % on the most important views.

Acknowledgement

The work presented in this dissertation was carried out as a member of the academic staff at the Chair of Media Technology (LMT) at the Technical University of Munich. Many people have supported me, personally as well as professionally, during the past years.

First of all, I would like to express my deep gratitude to my supervisor Prof. Dr.-Ing. Eckehard Steinbachfor the opportunity to join his research group. He provided me with endless support, crucial remarks, and the freedom to develop my own ideas that helped me to grow personally and reach all my goals at university.

Furthermore, I would like to thank Prof. Dr.-Ing. Markus Lienkamp for agreeing to become the second examiner and apl. Prof. Dr.-Ing. Walter Stechele for chairing the thesis committee.

Many thanks go also to our industry partner at BMW, Dr. Goran Petrovic for initiating my research project, his support during our interesting and fruitful collaborations, and his mentorship during my Ph.D.

Most of all I would like to thank the great team at LMT that I had the pleasure to spend the last three years with. While I highly value all my colleagues at LMT, I feel obliged to highlight a few people. I am particularly grateful to my colleague, office mate, and friend Christopher Kuhn for our great collaboration and fruitful discussions on a daily basis. I am looking forward to our future challenges and collaborations. Furthermore, I would like to extend special thanks to Dr. Christoph Bachhuber for great discussions and support even after he had left the LMT. I thank Dr. Tamay Aykut for interesting and joyful discussions during and beyond our common time at LMT and look forward to future collaborations. Furthermore, I would like to thank Dr. Rahul Chaudhari, Martin Piccolrovazzi, Yuankai Wu, and Andreas Noll for interesting discussions and fruitful collaborations. I am also grateful to all my former students, especially Lukas Püttner and Mariem Khlifi, who contributed to our publications.

My appreciation also goes to the professional administrative support provided by Marta Giunta, Simon Krapf, and Etienne Meyer. Special thanks are devoted to Dr. Martin Maier, for his immediate and ongoing support to me and all my colleagues throughout his time at LMT.

Finally I would like to thank my family, in particular my beloved parents, my friends Robin Beutner and Kevin Meyer, and my wonderful girlfriend Sophia, who supported me throughout my time at LMT.

Markus Hofbauer, Munich, March 15, 2022

Contents

Abbreviations

ADAS	advanced driver-assistance system	1
API	application programming interface	115
AuC	area under the curve	78
AVC	Advanced Video Coding	80
BD-VMAF	Bjøntegaard Delta VMAF	
BD	Bjøntegaard Delta	119
BDD-A	Berkeley Deep Drive Attention	35
BDR	Bjøntegaard Delta Rate	121
BOLA	Buffer Occupancy based Lyapunov Algorithm	20
C-SAS	Cranfield situation awareness scale	14
C4i	command, control, communication, computers, and intelligence	13
CARLA	Car Learning to Act	117
CARS	crew awareness rating scale	14
CIF	common intermediate format	121
CQP	constant quantization parameter	119
CRF	constant rate factor	22
CS-QP	cost-saving per QP	119
CS	cost-saving	76
CTC	common testing conditions	58
CTU	coding tree unit	66
CVE	cross-validation error	84
CWDM	Coarse Wavelength Division Multiplexing	7
DASH	dynamic adaptive streaming over HTTP	19
DBSCAN	density-based spatial clustering of applications with noise	45
DCT	discrete cosine transformation	74
SA	situation awareness	117
ELASTIC	fEedback Linearization Adaptive STreamIng Controller	20
EVQA	Ensemble-learning-based Video Quality Assessment index	23
FCL	fully connected layer	120
FoV	field of view	53
FVQA	Fusion-based Video Quality Assessment index	23
G2G	Glass-to-Glass	21
GLM	generalized linear regression method	84
GoP	group of pictures	121

GPU	Graphics Processing Unit	107
GTAV	Grand Theft Auto V	9
GUI	graphical user interface	113
HAS	HTTP adaptive streaming	117
HD	high definition	121
HEVC	High Efficiency Video Coding	80
HMD	head-mounted display	7
HTTP	hypertext transfer protocol	19
HVS	human vision system	67
IoU	intersection over union	121
ITU	International Telecommunication Union	80
JPEG	Joint Photographic Experts Group	119
JVET	Joint Video Experts Team	58
LED	light-emitting diode	16
LiDAR	light detection and ranging	117
LOOCV	leave-one-out cross-validation	84
LSNF	least square non-linear fitting	83
LSTM	long short-term memory	38
LTE	Long Term Evolution	12
MAC	medium access control	20
MAD	mean absolute difference	121
MAE	mean absolute error	121
MARS	mission awareness rating scale	14
MB	macro block	66
MDA	multi-dimensional adaptation	121
MDVQM	multi-dimensional video quality metric	68
ML	machine learning	121
MSCR	mean saving-cost ratio	121
MV-HEVC	Multiview High Efficiency Video Coding	19
MV-ROI	multi-view region of interest	121
MV	motion vector	91
MVC	Multiview Video Coding	19
N-HEVC	NVIDIA HEVC	79
PC	Pearson correlation	119
PR	Precision-Recall	78
PSNR	peak signal-to-noise ratio	121
PVS	processed video sequence	81
QCIF	quarter common intermediate format	81
QoE	quality-of-experience	17

Symbols

Driving Symbols

α	Vehicle steering angle [rad]
γ	Camera yaw angle [rad]
C	Camera view
c	Vehicle speed constant [s/m]
N	Number of cameras
S	Vehicle state information
v	Vehicle velocity [m/s]

Math Symbols

ρ	Spearman's rank correlation
σ	Standard deviation of a distribution
a	Generalized linear regression method model single feature weight
b	Generalized linear regression method model interaction term weight
$f(x)$	Generalized linear regression method model feature
I	ML model input
y	Generalized linear regression method model value

Situation Awareness Symbols

γ	Distraction punishment factor [rad]
d_{max}	Maximum distance of points used for the DBSCAN algorithm
n_g	Number of gaze samples
n_{min}	Minimum number of gaze samples used for the DBSCAN algorithm
o	Number of non-ROI clusters
p_{idt}	Perception level of a desired situation element at time t
p_{irt}	Perception level of a required situation element at time t
SA	Situation awareness
SA_{actual}	Current situation awareness of the driver
$SA_{optimal}$	Optimal situation awareness the driver can achieve

SA_{ratio}	Ratio of actual and optimal situation awareness
SE	Situation element
$SE_{act,d}$	Actual desired situation element
$SE_{act,r}$	Actual required situation element
$SE_{opt,d}$	Optimal desired situation element
$SE_{opt,r}$	Optimal required situation element
t	Time [s]
t_c	Comprehension time required for the operator to unerstand an SE [s]
t_{ac}	Alive time of a comprehended SE [s]
t_{ad}	Alive time of a detected SE [s]

Video Symbols

α	a_{st} model weight
δ	d_{st} model weight
ϵ	e_{st} model weight
γ	c_{st} model weight
λ	l_{st} model weight
ν	v_{st} model weight
ρ	$R_{max,I}$ model weights [kbit/s]
σ	Standard Deviation e.g. of a Gaussian low-pass filter
τ	t_{st} model weight
υ	u_{st} model weight
φ	j_{st} model weight
a	SCF content dependent parameter
a_{st}	a modeled by SA and TA
B	Size of the bitstream from an encoded frame [kbit]
b	TCF content dependent parameter
b_{st}	b modeled by SA and TA
C	Quality cost (difference of two distortions)
c	NCF content dependent parameter
c_{st}	c modeled by SA and TA
Cb	Blue-difference channel of a video
CH	Color channels of the video

Cr	Red-difference channel of a video
D	Video distortion (quality of a video)
d	NCF content dependent parameter
D_{filter}	Video distortion with preprocessing applied
D_{none}	Video distortion without preprocessing
d_{st}	d modeled by SA and TA
E	Encoder
e	NCF content dependent parameter
E_{SFV}	Encoder of the superframe video
e_{st}	e modeled by SA and TA
F	Frame
f	Video frame rate $\left[\text{fps}\right]$
f_{max}	Maximum video frame rate $\left[\text{fps}\right]$
GCF	Gaussian Correction Factor for the kernel size
$GSDCF$	Gaussian Correction Factor for the standard deviation
I	Image
j	RCF content dependent parameter
j_{st}	j modeled by SA and TA
k	Kernel size e.g. of a Gaussian low-pass filter
l	GCF content dependent parameter
l_{st}	l modeled by SA and TA
m	GCF content dependent parameter
m_{st}	m modeled by SA and TA
$MSCR$	Mean saving-cost ratio [kbit/s]
n	Group of pictures Length
NCF	Group of pictures Length Correction Factor
o	GCF content dependent parameter
o_{st}	o modeled by SA and TA
Q	JPEG quality level
q_p	Quantization parameter
$q_{p,min}$	Minimum quantization parameter
$q_{p,SFV}$	Quantization parameter of the superframe video

R	Video bitrate [kbit/s]
r	Video resolution [px]
R_{filter}	Video bitrate with preprocessing applied [kbit/s]
$R_{max,I,st}$	$R_{max,I}$ modeled by SA and TA [kbit/s]
$R_{max,I}$	Maximum video bitrate at an I-frame only GoP structure [kbit/s]
r_{max}	Maximum video resolution [px]
R_{none}	Video bitrate without preprocessing [kbit/s]
R_{SFV}	Video bitrate superframe video [kbit/s]
R_{total}	Total available transmission bitrate [kbit/s]
RCF	Resolution Correction Factor
S	Rate saving (difference of two bitrates) [kbit/s]
SA	Spatial activity of a video
SCF	Spatial Correction Factor
SR	Spatial resolution (synonym for r) [px]
t	$GSDCF$ content dependent parameter
t_{st}	t modeled by SA and TA
TA	Temporal activity of a video
TCF	Temporal Correction Factor
TR	Temporal resolution (synonym for f) [fps]
TR_{SFV}	Temporal resolution of the superframe video [fps]
u	$GSDCF$ content dependent parameter
u_{st}	u modeled by SA and TA
v	$GSDCF$ content dependent parameter
v_{st}	v modeled by SA and TA
VP	View priority of a camera view
Y	Luminance channel of a video

Chapter 1

Introduction

Autonomous vehicles have become an increasingly popular research topic over the last years. The continuing development of self-driving vehicles aims to increase driving safety and reduce the number of accidents and injuries [1]. The Society of Automotive Engineers (SAE) has classified the stages of autonomous driving into six levels, ranging from Level 0 (manual driving) to Level 5 (fully automated driving) [2]. With increasing levels of autonomy, the vehicle's advanced driver-assistance system (ADAS) is required to make increasingly complex and safety-critical decisions. Despite continuous advances of ADAS, failures of such systems are still inevitable [3].

In case a failure occurs, the car has several options to cope with the situation [1]. The vehicle can attempt to reach a safe state on its own, for example by parking at the side of the road. Another option is to request a takeover by a human driver. For a system operating on SAE Level 3, the human driver inside the vehicle can take over the control of the vehicle. However, studies have shown that it can take up to 40 s for human drivers to regain control of vehicles previously driving autonomously [4]. At SAE Level 4 and higher, no human driver has to be in the vehicle at all. To still be able to bring a human driver back into the loop, the self-driving vehicle can return the control to a human driver located in a remote control center [5]. In the state of California, law even requires vehicles without drivers inside them to be able to be remotely controlled [6]. The task of controlling a vehicle from a remote location is commonly referred to as teleoperation or teleoperated driving. Figure 1.1 shows the general workflow of how teleoperation can be used to resolve failures of autonomous vehicles.

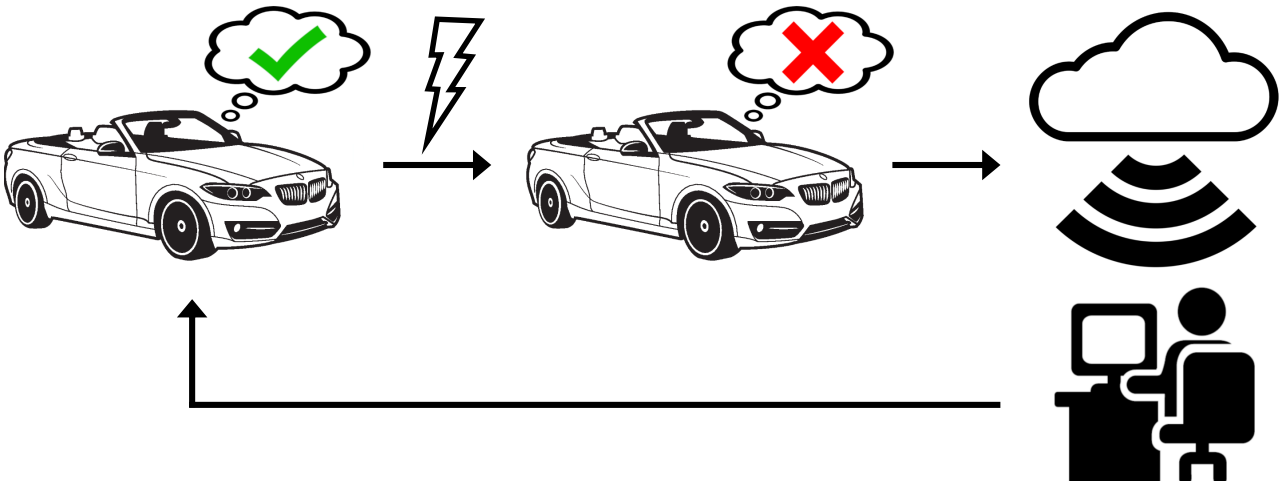


Figure 1.1 Schematic overview of a teleoperation workflow to resolve failures of autonomous vehicles. A vehicle driving autonomously encounters an erroneous situation it cannot handle on its own. The vehicle requests support by a human operator in a remote location. The human operator connects to the vehicle, resolves the erroneous situation from remote, and returns the control back to the vehicle. The vehicle continues to drive in autonomous mode.

When a failure occurs, the remote operator will take over the control of the vehicle, resolve the problematic situation, and then return the control back to the autonomous system. To control the system safely even in complex traffic situations, the operator has to be fully aware of the current situation. In teleoperated driving, the remote operator perceives the traffic situation via video streams from multiple cameras and transmits the control commands from their remote workspace back to the vehicle [7]. Operating from a remote location introduces new challenges, such as the additional delay caused by the communication link or the limited transmission resources of the mobile network [1].

To address these challenges, several research groups have implemented test setups for teleoperated driving. A typical teledriving setup consists of a vehicle equipped with multiple cameras to capture the traffic environment [7]. The video data of all camera views are compressed and transmitted over a mobile network to a remote operator workspace. The remote operator perceives the traffic situation via multiple displays or a head-mounted display (HMD) [8]. Similar to a racing simulation, the operator enters the control commands via a steering wheel and pedals to control the vehicle. The control commands entered are then transmitted to the vehicle. In [7], the authors implemented such a typical system setup for controlling a vehicle from remote. This vehicle is then used for further investigations such as comparing regular displays and HMDs [8], or visually compensating the delay by augmenting the scene representation [9].

Without any augmentation to support the operator, a delay of 300 ms already affects the operator performance [10]. A variable delay further increases the workload of the operator. In the literature, many approaches focus on improving the operator performance by augmenting the visual representation [12, 11] or providing haptic feedback [13]. The actual transmission of sensor information has received less attention. Feasibility studies have shown that today's 4G networks already provide sufficient latency and transmission capacities to allow for teledriving on public roads in selected areas [14, 15]. In these areas, all video streams can be transmitted in a sufficient quality and no special adaptation strategy for the video streams is required. However, even in such selected areas, the available transmission rate cannot be guaranteed by the network provider. These edge case situations, in which the available transmission rate does not allow for transmitting all camera video streams in sufficient quality, are the main motivation for this thesis.

All contributions made in this thesis intend to distribute the limited available transmission rate in a way that the most important camera views for the current driving situation are still transmitted in a sufficient quality to control the vehicle safely. Based on these conditions, we design a special adaptation strategy that considers the current traffic situation to control the individual video streams. Finally, the hardware limitations of commercial vehicles are considered. We design a preprocessing concept that allows for using the adaptation strategy developed before with the limited encoding hardware available in vehicles. The resulting major contributions made in this work are summarized next.

1.1 Major Contributions

This thesis contains four main contributions that constitute a framework for teleoperated driving as summarized in Figure 1.2. All modules can be integrated into autonomous vehicles, which enables safe remote control of the vehicle by considering the current network and traffic situation.

As first contribution, we design a general teleoperation framework for driving in simulated environments. The framework is represented in Figure 1.2 by the background in gray outlined with dashed lines. A video streaming pipeline with an interface for live video stream adaptation is the central component of this framework and is shown in blue. The second contribution of this thesis focuses on the online assessment of the operator situation awareness (SA), shown in green. In the third contribution, we propose a traffic-aware multi-view adaptation (TAMVA) scheme that controls the individual video streams to increase the operator SA. The TAMVA scheme is shown in orange in Figure 1.2. As last contribution, we propose a preprocessing concept to enable individual video

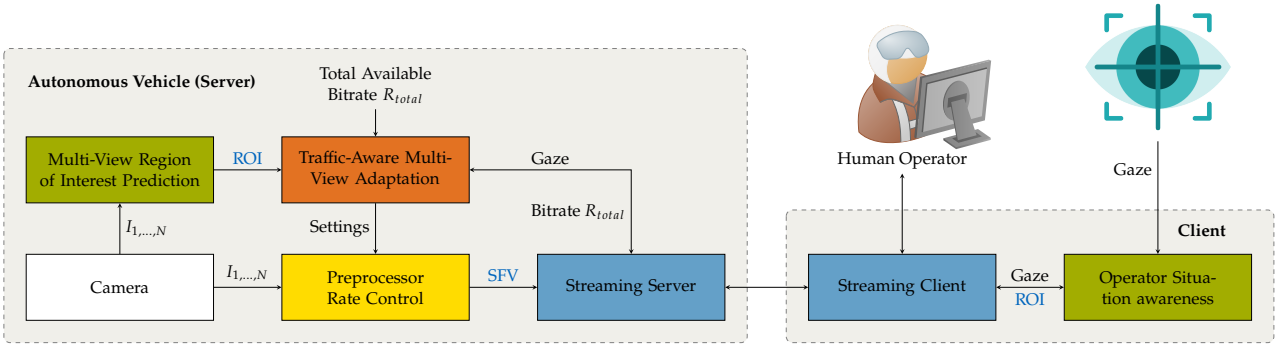


Figure 1.2 Overview of a general teleoperated driving system setup and the proposed contributions. The general adaptive streaming framework proposed in Chapter 3 is shown in blue. The situation awareness (SA) assessment introduced in Chapter 4 is shown in green. Chapter 5 presents the traffic-aware multi-view adaptation (TAMVA) scheme shown in orange. The preprocessor-based rate control proposed in Chapter 6 is shown yellow.

stream adaptation if there is only limited encoding hardware available. This is shown in yellow in Figure 1.2. Next, each contribution is summarized in more detail.

1. **Teleoperated Driving in Simulated Environments:** The design and evaluation of adaptive streaming solutions for teleoperated driving require controllable network and traffic conditions. For this purpose, we design a general teleoperation framework for driving in simulated environments in Chapter 3. The framework can be used to extend existing driving simulators via a standard interface for teleoperated driving. This comprises a hardware setup and a customizable graphical user interface (GUI) for the operator workspace, a low delay streaming pipeline with an interface for live stream adaption, and a communication interface for exchanging control commands and status information between the vehicle and the operator control station. We use the proposed teleoperation framework extending the Car Learning to Act (CARLA) driving simulator [16] to implement and evaluate the remaining contributions made in this thesis.

2. **Driver Situation Awareness Assessment:** An important aspect for both in-car driving and teledriving is the observation and measurement of the driver situation awareness (SA). Especially during the transition to human control, the vehicle has to ensure that the driver is sufficiently aware of the current situation. To the best of our knowledge, no direct method to explicitly estimate driver awareness exists in the literature. To this end, we introduce a metric to measure the SA of a human driver in realtime for both in-car drivers and remote operators in Chapter 4. The proposed approach is inspired by methods used in aviation. An eye-tracking device measures the actual SA of the human driver. Region of interest (ROI) prediction allows for estimating the optimal SA that defines all elements the driver should have perceived. By comparing the actual SA with the optimal SA, the driver awareness can be modeled based on the current traffic situation. The proposed approach has been evaluated in a user study on eight driving scenarios and is able to accurately measure the participants' SA while driving.

3. **Traffic-Aware Multi-View Video Stream Adaptation:** Next, the actual adaptation of the video streams is addressed. While matching the available network resources, the video streams need to be automatically adapted to provide the operator with the best possible scene understanding. To automatically adapt the video streams, we develop a traffic-aware multi-view adaptation (TAMVA) scheme in Chapter 5. The importance of each camera view is estimated by the vehicle's real-time movement in traffic. The resulting prioritization together with the total available transmission rate determines a specific bit budget for each camera view. The video quality of each individual video stream is then optimized for the given bit budget using a quality-of-experience-driven multi-dimensional adaptation (MDA) scheme. To remove less important

areas from the image and further reduce the bitrate required for streaming the video content, the rear-facing camera views are filtered by applying ROI masks. All modules introduced in this chapter are implemented in the proposed teleoperation framework. The ROI masking achieves Bjøntegaard Delta Rate (BDR) savings of at least 19.8 % compared to streaming the full camera view. The overall system improves the Video Multi-Method Assessment Fusion (VMAF) score by 5 % on average for each camera when the importance of the individual camera views as rated by the users is considered. The quality gains achieved by using such a prioritization of important camera views can still provide the operator with a sufficient quality [18, 17] for the most important camera views to control the vehicle safely in traffic.

4. **Adaptive Multi-View Live Video Streaming Using a Single Encoder:** The proposed TAMVA scheme controls the individual camera video streams to provide the operator with the best possible scene understanding. Limited by cost and size, autonomous vehicles are often restricted to a single video encoder. To still encode all camera views at the same time using a single encoder, the individual frames have to be combined into a single superframe video (SFV). While this allows for streaming multiple camera views, it prevents the adaptation of the individual camera views based on the current traffic situation. To enable the individual adaptation while using a single encoder, we propose a preprocessing concept in Chapter 6. The preprocessing filters control the individual camera views before the combination into the SFV, which is then encoded and transmitted.

   a) First, the general idea of the preprocessing concept is introduced. In this thesis, we present four preprocessing filters to control the frame rate, frame size, rate/quality, and color channels of the video sources. A proof of concept is implemented to evaluate the influence of the different preprocessing filters. The results demonstrate that the frames preprocessed by the filters do not cause any side effects on neighboring segments when encoding the SFV. We evaluate the usability of the proposed concept in an exemplary driving scenario with a six-camera setup. While achieving a similar bitrate for the most important views, the proposed approach requires a total bitrate that is 40 % smaller compared to a single encoder without preprocessing.

   b) The preprocessing filters used to control the rate/quality of the individual camera views can be summarized as spatial low-pass filters. This category of filters applied on the video sources before encoding has a direct influence on the resulting rate-distortion (RD) performance of the encoder. To identify the most suitable preprocessing algorithm, we conduct an extensive RD-performance analysis. In a regular video encoding scenario, the RD curve of a certain encoder is only determined by a set of quantization parameters (QPs). The preprocessing filters introduce a new dimension of input values in the form of the additional preprocessing parameters. To effectively compare the RD performance of the encoder including the preprocessing dimension, we introduce two evaluation methods. First, we propose the novel mean saving-cost ratio (MSCR) as the logarithmic mean ratio of maximum bitrate savings over maximum quality cost for all parameters of a preprocessing filter. The MSCR acts as a single-score metric to compare different preprocessing algorithms. Second, the Bjøntegaard Delta (BD) curve compares two preprocessing filters over a range of QPs. According to both proposed evaluation methods, the Gaussian low-pass filter achieved the best performance in a comparison of different preprocessing algorithms.

   c) Then, we use the insights gained from the proof of concept and the filter performance evaluation to design a preprocessor rate control. The TAMVA scheme estimates the frame rate, frame size, and target rate/quality to control multiple encoders. The preprocessing filter concept enables the individual adaptation of multiple camera views using a single encoder. The preprocessor rate control connects these two components by estimating the parameters required to control the preprocessing filters from the parameters estimated by the TAMVA scheme to control multiple encoders. As the core part of the rate control,

we extend an existing analytical bitrate model and propose a novel bitrate model based on machine learning. Both models specifically consider the influence of the Gaussian low-pass filter as well as the influence of the QP, the frame size, the frame rate, and the group of pictures (GoP) length on the video bitrate. Because both models show similar performance, either one can be used as the core part of proposed rate control. Both rate models outperform state-of-the-art bitrate models by at least 22 % regarding the overall root mean square error (RMSE).

d) Lastly, we evaluate the proposed preprocessing approach for three representative driving scenarios. We compare the bitrate savings and quality gains of the preprocessing approach to a reference solution that uses multiple encoders and a regular superframe composition approach without preprocessing. The proposed approach achieves a comparable rate and quality for the most important views compared to using multiple encoders, but reduces the quality on the remaining views by only 1.8 % VMAF score. In comparison, an approach using a single encoder without preprocessing causes quality degradations of 4.5 % VMAF score for the remaining views and even 5.6 % VMAF score on the most important views.

1.2 Thesis Organization

This thesis is organized as follows. Chapter 2 introduces the background information necessary for understanding the contributions made in this thesis. We summarize related work with respect to teleoperated driving, situation awareness assessment, and video processing and transmission. Shortcomings of the state of the art are discussed to motivate the methods proposed in this work. Chapter 3 presents the design of a general teleoperation framework that allows for driving in simulated environments. The framework builds the basis for the remaining contributions made in this thesis. Most notably, the framework includes a video streaming pipeline that provides an adaptation interface for controlling the respective video encoder. In Chapter 4, we propose a method for assessing the situation awareness of a driver or remote operator in realtime. This includes the extension of a state-of-the-art region of interest prediction method from single-view to multi-view. In Chapter 5, we propose a traffic-aware multi-view adaptation scheme to increase the driver situation awareness. Relying on the teleoperation framework, the proposed adaptation scheme controls the individual video streams of every camera view to provide the operator with the best possible scene understanding. Chapter 6 addresses the hardware limitations of commercial vehicles that prevent individual traffic-aware video stream adaptation. To this end, we propose a novel preprocessing concept that enables the individual adaptation of multiple camera views under the given hardware limitations. Further, we design a preprocessor rate control to configure the preprocessing filters based on the adaptation parameters used for controlling multiple encoders. Chapter 7 concludes this thesis. The main contributions are summarized and their limitations are discussed. Finally, we outline possible directions for future research.

Parts of this work have been published in international, peer-reviewed scientific journals and conferences [19, 20, 22, 21, 24, 23, 25].

Chapter 2

Background and Related Work

In this chapter, we summarize the background of this thesis. First, we give an overview of state-of-the-art teleoperated driving (ToD) systems. Then, we discuss existing approaches for measuring the driver's awareness of the current traffic situation. Finally, we summarize methods for video processing and transmission.

2.1 Teleoperated Driving Systems

Teleoperation is defined as the interaction of an operator with the physical world without being physically present [26, 27]. The increasing transmission capacity and distribution of mobile networks increase the potential of ToD as a feasible mobility concept for on-off-road applications.

Despite high-rate wireless connections not being available then, Ross et al. [28] set up an advanced teleoperation testbed in 2007. They used an 1 km single mode Coarse Wavelength Division Multiplexing (CWDM) optical fiber for connecting their off-road vehicle with the operator workstation. Providing the operator with five camera views streamed at 25 Hz, they mainly suffered from high delays up to 960 ms. Their study demonstrated that a broad field of view (FoV) with an average quality should be used in most cases, while complex situations require high resolution at specific areas. In 2011, Khan et al. [29] proposed a wireless teleoperation vehicle. The authors addressed the varying time delay and quality of service (QoS) of the network by using a Support Vector Machine (SVM) to predict the delay and packet loss. The resulting predictions are then used to adapt the video content. In 2013, Gnatzig et al. [7] presented a system design for teleoperated road vehicles that uses multiple static cameras, sensor fusion display and a network-based video transmission and communication architecture. Shen et al. [30] presented a teleoperation vehicle in 2016 that was exclusively created from off-the-shelf components. The system included a head-mounted display (HMD) view, wheel and pedal control, and a Real-Time Transport Protocol (RTP)-based video communication with 30 fps update rate and 42° FoV.

The main issue of most teleoperation systems is the delay introduced by the transmission link. This delay affects the operator performance and immersive feeling of being physically there [12]. Next, we discuss the main components of typical ToD systems as well as current solutions for improving the operator performance and for addressing the issue of delay.

2.1.1 General System And Use Cases

The Society of Automotive Engineers (SAE) divides the task of driving into three main types of activities [31]:

- **Strategic level operation** includes travel planning, consideration of available options, and cost-risk analysis.
- **Tactical level operation** includes maneuver planning, lane selection, speed selection, and reactions to objects and events.

- **Operational level operation** includes longitudinal and lateral control as well as object detection, event detection, and classification.

Based on these definitions for a driving tasks, the 5GAA Automotive Association categorizes the role of the operator into four types [32]:

- **Non-ToD (No Role):** The operator is not engaged in the driving task.
- **Dispatch ToD (Dispatcher):** The operator performs strategic decisions such as the selection of the route while the systems plans and executes all operations.
- **Indirect Control ToD (Indirect Control or Remote Assistant):** The operator additionally takes over the tactical tasks such as the planning of a route, while the system still executes the tasks planned by the operator.
- **Direct Control ToD (Direct Controller or Remote Driver):** The operator is in full control of the vehicle and executes all operations in realtime.

In this thesis, we focus on the direct control of a remote vehicle. A typical ToD setup for direct control such as proposed in [7] consists of three main components: the vehicle that is controlled remotely, the operator workspace with a human operator, and the network link in between. Figure 2.1 visualizes such a general system setup.

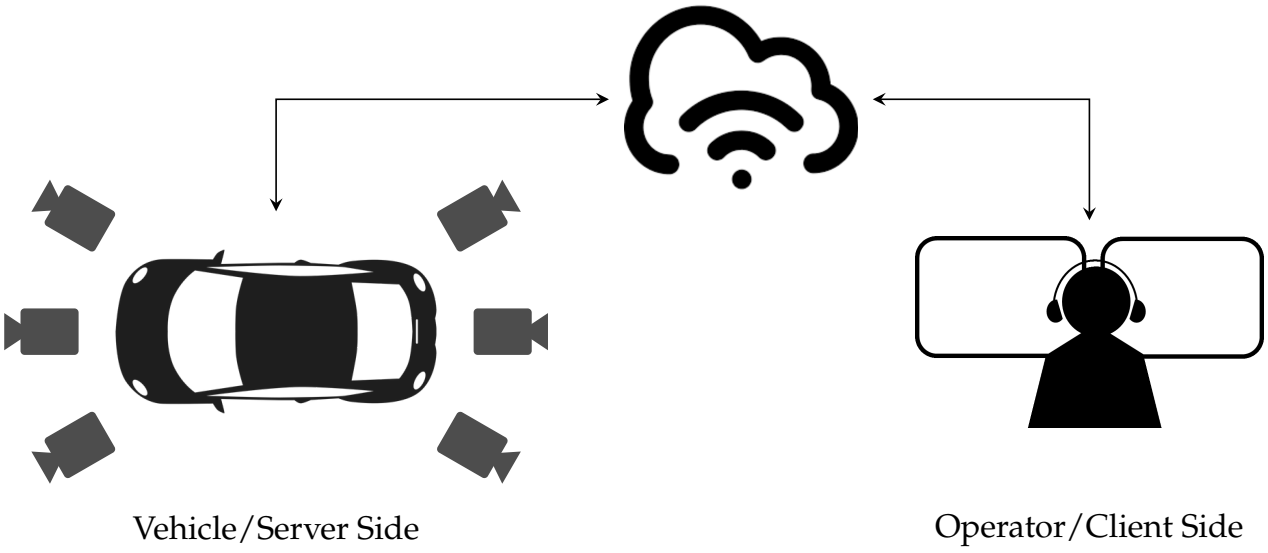


Figure 2.1 Overview of a general teleoperated driving (ToD) system setup consisting of a vehicle, the operator workspace, and the network link in between.

The vehicle is equipped with multiple cameras that capture the environment. Then, the video data are encoded and transmitted over a mobile network to the remote operator workspace. The operator perceives the traffic situation via one or multiple displays and enters the control commands. The commands are transmitted back to the vehicle and executed accordingly. In this thesis, we focus on the transmission of sensor information, in particular video data, from the vehicle to the operator workspace.

2.1.2 Driving Simulations

Most teleoperation systems are initially developed in simulation, since real vehicles are costly and time consuming in their setup. ToD simulation systems are usually based on vehicle simulators with

custom extensions for the streaming functionality. Hosseini et al. [9, 11] used SILAB [33], a closed source vehicle simulation software. SILAB can be purchased together with the related hardware components to perform realistic vehicle and driving simulations. The extension of SILAB from local driving to teledriving is done by connecting the simulator to an operator workstation via the User Datagram Protocol (UDP). A major drawback of this closed source system is the missing possibility for custom modifications. Camera data, for instance, have to be acquired using a screen grabbing software.

With increasing popularity of autonomous driving while no suitable simulations were available, video games have been used to implement and test existing solutions. Modern video games such as Grand Theft Auto V (GTAV) already provide photorealistic graphics and physics simulation. A publicly available extension called DeepGTAV [34] supports autonomous driving scenarios. Similar to closed source vehicle simulators such as SILAB, video games have the drawback of extensibility.

Open source solutions such as DeepDrive [35] provide simulations for autonomous driving that allow for extension. However, DeepDrive provides only a small set of features. Microsoft developed the simulator AirSim [36] for autonomous driving and drone simulations. Its main focus is on dataset generation for machine learning (ML). AirSim's detailed physics simulation is well suited for aviation simulations [37]. The Advanced Platform Lab at the LG Electronics America R&D Center, formerly the LG Silicon Valley Lab, developed the LGSVL simulator [38]. LGSVL is fully integrated into the popular open source driving platforms Apollo [39] and Autoware [40]. Finally, Car Learning to Act (CARLA) [16] is an open source driving simulator with special focus on autonomous driving. CARLA provides a wide range of sensors and pretrained autonomous driving models, while supporting common map standards such as OpenDrive. Similarly to AirSim, CARLA uses the Unreal Engine [41] for physics simulation and rendering. CARLA's flexible application programming interface (API) and the ongoing development make it one of the most popular driving simulators [42]. With the *CARLA Leaderboard* [43], CARLA provides its own benchmark system for autonomous driving models. While these open source simulators allow for extension, none of the above can be directly used for ToD.

Bodell et al. [44] proposed a 360°-view ToD vehicle based on the Robot Operating System (ROS) [45] and the Gazebo simulation [46]. While ROS is widely used in autonomous driving [47], the main focus of Gazebo is on robotics with configuration options down to joint level. In [48], the authors developed the OpenROUTS3D simulator for ToD based on the Unity [49] game engine. OpenROUTS3D provides a user study mode and configuration options for different network QoS aspects. However, OpenROUTS3D does not include the actual compression and transmission of video data. Instead, it simulates this aspect by applying pixelation, delay, and packet loss based on configurable probabilistic distributions. The lack of actual network communication prevents OpenROUTS3D to be used for approaches that focus on the transmission of sensor information. Hence, we introduce a teleoperation framework that includes a video streaming pipeline as contribution in Chapter 3. The proposed framework extends the CARLA simulator for a ToD component and will be used for the evaluation of the remaining contributions made in this thesis.

2.1.3 Operator Perception

The operator perceives the current driving scene based on the sensor data transmitted from the remote vehicle or driving simulator. Labonente et al. [50] evaluated different visualization methods focusing on novice users. Two view concepts achieved the best performance for a task completion evaluation: an ego-perspective 3D representation of the current scene, augmented with an ego-perspective camera view at the central FoV and a 3D third-person view, combined with the same central ego-perspective camera view used before. Fong et al. [51] showed that using multiple sensor information within a sensor fusion display could improve the operator's depth recognition. While both sensor representations were mainly designed for teleoperation in robotics, they highlight that the presentation of the sensor data plays an important role for how well the operator understands the remote situation.

In [7], the authors visualized the video streams of multiple cameras attached to a vehicle with three regular displays. While they were able to successfully control a vehicle from remote, they faced several significant problems. First, the limited immersive telepresence caused a nonrealistic driving feeling and second, the data processing and transmission introduced a time delay. Tang et al. [52] proposed a zoom-blur model to improve the operator's speed perception and hence the operator's immersion of being physically inside the vehicle. Without this physical presence and experiencing the vehicle's movement, the actual speed is often underestimated. The zoom-blur model blurs the outer regions of the front camera and the side facing cameras for higher speeds. This supports the operator in estimating the actual speed and controlling the vehicle.

The effects of time delay are addressed in [53] by using a predictive display. Due to the data transmission, the operator perceives only a delayed representation of the current driving scene. The predictive display augments this representation with the actual position of the vehicle. The actual position of the vehicle is estimated based on the vehicle's movement and the transmission delay. In [54], the authors evaluated different presentation modes for the predictive display. The *Frame Prediction* display method achieved the best performance in a driving user study. It visualizes the path and actual front position of the vehicle. Hosseini et al. [9] extended the idea of the predictive display by showing the actual position of the ego vehicle as well as the positions of other traffic participants. The actual positions are predicted on the client side using optical flow and semi-global matching techniques. Figure 2.2 shows the predictive display for the ego vehicle and other traffic participants in two traffic situations.

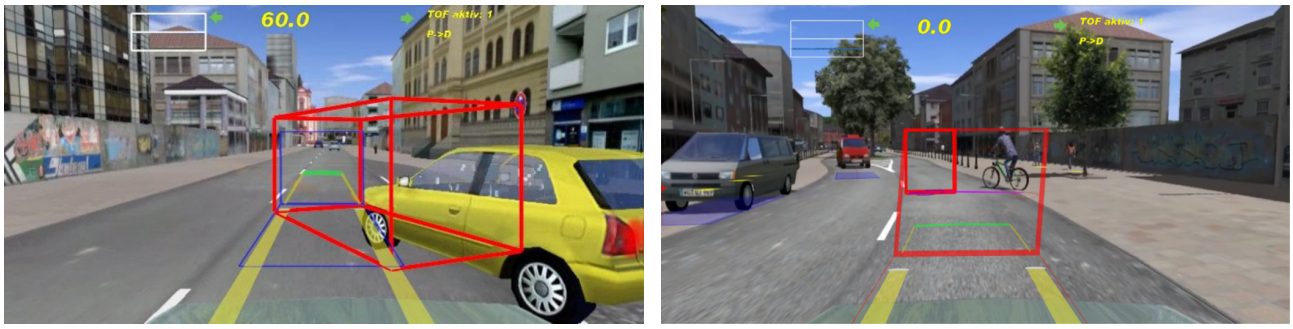


Figure 2.2 Predictive display visualizing the path of the ego vehicle as well as the actual front position of the ego vehicle and other traffic participants (source: [9] © 2016 IEEE).

Equipped with the predictive display, Hosseini et al. [11] further extended their visualization safety concept by using HMDs to prevent distractions of the operator. Within the HMD view, there exist uncovered areas due to the fixed camera views. These uncovered areas are in particular harmful in narrow driving situations. To enhance the operator's telepresence, they extended the 360° view including the uncovered areas with the vehicle's shape and bounding boxes of other traffic participants. The bounding boxes are calculated with a fixed height on the client side, based on single-layer light detection and ranging (LiDAR) data transmitted from the remote vehicle. A comparison of this safety concept on regular displays and HMDs demonstrated a better depth perception using HMDs when driving in simulation.

Georg et al. [8] further evaluated the safety concept of [11] by comparing regular displays and HMDs on an actual test vehicle. The authors conducted a user study, where the participants had to complete six driving maneuvers. Both setups evaluated in the user study produced similar results in driving quality, workload, and the feeling of immersive telepresence. Based on their results, the authors proposed an immersive interface for ToD in [55]. The interface is based on an omnidirectional sensor representation that allows for a fast switching between different visualization types. This enables the operator to select a suitable representation based on the current driving scenario. Figure 2.3 shows three visualization types that can be selected by the human operator.

Figure 2.3 Three visualization types supported by the ToD interface. A LiDAR point cloud augmented with camera data is shown on the left. In the middle, all fisheye cameras, a front camera, and LiDAR data are combined in to a single representation. The right view shows a birds-eye view based on the fisheye cameras (source: [55] © 2019 IEEE).

The left view visualizes the LiDAR points and all cameras as rectangular views. All fisheye cameras, a front camera, and the LiDAR data are combined in a single view in the middle. The right image shows a birds-eye view created from two fisheye cameras.

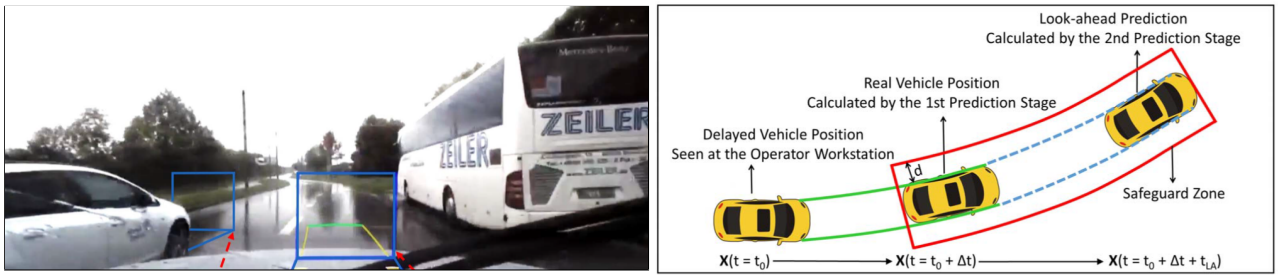
In [56], Georg et al. use this interface in a long-time user study to measure the impact of different displays and video quality levels on the operator situation awareness (SA). The study showed that the video quality has a significant influence on several factors that contribute to the SA. These factors include the detection of traffic signs or the perception of road users in situations with changing light conditions. The goal of this thesis is to adapt the individual video streams in a way that the operator perceives the views most related to the current traffic situation with the best possible video quality. We present the proposed adaptation scheme in Chapter 5.

2.1.4 Vehicle Remote Control

Besides the perception of the environment, the actual control of the vehicle is another important aspect of ToD. Kebira et al. [57, 58] augment force models to manage variable time delay and uncertainties of internet-based teleoperation for robotics. While a delayed transmission of the control commands is critical in teleoperation for robotics, the vehicle's inertia makes it less sensitive to delayed control commands. On the other hand, a complete drop of the connection is a severe security issue. Tang et al. [59] proposed a concept called *Free Corridor* to address such connection losses. The *Free Corridor* shows the free space that is required for safely stopping the vehicle. It is calculated based on the vehicle's movement and augments the operator's display. The operator has to ensure that the augmented area is always free from any other traffic participants. In case of connection loss, the vehicle can stop itself within this area without the risk of a collision.

In a follow-up work, Hosseini et al. [13] used their concept of predictive displays [9] and combined it with the free corridor concept [59]. This results in two display augmentations such as shown in Figure 2.4a. The first blue frame visualizes the actual position of the vehicle as introduced with the predictive display. The green line together with the yellow lines represents the free corridor. To support the operator in keeping the corridor free, the authors provided haptic feedback to the operator. Whenever another road user enters the safeguard zone shown in Figure 2.4b, the steering wheel provides the operator with haptic feedback by steering away from the user entering the safeguard zone. Especially in narrow driving situations, this concept improved the lateral control of the vehicle.

While these safety concepts try to compensate the transmission delay or connection loss for a single vehicle-to-operator connection, Gohar et al. [60] suggested to rely on multiple remote drivers. The respective remote driver with the lowest latency to the vehicle is selected for controlling the vehicle. A critical problem of this approach is that overly frequent handovers between different operators can negatively affect their SA.



(a) Predictive display and free corridor (source: [13])

(b) Safeguard zone (source: [13])

Figure 2.4 Safe lateral control for ToD using free corridor, predictive display, and haptic feedback (source: [13] © 2016 IEEE).

2.1.5 Network Communication and Sensor Data Transmission

The major challenge in ToD is a reliable low delay data transmission that enables the operator to safely interact with the remote environment [9]. So far, different visualization and control strategies have been discussed. In this section, we summarize concepts for the actual transmission of sensor data and the challenges caused by mobile networks.

In [61], the authors implemented the predictive display, free corridor, and haptic feedback such as proposed by [13] and analyzed the communication challenges when driving via commercial Long Term Evolution (LTE) networks. Test drives showed an average delay of 138 ms for transmitting the camera data with a maximum delay of 445 ms. To avoid jitter, the authors buffered the video stream for a constant time delay of 500 ms. This strategy follows the investigations of [12], which showed stronger decreases in driver accuracy for variable time delays compared to constant time delays. Similarly, Neumeier et al. [14] evaluated the feasibility of ToD in LTE networks. Based on a user study [48], they defined 250 ms as the maximum tolerable network delay, plus 50 ms delay introduced by the sensors. Further, they suggested to use at least 1 Mbit/s per camera video stream. Test drives on public roads over 5200 km showed that ToD may be feasible in special areas which have shown a sufficient network quality. The authors refer to these areas as whitelisted areas. In [18], a user study was conducted to investigate the required quality for ToD. Depending on the driving situation, 300 kbit/s to 800 kbit/s per camera video stream could be sufficient for ToD. While the authors evaluated which quality level is required, they did not focus on how to reach that rate/quality or how to distribute it within each frame.

Dror et al. [62] proposed a content adaptive video compression scheme to increase the video quality of important regions within the frame compared to the remaining video content. The authors encoded object regions, detected by a state-of-the-art object detection network, with a higher video quality by spending more bit budget on these regions than on the background. This content adaptive video compression improved the peak signal-to-noise ratio (PSNR) for object regions within a single camera view based on a given target bitrate. Similar to Neumeier et al., Dror et al. consider the same bit budget for each camera view. We contribute to this field by proposing novel methods for how to distribute the total available bit budget over multiple individual camera views, since not all camera views are equally important for a respective driving situation. We introduce our methods for how to distribute the bitrate in Chapter 5.

2.2 Situation Awareness

The solutions discussed in the previous section provide the operator with an immersive feeling of being physically inside the vehicle. The requirements for the operator to successfully control the vehicle then become similar to the ones of an in-car driver. Same as an in-car driver, the operator has to be fully aware of the current traffic situation. Several studies have shown that higher levels of automation lead to a reduced driver attention [63, 64, 65]. A human driver who is not in control

of the vehicle may need up to 40 seconds to obtain full control of the vehicle [4]. In particular, the transition of vehicle control from the system to the human driver requires the driver to be ready for taking over [64]. Before transferring the control to the driver, the system has to guarantee the driver's ability to control the vehicle safely [4]. An operator connecting to a remote vehicle performs a similar takeover process, having to assess an initially unknown situation.

Van de Beukel et al. [66] demonstrated that the SA of the driver is an important factor for how long it takes to take over control. Measuring and potentially increasing the SA is therefore an important aspect of telepresence and ToD [67]. After defining the term SA, we therefore discuss methods for assessing the SA and monitoring the driver next.

2.2.1 Definition

In 1988, Endsley [68] defined situation awareness (SA) as "perception of elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future" (p. 97). In 1995, Endsley [69] further proposed a theoretical model of SA based on three levels:

- **Level 1: Perception** of all elements in the current environment, e.g., perception of other traffic participants or the current movement of the ego vehicle.
- **Level 2: Comprehension** of the current situation, e.g., children at the side of the road requiring the car to drive slower.
- **Level 3: Projection** of future status, e.g., children running towards the road will also enter the road.

This model tries to describe the cognitive state of humans in dynamic environments and their effectiveness in decision making when operating these systems. Next, we discuss methods for assessing the SA based on this definition.

2.2.2 Assessment Methods

SA assessment can be categorized into objective and subjective methods. Riley et al. [67] used methods of both categories for measuring SA as an alternative measure of telepresence. Salmon et al. [70] performed a general review of SA assessment methods in command, control, communication, computers, and intelligence (C4i) environments. Nguyen et al. [71] reviewed recent approaches of SA assessment in aviation environments. Salmon et al. [70] as well as Nguyen et al. [71] classified SA assessment approaches into six categories:

- **Freeze probe techniques:** While the operator is performing a certain task in simulation, the scene is randomly frozen and all display content is removed. Then, the subject has to answer questions about the current situation based on their knowledge from the point of freezing the scene. The responses are recorded and compared to the status of the actual environment. Popular freeze probe techniques are the situation awareness global assessment technique (SAGAT) [68], situation awareness control room inventory (SACRI) [72] and SA of en-route air traffic controllers in the context of automation (SALSA) [73]. While freeze probe techniques directly assess the current understanding of the situation, they can only be implemented in simulation since the real world cannot be frozen.
- **Real-time probe techniques:** Similar to freeze probe techniques, the operator performs a certain task while being observed by an expert. The expert prepares queries and administers them online without freezing the task. The response time or the content of the answer are taken as a measure of the subject's SA. Typical real-time probe techniques are the situation present assessment method (SPAM) [74], Solutions for Human-Automation Partnerships in European

Air Traffic Management (SA for SHAPE) [75], and SA for SHAPE on-Line (SASHA_L) [75]. While these techniques can be applied in realtime, they are still intrusive to the task and require an expert observing the task.

- **Post-trial self-rating techniques:** Here, subjects rate themselves on a rating scale after executing the task. Established methods include the situation awareness rating technique (SART) [76], the situation awareness rating scales technique (SARS) [77], the Cranfield situation awareness scale (C-SAS) [78], the crew awareness rating scale (CARS) [79], the mission awareness rating scale (MARS) [80], and the quantitative analysis of situation awareness (QUASA) [81]. While post-trial self-rating techniques are easy to implement and non-intrusive to the task, they have several problems. Users might rate themselves too highly, since they might not even recognize having a poor SA. Further, post-trial techniques measure the subject's SA at the end of the task, at which time the participants might not recall all events.

- **Observer rating techniques:** The same experts participating in the real-time probes observe the user during task execution and provide a rating for them. The situation awareness behavioral rating scale (SABRAS) [79, 80] is a commonly used technique in this field. While these techniques are not intrusive, the rating is subjective to the expert and cannot directly assess the SA of the subject.

- **Performance measures:** Some aspects of the performance can be measured directly, such as the time to complete the task or the number of errors during task execution. Georg et al. used performance measures in their user studies on the driver SA for different displays and sensor representations [8, 55, 56]. While performance measures do not interrupt the natural task workflow, they can only provide an indirect measure of the subject's SA. A good task performance may not always correlate with the SA.

- **Process indices:** Process indices comprise the recording, analysis, and rating of the processes that the subject follows to establish the SA during the task performance. A typical process index is to record the subject's gaze using an eye-tracking system and to analyze the user behavior during task execution [82]. While eye-tracking devices directly measure the subject's fixation, their implementation outside of a laboratory environment is often difficult.

Objective sources about the SA are diverse and oftentimes difficult to model or assess. Subjective methods are therefore still dominating in the field of SA assessment. Salmon et al. [70] reported that most of the methods from the literature are offline measurements and questionnaires, while there is a lack of suitable online measurement systems. To overcome the need for post-drive questionnaires or freeze probes, Martelaro et al. [83] proposed the DAZE app. DAZE uses real-time in-event alerts to enable online SA measurements [84]. In case of a critical situation, the driver has to first acknowledge noticing the event and then locate the event by clicking on the map displayed by the DAZE app. Possible events cover questions such as *Did you see the police car?*. Although DAZE can be used while driving the car, it only allows for a reactive event measurement and is not suited for constantly monitoring the driver's awareness. However, dynamic tasks such as driving and teleoperation require constantly monitoring the operator to ensure safety.

Desvergez et al. [85] and De Winter et al. [86] showed that visual attention measurements have a higher correlation to the task performance than freeze probes. Hasanzadeh et al. [87] analyzed the awareness of construction workers and showed that eye tracking can be directly used for awareness measurement. Van de Merwe et al. [88] demonstrated the benefits of using eye movements to assess SA by observing the pilot's gaze in different aviation scenarios. Based on the methods discussed, eye tracking is the most suitable technique to be used for constantly monitoring the driver and assessing their SA [89]. The first step in any assessment of SA should be a requirements analysis to determine which elements determine the operator's SA [70]. In the next section, we discuss which elements affect the driver attention and how the attention can be estimated.

Figure 2.5 Raw image and gaze heatmap accumulated from multiple drivers (source: [95]).

2.2.3 Driver Attention

Driver attention can be summarized as a collection of areas or regions the driver should focus on. A driver usually focuses on respective regions by looking at them for an extended period of time. These regions are commonly known as region of interests (ROIs). Predicting which parts of an image are most relevant is also known as ROI prediction. In the literature, there are several approaches that predict these ROIs.

In autonomous driving, object detection or classification models play an important role. Determining the regions a given model focuses on is therefore one option to predict ROIs. Visual backpropagation [90] is one possible method to visualize which parts of an image a model is currently using to determine its output. However, this approach is not trained to exclusively focus on ROIs. Xu et al. [91] introduced the concept of attention that allows actively learning attention maps. An attention module that adaptively directs the model's visual attention on the most important regions during learning [92] can be used to predict the ROI for a given image. While this allows for predicting salient regions of a specific model, the predicted regions might not match with what a human would consider interesting. To predict human gaze fixation, Cornia et al. [93] proposed a long short-term memory (LSTM)-based saliency model that learns to predict the distribution of human gaze fixation points. While their results indicate that human attention in general can be predicted, the task of driving raises additional constraints. Not all salient regions can be considered interesting. Free-space regions without salient characteristics might be relevant for the driving task as well, such as monitoring whether the sidewalk is devoid of pedestrians. On the other hand, areas of an image such as advertisements are likely to be predicted as salient regions, but fixations on the advertisement can be harmful for the driver's SA.

The Dr(eye)ve project [94] was one of the first projects that specifically addressed the issue of ROI prediction for driving. The authors created a dataset with six hours of driving, recorded with a single front camera. An eye-tracking device captured the driver's gaze. The gaze data was then used to generate the ROI labels. However, this way of recording and labeling data has several drawbacks. First, an eye-tracking-based labeling process only allows for a single ROI at a time. Second, distractions such as caused by advertisements could introduce false ROIs not relevant for the current driving situation. Lastly, critical driving situations are rare in regular driving. The single ROI at a time together with a large amount of regular driving scenes leads to the model predicting mainly the vanishing point as the ROI.

Xia et al. [95] addressed these issues. The authors proposed the Berkeley Deep Drive Attention (BDD-A) dataset, which was labeled by collecting the gaze data of multiple independent participants watching recorded driving scenes. Accumulating their gaze patterns results in attention maps that consist of multiple ROIs, as visualized in Figure 2.5. The driving scenes were selected to contain mainly videos of braking, which increases the amount of critical situations. Then, the authors designed an ROI prediction framework based on an object detector. During training, they used human weight sampling to assign a higher weight to critical situations. While this approach allows for multiple ROIs at the same time, the labels are still based on human gaze data. Pal et al. [96] proposed to combine the gaze information with driving specific context information learned from a saliency model. This approach increased in particular the ROIs predicted on the periphery. Their approach as well as the

BDD-A model is limited by the focus on a single front camera. Rear-facing camera views, for instance, have different ROIs such as the free space that is usually covered by the shoulder check, while traffic lights are no longer relevant. In Chapter 4 of this thesis, we propose a concept for extending available ROI prediction methods from single view to multi view.

2.2.4 Driver Monitoring and Online Assessment

ROI prediction allows estimating *which* areas the driver should be aware of to perform the driving task. A continuous monitoring of the driver additionally measures *if* the driver is aware of those regions. Yang et al. [63] proposed to continuously keep the driver's attention on a high level using a colored light-emitting diode (LED) status bar below the front window. When driving in autonomous mode, the visual effects inform the driver through their peripheral view about the vehicle's state, even when the driver is not looking at the road. This reduces the time to take over control from the vehicle. However, this approach conflicts with the idea of removing the human from the driving task and is not feasible for a remote driver that only connects to the vehicle in case the autonomous system has to return the control.

Actively monitoring the driver is still an open challenge [97]. Murphy-Chutorian et al. [98] considered the head pose as a useful indicator for the driver's attention. Estimating the driver's attention based on their head position was proposed by Tawari et al. [99]. The duration that the driver is not looking ahead is used as a measure for distraction. However, looking ahead onto the road is only a rough estimation of the driver's attention. Further, such an approach cannot distinguish between the perception of relevant and irrelevant objects or events inside the driver's FoV.

Fletcher et al. [100] proposed to focus on relevant objects by detecting speed signs and comparing them to the driver's gaze. This provides an immediate feedback if a speed sign has been missed by the driver. Mori et al. [101] proposed the calculation of an awareness score for the driver based on the potential risk caused by surrounding vehicles and the gaze behavior of the driver. They classified the environment into rough segments and assumed that more gazes on potentially critical areas imply a higher level of SA. However, the gaze behavior of the test participants was manually labeled from video recordings, which prevents this system from being used for online assessment. Langner et al. [102] proposed a system to measure the driver's awareness or unawareness in critical situations using eye tracking and object detection. If the driver does not look at an object detected by the system, they receive an acoustic warning. Similarly, Amadori et al. [97] used gaze data and the driver's head pose as input for their decision anticipation model. The model then predicts the likelihood of the driver making correct or wrong decisions. While such systems continuously monitor the driver and indicate that something has been missed or a decision was wrong, they only offer a binary output. To the best of our knowledge, there is currently no dedicated metric for real-time estimation of an SA score in driving.

Hooey et al. [103] modeled a pilot's SA as the ratio of perceived/actual situation elements (SEs) and required/optimal SEs. Shuang et al. [104] further evaluated this approach by comparing the results of the SA model with the two subjective techniques SAGAT and SART. In the aviation context, optimal SEs can be considered to include all cockpit instruments. Actual SEs can be estimated by analyzing the pilot's gaze. If the gaze stays on the same region/instrument for a certain time, it can be counted as a perceived SE. As part of this thesis, we transfer the concept of Hooey et al. [103] from the static cockpit environment to the highly dynamic environment of driving in Chapter 4.

2.3 Video Streaming

In the previous sections, we discussed solutions that provide the operator with an immersive feeling and allow for assessing the operator's SA. As demonstrated by Georg et al. [56], a good video quality supports the operator in gaining a better SA of the remote situation. In this section, we summarize

approaches for adaptive video streaming designed to provide the operator with the best possible video quality.

The main goal of adaptive streaming is to effectively use the available transmission resources to achieve the best possible quality and a continuous video stream without stalling. In applications such as video streaming for home entertainment, stalling only affects the user's quality-of-experience (QoE). In ToD, stalling becomes a severe security issue. The concepts discussed in this section aim to avoid stalling while delivering the best possible quality. We first introduce a general pipeline for video communication as well as the fundamentals of video coding for one or multiple camera views. Then, we review high level methods for adaptive and low delay video streaming before focusing on methods that consider the special requirements of ToD and the hardware limitations of vehicles. Finally, we discuss how to control the video encoder to match these requirements and how to measure the resulting quality.

2.3.1 General Video Communication Pipeline

Streaming is the simultaneous transmission and usage of video and potentially audio data. This means that novel content is continuously transmitted from the server to the client. Simultaneously, content that is already available on the client is relayed to the user. Figure 2.6 shows the schematic overview of a general streaming pipeline.

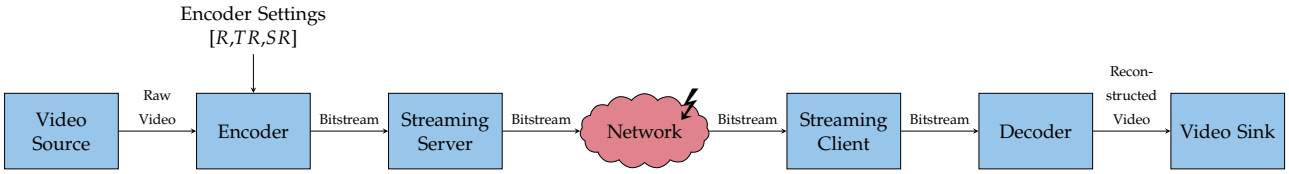


Figure 2.6 Overview of a general streaming pipeline consisting of a video source, video encoder, and streaming server. The client side consists of the streaming client, a video decoder, and a video sink such as a display.

A video source such as a camera or video file provides the raw video content. This raw content is compressed by a video encoder based on the encoder settings. A streaming server transmits the encoded bitstream over a network connection to a client. The decoder on the client side decodes the bitstream received from the server. The reconstructed video content decoded from the bitstream ends in a video sink such as a display, where it is shown to the user. The methods discussed in the remainder of this section all refer to components of the general pipeline shown in Figure 2.6.

2.3.2 Fundamentals of Video Compression

Before we discuss existing video streaming solutions, we first introduce the fundamentals of video compression that are required for understanding the related work and contributions presented in this thesis. The goal of video compression is to minimize the data rate (bitrate) of the video content while affecting the visual quality as little as possible. The video community is continuously improving video compression and has agreed on video coding standards such as H.264/Advanced Video Coding (AVC) [105], H.265/High Efficiency Video Coding (HEVC) [106], and H.266/Versatile Video Coding (VVC) [107]. All video coding standards established today rely on the same fundamental techniques for encoding (compressing) and decoding (decompressing) a video. Next, we discuss the most important concepts typically included when compressing a video.

- **Color Conversion:** Humans usually consume video content as color images based on red, green, and blue (RGB) color components. Since the human vision system (HVS) is less sensitive to changes in color than to changes in brightness, the RGB content is first converted into the YCbCr color space [105]. The Y component represents the brightness, also known as *luminance*, while the color or *chrominance* components Cb and Cr represent the blue-difference and red-difference

relative to gray. The *chrominance* channels are often used in lower resolution compared to the *luminance* channel, taking advantage of the HVS's sensitivity to brightness. This technique is also known as *Chroma Subsampling* [105].

- **Block Partitioning:** Video codecs divide the image into smaller blocks before further processing. For the H.264 standard, for instance, these blocks are called macro blocks (MBs) with a size of 16×16, 8×8, or 4×4 pixels [105]. The encoder as well as the decoder process these blocks, usually from top left to bottom right. The successors of H.264 [105], H.265 [106] and H.266 [107] allow for larger and more flexible block sizes and shapes.

- **Frame Types and Group-of-Pictures Structure:** A group of pictures (GoP) describes a sequence of frames. A single frame can be categorized into one of the following three types:
  - **I-Frame:** An I-frame, also known as intra-, reference-, or key-frame, is a self-contained frame that can be decoded exclusively by the information it contains, similar to a single picture. The first frame of every video sequence is usually an I-frame.
  - **P-Frame:** A P-frame or predicted frame is a frame that can be predicted using only the difference to the previous I-frame or P-frame. Since it only contains the difference to the previous frame, the data consumption is much lower than for an I-frame.
  - **B-Frame:** A B-frame or bi-predictive frame is a frame that relies on past and future I- or P-frames. Since the encoder has to wait until the next I- or P-frame frame arrives to encode the respective B-frame, this type of frame inherently introduces a delay which makes them unsuitable for live streaming applications [108, 109].

  The length of a GoP is defined as the interval length between two I-frames.

- **Predictions:** In video coding, there are usually two types of predictions that make use of the spatial and temporal redundancies of the video.
  - **Intra Prediction** utilizes the spatial redundancies within a single frame. The values for the current block can be approximated based on neighboring blocks already processed by the encoder. This approximation exploits the fact that in natural images, the colors of neighboring pixels and blocks are changing slowly. Then, the difference (residual image) between the predicted blocks and the true blocks is further processed, since the residual contains values close to zero which require fewer bits for representation in the further processing steps.
  - **Inter Prediction** utilizes the temporal redundancies between two frames. Similar to the spatial redundancies, the encoder tries to represent a block that changed as a translation relative to neighboring blocks of a temporal adjacent frame using so-called motion vectors (MVs). The residual of predicted and true blocks is used again for further processing.

- **Transform:** The prediction residuals are transformed into the frequency domain using transforms such as the discrete cosine transformation (DCT) [105]. Most of the signal's energy is concentrated in the lower frequencies, which allows for removing higher frequencies without affecting the perceptual image quality.

- **Quantization:** The removal of higher frequencies is implemented as a quantization. The coefficients of the transformed image are quantized based on a configurable quantization parameter (QP). The QP can be either directly specified for the encoder when using the constant quantization parameter (CQP) mode or it is calculated by the encoder's rate control.

- **Entropy Coding:** Finally, entropy coding compresses information of the transformed image such as the quantized coefficients or MVs losslessly.

- **Rate-Distortion Optimization:** To achieve the best possible trade-off between bitrate and quality or distortion, the encoder employs rate-distortion optimization (RDO). The RDO process identifies the optimal point, depending on a given Lagrangian weight, at which the video shows acceptable distortion at a sufficiently low bit rate.

- **Rate Control:** Rate control is a mechanism that controls the quantization of the encoder, for instance, to achieve a constant bitrate. Since the video content and hence the spatial and temporal complexity vary over time, the rate control estimates the quantization level required to achieve a certain bitrate. In particular for low delay video streaming such as required for teleoperation, a precise rate control enables small buffers and hence keeps the delay small [110].

The resulting bitstream is then transmitted over the network. In ToD, there is usually more than a single camera view that has to be encoded. Therefore, we discuss existing solutions and extensions to the available video coding standards for multi-view video coding and streaming next.

2.3.3 Multi-View Video Streaming

Most video coding standards available such as H.264/AVC [105] or H.265/HEVC [106] provide multi-view extensions that directly allow for encoding stereo/multi-view representations of the same scene. The multi-view extension for H.264/AVC is Multiview Video Coding (MVC) [111] while Multiview High Efficiency Video Coding (MV-HEVC) [113, 112] extends H.265/HEVC. These extensions transfer the temporal concept of inter-frame prediction for a single view to the inter-view prediction of multiple views in the spatial domain. One view is selected as the base view, while the other views are encoded relative to this base view. A requirement for these multi-view extensions are overlapping regions where the inter-view prediction can benefit from similarities between neighboring views [114]. In ToD, the cameras mounted on the vehicle are arranged to capture different parts of the surrounding environment, resulting in few overlapping regions. Since there are only few similarities between neighboring views, using multi-view video coding is less beneficial.

A better solution for processing multi-view video streams showing different scenes such as in ToD is to use multiple encoders. Multiple encoders processing the individual camera views with different content will result in different visual quality levels for the individual views. An additional joint rate control algorithm can be used to equalize the resulting perceptual quality of the individual streams [116, 115]. A common joint rate controller allocates the bitrate for every individual view while also controlling a separate encoder for that view. In ToD, the importance of individual camera views highly depends on the traffic situation. Rather than trying to maintain a similar quality for all camera views, the goal of this thesis is to favor specific camera views with a higher quality based on the current traffic situation. This is important for ensuring that a human driver is aware of the most important views in the current situation. Our proposed solution is introduced in Chapter 5. Since using individual encoders for each camera view is the most suitable solution in ToD, we next discuss existing approaches for transmitting video streams adaptively based on the currently available transmission rate of the network.

2.3.4 HTTP Adaptive Video Streaming

The broadest distribution of video streaming applications is in the entertainment sector with its various video-on-demand providers and social media applications. The state-of-the-art technology in this field is HTTP adaptive streaming (HAS) with its standardization of dynamic adaptive streaming over HTTP (DASH) in 2011 [117]. In HAS, the original video is partitioned into multiple segments, or chunks, of equal playback length [118]. Every chunk is encoded in multiple representations of bitrate, resolution, and quality, which are then stored on the server. Additionally, a manifest file stores the information about the available representations as well as hypertext transfer protocol (HTTP) uniform resource locators (URLs) to the individual segments. HAS utilizes HTTP with the Transmission

Control Protocol (TCP) as the transport-layer protocol to transmit the chunks requested by the client. Figure 2.7 shows a typical HAS streaming pipeline.

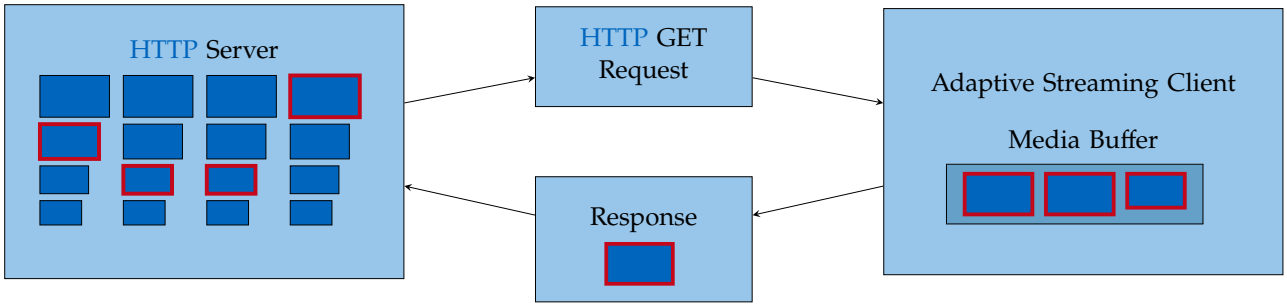


Figure 2.7 Overview of a typical HTTP adaptive streaming (HAS) streaming pipeline with chunks of the video content encoded in multiple resolutions and bitrates. Based on the available transmission rate of the network, the client selects the next chunk most suitable for ensuring a high video quality without stalling.

At the beginning of a streaming session, the client downloads the available manifest file from the server. With the knowledge about the available video representations, the client continuously requests the next chunk in the appropriate rate that matches the current network conditions, while simultaneously playing the already downloaded content. With this concept, the client tries to show the highest possible quality with a minimum number of stalling events [118]. Since the client requests the content, this kind of streaming can be classified as pull-based streaming [118]. The client side adaptation is an important advantage of HAS and leads to an excellent scalability, since the server's workload is almost independent of the number of clients. While there also exist adaptation schemes on the server side or hybrid solutions, most adaptation schemes available reside on the client side [118].

These client-side adaptation schemes select the appropriate video bitrate for the next chunk to be downloaded based on one or more metrics such as the current playback buffer size or the available transmission rate. Buffer-based systems such as Buffer Occupancy based Lyapunov Algorithm (BOLA) [119], which is also used in the standard DASH client, observe the client buffer and aim to keep the buffer at a constant fullness. Rate-based adaptation algorithms such as fEedback Linearization Adaptive STreamIng Controller (ELASTIC) [120] try to estimate the currently available transmission rate and select the best possible chunk that can be downloaded in time. Spectrumbased QUality ADaptation (SQUAD) [121] jointly uses the available transmission rate and buffer information while trying to minimize the number of quality switches. Instead of using fixed control rules for selecting the best segment, PENSIVE [122] is based on a neural network. Its goal is to show the best possible QoE, outperforming the other approaches mentioned above.

The operational design domain of most solutions presented is entertainment and video on demand [118]. In video-on-demand applications, the video content is already known in advance and the additional delay caused by chunk-sizes of a few seconds is irrelevant. In contrast, the requirements for teleoperation are critical to delay [12]. Lottermann et al. [123] proposed a DASH-based solution for providing live video content streamed from a vehicle. To reduce the delay introduced by the video-content segmentation, the segment size was reduced to about 0.5 s. Since DASH-based adaptation schemes require different encoded versions of the same video content, the time for encoding the video also contributes to the delay. The authors addressed this by reducing the set of available levels based on the current network conditions in order to reduce the time required for encoding. With a chunk size of 0.5 s, the delay introduced by this streaming solution is still too large for ToD [12, 14].

Further approaches for low latency streaming in the automotive context use HEVC encoding, a medium access control (MAC) layer adaptation and communication over vehicular ad-hoc networks (VANETs) [124, 125]. However, VANETs provide only a limited range since they were designed for vehicle to vehicle (V2V) communication. Other applications providing live content uplink streaming usually use RTP-based methods [118]. Yu et al. [126] compared the three popular video conferencing services Microsoft Skype, Google Hangout, and Apple Facetime. All of them mainly use RTP/UDP

as the underlying streaming protocol, while Google Hangouts switches to TCP in case of blocked UDP connections. Favario et al. [127] compared mobile live streaming approaches using the Real-Time Messaging Protocol (RTMP) and HAS. They demonstrated that real-time streaming can be implemented in the best way when using TCP with RTMP-based solutions for a small number of clients and HAS-based approaches for a large number of users. Especially for live content, there is the drawback of TCP's acknowledgement concept [128]. Delay critical applications such as video conferencing systems or teleoperation systems prefer UDP or RTP-based communication [129]. The ToD setup of [7] as well as the one proposed in Chapter 3 use RTP/UDP-based video communication. Next, we discuss the specific requirements for a low delay streaming pipeline as well as existing approaches.

2.3.5 Low Delay Video Streaming

The delay required for safe ToD should be less than 300 ms based on the results of Neumeier et al. [48]. Here, the video transmission is the main contributor to the delay due to the large amount of data. The authors distinguished between 50 ms introduced by the sensors and 250 ms as network latency. However, this is only a coarse distinction of the delay contributors and there exist further aspects that need to be considered.

Bachhuber et al. [130] performed a general end-to-end analysis on the delay contributors in video streaming. The authors defined this end-to-end delay as Glass-to-Glass (G2G) delay, which starts at the camera lens and ends at the screen glass. Additionally, they proposed a low complexity system for G2G delay measurement [131]. The results of their delay analysis demonstrated that cameras and displays with a low update rate cause significant delay that has to be addressed. Figure 2.8 shows a general video communication pipeline and the main delay contributors such as analyzed in [130].

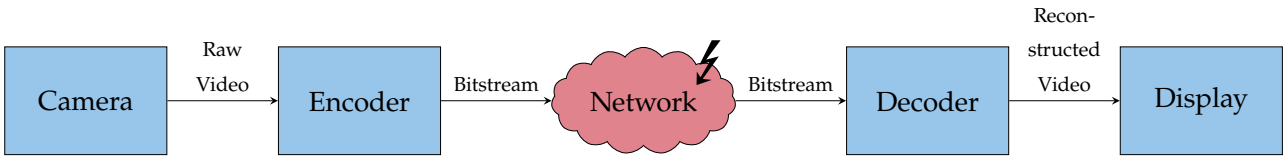


Figure 2.8 General video communication pipeline and delay contributors.

In addition to the delay of camera and display, the encoding and decoding delay as well as the transmission delay have to be considered. Bachhuber et al. [130] further distinguish between camera, encoder, decoder, and display as sources of delay. In this thesis, we consider the hardware used in the vehicle as well as the operator workstation as given and only focus on the encoder and configuration options to keep the encoder delay small.

Loschky et al. [132] investigated the impact of different levels of delay and pointed out that delays below 60 ms are typically not perceivable by humans. Schreier et al. [108] proposed the usage of forward predictive frames only (I/P-frames) and to avoid bi-directive frames (B-frames), since they introduce an additional encoding delay due to their dependency on future frames. Further, the encoder should be configured for low latency encoding [130]. While this increases the encoding speed, it prevents the encoder from doing extensive searches and hence results in a higher bitrate compared to slower, but more efficient coding modes. Based on these restrictions for live video streaming, the H.264 codec [105] is still widely used, although its successors H.265 [106] and H.266 [107] are already available [133].

Besides encoding the video as fast as possible, small buffers are another way to keep the delay low. Exceeding the available transmission rate results in additional delay and likely congestion. This is particularly critical for small buffers that intend to compensate network issues. Using small buffers while still matching the available transmission resources requires precise bitrate control. Constant bitrate control that aims to match the video bitrate constantly over time should be preferred to

variable bitrate control, which is mainly used for on-demand video content to achieve a constant quality level [123].

The default x264 [134] constant bitrate control achieves a bitrate that is on average constant over a certain time. For low delay video streaming, a constant output bitrate at any time is mandatory. Rate control approaches such as the ρ-domain rate control [135] and its extensions [110] estimate the bitrate more precisely and hence allow for matching the bitrate precisely at any time. Gao et al. [136] further model the ρ-domain output as an exponential function to avoid the evaluation of all quantization levels, which reduces complexity. The precision of the rate control in general depends on the accuracy of the bitrate model used. The encoding hardware such as used in vehicles usually provides only limited accessibility to features of the underlying codec, which prevents using approaches such as ρ-domain rate control. In the next section, we discuss existing bitrate models that allow for a precise estimation of the video bitrate with respect to the limitations and requirements of vehicles and ToD.

2.3.6 Bitrate Models

The core part of every rate control is the bitrate model estimating the resulting bitrate for a set of encoding parameters without the need to actually encode the video. The bitrate model proposed by Yao et al. [137] considered both the influence of quantization and frame rate and uses content-dependent model parameters. Ma et al. [138] extended this work by estimating the two video-specific parameters based on three content-dependent features. In [139], the authors further updated their model to additionally consider the impact of the frame size. Downsides of their bitrate models are the content-dependent features, which are computationally complex and based on the motion estimation and frame difference information of the underlying codec [140]. Such an approach is therefore difficult to implement for embedded systems, where the encoder can be considered as a black box and accessing the encoder apart from a given interface is not possible.

Lottermann et al. [141, 142] addressed this problem by estimating the bitrate independently of codec-dependent features. This approach allowed them to decouple the rate control from the actual video encoder. In [141], the authors proposed a bitrate model that considers the impact of frame rate and the QP. Then, they modeled the impact of the frame rate and the QP based on spatial and temporal activity measures. Other than for the content-dependent approach used by Ma et al. [138], it is possible to extract both activity measures from the uncompressed video sources. Lotterman et al. [142] extended their previous work [141] to include GoP characteristics, since they significantly influence the overall video bitrate. While these models do not consider all influence factors such as the frame size, they inspired the analytical bitrate model proposed in Chapter 6.

The bitrate models discussed estimate the bitrate analytically and consider the influence of the QP. This simplifies the modeling process, since no additional rate control by the encoder affects the bitrate. However, it requires the encoder to be used in CQP mode. Covell et al. [143] followed the idea of Ma et al. [138] and modeled the bitrate for the constant rate factor (CRF) encoding mode. They proposed a neural network to estimate the content-related model parameters and the CRF. Sun et al. [144] improved the linear model of Covell et al. [143] to increase the bitrate accuracy. Similar to the original model, the content-related parameters are estimated using a neural network. Both networks use MV information, which is available since the models were mainly designed for transcoding applications. In a ToD context where the encoder is a black box, such information is likely unavailable.

2.3.7 Video Quality Assessment

Besides estimating the bitrate to control the video encoder, assessing the resulting quality is another approach for controlling the video encoding pipeline [145]. Additionally, any kind of video processing requires quality assessment to rate and compare different processing techniques. Most objective video quality metrics (VQMs) are based on subjective user studies where the VQM tries to match the results

of the user study as closely as possible [146]. In general, objective VQMs can be classified into three categories [145]:

- **Full-reference** quality metrics estimate the quality relative to the undistorted reference signal. Hence they require access to the original video source, which limits their applicability to the receiver side.
- **Reduced-reference** quality metrics do not require access to the full original image content, but to a set of features extracted from both videos. These features are used when the original video is not available or when the transmission of the full video is not possible.
- **No-reference** quality metrics can be calculated from the distorted image or video signal without having information about the original content.

The PSNR is one of the most commonly used full-reference metrics for video quality assessment. However, it does not represent the HVS closely [145]. In 2004, Wang et al. [147] proposed the Structural Similarity Index (SSIM) as a quality metric superior to the PSNR. SSIM exploits the structured information of natural images under the assumption that structural distortion decreases the visual quality. Both PSNR and SSIM are image quality metrics that do not consider the temporal information of a video. Wang et al. [148] extended their SSIM to consider the temporal domain as well. This improved the correlation of SSIM on the video quality, but comes at the cost of increased computational complexity.

Peng et al. [149] proposed the spatio-temporal video quality metric (STVQM) to predict the video quality by considering both spatial and temporal impairments. STVQM is based on the PSNR to measure the spatial quality per frame and a correction factor to model the temporal quality perception. This temporal correction factor is estimated by the frame rate as well as the spatial and temporal activity measures obtained from the undistorted video. Ou et al. [140] proposed a similar approach to model video quality as an independent product of spatial and temporal quality factors. The spatial factor is based on the PSNR, while the temporal factor considers the frame rate of the video. Both factors depend on a respective single content-dependent parameter. In [150], the authors extended their spatio-temporal quality model to further consider the influence of different frame resolutions. This frame resolution factor depends on another single content-dependent parameter. However, the content-dependent parameters are calculated from motion information and frame difference, which are computationally complex and only available with access to the underlying codec. This makes them less suitable for ToD and live streaming applications compared to the STVQM.

Wichtlhuber et al. [151] proposed the real-time video quality metric (RT-VQM), which is specifically designed for live streaming applications. RT-VQM is a Graphics Processing Unit (GPU) supported version of the precise and standardized VQM proposed by Pinson et al. [152]. Slicing the video content before loading it into the GPU's memory allows for a fast processing and estimation of the video quality. The underlying VQM itself is a reduced-reference metric based on three low-bandwidth features. This enables the RT-VQM to be used for live streaming applications as well as on streaming servers for transcoding evaluation, where no access to the raw video is available.

Zhang et al. [153] proposed the multi-dimensional video quality metric (MDVQM), a no-reference VQM closely representing the HVS. The authors modeled the MDVQM as a product of a signal-to-noise ratio (SNR) model and two correction factors that include the influence of spatial and temporal resolution. The features used in the VQM are based on the spatial and temporal activity measures estimated from the encoded video sequences.

Besides designing a metric that closely matches the HVS in all scenarios, an alternative approach is to combine the strengths of multiple individual VQMs. Different VQMs have special strengths and weaknesses depending on the video content. Lin et al. [154] proposed the Fusion-based Video Quality Assessment index (FVQA) to estimate the video quality by fusing the quality scores of multiple individual VQMs. FVQA achieves a meaningful performance for a wide range of video contents and codecs compared to state-of-the-art VQMs. With the Ensemble-learning-based Video Quality

Assessment index (EVQA) [155], the authors proposed another fusion-based VQM. In contrast to the FVQA learning static fusion rules for multiple video content groups, the EVQA adopts a more dynamic learning-based approach. In cooperation with Netflix, Lin et al. [154, 155] proposed Video Multi-Method Assessment Fusion (VMAF) [156] to estimate the subjective video quality. VMAF combines the strengths of multiple elementary quality metrics such as PSNR and SSIM. An SVM regressor fuses the individual metric scores to a single final quality score. VMAF outperforms traditional VQMs and has been established the state of the art for video quality assessment [157]. In this thesis, we use PSNR, SSIM, STVQM, MDVQM, and most importantly VMAF to evaluate the methods proposed in Chapter 5 and Chapter 6.

2.3.8 Multi-Dimensional Stream Adaptation

Most stream adaptation approaches focus on controlling the bitrate or quality of the video. Similar to spatio-temporal VQMs that measure the influence of the frame rate and the frame size on the video quality, these parameters can also be used to control the video stream. Reed et al. [158] proposed a multi-dimensional bitrate control scheme that considers the frame rate and the frame size besides the QP to control the buffer level. Afonso et al. [159] proposed a spatio-temporal resolution adaptation scheme that dynamically resamples the input video during encoding. The authors apply multiple perceptual VQMs to select the optimal combination of frame rate, frame size, and QP. In [145], a QoE-driven multi-dimensional adaptation (MDA) scheme was proposed that uses the spatio-temporal MDVQM [153] to select the optimal combination of spatio-temporal resolution and quantization for adapting the video stream. In this thesis, we use the QoE-driven MDA approach of [145] as part of our contribution in Chapter 5.

2.3.9 Image Preprocessing

Lastly, we discuss spatial image filtering as a preprocessing technique to manipulate the image content [160]. This is relevant for the contributions of Chapter 6 that address the hardware limitations of vehicles to still allow for individual rate/quality adaptation of multiple video streams when only a single encoder is available.

Popkin et al. [160] proposed a computationally efficient algorithm for space-variant Gaussian blurring of images. Encoding the residuals preprocessed this way reduces artifacts, improves the quality, or reduces the bitrate for the encoded image or video. Sun et al. [161] as well as Astle et al. [162] proposed to filter the entire image in a preprocessing step to reduce the perceptibility of coding artifacts for higher quantization during encoding. Karlsson et al. [163] proposed ROI video coding using Gaussian preprocessing filters. Applying the filter on the background reduces the bitrate required for the encoder to encode the image, while the foreground stays at a constant quality. Alternatively, the quality of the foreground can be increased by keeping the bitrate of the entire image constant. To reduce the bitrate required for video streaming in mobile environments, Huang et al. [164] used a similar preprocessing approach by applying a low-pass filter on the image background. As an extension of [163], Karlsson et al. [165] proposed halving the temporal resolution (frame rate) of the background. This results in reusing blocks of the odd-indexed frames for the even-indexed frames during the encoding process. Budagavi et al. [166] used a Gaussian low-pass filter on the outer regions of 360° videos to reduce the bitrate required for encoding less important parts of the image.

Grois et al. [167] proposed a complexity-aware adaptive ROI prefiltering scheme for scalable video coding. The authors applied the prefilters dynamically using a transition region between foreground and background to increase the visual presentation quality of the ROI. In a comparison of different preprocessing filters such as Gaussian, Wiener, and Wavelet filters, the Gaussian low-pass filter performed best due to its low computational complexity [168, 169]. Additionally, Grois et al. [167] have applied predefined filter configurations of specific regions in the frame to achieve the best possible quality for the ROI. However, such an extension depends highly on information of the underlying codec such as the motion estimation result, which requires direct access to the encoder.

Embedded devices are usually equipped with a hardware encoder with only limited configuration possibilities provided by the encoder interface. In this case, the input data has to be preprocessed before the actual encoding step. In Chapter 6, we propose a preprocessing scheme that enables rate/quality adaptation of multiple camera views when only a single encoder is available.

2.4 Summary

In this chapter, we summarized the background as well as the related work most relevant to the contributions made in this thesis. Firstly, we gave a general overview of state-of-the-art ToD systems and how they addressed the task of providing the operator with an immersive feeling of being physically in the vehicle. Several approaches were discussed that propose to compensate the transmission delay by augmenting the display, providing control feedback, or improving the transmission by itself. Our first contribution in Chapter 3 is based on these considerations and offers the first ToD setup for the CARLA simulator. Next, after defining the term SA, we discussed methods for assessing and monitoring the driver's SA, which are related to our contributions in Chapter 4. Finally, we introduced a general pipeline for video communication as well as the fundamentals of video coding for one or multiple camera views. We reviewed methods for adaptive and low delay video streaming, video quality assessment, and methods that consider the special requirements and hardware limitations for ToD. These methods build the basis for the contributions of Chapter 5 and Chapter 6. In the next chapter, we introduce a teleoperation framework to extend existing driving simulators for vehicle remote control as the first major contribution of this thesis.

Chapter 3

Teleoperated Driving in Simulated Environments

In this chapter, we present the teleoperation framework TELECARLA for driving in simulated environments. The proposed framework consists of an online-configurable low delay streaming pipeline, a customizable graphical user interface (GUI), and a scenario evaluation module. TELECARLA builds the basic framework for the implementation and evaluation of the subsequent contributions made by this thesis.

Most teleoperated driving (ToD) simulation systems currently available extend vehicle simulators with a video streaming component. The commercial driving simulator SILAB [33], for instance, provides hardware components as well as the software to perform realistic vehicle driving simulations. However, using this simulator for ToD requires a screen grabbing software to access the frames due to limited accessibility of the simulator [9, 11].

In contrast to such closed source systems, open source driving simulators are usually open for extension and allow for accessing sensor information. We hence propose TELECARLA, an open source extension of the Car Learning to Act (CARLA) simulator [16] for ToD research using low-cost off-the-shelf components. Before introducing the design of the teledriving framework, we present the hardware setup that enables control of the ego vehicle in the simulated environment based on a wheel and pedals.

The teledriving framework presented in this chapter has been published in [19] with the source code available on GitHub[1]. Additionally, TELECARLA has been integrated as an official plugin to the CARLA simulator ecosystem [170] and is actively being maintained. Further implementation details of TELECARLA are given in Appendix A.

3.1 Hardware Setup

The basic hardware setup for controlling the vehicle in the simulation consists of a gaming seat as well as the Logitech G29 gaming wheel and pedals. These components have a price of less than 500€, which enables an easy, cheap, and reproducible setup. Three 24 ” displays arranged in a setup similar as in [7] visualize the simulated environment. The number, type, and arrangement of the displays can be customized according to the user's needs. A regular desktop PC extended with an NVIDIA GTX 1050 Ti graphics card runs the CARLA simulator. Figure 3.1 visualizes the complete hardware setup.

Controlling the vehicle in the CARLA simulator via a wheel and pedals requires small modifications to the client, as well as parameter tuning to customize the sensitivity of the wheel and the pedals for an optimal driving experience. The default driving setup uses a single RGB camera front view with 90° horizontal field of view (FoV) for controlling the vehicle in the simulation. Further sensor information such as ground-truth semantic segmentation, optical flow, and depth views are available as well. Next, we present the proposed teleoperation framework TELECARLA that extends the CARLA simulator setup with a teledriving mode.

[1] https://github.com/hofbi/telecarla

Figure 3.1 Teleoperation workstation running the CARLA simulator with wheel and pedals control, gaming seat and three displays.

3.2 TELECARLA

In this section, we present TELECARLA, a framework that allows for extending existing open source driving simulators such as CARLA with a teleoperation mode. TELECARLA comprises the hardware setup presented in Section 3.1, a customizable GUI for the operator workspace, a communication interface for exchanging control commands as well as status information between the server and client, and a low delay streaming pipeline with an interface for live video stream adaptation. The entire system uses the Robot Operating System (ROS) [45] as middleware providing common interfaces to design a modular and scalable micro-service-based architecture. Any driving simulator that uses ROS can be extended for teledriving using the system proposed in this chapter. Here, we use the CARLA ROS bridge [171] to connect the CARLA simulator using the proposed framework. Since the CARLA simulator is the first simulator that has been extended for teledriving, we name our system TELECARLA.

3.2.1 CARLA Network Architecture Analysis

We first analyze CARLA's network architecture to motivate the need for designing a new teleoperation framework. Internally, CARLA already uses a client-server architecture. In CARLA as well as other comparable simulators, the scenes are rendered on the server and then streamed to the client. This design choice allows for the simultaneous connection of multiple clients to a single CARLA server. For its clients, CARLA provides a Python application programming interface (API) for fast and simple usage and the implementation of various tasks.

By default, the client-server connection is via localhost on the local machine, but the client can also be connected to a remote server. Connecting the client to a remote server already allows for ToD in the simulation. However, the data transmission of CARLA is by design uncompressed, resulting in large data transmission rates. Figure 3.2 shows an exemplary network trace of CARLA's raw data transmission.

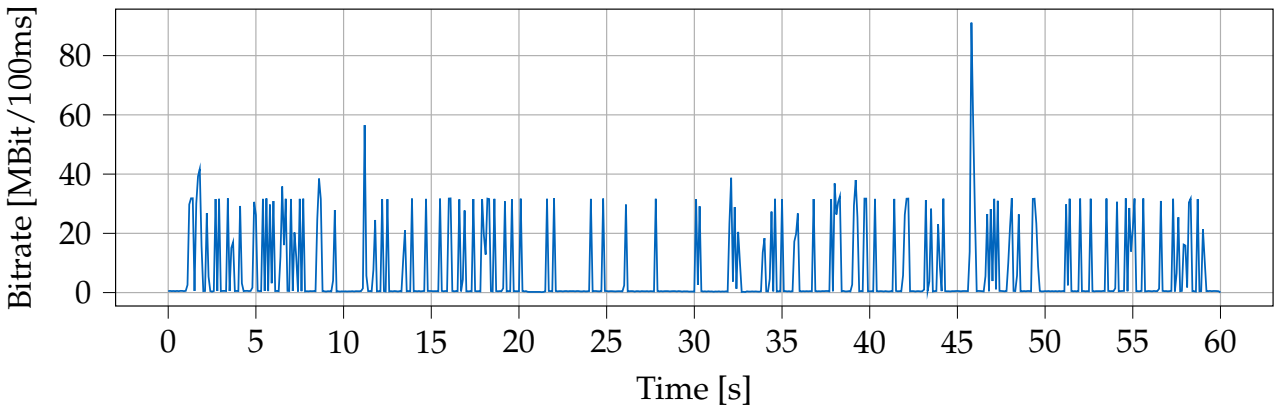


Figure 3.2 Network trace of CARLA's raw data transmission.

This huge amount of data, transmitted without any compression, results in a very low frame rate of 3 fps to 4 fps, which makes ToD nearly impossible. CARLA transmits the sensor data without compression, since it has been designed for autonomous driving research where most algorithms require raw data input. Since CARLA is open source, this issue could be addressed by integrating video compression into their streaming pipeline. However, this approach would require rebuilding the entire simulator library for changes in the video streaming system or adaptation algorithms running on the server side. Furthermore, such modifications on the core part of CARLA would be less flexible compared to an external solution. Hence, we design such an external solution that can be used with CARLA as well as other simulators.

3.2.2 Proposed Concept

The proposed framework TELECARLA uses the Robot Operating System (ROS) [45] as the underlying middleware, which is widely used in the area of autonomous driving [47]. ROS provides common interfaces that allow for creating a modular and scalable system. The ROS-based connection of TELECARLA with the CARLA simulator is established via a ROS bridge available for the CARLA simulator [171]. Then, TELECARLA encodes and transmits the video data using GStreamer [172] from the TELECARLA server running on the same machine as CARLA to a TELECARLA client running on a separate machine. Vehicle status information and the control commands entered by the operator are transmitted between these two machines using the Remote Procedure Call (RPC) protocol. Figure 3.3 shows the proposed architecture with the available CARLA components in blue and the proposed TELECARLA modules in green.

Every green block represents a separate ROS module, or node, which can be used multiple times in the same system. This allows for streaming multiple camera views by creating pairs of TELECARLA servers and clients for each camera available in the vehicle. Every TELECARLA streaming server provides an adaptation interface for controlling the video encoder. On the client side, the video stream is decoded and shown to the user. Details for the user interface (UI) and streaming module are presented in the following sections.

TELECARLA connects to the CARLA server via localhost and its ROS bridge similar to any other client available. This allows for using all existing functionality of CARLA such as traffic management, weather conditions management, as well as scenario evaluation. Further, custom CARLA clients can be implemented depending on the use case.

For a fully controllable network, we suggest using existing network emulation software to control the network conditions. Available network emulation software such as `tc-netem` allows for controlling the network parameters such as the available transmission rate, the transmission delay, or the packet-loss rate to evaluate different adaptation algorithms.

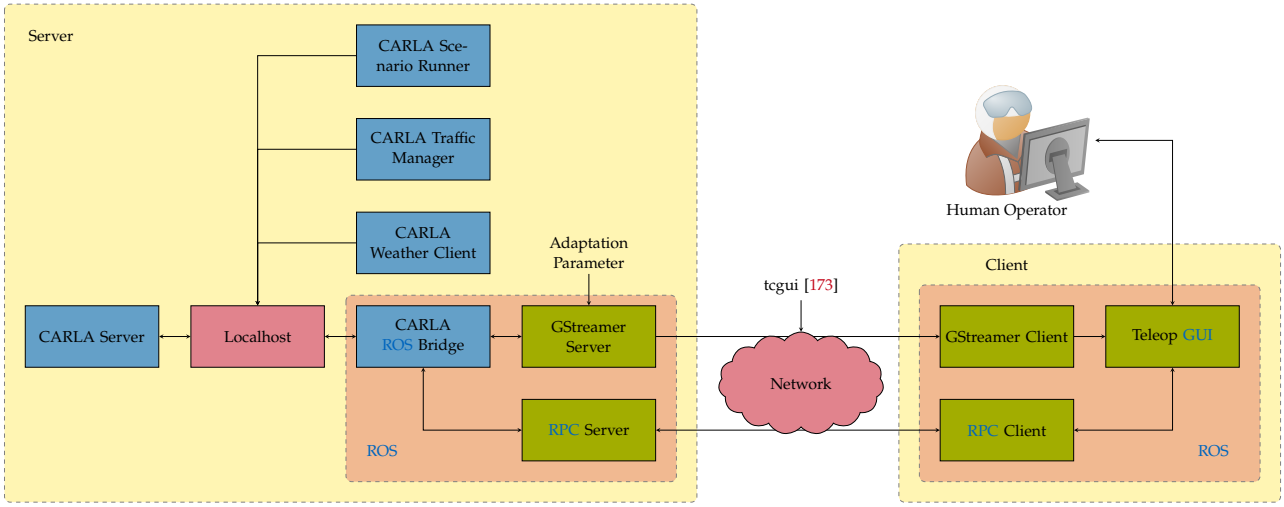


Figure 3.3 TELECARLA architecture extending the CARLA simulator for ToD. CARLA components are visualized in blue, TELECARLA modules are shown in green.

3.2.3 Teleoperation User Interface

We design a GUI as an interface between the human operator and the TELECARLA client receiving the sensor data. The GUI displays the sensor information such as the video streams and reads the vehicle control commands entered by the human operator via wheel, pedals, and keyboard. The GUI is specifically designed to render the video streams without any buffering to avoid additional delays introduced when displaying the content [130]. Additionally, the GUI renders force feedback on the steering wheel when the vehicle is controlled by an autonomous agent to synchronize the steering-wheel angle with the actual steering angle of the vehicle. The synchronized angles are in particular important for takeover situations where the human operator takes over manual control of the vehicle.

The layout and type of sensor information rendered on the screen is defined by a simple configuration file to customize the GUI according to the available sensor data. Figure 3.4 shows an exemplary setup with six camera views.

Figure 3.4 TELECARLA GUI with a six camera setup and vehicle status information.

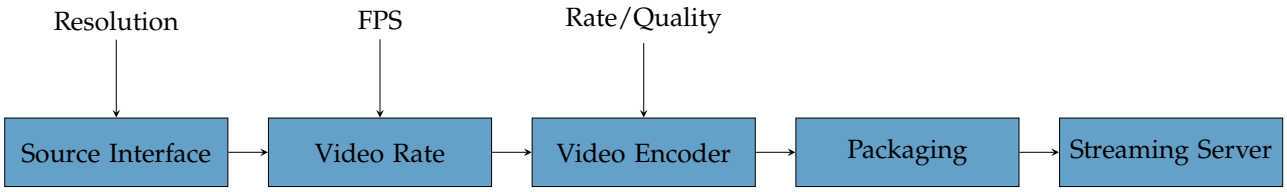
Three front-facing camera views in the bottom row and three rear-facing camera views in the top row cover the surrounding traffic environment. Each camera view is augmented with its own video status information located in the top left corner of the respective video segment. The video status information covers the camera name and a timestamp as well as the spatial and temporal resolution of the input video stream. The GUI re-scales the input camera streams to fit into the desired GUI

segment defined by the user. To keep the delay introduced by the visualization as small as possible, each segment is independently updated as soon as new data arrives. This update concept for the individual segments allows for an easy implementation of additional display augmentations such as a predictive display [9] or free corridor [59] to further improve the performance of the operator. The dark segment on the left of the GUI gives an overview of the control signals available for the steering wheel and the keyboard. The block on the top right displays the vehicle status information such as the current speed or the gear position.

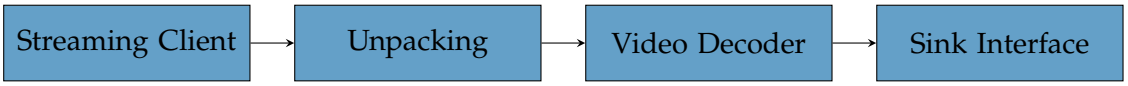
The flexible and modular design of the GUI allows to easily change the layout of the GUI as well as to implement new visualization modules. This allows to visualize a map or a top-level view with the vehicle surroundings, for example. Further implementation details of the TELECARLA GUI can be found in Appendix A.1.

3.2.4 Video Streaming System

The video streaming system is the core part of the proposed TELECARLA system. We designed the streaming pipeline focusing on a low delay video communication such as required for ToD. The streaming pipeline provides an interface that allows for a live adaptation of the video stream. Using the adaptation interface, the video frame rate and frame size as well as the target rate/quality can be updated dynamically to control the video stream. The adaptation can be controlled either manually by the human operator or automatically by an adaptation algorithm such as proposed in Chapter 5. Figure 3.5 shows the two proposed streaming pipelines, Figure 3.5a for the server side and Figure 3.5b for the client side.



(a) TELECARLA server pipeline.



(b) TELECARLA client pipeline.

Figure 3.5 TELECARLA streaming pipelines. The server pipeline exposes the available parameters that can be controlled by the adaptation interface.

A source interface on the server side buffers the arriving frame, parses its resolution as well as the color format, and optionally downscales the frame if specified by the adaptation interface. As soon as a complete frame is in the buffer, the frame is processed by the pipeline to avoid artificial delays introduced when buffering multiple frames. Based on the target frame rate specified, the temporal resolution of the video stream is downsampled if required. The resulting video stream is then encoded using a video encoder configured for low delay video coding. For a given target bitrate, the encoder processes the video using its internal rate control algorithm to achieve a constant bitrate for the encoded video stream. Alternatively, the encoders use the constant quantization parameter (CQP) mode when a certain quality in form of a quantization parameter (QP) is specified. By default, we use the x264 [134] software video encoder due to its speed advantages for low delay transmission compared to more advanced video coding standards such as H.265 [106] or H.266 [107]. This speed advantage is of higher priority in ToD than achieving the best possible compression rate [133]. The encoded bitstream is then streamed to the client using the Real-Time Streaming Protocol (RTSP) protocol. On the receiver side, the encoded bitstream is decoded and then published via the ROS interface for displaying and further processing.

The proposed streaming system uses GStreamer [172], an open source, multi-platform, plugin-based multimedia framework. The individual modules of the streaming pipeline presented in Figure 3.5 are based on available GStreamer plugins. Both streaming pipelines are integrated into a ROS node to rely on the flexible ROS interface and enable scalability with respect to the multi-view streaming system. Further implementation details of the TELECARLA streaming pipelines can be found in Appendix A.2.

3.3 User Study Design

To evaluate the basic functionality of the proposed TELECARLA framework, we compare the TELECARLA streaming pipeline with the original CARLA client-server connection for local and remote driving in a user study. We use the CARLA scenario runner module [174] to conduct the user study and to record the performance of all participants. The scenario runner module contains predefined traffic scenarios that can be executed in the CARLA simulator.

For the user study, we selected a set of six scenario classes. Each scenario covers different turning and collision-avoidance tasks in urban environments with other traffic participants present. Each scenario class contains multiple scenarios of the same type to avoid repeating the same scenario, which could be memorized by the participants. All participants have a regular driving license and began the experiment with 3 min of local (no delay, no compression) driving in the simulator to become familiar with the general setup and the vehicle control. After the introduction phase, we evaluated the three different driving modes listed in Table 3.1.

Local	Local driving (no delay and no compression)
Carla	Remote driving with CARLA's remote connection (delay and no compression)
TeleCarla	Remote driving with TELECARLA streaming pipeline (delay and compression with a target bitrate of 2 Mbit/s)

Table 3.1 Driving modes used for the evaluation.

Every participant had to complete a set of six scenarios for each driving mode: *Local*, *Carla*, and *TeleCarla*. In the *Local* mode, the participants drive without video compression or transmission delay, as if driving the car as an in-car driver. In comparison, the remaining two modes evaluate the remote driving case. The *Carla* mode uses the regular uncompressed client-server communication available for CARLA between two separate machines. In the *TeleCarla* mode, we use the TELECARLA streaming system to compress and transmit the video data between two physical machines. Each set of scenarios evaluated for one of the three modes covers the same scenario classes, but different scenarios. In total, the participants completed 18 test drives, six for each mode.

For the evaluation, we used two Dell Optiplex 9020 with an Intel Core i7-4790 8x3.6 GHz processor, an NVIDIA 1050Ti Graphics Processing Unit (GPU), and 16 GB DDR3 RAM for the server as well as 24 GB DDR3 RAM for the client. Both systems were running with a 64-bit Ubuntu 18.04.3 LTS, ROS melodic, and the GStreamer version 1.16.1. The scenarios were evaluated with CARLA version 0.9.6, using a single front camera with a resolution of 1270×720 pixel at 20 Hz. 20 Hz is the maximum frame rate that can be achieved by the CARLA ROS bridge.

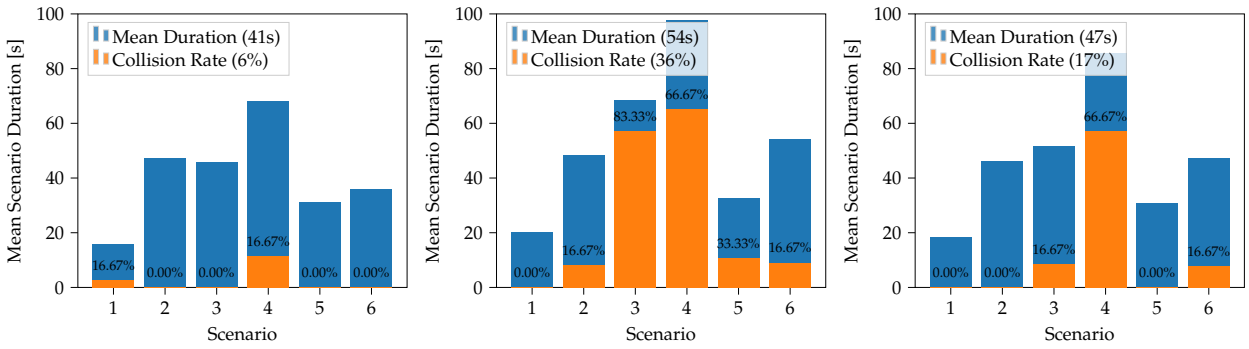
3.4 Results

In this section, we present the results of our experiments. Table 3.2 summarizes the results for the average scenario duration, collision rate, transmission rate, and frame rate of the three driving modes evaluated in the user study.

Mode	Scenario Duration	Collision Rate	Bitrate	Frame Rate
Local	41 s	6 %	-	20 Hz
Carla	54 s	36 %	69 Mbit/s	3.6 Hz
TeleCarla	47 s	17 %	2 Mbit/s	20 Hz

Table 3.2 Results for the driving mode evaluation.

The *Local* driving mode runs at the maximum frame rate of 20 frames per second defined by the CARLA ROS bridge. With 41 s average scenario duration and a collision rate of 6 %, the *Local* mode acts as the baseline for the evaluation. The raw data transmission of CARLA results in an average bitrate of 69 Mbit/s and an average frame rate of 3.6 frames per second for the *Carla* mode. Such a low frame rate leads to a stop-and-go driving behavior of the participants and hence increased the average scenario duration by 13 s or 32 %. Although driving at reduced speed, the remote driving is challenging at such low frame rates, which is highlighted by a collision rate increased by a factor of 6. The *TeleCarla* mode reaches the target bitrate of 2 Mbit/s such as specified for the encoder. While the compression of the video stream introduces an additional processing step, the *TeleCarla* mode still reaches the available update rate of 20 frames per second. The delay introduced by the transmission as well as the visual impairments introduced by the compression increased the average scenario duration by 6 s (15 %). The collision rate for the *TeleCarla* mode is 17 %, which is only half the collision rate of the *Carla* mode. Figure 3.6 shows the average scenario duration and collision rate individually for every scenario class and driving mode.



(a) Driving Mode *Local* **(b)** Driving Mode *Carla* **(c)** Driving Mode *TeleCarla*

Figure 3.6 Average scenario duration in blue and the average collision rate in orange for each driving scenario of the evaluated driving modes: *Local*, *Carla*, and *TeleCarla*. The scenario classes used for the evaluation are: *DynamicObjectCrossing* (1), *FollowLeadingVehicleWithObstacle* (2), *HeroActorTurningRightAtSignalizedJunction* (3), *ManeuverOppositeDirection* (4), *Stationaryobjectcrossing* (5), and *TurnLeftAtSignalizedJunction* (6). Adopted from [19] © 2020 IEEE.

Each bar represents the average duration required to complete the respective scenario. The orange segment within a blue bar represents the average collision rate of all participants for this scenario. Figure 3.6a shows the *Local* mode, Figure 3.6b the *Carla* mode, and Figure 3.6c the *TeleCarla* mode using the proposed streaming system.

Notably, most of the collisions for the *TeleCarla* mode occurred for scenario (4) *ManeuverOpposite-Direction*. In this scenario, the ego vehicle lane is occupied by an obstacle that has to be passed on the opposite lane with continuous oncoming traffic. This requires fast and accurate steering actions, which was difficult for the mainly untrained participants experiencing transmission delay for the first time. However, existing works such as [8, 56] mention that safe ToD requires operators who are specifically trained for such situations.

3.5 Summary

In this chapter, we presented the teleoperation framework TELECARLA for driving in simulated environments. The framework proposed in this chapter extends the open source CARLA simulator with a teleoperation mode. TELECARLA consists of a low delay streaming pipeline providing an interface for live video stream adaptation as well as a customizable GUI and a low-cost hardware setup using off-the-shelf components. All modules of TELECARLA are implemented as separate ROS nodes to maximize modularity and scalability. The modular system design allows to extend any driving simulator that provides a ROS bridge.

An initial analysis of CARLA's streaming architecture suggested that the default client-server connection of the CARLA simulator is not directly applicable for ToD research. This is mainly caused by the raw data transmission between the CARLA server and its clients, which exists since CARLA was designed mainly for autonomous driving research where algorithms typically require raw input data.

We evaluated the proposed system in a user study, comparing it to local driving as well as the available CARLA client-server connection between two machines. The results demonstrate that the proposed framework enables the operator to control the vehicle remotely at the same video frame rate as the frame rate generated by the simulator when driving locally. Further, it significantly outperforms the available CARLA remote connection. TELECARLA builds the basic framework for the implementation and evaluation of the following contributions proposed in this thesis.

Chapter 4

Driver Situation Awareness Assessment

In this chapter, we introduce a method to measure the situation awareness (SA) of a human driver in realtime. Constantly observing the driver and measuring their SA is an important aspect for both in-car driving as well as teledriving. In particular during the transition to human control, the vehicle has to ensure that the driver is sufficiently aware of the current situation [63, 64, 65]. To the best of our knowledge, no method for explicitly estimating the driver SA exists in the literature. We use the driver's gaze as a source of information for assessing their SA such as introduced in Section 2.2.2. The proposed approach is inspired by methods used in aviation and can be applied for both in-car drivers as well as remote operators. Additionally, we use a machine learning (ML)-based approach to predict all relevant elements the driver should perceive in a given scene. Comparing the elements the driver actually focuses on with the elements the driver should have perceived allows for quantifying the driver's SA in realtime.

In the following sections, we first propose a multi-view region of interest (MV-ROI) prediction concept to predict regions most relevant for driving for multiple camera views such as used in teleoperated driving (ToD). Then, we present a method to assess the driver SA in realtime, which was inspired by methods used in aviation. By comparing the driver's gaze with the region of interest (ROI) predictions, the driver awareness can be modeled for the current traffic situation.

Some of the concepts and contributions of this chapter have been published in [20, 22]. An implementation of the system to measure the driver SA presented in this chapter is available as open source on GitHub[1]. The module extends the basic teledriving framework TELECARLA presented in Chapter 3. Implementation details are given in Appendix A.

4.1 Multi-View Region-of-Interest Prediction for Autonomous Driving

The ability to predict the attention of human drivers is the basis for a range of autonomous driving functions. The predicted attention is also an indicator for what a human operator controlling the vehicle should focus on. State-of-the-art approaches for predicting the driver's attention use only a single front-facing camera and rely on automatically generated training data [94, 95, 96]. However, vehicles used in autonomous driving as well as ToD are typically equipped with multiple cameras to cover the complete surroundings of the vehicle. Different camera views, such as the rear-facing views, result in different ROIs compared to the front view. Using a model which is trained on front view data only is unlikely to achieve the same prediction accuracy for the rear views.

The Berkeley Deep Drive Attention (BDD-A) dataset proposed by Xia et al. [95] is based on a single front camera. The authors created the labels for the BDD-A dataset by collecting the gaze data of multiple independent participants watching driving scenes recorded on public roads. The data collected from all participants is accumulated to obtain the ROI annotations. Then, the authors designed an ROI prediction network based on an object detector and trained the network on the BDD-A dataset. In the context of driving, empty space devoid of any objects can also be an ROI. For instance, the adjacent lane during lane changes or the free space that is usually covered by the

[1] https://github.com/hofbi/driver-sa

shoulder check can be highly interesting regions regardless of their contents. Those regions will not be recognized by a typical object-detector-based model. Further, objects such as traffic lights that are highly relevant for the front-facing camera views should no longer be considered as ROIs in the rear-facing camera views. We hence argue that ROIs labeled by humans for a dataset that contains multiple camera views are more complete than the ROIs in a single-view data set. Having dedicated ROIs for each view allows for more accurate ground-truth data for training ROI prediction models. Such a dataset addresses both the issues of multi-view camera setups and the detection of free space.

While a manually annotated dataset results in a better ground truth, the labeling process is significantly more time consuming and expensive compared to automatically generated labels using eye tracking. To still benefit from high-quality ROI labels, we created a comparably small human-labeled dataset which contains six camera views. We refer to this dataset as the MV-ROI dataset. Then, we use the state-of-the-art BDD-A model trained on data for a single front camera as a baseline and finetune the baseline model using our manually labeled MV-ROI dataset. To further distinguish the requirements for the different camera views, we train two separate models with special focus on either front- or rear-view data. Figure 4.1 presents a general overview of the proposed MV-ROI prediction approach.

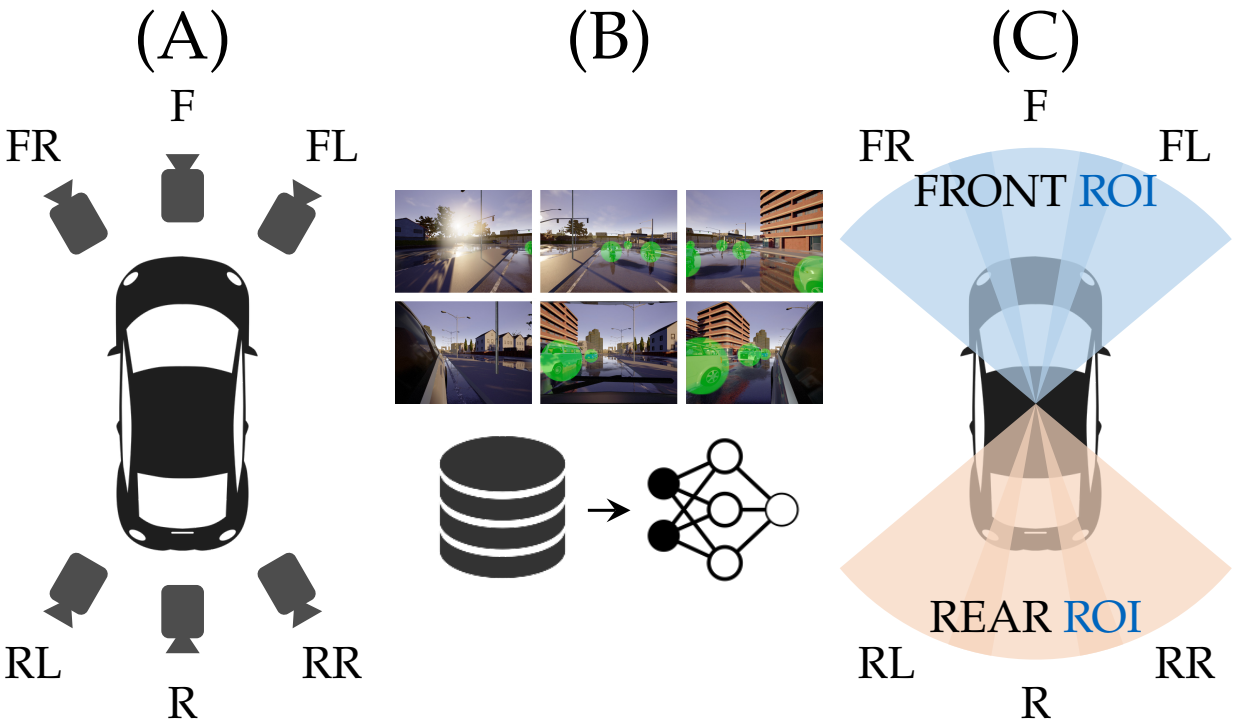


Figure 4.1 Overview of the proposed multi-view region of interest (MV-ROI) prediction. A vehicle equipped with six cameras (A) records the data which are manually labeled (B). Using these data, we train two separate ROI prediction models (C).

We first record driving video sequences in the CARLA [16] simulator using a vehicle equipped with six cameras (A). Next, we manually annotate these video sequences with ROI labels for all camera views (B). To support further manual labeling, we designed a semi-supervised annotation framework to increase the labeling speed while keeping the high-quality labels created by human data annotation. Finally, we train two separate ROI prediction models using our annotated video sequences (C). The MV-ROI dataset as well as the annotation framework are publicly available as open source on GitHub[2].

[2] https://github.com/hofbi/mv-roi

4.1.1 Multi-View Region-of-Interest Dataset

To generate the multi-view region of interest (MV-ROI) dataset, we recorded all data using the CARLA driving simulator [16] under sunny weather conditions. Initially, we recorded ten driving sequences covering different traffic situations such as turning at an intersection. Every sequence was recorded with the six camera setup outlined in Figure 4.1 at a frequency of 10 Hz and a duration of around 10 s. All images were labeled by hand without any model suggestions to obtain high-quality ground-truth data. We decided to use circular ROI labels. Each ROI is thus defined by the center and radius of a circle, which avoids unnecessary complexity of more elaborate shapes.

The ROIs can vary depending on the intended driving maneuver. Since the ROI prediction model does know in advance which maneuver will be executed, we decided to label all ROIs that could be relevant in each driving scenario. In driving, it is safer to predict more regions than required compared to missing one. For instance, at an intersection, we label all areas that would be ROIs for turning left, going straight, and turning right. Figure 4.2 shows one sample of the MV-ROI dataset with circular annotations for every camera view.

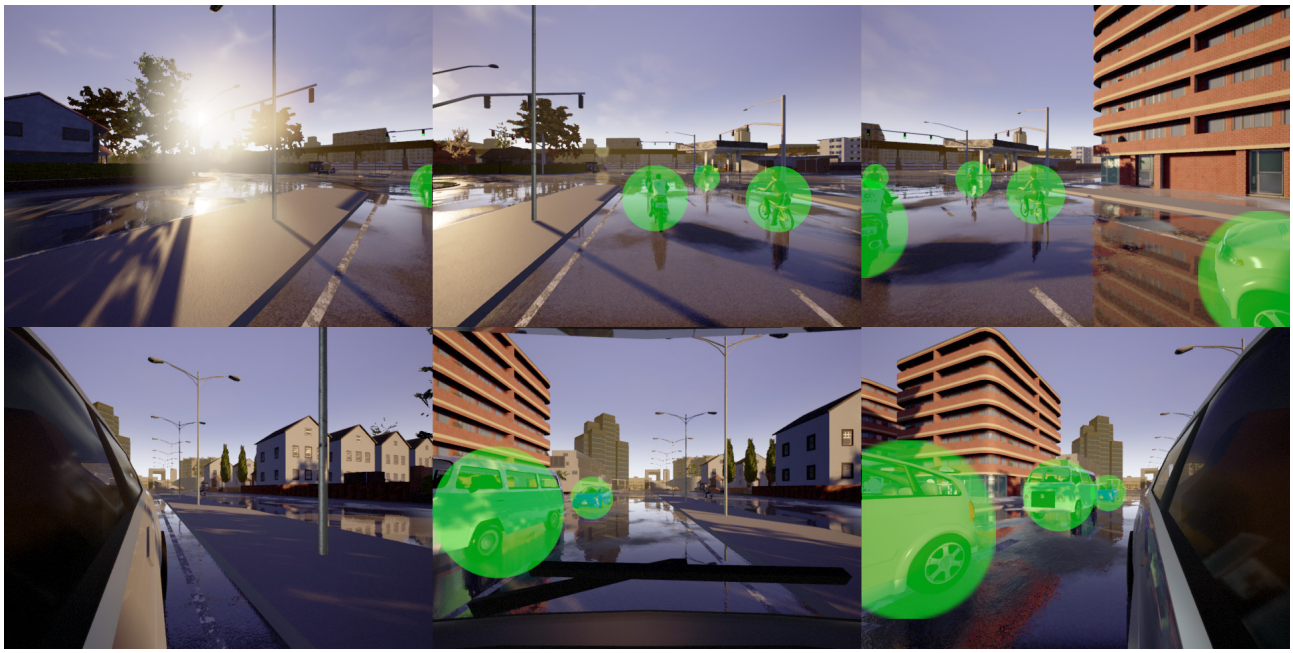


Figure 4.2 Exemplary sample of the MV-ROI dataset with manually labeled images for six camera views. The top row consists of the three front views: front left (FL), front (F), and front right (FR). The bottom row shows the rear facing camera views: rear left (RL), rear (R), and rear right (RR). Adopted from [20] © 2020 IEEE.

To further increase the amount of data and include various weather conditions in the dataset, we used a special feature of the CARLA simulator which enables us to replay the same state log recorded in the simulator for different weather conditions. These additional recordings have been labeled with the proposed semi-supervised annotation framework which will be introduced in Section 4.1.3. The contents of the MV-ROI dataset are summarized in Table 4.1.

The entire dataset consists of 20 driving sequences recorded at 10 Hz with a duration of around 10 s. Every driving sequence consists of video sequences from six camera views, resulting in a dataset of more than 17 000 images. Compared to the BDD-A dataset which contains more than 1200 videos with a total of 3.5 hours of driving, the size of the MV-ROI dataset is rather small, but sufficient for the finetuning approach discussed next.

Table 4.1 Summary of the multi-view region of interest (MV-ROI) dataset.

Views	6
Sequences	20
Sequence Length	~10 s
Frame Rate	10 Hz
Images	~17 000
ROI Annotations	~32 000

4.1.2 Multi-View Region-of-Interest Prediction

To extend state-of-the-art ROI prediction networks from single view to multi-view scenarios, we propose to finetune them with our manually labeled MV-ROI dataset introduced in the previous section. For this, we use the state-of-the-art BDD-A model and finetune it twice for front-facing and for rear-facing camera views to create two expert models. The BDD-A model predicts ROIs by generating a pixel-wise heatmap with probability scores, while we are only interested in low-complexity circular ROIs. We therefore design a postprocessing step to create such low-complexity labels from the heatmaps predicted by the finetuned BDD-A models.

4.1.2.1 Model Design

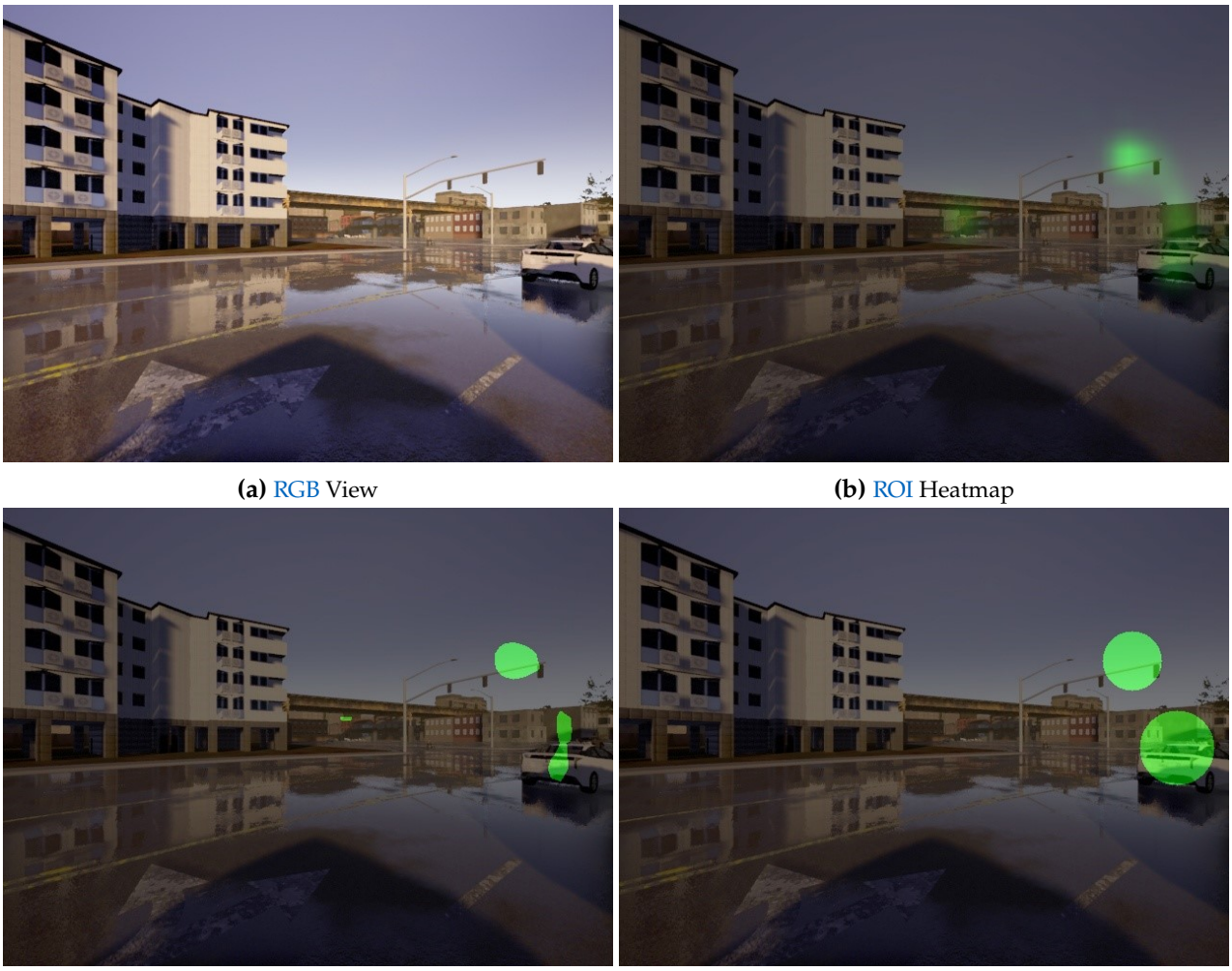
We use the BDD-A model [95], referred to as *BDD-A* from here on, as the baseline ROI prediction model. The authors generated the ROI ground-truth labels for training *BDD-A* automatically, using a combination of object detection and human gaze detection, which they averaged over multiple human viewers. Then, they trained a long short-term memory (LSTM) model to predict new ROIs. The training sequences are weighted towards more important scenarios such as turning to include a special focus on situations that are more complex and thus more relevant for driving. While the resulting *BDD-A* model allows for robust ROI predictions, it was only trained for a single front-facing camera. Here, we extend *BDD-A* to multi-view ROI prediction.

The basic prediction task for a single view and multiple views is similar, with only a few ROIs being specific to individual views. The rear views, for instance, do not contain ROIs for traffic lights since they are not relevant anymore. We therefore finetune the *BDD-A* model using our manually labeled MV-ROI dataset introduced in Section 4.1.1. For this, we continue training the pretrained *BDD-A* model until convergence as suggested in [95].

From the ten manually labeled sequences that were initially labeled for the MV-ROI dataset, nine are used for finetuning and one is used for testing. We refer to the model obtained from finetuning *BDD-A* with all training images from all perspectives as *ROI-single*. While *ROI-single* is finetuned with accurate ground-truth data, it does not specifically consider the different structures of ROIs in front- and rear-facing camera views. We therefore finetune *ROI-single* a second time, once using only front-facing images and once using only rear-facing images. The resulting two models are referred to as *ROI-multi-2* since they were finetuned twice. As an alternative approach to include a special focus on the different camera views while avoiding two training steps, we finetune only once with all data, but with different weightings. The front-facing model is shown each front-facing image twice as often as the rear-facing images during training. The rear-facing model is trained accordingly for the rear-facing images. We refer to the resulting two models as *ROI-multi-1*, since they were fintuned only once. Preliminary results showed that training more models such as an expert model for each of the six camera views does not further improve the performance. This can be explained by the high similarity of the three camera views facing in the same direction.

4.1.2.2 Postprocessing Label Generation

The *BDD-A* model as well as our finetuned models generate pixel-wise heatmaps with values ranging from 0 to 1, where 1 indicates the highest probability of an ROI. In our context, we are only interested in low-complexity circular ROIs which were used as labels in the MV-ROI dataset. Hence, we design a postprocessing step to convert the resulting heatmaps into low-complexity circular labels. Figure 4.3 shows the four processing steps to generate the circular labels.



(a) RGB View **(b)** ROI Heatmap

(c) Connected Components **(d)** Filtered ROI Circles

Figure 4.3 Pseudo label generation workflow. Adopted from [20] © 2020 IEEE.

Figure 4.3a shows a single RGB image, captured from the front-left camera. The image is fed into one of the finetuned models to predict an ROI heatmap such as shown in Figure 4.3b. Then, the resulting predictions are binarized to either 1 or 0 based on an empirically determined threshold. The binarized heatmap is used to calculate binary connected components, visualized in Figure 4.3c. Next, the connected components are filtered to remove small areas. Finally, the remaining components are encircled to create the final low-complexity labels shown in Figure 4.3d.

4.1.3 Semi-Supervised Data Annotation

To finetune the *BDD-A* model as discussed in the previous section, we used ten driving video sequences of the MV-ROI dataset that have been manually labeled. However, large-scale labeling by hand is undesirable, since it is slow and expensive. Instead, semi-supervised labeling can be used, a strategy also employed in other large-scale data sets such as BDD100k [175]. The idea is to use

automatic recommendations from a model and finetune the model proposals by hand. To increase the size of the MV-ROI dataset and benefit from the already finetuned models, we propose a semi-supervised label generation pipeline designed for ROIs in multiple camera views. Figure 4.4 shows the entire end-to-end workflow of the labeling pipeline and data generation process.

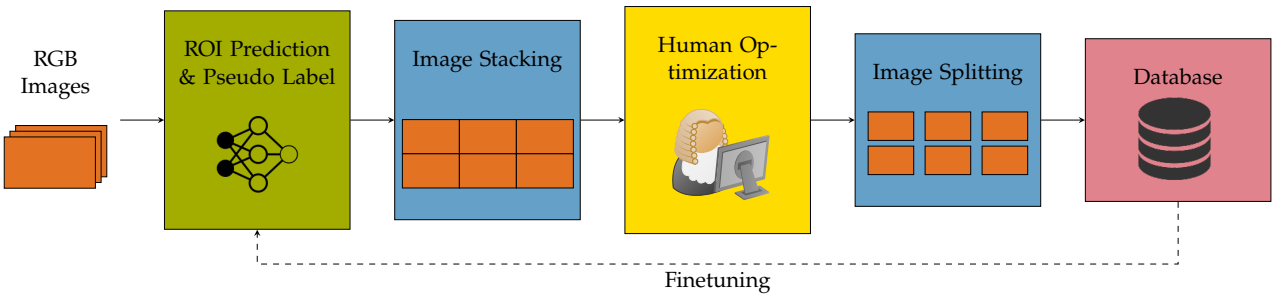


Figure 4.4 Semi-supervised data generation pipeline. Adopted from [20] © 2020 IEEE.

Unlabeled RGB images are fed into the pipeline and processed by the desired ROI prediction model for either front or rear view. The heatmap predicted by the model is converted into the circular labels as explained in Section 4.1.2.2. These circular labels are used as recommendations for the human inspector. The labels can be inspected and modified using the open source labeling tool labelme [176]. Since every sample of the dataset consists of six camera images, multiple views of the same scene have to be inspected. The sequential inspection of multiple views is slow and it might be difficult for humans to understand the context of the current driving situation from a single view. Hence, we suggest to combine all camera views of the same scene into a single frame as shown in the *Image Stacking* block to help the human inspector better understand the driving scene. After human inspection and correction, the image and the corresponding labels are split back into the separate camera views and stored. The new labeled data can be directly reused for further finetuning and improving the suggestions of the ROI prediction model.

Using this workflow, we extended the MV-ROI dataset for the initial ten driving sequences to the values listed in Table 4.1. The workflow can also be transferred to real world data, allowing to easily create data and finetune a custom ROI prediction model.

4.1.4 Results

In this section, we evaluate the different finetuning strategies and present the results. We compare the three ROI prediction models finetuned with nine driving sequences of the MV-ROI as explained in Section 4.1.2.1 with the *BDD-A* model as a baseline. For the baseline, we used the pretrained model of [95] without further modifications. Table 4.2 shows a summary of the models evaluated and the training procedures.

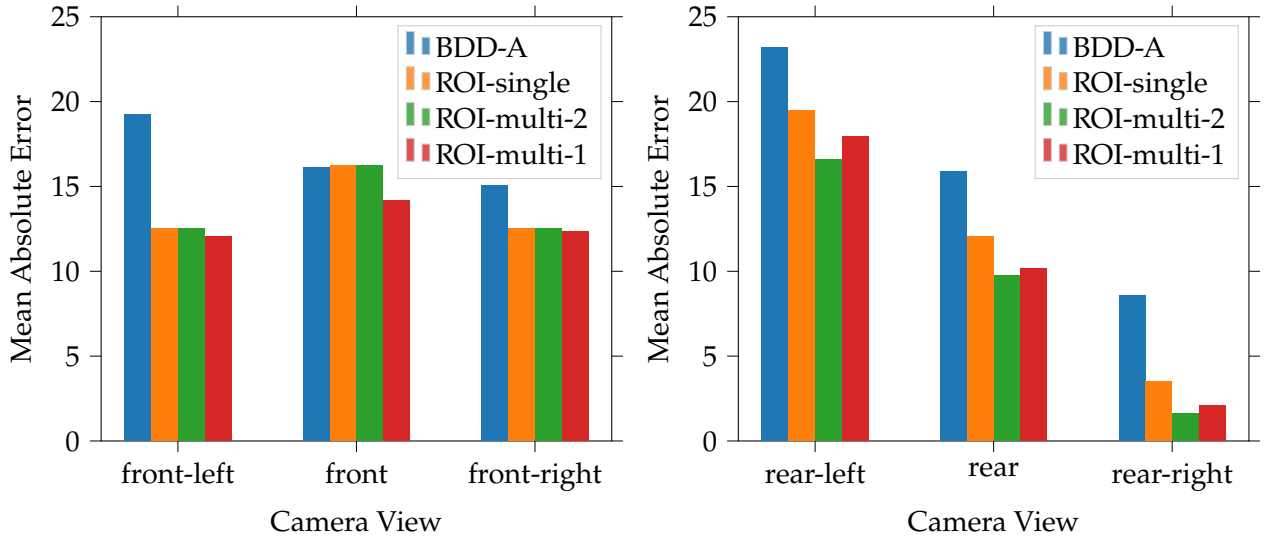
Table 4.2 Naming and training overview for the models evaluated.

Model	Training Process
BDD-A	Unchanged BDD-A model from [95]
ROI-single	*BDD-A* finetuned with manually labeled data
ROI-multi-2	*ROI-single* finetuned again with manually labeled data front or rear view
ROI-multi-1	*BDD-A* finetuned with manually labeled data and doubled weight on front or rear view

ROI-single is the baseline model finetuned once with all training images. To allow the ROI prediction to specialize on different view perspectives, we also implemented two approaches which use a model trained separately for either front or rear view. First, we finetuned the *ROI-single* a second time with

either front- or rear-view data, named *ROI-multi-2*. Finetuning the baseline model in a single step with doubled weight on either front or rear view is referred to as *ROI-multi-1*.

From the ten sequences of the initial dataset, we selected one sequence at random and reserved it for testing. This test sequence is used to create ROI predictions of all four models listed in Table 4.2. The mean absolute error (MAE) calculated from the models' predictions and the ground-truth labels for every camera view and model are shown in Figure 4.5.



(a) Front view evaluation

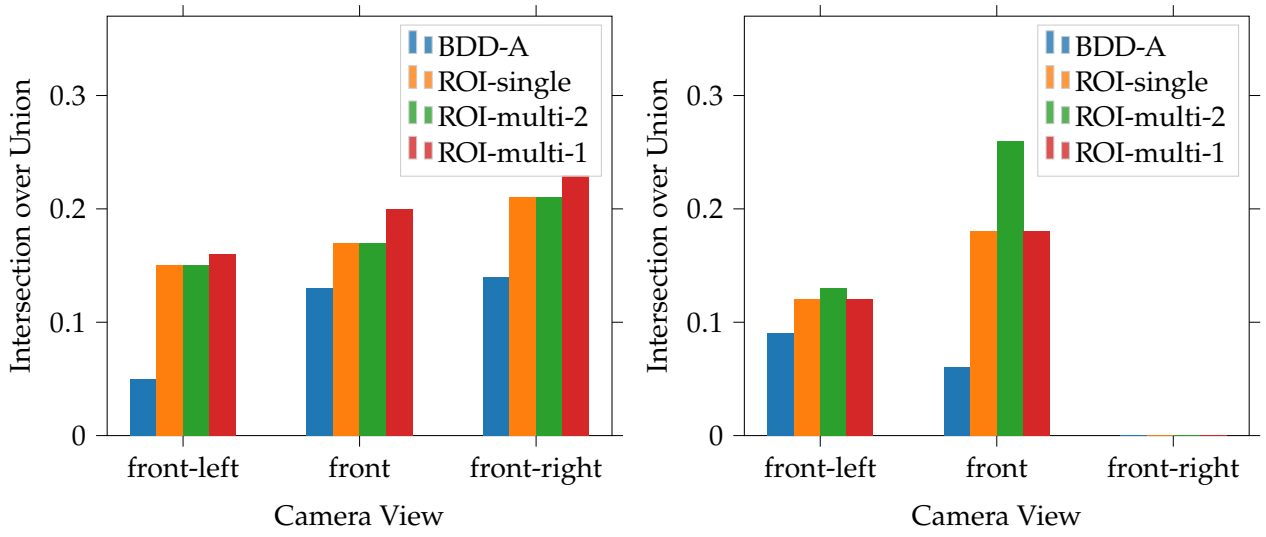
(b) Rear view evaluation

Figure 4.5 MAE evaluation results of the baseline *BDD-A* model and the proposed finetuned ROI prediction models. Adopted from [20] © 2020 IEEE.

The baseline model, which was only trained for front view data, already performs well for the front view testing data. The results highlight that the *BDD-A* model can barely be improved with finetuning using the relatively small finetuning data set. However, the remaining camera perspectives show an MAE reduced by 4.9 % after finetuning. Using two separate models for front and rear view further reduces the error compared to using a single model for all views. This is especially significant for the rear views, which show improvements by 1.2 % compared to the *ROI-single* model. These results suggests that there is a remarkable difference between front and rear view data. Traffic lights, for instance, are interesting regions for the front view, but not in the rear view. For the front views, the model of *ROI-multi-2* shows no further improvements compared to *ROI-single*. This can be explained by the overfitting introduced by further finetuning since the front-left and front-right view are closely related to the straight front view. Finetuning the baseline model in a single step such as for *ROI-multi-1* shows improvements of 11 % compared to *ROI-single*. However, for the rear-view data, *ROI-multi-1* performs worse than the two step finetuning approach *ROI-multi-2*. A possible explanation could be that a single finetuning step is not enough to let the model learn to distinguish between front and rear ROIs, which results in a overall error rate slightly higher than *ROI-multi-2*.

Besides the MAE, we evaluate the intersection over union (IoU) as an alternative metric to validate the results. Figure 4.6 shows the IoU calculated for every camera and model.

The results confirm the observations made for Figure 4.5, with only slight differences compared to the MAE. The *ROI-multi-1* shows the highest IoU for the front views, while the *ROI-multi-2* again reaches the best results for the review views. For the rear right view, the IoU is zero for all models, since the ROIs of the scenario used for testing are small and only appear for a short time. Hence, there is no intersection between the ROIs predicted and the ground truth. Nevertheless, the MAE calculated in Figure 4.5 still allows for comparing the performance of the individual models for the rear right view.



(a) Front view evaluation

(b) Rear view evaluation

Figure 4.6 IoU evaluation results of the baseline *BDD-A* model and the proposed finetuned ROI prediction models.

Table 4.3 Mean absolute error (MAE) and intersection over union (IoU) averaged over all six camera views for each evaluated model.

	BDD-A	*ROI-single*	*ROI-multi-2*	*ROI-multi-1*
MAE	16.3496	12.7229	11.5520	**11.4614**
IoU	0.0783	0.1383	**0.1533**	0.1483

Finally, the average MAE and IoU of all six camera views for every model are summarized in Table 4.3. The results suggest that in general, two separate models for front and rear view outperform a single one-fits-all model. The IoU values show the same general effect as the MAE. A single finetuning step improves the model, but using two separate models performs best. In conclusion, we thus recommend to include multi-view data into the training process and train two separate ROI prediction models for the front and rear view.

4.2 Real-Time Driver Monitoring And Situation Awareness Assessment

After having presented a method to predict the interesting regions the driver should focus on, we next propose a method that uses these predictions together with the actual gaze information of the driver to estimate the driver's SA. First, we introduce a concept for assessing the driver's SA in realtime, which is inspired by methods used in aviation. Then, we discuss the challenges when transferring an SA model designed for aviation into the driving context and propose an SA method for driving. Finally, we present a framework that uses the proposed system to assess the driver SA in realtime, which is used for evaluating the proposed approach in a user study.

4.2.1 Concept

To motivate the proposed concept, we first summarize how the topic of SA has been approached in the literature. Several studies have shown that higher levels of automation lead to a reduced driver attention [63, 64, 65]. A human driver who was previously not in control of the vehicle may need up to 40 seconds to obtain full control of the vehicle when asked to take over control [4]. Constantly

monitoring the driver is required for determining the driver's ability for a successful takeover and can be an additional safety feature by proactively guiding the humans' attention to important, yet overlooked areas [102].

Existing approaches assess head pose estimation as well as the driver's gaze as good indicators of driver attention [98, 99, 102]. However, head-pose-based systems are just a rough estimation of the driver attention and cannot distinguish between the perception of relevant and irrelevant objects or events inside the driver's field of view (FoV). The eye tracking-based system proposed in [102] only offers an acoustic warning in case the driver misses an object. Active driver observation and the assessment of their SA is still an open challenge due to the high variability of the input data [97].

Hooey et al. [103] proposed a concrete method to measure the SA of a pilot. The authors model the pilot's SA score as a ratio of the actual situation elements (SEs) the pilot has perceived and the optimal SEs the pilot should have perceived. This approach has been validated by Shuang et al. [104]. The authors compared the estimations of the pilot SA model with the two subjective measurement techniques SAGAT [68] and SART [76]. We transfer this idea of modeling the SA as the ratio of actual and optimal SEs from an aviation environment into the driving context to measure the SA of a human driver in realtime. Figure 4.7 visualizes the general concept of the proposed driver SA model.

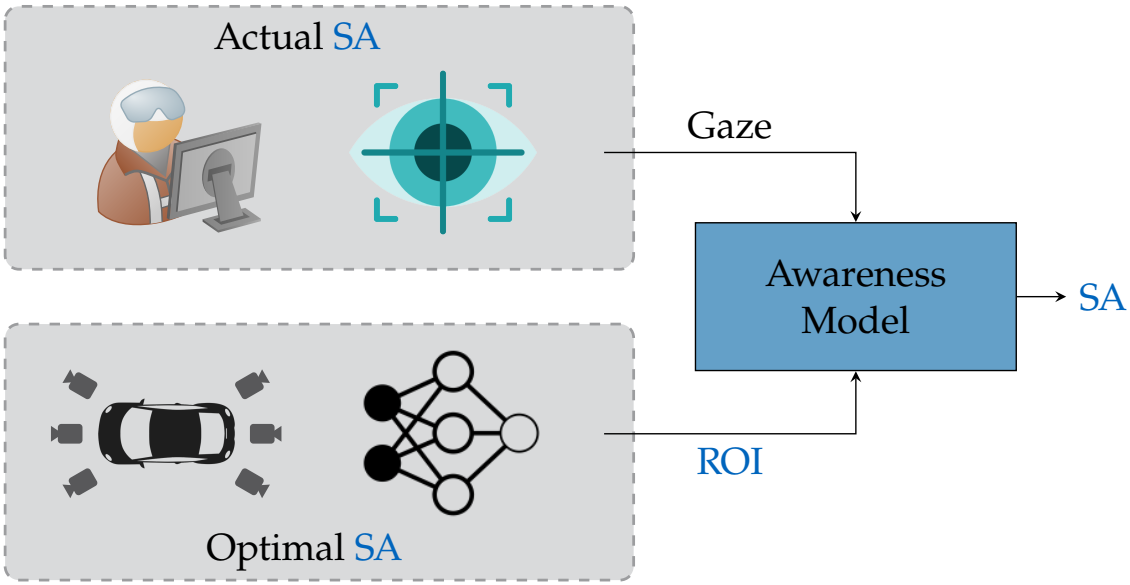


Figure 4.7 Concept of the proposed driver SA model. Region of interest (ROI) prediction defines the optimal awareness which is compared to the actual SA of the driver captured by an eye-tracking device.

We use the MV-ROI prediction introduced in Section 4.1 to estimate all SEs the driver should have perceived. Then, we use eye tracking to measure the actual SA of the human driver. Comparing the actual SA with the optimal SA allows to estimate the awareness of the driver based on the current traffic situation. The proposed metric can be used, for instance, to measure the takeover ability of a driver in a system-to-human transition or to constantly monitor the driver, both in-car and remote. Next, we introduce the pilot SA model presented in [103] in detail. Finally, we discuss the challenges of transferring this approach to the driving context and propose solutions.

4.2.2 Driver-Awareness Model Design

While the SA model proposed in this thesis is inspired by the model of [103], there are several challenges to solve when transferring the model from aviation to driving. To elaborate these challenges, we first discuss the SA modeling approach introduced in [103]. Then, we transfer the SA model to the driving context and propose our solutions for the challenges introduced by the new domain.

4.2.2.1 Situation Awareness in Aviation

SA assessment has long been a relevant topic in the field of aviation. While the task of aviation is inherently different to driving, results from this field can be used as inspiration for SA measurement in driving. Hooey et al. [103] define the SA score of a pilot at a time t_i as the ratio of actual SA and optimal SA:

$$SA_{ratio}(t_i) = \frac{SA_{actual}(t_i)}{SA_{optimal}(t_i)}. \tag{4.1}$$

The optimal SA defines the awareness as the situation where the operator perceives and comprehends all SEs. The authors define the optimal SA at a time t_i as the weighted sum of required and desired SEs:

$$SA_{optimal}(t_i) = \sum_{r=1}^{m} 2 \cdot SE_{opt,r} + \sum_{d=1}^{n} SE_{opt,d}. \tag{4.2}$$

In the context of aviation, such SEs are mainly cockpit instruments visualizing state information of the airplane. Usually, there are no SEs outside of the cockpit that can appear or disappear. This means all optimal SEs required for this approach have static positions and are known in advance.

The actual SA describes all SEs in the current situation that are actually perceived and comprehended by the pilot. This current awareness status of the operator is defined as the weighted sum of required and desired SEs, multiplied by their perception level:

$$SA_{actual}(t_i) = \sum_{r=1}^{m} 2 \cdot SE_{act,r} + \sum_{d=1}^{n} SE_{act,d} = \sum_{r=1}^{m} 2 \cdot p_{irt} + \sum_{d=1}^{n} p_{idt}. \tag{4.3}$$

The probability scores p_{idt} and p_{irt} describe the perception level of a desired SE or required SE, respectively. Both scores can be either 0 for an *undetected* SE, 0.5 for a *detected* SE, or 1 for a *comprehended* SE. The scores are assigned based on the pilot's gaze, which is measured with an eye-tracking device.

Transferring this approach from the aviation context into the driving context introduces several problems. In the aviation context, the SEs are inside the aircraft, static, and known in advance. The SEs in driving are outside of the vehicle and can be moving elements. They can appear and disappear and are not known in advance. Actual SEs can be detected with an eye-tracking device as suggested in [103]. Different to the static SE regions looked at by the pilot, the detected and comprehended regions outside of the vehicle need to be tracked to compensate their movement and ensure a correct mapping of their perception level. Furthermore, driving is a highly dynamic tasks with higher update rates compared to the aviation model. In [103], the authors also report a duration of 300 s until an SE drops into a fully undetected state. Since the environment in driving changes much more frequently, these durations need to be lower. Hence, driver distraction is another critical part which should be considered. We discuss how to address these challenges next.

4.2.2.2 Awareness in Driving

To cope with the issues caused by changing to a new domain, we propose the following solution. We use an ML-based approach to predict the frequently changing, optimal SEs. This can be achieved by using an ROI prediction network such as in [95] or [96]. Since teledriving usually requires multiple camera view, we use the MV-ROI prediction approach introduced in Section 4.1.

Measuring the SEs perceived by the driver can be done by evaluating the driver's gaze, similar to the aviation scenario. Additionally, the ROIs inspected by the driver need to be tracked over time to assign and update the correct perception level. Similar to [103], we assign every SE one of the following three perception levels using the same probability scores such as introduced in the previous section: 0 for *undetected*, 0.5 for *detected*, or 1 for *comprehended*. Figure 4.8 shows the general process and update policy for all different states of a single SE.

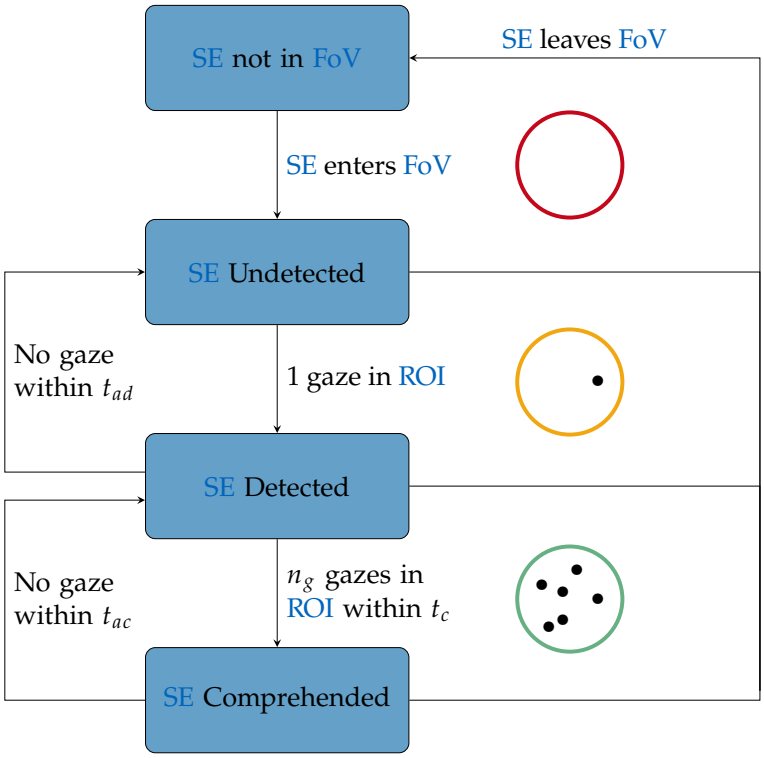


Figure 4.8 Workflow of the proposed driver SA model from the appearance to the comprehension of an SE. Adopted from [22] © 2020 IEEE.

An SE can either appear or enter the FoV. Then, the SE is recognized by the ROI prediction network and classified as *undetected*. As soon as the first gaze point falls into the ROI, this SE is classified as *detected*. The *detected* SE is classified as *comprehended* if there are more than n_g gaze samples within a certain comprehension time t_c within the ROI. A *comprehended* SE keeps its status *comprehended* for a comprehension alive time t_{ac}, which is refreshed every time a new gaze point hits the respective SE. If the SE is unseen until the alive time t_{ac} expires, the status of the SE changes from *comprehended* to *detected*. A *detected* SE has a detection alive time t_{ad}, which is refreshed as well if new gazes hit the SE region. Similarly, if the alive time t_{ad} expires, the SE falls back into the status *undetected*. If the SE leaves the FoV, the SE is removed from the measurement regardless of its state.

Since driving is a highly dynamic task within a frequently changing environment, both the comprehension alive time t_{ac} as well as the detection alive time t_{ad} have to be much lower than in the aviation context. In driving, even a short distraction can be a severe safety issue. This is a novel problem compared to the aviation context, where all SEs are inside the cockpit and have a maximum update time of 300 s [103]. To this end, we address distraction of the driver by introducing a penalty system. We define distraction as the accumulation of gaze points in an area not predicted as an ROI. We detect these accumulations by clustering all non-ROI gaze points using the density-based spatial clustering of applications with noise (DBSCAN) algorithm [177] with a maximum point distance d_{max} and a minimum number of gaze samples n_{min}. A minimum number of gaze samples required for clustering avoids punishments from single or few non-ROI gaze points. These points usually occur when the driver's gaze moves from one ROI to another caused by the sampling frequency of the eye-tracking device. To represent distraction in the SA model, we extend the calculation of the actual SA in Equation 4.3 with a punishment factor γ:

$$SA_{actual}(t_i) = \sum_{r=1}^{m} 2 \cdot p_{irt} + \sum_{d=1}^{n} p_{idt} - o \cdot \gamma, \tag{4.4}$$

with o as the number of non-ROI gaze clusters. Considering all these modifications, the SA score of the driver can be calculated from Equation 4.1, 4.2, and 4.4.

4.2.3 System Design

In this section, we propose a system design implementing the driver SA model introduced in the previous section. We implement the system proposed for driver awareness measurement using the Robot Operating System (ROS). This allows for a versatile usage in both vehicles and simulations as well as a seamless integration in the TELECARLA framework proposed in Chapter 3. Figure 4.9 highlights the architecture of the proposed system.

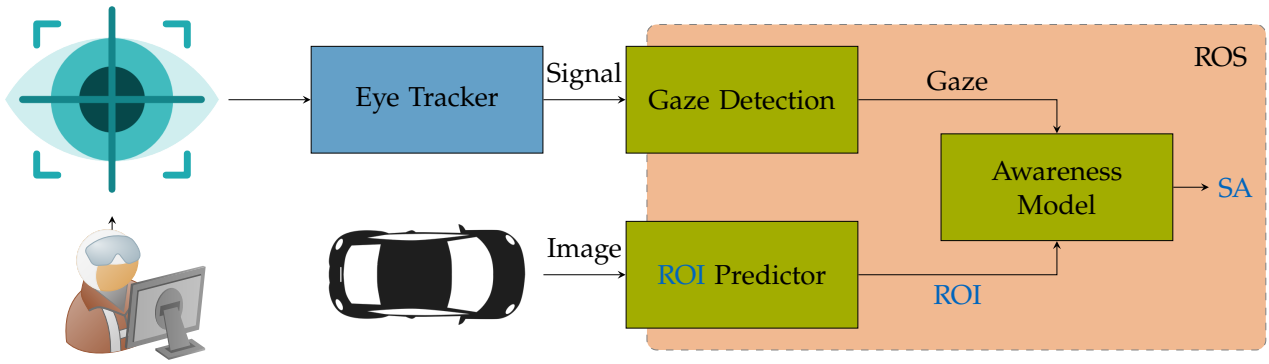


Figure 4.9 Architecture of the proposed driver SA approach. ROI prediction defines the optimal awareness which is compared to the driver's actual awareness captured by the eye tracking device.

An eye-tracking device captures the gaze information of the operator. The gaze information is processed and forwarded to the driver SA module. From the vehicle or simulation, the RGB images are used as input for the ROI prediction module. The ROI data define the optimal SA using the MV-ROI model introduced in Section 4.1. The frame rate of the RGB images defines the sampling frequency of the SA model. Since existing eye-tracking devices usually work at much higher frequencies, the gaze data are buffered until a new frame arrives.

On every new ROI sample, the SEs are tracked by comparing their Euclidean distance to the SE positions of the previous frame. With a distance value below a certain threshold, an SE is considered as known from the previous frame and the perception level of the SE in the previous frame is used. If the distance closest to an SE in the previous frame exceeds the threshold, the SE is considered as unknown and a new SE with the status *undetected* is created. After tracking and updating the optimal SEs, the buffered gaze data are evaluated using the currently existing SEs according to the process in Figure 4.8. All gaze points within an SE contribute to its perception level. Gaze points that do not fall into an SE are considered as distraction gaze samples. All distraction gaze points are clustered using the DBSCAN algorithm with a maximum point distance d_{max} and a minimum number of gaze samples n_{min}. The resulting number of clusters o multiplied with the punishment factor γ contributes to the distraction punishment as introduced in Equation 4.4. Figure 4.10 shows an exemplary driving scene recorded in the CARLA simulator with all types of SEs as well as a distraction gaze cluster.

The green circle shows a vehicle in front of the ego vehicle which is already *comprehended* by the operator. A certain number of gaze points within a short time fell onto this SE. The cyclist on the right marked with the yellow circle is *detected*, meaning only a few gaze points fell into this region. The traffic light highlighted with the red circle is so far *undetected*, indicating that the operator never looked at this traffic light. Finally, the blue dots on the advertisement representing the driver's gaze points cause a distraction punishment, since the advertisement is not relevant for driving.

When all three elements are required SEs, the final SA score in this situation would be less than 0.5. The current states for the SEs result in exactly 0.5, which are then reduced by the distraction

Figure 4.10 Visualization of the circular ROIs representing the optimal SEs and their perception level. The *undetected* SE is visualized in red, the *detected* SE in yellow, and the *comprehended* SE in green. The accumulation of gaze points in blue on a non-ROI area will be considered by the distraction penalty. Adopted from [22] © 2020 IEEE.

punishment depending on the punishment factor γ. Further implementation details of the system can be found in Appendix A.4. Next, we evaluate the proposed approach using this system in a user study.

4.2.4 Results

In this section, we design a user study to evaluate the proposed driver SA model and present the results. We first present the experimental setup and modalities of the user study. Then, we systematically analyze one exemplary scenario selected from the user study. Finally, we present the overall results for all scenarios evaluated.

4.2.4.1 Experimental Setup

We use the hardware setup of the proposed TELECARLA framework extended with the system designed in Section 4.2.3. Implementation details of the driver SA system integrated in the TELECARLA architecture can be found in Appendix A.4. We measure the driver's gaze using the *Pupil Core* wearable eye-tracking device [178].

In total, eight users participated in the user study. First, the eye-tracking device is calibrated for the respective participant. Then, we explain the general setup and tasks. Since all participants should see exactly the same driving scenarios to get reproducible results and to avoid biasing influences due to difficulties in controlling a vehicle in the simulation environment, we used scenario recordings rather

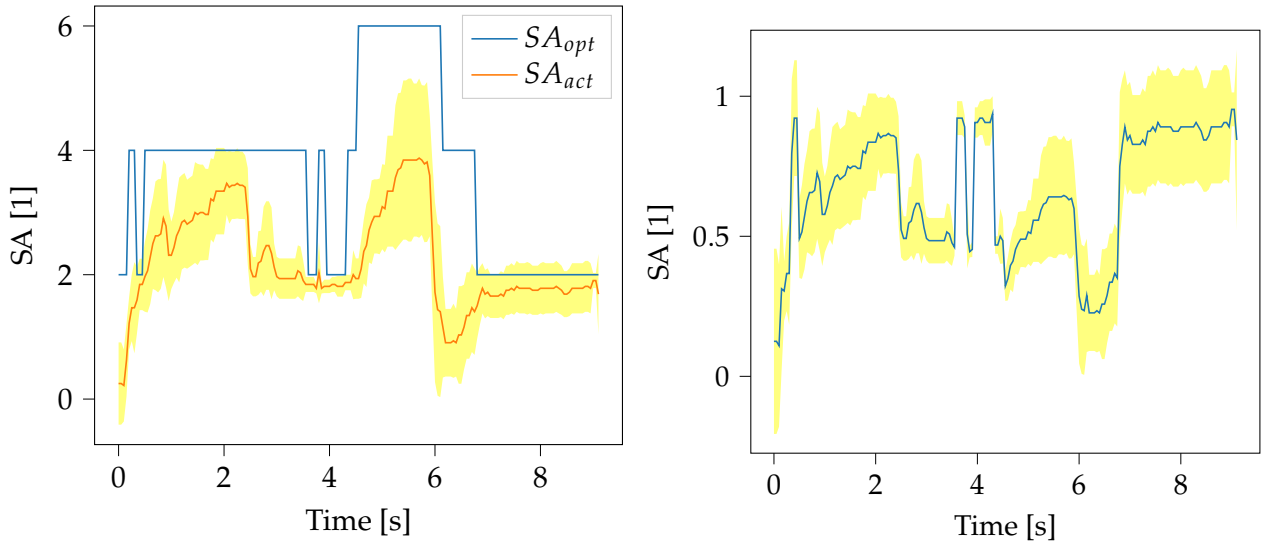
than active driving in the simulation. Monitoring the driving scene without active driving is also a typical scenario for in-car drivers as well as remote operators.

We showed all participants recordings of eight unique driving scenarios selected from the MV-ROI dataset introduced in Section 4.1.1. The ROI labels available in the dataset define the optimal SA. This guarantees exactly the same optimal SEs for all participants. Then, every participant monitored each scenario from the perspective of a single front view camera. On average, the driving scenarios have a duration of about ten seconds and cover a range of relevant driving situations.

In aviation, the optimal SEs such as the altitude meter or other cockpit instruments are static and well known in advance. This allows for a precise definition of questionnaires and comparison of the given answers. Driving as a highly dynamic task is more difficult to be evaluated with precise questions suitable for a questionnaire. Since the concept to measure a pilot's SA was already evaluated in [104], we do not compare the SA measurements of the user study with additional questionnaire-based SA assessment methods. Next, we first visualize and discuss the SA measured for all participants and a single scenario selected from the user study. Then, we compare the overall SA scores for all scenarios to validate the ROIs as a proper definition of the optimal SEs.

4.2.4.2 Single Scenario Analysis

First, we analyze the SA measured for the single scenario *turn right with three vehicles*. In this scenario, the ego vehicle approaches an intersection with three vehicles in front. After the traffic light switches to green, one leading vehicle as well as the ego vehicle turn right into another street. Since the scenario was selected from the MV-ROI dataset, the optimal SA is already defined by the ROI labels. The actual SA as well as the final SA score are measured and estimated using the eye-tracking based system presented in Section 4.2.3. Figure 4.11 visualizes the actual SA and optimal SA in Figure 4.11a as well as the overall SA score in Figure 4.11b averaged for all participants.



(a) Actual SA and optimal SA. Adopted from [22] © 2020 IEEE.

(b) Overall SA score.

Figure 4.11 Average SA for the selected driving scenario *turn right with three vehicles*. The actual SA follows the optimal SA shifted by a small delay of about 1 s. The overall SA score is the ratio of actual SA and optimal SA.

Within the first four seconds, two leading vehicles are visible. The optimal SA frequently changes in the first second as well as the fourth second, since there are occlusions between the vehicles in front when approaching the intersection. Most users comprehend both required SEs after at most two seconds. As soon as the ego vehicle reaches the intersection, the third vehicle becomes visible, resulting in the maximum optimal SA. With a small delay, the participants start perceiving the additional SE, which is visible until the vehicle turns right into the new street. Caused by the turn,

both optimal SA and actual SA suddenly drop, since there is only one SE in the new street after the turning maneuver. The actual SA drops even further as the driver first has to adapt to the new traffic situation. After the turning maneuver, the participants require about one second to detect and comprehend the single leading vehicle. As a result, the actual SA reaches the optimal SA score.

The shaded yellow area represents the standard deviation σ of the actual SA and the SA score. Since the optimal SA is based on ground-truth labels available in the MV-ROI dataset, σ is zero for all scenarios. In more complex or fast changing situations such as the turning maneuver at the intersection, the variance is higher due to the maximum number of vehicles visible in the scene and the individual drivers adapting differently to the new situation. At the end of the scenario, the variance is consistently small due to the lower complexity of a single leading vehicle. Figure 4.12 presents the SA score of the individual participants.

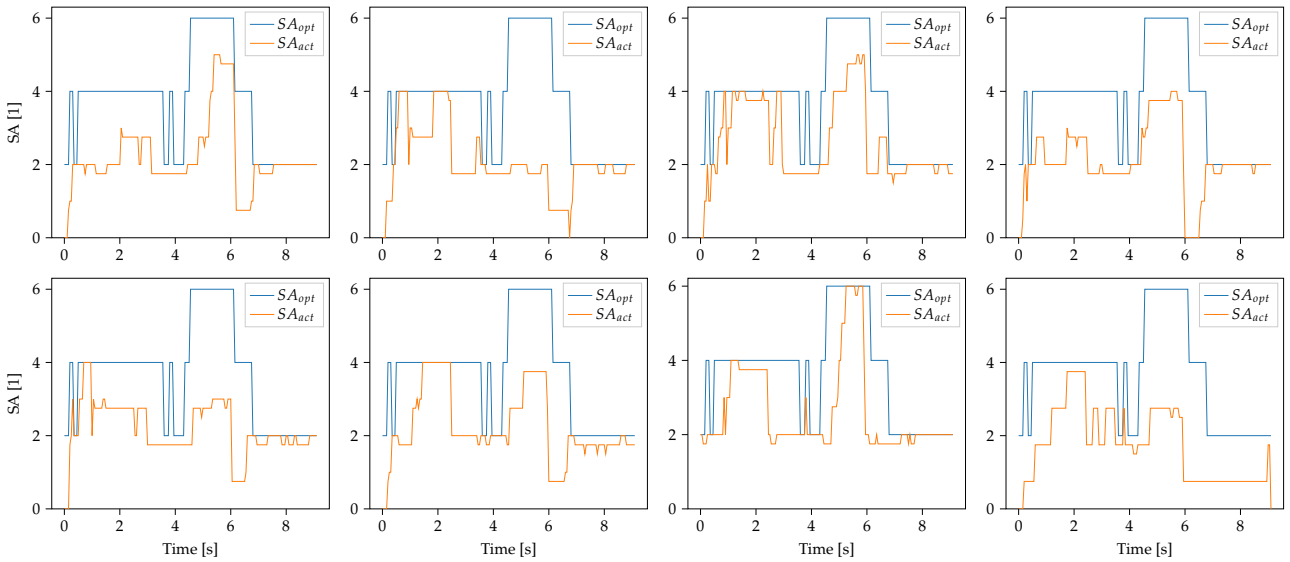


Figure 4.12 Actual SA and optimal SA of the individual participants for the selected driving scenario *turn right with three vehicles*.

When approaching the intersection, only two of the eight participants (1 and 4) detect both, but comprehend only one of the two vehicles visible in the scene. After the turning maneuver, only one participant (8) detects, but does not comprehend the single leading vehicle. Close to the intersection, three participants (2, 5, and 8) miss the third vehicle completely. The results of the user study demonstrate that users typically require around a second until a new SE is detected and comprehended. The SA measured by the proposed system is consistent with the changes in the environment and allows to assess the reaction speed and awareness of each individual user, demonstrating its potential for driver monitoring. Next, we present the results for all scenarios evaluated in the user study.

4.2.4.3 Overall Scenario Analysis

Eight scenarios including the *turn right with three vehicles* discussed in the previous section have been evaluated in the user study. The SA scores measured for all participants and driving scenarios are summarized in Table 4.4, ordered by their optimal SA.

Independently from the number of optimal SEs, the average SA score of all scenarios is almost identical. This suggests that the drivers were able to eventually fully comprehend all traffic situations, but need some time to react to sudden changes. As the definition of the optimal SA is not trivial in the context of driving, we analyzed the variance of the SA measurements. A small variance score suggests that all participants follow a similar behavior and try to focus on similar areas. We conclude that this supports our assumption that ROI prediction is a proper definition of the optimal SA.

Table 4.4 Average SA measurements and standard deviation for all evaluated scenarios.

Scenario	$\overline{SA}_{opt}$	$\overline{SA}_{act}$	$\overline{SA}_{ratio}$ ($\pm\sigma$)
Turn right, uphill	2.77	1.78	0.60 ±(0.30)
Turn right, three vehicles in front	3.63	2.21	0.66 ±(0.16)
Turn left, downhills	3.79	2.00	0.59 ±(0.26)
Straight, stop at intersection	4.16	2.07	0.52 ±(0.31)
Turn left, three vehicles in front	5.08	2.42	0.48 ±(0.23)
Straight, one vehicle in front	6.28	3.54	0.63 ±(0.14)
Straight, pedestrians passing by	9.81	4.50	0.51 ±(0.11)
Stopped, vehicles passing by	10.02	4.82	0.55 ±(0.17)

4.3 Summary

In this chapter, we introduced a novel metric to measure the SA of the driver both online and in realtime. For this, we used the driver's gaze as a source of information for assessing their SA as well as regions relevant for driving that are visible in the scene.

We first presented an approach to extend existing ROI prediction networks from single view to multiple views. A state-of-the-art ROI prediction network trained with a single front view camera is finetuned multiple times to create two expert networks specialized on either front views or rear views. For finetuning the model, we created the multi-view dataset MV-ROI, which is manually labeled with low-complexity ROI annotations. The ROIs can be used to define all relevant elements the driver should have perceived in the scene.

Next, we developed a system to measure the SA for both in-car drivers as well as remote operators. We defined the SA score as a ratio of actual SA and optimal SA inspired by related work in the field of aviation. The actual SA and optimal SA are defined as a weighted sum of SEs. We use eye tracking to measure the driver's actual SA. Then, the SA score is estimated in realtime by comparing the elements the driver actually focuses on with the elements the driver should have perceived. Unlike the aviation case with a static and pre-defined optimal SA, we determined all frequently changing optimal SEs using ROI prediction. Further, the driver SA model considers distraction caused by gazes on elements that are irrelevant for driving by using a distraction penalty system.

Evaluating the proposed concept in a user study for different traffic scenarios demonstrated an intuitive human behavior. The actual SA follows the optimal SA with a small delay and drops after maneuvers that change the scene. After a short time, the driver adapts to the new traffic situation and again reaches a higher level of SA. The results of all scenarios suggest that this behavior is independent of the number of SEs and that the definition of the optimal SEs based on ROIs was an appropriate decision.

Chapter 5

Traffic-Aware Multi-View Video Stream Adaptation

The TELECARLA framework presented in Chapter 3 enables teleoperated driving (ToD) in simulation, with the possibility to dynamically adapt multiple video streams. The driver situation awareness (SA) model introduced in Chapter 4 allows for measuring the awareness of the driver in realtime for the current traffic situation. Given the multiple camera views available in commercial vehicles and the limited transmission capacity of mobile networks, these multiple camera views have to be transmitted over the single network channel. While all video streams together have to match the currently available transmission rate, individual streams can be encoded differently based on their importance for the current traffic situation. The goal of an individual stream adaptation is to optimize the visual quality of the video streams as well as the SA of the operator.

In this chapter, we propose a traffic-aware multi-view adaptation (TAMVA) scheme that controls the individual video streams based on the current traffic situation. We first introduce the general concept of the TAMVA scheme and present an approach to estimate the importance of each camera view using the vehicle's state information. Then, we use a multi-dimensional adaptation (MDA) streaming module to select the optimal combination of temporal resolution (frame rate), spatial resolution (frame size), and target rate/quality for each camera view to control the encoder for the respective video stream. Additionally, we propose a dynamic region of interest (ROI) masking approach to improve the video quality of individual camera views by filtering less important regions in the respective camera view. Due to this filtering, the encoder is able to spend more bits for the important regions which results in a higher video quality of the important regions.

The TAMVA scheme presented in this chapter has been published in [23]. The code is available as open source on GitHub[1] and integrated in the basic teledriving framework TELECARLA, which has been presented in Chapter 3. Implementation details of the respective TELECARLA module are given in Appendix A.

5.1 Concept

The current traffic situation determines which parts of the driving scene and hence which camera views are most relevant for the operator. To use the available transmission rate in the best way and to support the operator in understanding the remote environment, the camera views most relevant for the current driving situation should be transmitted with a higher quality than less important camera views. In particular in situations where the available network transmission rate is low, a prioritization of important camera views can still provide the operator with a sufficient quality for the most important camera views to control the vehicle safely in traffic [18, 17].

Existing approaches in the field of ToD mainly focus on the adaptation of a single camera view [62] or try to achieve an equally distributed quality for all camera views [116]. To adapt the video streams of multiple camera views individually based on their importance for the current traffic situation, we propose a traffic-aware multi-view adaptation (TAMVA) scheme. Figure 5.1 shows a general overview of the proposed TAMVA scheme.

[1] https://github.com/hofbi/tamva

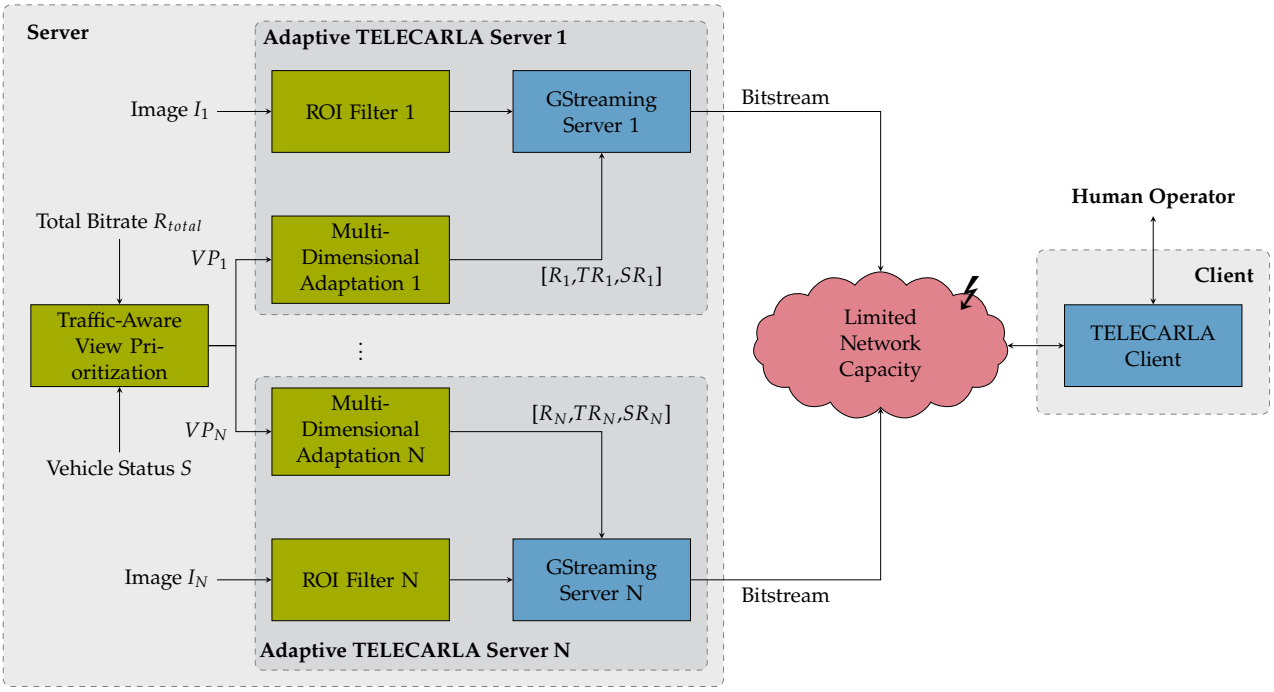


Figure 5.1 Overview of the proposed traffic-aware multi-view adaptation (TAMVA) scheme. First, the view priority $VP_1, \ldots, VP_N$ and the resulting bit budget $R_1, \ldots, R_N$ for each camera view $C_1, \ldots, C_N$ are estimated based on the total available transmission rate R_{total} and the vehicle's status information S. Then, the optimal parameters R_i, TR_i, SR_i, with $i \in \{1, \ldots, N\}$ are estimated to control the rate/quality and the spatio-temporal resolution of the video stream.

We estimate the view priority VP_i of each camera view C_i using the orientation of the respective camera γ_i as well as the vehicle's state information S. The state information S includes the steering angle α, the velocity v, and the gear position of the vehicle. Considering the total available transmission rate R_{total} and the priority of the individual camera view VP_i, we assign each camera view C_i a certain bit budget R_i. Then, an MDA streaming module selects the optimal combination of temporal resolution TR_i (frame rate), spatial resolution SR_i (frame size), and target rate R_i for each camera view C_i to control the encoder of the respective video stream. Lastly, we use the adaptation interface of the TELECARLA framework introduced in Chapter 3 for the dynamic rate control and stream adaptation of the individual video streams. Next, we present traffic-aware view prioritization in detail.

5.2 Traffic-Aware View Prioritization

The importance of individual camera views for the operator depends directly on the current driving situation. Given is a vehicle equipped with N cameras. The traffic-aware view prioritization system proposed in this thesis estimates the relative view priority VP_i in percent for each camera view C_i, with $i \in 1, \ldots, N$. Then, the bit budget R_i available for each camera view C_i is defined based on the total available transmission rate R_{total}:

$$R_i = VP_i \cdot R_{total}. \tag{5.1}$$

The total available transmission rate R_{total} can be estimated by approaches such as [179]. The bit budget R_i estimated for each camera view C_i specifies the target bitrate for the adaptive streaming module, as outlined in Figure 5.1.

The view priority VP_i is estimated based on the relative orientation γ_i of the respective camera C_i and the vehicle's state information S. The state information S of the vehicle includes the steering

angle α, the velocity v, and the gear position of the vehicle. Figure 5.2 shows a vehicle setup with $N = 6$ camera views and camera yaw angles γ_i.

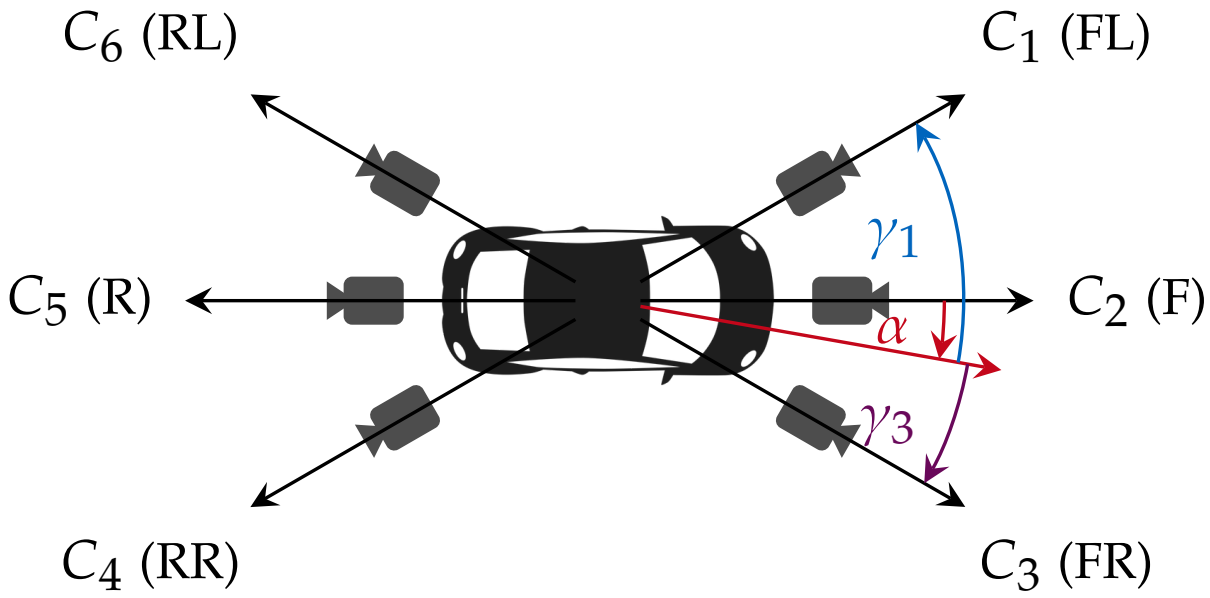


Figure 5.2 Definition of the camera yaw angles γ_i for a camera setup with $N = 6$ cameras. Adopted from [23] © 2022 IEEE.

We define γ_i as the yaw angle of a camera view C_i relative to the vehicle's steering angle α. Then, we assign every camera view C_i a specific view priority score VP_i based on γ_i, for $\gamma \in (-\pi, \pi]$:

$$VP_i = \begin{cases} 4 + \max(0, c \cdot v) & \text{for } |\gamma_i| < \frac{\pi}{4} \\ 2 & \text{for } |\gamma_i| < \frac{\pi}{2} \\ 1 & \text{else,} \end{cases} \tag{5.2}$$

with the speed constant $c = 0.1\,\mathrm{s/m}$ and the velocity v. Finally, we normalize all view priorities VP_i by the sum of all view priority scores $\sum_{i=1}^{N} VP_i$.

The definition in Equation 5.2 results in a higher visual quality for camera views pointing towards the current trajectory of the vehicle. Since the driver focuses more towards the vanishing point for higher speeds, we additionally weight camera views pointing towards the current driving direction with a velocity-dependent offset $c \cdot v$ compared to the other camera views [52]. To restrict the velocity-dependent offset to forward driving only, we set this offset to zero for negative velocities. For reverse driving, the average driving speeds are usually slower compared to forward driving and a video quality distributed for a broader field of view (FoV) is preferable. To favor the rear-facing camera views for reverse driving, we consider the gear position that inverts the prioritization for reverse driving with the velocity-dependent offset set to zero.

The specific priority scores of Equation 5.2 have been empirically selected for the ToD use case and the evaluation in this thesis. For other use cases, these scores can be tuned to match the requirements of the respective application.

5.3 Multi-Dimensional Stream Adaptation

The bit budget R_i estimated for each camera view C_i based on the view priority VP_i defines how an individual encoder processes the video stream of the respective camera view C_i. R_i either specifies the target bitrate for the respective video encoder directly or is used to determine the target quality.

The target quality can be controlled with a quantization parameter (QP) when using a user-defined rate control algorithm that estimates the QP based on the available bit budget R_i. In this chapter, we directly specify the target bitrate R_i for the video encoder.

In video coding, the spatial and temporal resolution can be used in addition to a target bitrate or target quality to control the video stream. MDA schemes such as proposed in [145] consider the spatial and temporal resolution of the video stream to further optimize the perceptual video quality of the video. For this, the MDA scheme selects the best possible combination of spatio-temporal resolution (frame size and frame rate) for the encoder to maximize the resulting video quality. In [153], the authors proposed the multi-dimensional video quality metric (MDVQM) to estimate the perceptual video quality. MDVQM was specifically designed to consider the influence of the spatio-temporal resolution on the video quality. In [145], the authors use their MDVQM in a data-driven approach as part of their MDA scheme to optimize the video quality. We integrate this data-driven MDA scheme in the proposed TAMVA scheme as outlined in Figure 5.1. The parameter set estimated by the MDA scheme that consists of the target bitrate R_i, the spatial resolution SR_i, and the temporal resolution TR_i, controls the encoder for each camera view C_i individually.

5.4 Adaptive Region-of-Interest Masking

In ToD, the transmission rate available in mobile networks can vary drastically depending on the location and time [14]. In case of a low transmission rate, the rate and quality of the respective video streams has to be adapted to the available transmission resources to ensure a reliable low delay video transmission and avoid stalling of the video stream. However, a low target bitrate specified for the encoder results in a low visual quality of the resulting video stream. Such a low visual quality can be a critical issue for the operator responsible for controlling the vehicle safely in traffic. While the visual quality of the image is reduced for all regions of the image in situations with a low transmission rate, not all parts in the image are of similar importance for the operator.

Furman et al. [180] suggested to apply a static ROI mask to the camera frame to cut off less important areas of the frame. This ROI masking approach reduces the image content that needs to be encoded. Following this idea, we apply such a masking approach in a preprocessing step by setting all pixels outside of the mask to zero. The pixels inside the mask are not affected by the preprocessing.

Encoding the frames preprocessed this way increases the image quality of the unaffected image parts when the target bitrate remains unchanged. The reason for this behavior can be expained by the fact that the encoder treats all zero-regions created in the preprocessing step with skip-mode, since they do not change compared to the previous frame. Three exemplary ROI masks applied to three front-facing camera views are visualized in Figure 5.3.

Figure 5.3 Three camera views with ROI masks. Left: A left-facing camera view. Center: A front-facing camera view. Right: A right-facing camera view.

While irrelevant regions such as the sky are removed by the ROI mask, the relevant parts of the frame such as the road as well as other traffic participants are clearly visible. Similar to the view priority, the regions of the frame that are important for the driver change depending on the driving situation. When driving straight with a higher speed, the driver focuses towards the vanishing point [52]. For turning maneuvers or when driving at lower speeds such as in urban environments,

the FoV required for the human driver increases. To this end, we extend the idea of [180] from a static mask that is identical in all situations to a dynamic mask that changes its shape based on the vehicle's steering angle.

The static mask such as proposed in [180] as well as the proposed dynamic ROI mask both have an elliptical shape. A video encoder processing the frame first splits the respective frame into blocks. If preprocessed with the ROI mask, this will result in blocks that are partially masked along the curve of the ellipse. To improve the masking process optimized for the block-based processing of the video encoder, we design a block-based masking mode. Figure 5.4 visualizes the proposed block-based mask for an exemplary front view.

Figure 5.4 ROI filtered front view using the block-based masking mode.

In the block-based masking mode, the elliptical curve of the ROI mask is approximated with blocks. The size of these blocks matches the internal block size of the video encoder. Applying such a block-based mask ensures that if the encoder processes the frame block-wise, a single block is either completely set to zero or completely unaffected by the ROI filter. This enhancement further reduces the bitrate required for encoding the video stream if filtered with the block-based ROI mask.

5.5 User Study Design

The TAMVA scheme proposed in this chapter allows to control the individual video streams based on their importance for the current traffic situation. To evaluate the concepts proposed with the TAMVA scheme, we integrate all modules implementing these concepts into the TELECARLA framework introduced in Chapter 3. Implementation details for the individual modules of the TAMVA scheme are available in Appendix A.3. Using this implementation integrated in the TELECARLA framework, we present the design of our user study conducted to evaluate the proposed TAMVA system next.

We evaluate all driving scenarios using the TELECARLA module for automatic scenario evaluation. For this, we create 14 driving scenarios using the OpenDrive standard[2], which is supported by CARLA and TELECARLA. All driving scenarios designed for this user study represent typical situations of

[2] https://www.opendrive.com/

ToD with a focus on urban driving [56, 9]. These scenarios include collision avoidance tasks as well as turning maneuvers with pedestrians or cyclists crossing the streets. Additionally, the scenarios include different weather conditions varying from sunny conditions in daylight to rainy conditions at night.

For the evaluation, we define two camera view prioritization modes. Both modes rely on the same six camera setup as outlined in Figure 5.2. In the *Uniform* mode, the available transmission rate is distributed uniformly over all six camera views. Each camera view hence gets the same bit budget for encoding the video stream. This bit budget specified for each view directly defines the target bitrate for the respective encoder. The *Prioritization* mode uses the proposed modules and strategies as outlined in Figure 5.1. The traffic-aware view prioritization module defines the bit budget R_i for each camera view C_i based on the total available transmission rate R_{total}. Then, the MDA scheme controls the encoder for each camera view C_i by specifying the frame size SR_i, the frame rate TR_i, and the target bitrate R_i. Additionally, the ROI filter is applied on the three rear-facing camera views (RL), (R), and (RR).

For a second experiment, we record three representative driving scenarios in the CARLA simulator. The users watch these recordings and rate the importance of each camera view for the current driving scenario. The driving scenarios have been recorded using the same six camera setup $C_1, \ldots, C_6$ with a resolution of 1920×1080 for all cameras. The three scenarios recorded for the user study consist of *left-turn*, *right-turn*, and *straight-driving*.

In total, ten users participated in the user study. To become familiar with the simulation environment as well as the general setup, all participants started with three minutes of free driving. Then, the 14 driving scenarios were evaluated in random order. For each of the scenarios, we randomly selected either the *Uniform* mode or the *Prioritization* mode for the evaluation. The overall transmission rate used for all driving scenarios was specified as 3000 kbit/s. Since a low transmission rate is the main use-case of the system proposed in this chapter, we selected the transmission rate in a way that it causes perceivable degradations in the video quality. During driving in the simulator, the scenario evaluation framework automatically tracks the user's performance. The performance indicators recorded by the framework include the scenario duration and the number of failures.

After completing the active driving tasks, we conducted the seconded experiment. The three driving scenarios recorded in the simulator were shown to each of the ten participants. The participants were asked to rate the importance of each camera view for the current driving situation on a scale of 1 to 10. A rating of 10 represents the highest priority for the current driving situation. Every participant rated each of the three scenarios that had been recorded with six camera views. In total, every participant therefore rates 18 camera views.

5.6 Results

In this section, we present results of the experiments conducted to validate the individual modules as well as the overall TAMVA system proposed. We first discuss the results of the user study. Then, we analyze the correlation of the proposed view prioritization approach with the user ratings. Lastly, we investigate the rate-quality gains for the ROI filter and the overall system.

5.6.1 Driving Performance

To analyze the driving performance, the scenario evaluation framework measured the scenario duration, the failure rate, and the number of missed red lights for each participant who participated in the user study. The performance indicators measured for both modes, *Uniform* and *Prioritization*, are summarized in Table 5.1.

The failure rate describes the average number of failures per scenario. The failures determining the failure rate are automatically registered by the scenario evaluation framework if a violation of the traffic rules is detected. The red light rate is the average number of missed red lights per scenario.

Table 5.1 Average performance of the participants completing the driving scenarios.

Mode	Duration ($\pm\sigma$)	Failure Rate ($\pm\sigma$)	Red Light Rate ($\pm\sigma$)
Uniform	**88 s (±36 s)**	**0.27 (±0.10)**	0.62 (±0.62)
Prioritization	97 s (±33 s)	0.30 (±0.10)	**0.55 (±0.65)**

The results suggest that neither mode consistently outperforms the other. While the *Prioritization* mode achieves a lower rate of missed red lights, the *Uniform* mode reaches a slightly lower scenario duration and failure rate. However, the results for all three performance indicators differ only by 10 % to 11.5 %. These small differences are likely caused by the variations in human performance, and not by the influence of the adaption. Furthermore, the time of the red light also influences the scenario duration.

A possible explanation for the comparable results of both modes is that the human drivers are still able to recognize the relevant features, even with a very poor image quality. This ability of the human drivers to cope with bad visual conditions was also observed in the longtime study conducted by Georg et al. [56]. Similar as in [56], we observe a slightly better perception of objects such as traffic lights for the *Prioritization* mode compared to the *Uniform* mode.

5.6.2 Traffic-Aware View Prioritization

After the participants had actively driven in the simulator for the first experiment, we conducted the second experiment as a monitoring and rating task. The participants had to observe video recordings of the three driving scenarios *left-turn*, *right-turn*, and *straight-driving*. The three scenarios have been recorded before the user study. Then, each participant had to rate the priority of the individual camera views for the traffic situation of the respective driving scenario. Table 5.2 shows the average priority scores and their standard deviation on a scale of 1 to 10, with 10 being the highest priority.

Table 5.2 Average priority scores of the individual cameras for the three representative driving scenarios rated by the participants of the user study.

Scenario	F ($\pm\sigma$)	FL ($\pm\sigma$)	FR ($\pm\sigma$)	R ($\pm\sigma$)	RL ($\pm\sigma$)	RR ($\pm\sigma$)
left-turn	9.25 ± (0.89)	9.38 ± (0.74)	6.13 ± (1.64)	4.75 ± (2.49)	5.38 ± (1.92)	4.88 ± (2.64)
right-turn	9.25 ± (0.89)	5.75 ± (1.49)	9.5 ± (0.76)	4.75 ± (2.49)	4.63 ± (2.45)	5.38 ± (2.13)
straight-driving	10.0 ± (0.00)	6.88 ± (1.36)	7.50 ± (1.51)	5.00 ± (2.45)	4.88 ± (1.81)	4.38 ± (1.92)

The front view (F) receives the highest scores of at least 9.25 for all scenarios. The participants rate the front-lateral views (FL) and (FR) only with a high priority score for the *left-turn* and *right-turn*, respectively. The front-lateral views that do not face in the direction of the turn receive a score of 5.9 on average. For the rear views, the behavior is similar as for the front views. However, the absolute priority scores are lower and the difference in the priority ratings is smaller.

To evaluate the proposed traffic-aware view prioritization model, we use the Spearman's rank correlation (ρ) and the Pearson correlation (PC). We calculate both correlations between the priority score estimated by the proposed model and the user rating listed in Table 5.2. Table 5.3 presents the resulting correlation scores.

We observe a high correlation for all scenarios. This high correlation indicates that the prioritization model successfully mimics the importance ratings given by the human drivers. For straight driving, the correlation is lower than for the turning maneuvers. A possible explanation is that the straight driving video was recorded at constant speed only. Considering multiple straight driving scenarios

Table 5.3 Correlation of the user ratings and the priority scores estimated by the proposed traffic-aware view prioritization module.

Scenario	left-turn	right-turn	straight-driving
ρ	0.93	0.93	0.83
PC	0.99	0.99	0.83

at different speed levels could increase the correlation or provide further information for finetuning the speed factor of the prioritization model.

5.6.3 Video Rate-Quality Assessment

After analyzing the driving performance of the participants and the correlation of the view priority ratings, we evaluate the resulting video quality and bitrate savings of the proposed methods. First, we evaluate the ROI masking approach for individual camera views. Then, we analyze the rate-quality savings of the overall TAMVA system.

All video sequences of the three representative driving scenarios are encoded using the x264 [134] software video encoder. Encoding the video sequences of three driving scenarios with the six camera views $C_1, \ldots, C_6$ results in 18 encoded video sequences. The video quality of the compressed video sequences compared to the uncompressed reference sequences is measured using the peak signal-to-noise ratio (PSNR) and the state-of-the-art video quality metric Video Multi-Method Assessment Fusion (VMAF) [157].

5.6.3.1 Region-of-Interest Masking

To evaluate the ROI masking approach, we apply the continuous and the block-based ROI filter on each of the 18 video sequences created for testing. In the following, we refer to the two different ROI filtering modes as *Continuous* and *Block-based*. *Baseline* indicates the reference mode without any ROI filter applied.

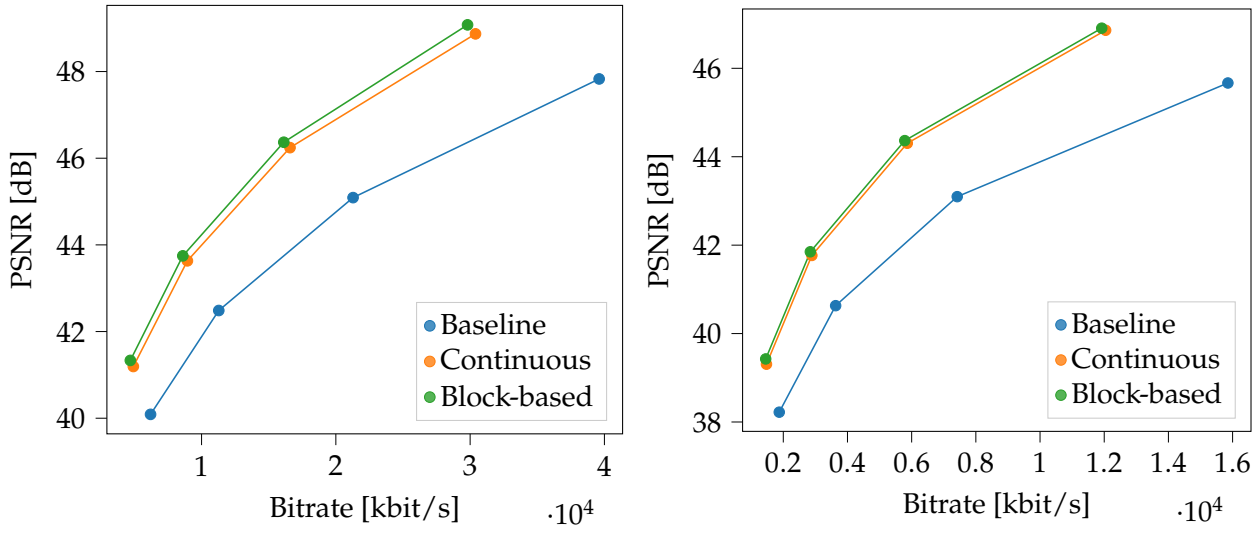
The 18 video sequences are encoded with four different quality levels and two group of pictures (GoP) lengths for the three modes. Following the Joint Video Experts Team (JVET) common testing conditions (CTC) [181], we select the four QPs to $q_p \in \{22, 27, 32, 37\}$ to define the quality levels of the encoded video sequences. For the GoP lengths, we select 1 as an I-frame only mode and 20 for an IPPP GoP structure. The IPPP GoP structure with a GoP length of 20 is a typical configuration which we also used in Chapter 3.

We measure the bitrate of the encoded video sequences as well as the video quality using the PSNR and VMAF. The bitrate and quality scores measured for the four QPs describe the characteristic rate-distortion (RD) curves of the two ROI filter modes and the *Baseline*. Figure 5.5 shows the resulting RD curves for all three modes and both GoP lengths using the PSNR as a quality metric to measure the distortion.

For both GoP lengths, the ROI filter modes outperform the *Baseline* for all QPs. The *Block-based* mode achieves additional quality gains compared to the *Continuous* mode. Figure 5.6 shows RD curves similar to Figure 5.5, but uses the VMAF score as quality metric to measure the distortion.

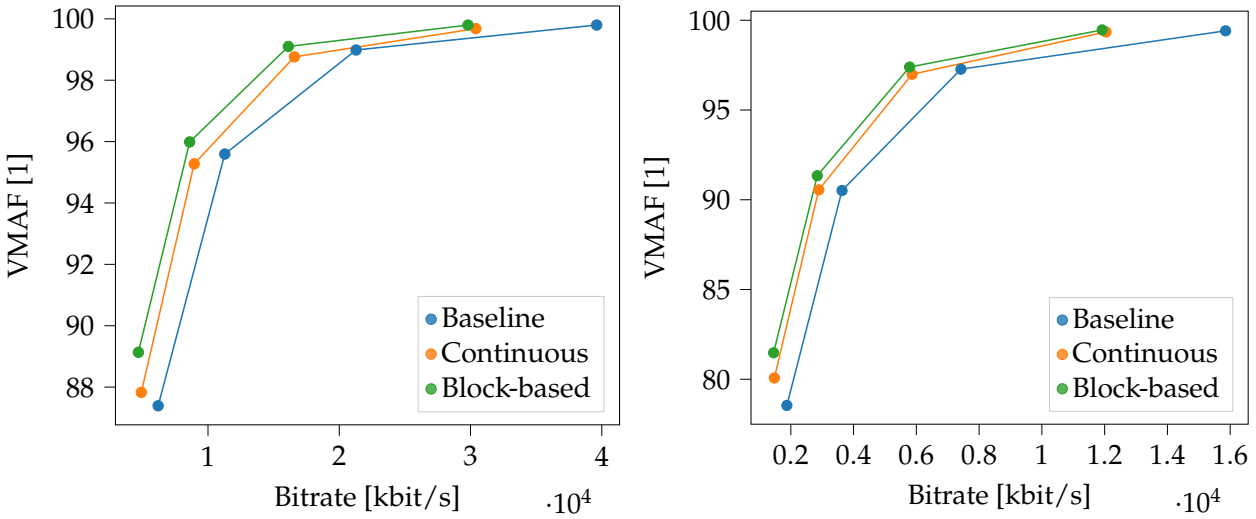
We further compare the RD curves of both ROI filter modes with *Baseline* using the Bjøntegaard Delta Rate (BDR) [182]. Table 5.4 shows the resulting BDR of the *Continuous* and the *Block-based* mode for the two GoP lengths and quality metrics.

The BDR in Table 5.4 shows the bitrate savings of the *Continuous* and the *Block-based* mode compared to the *Baseline* mode in percent. The proposed ROI filter achieves noticeable bitrate savings of at least 19.71 % for VMAF and a GoP length of 1. For a GoP length of 20, the bitrate savings increase to 23.81 %, since the encoder's inter-frame prediction benefits from the image regions filtered by the ROI



(a) GoP length 1 **(b)** GoP length 20

Figure 5.5 RD curves for *Baseline* and the two ROI filter modes with the distortion measured using the PSNR.



(a) GoP length 1 **(b)** GoP length 20

Figure 5.6 RD curves for *Baseline* and the two ROI filter modes with the distortion measured using VMAF.

Table 5.4 Bjøntegaard Delta Rate (BDR) [%] using the corresponding quality metric VMAF or PSNR and GoP-length of 1 or 20. The proposed ROI filter was used either in the *Continuous* or the *Block-based* mode.

Mode	VMAF-1	PSNR-1	VMAF-20	PSNR-20
Continuous	−19.71	−33.55	−23.81	−37.95
Block-based	**−28.48**	**−37.24**	**−28.84**	**−39.91**

mask. Using the *Block-based* mode further improves the rate savings by at least 3.69 percentage points. These improvements are due to the filter mask optimized for the encoder's block size, which avoids blocks that are only partially filtered.

5.6.3.2 Overall System Performance

Finally, we evaluate the quality gains of the overall system when using the *Prioritization* mode compared to using the *Uniform* mode. We first specify a total available transmission rate of 6000 kbit/s for both modes. We again selected the transmission rate in a way that it causes perceivable degradations in the video quality, since a low transmission rate is the main use-case of the TAMVA scheme. For the *Uniform* mode, this distributes the available transmission rate uniformly for all camera views $C_1, \ldots, C_6$, resulting in a target bitrate of 1000 kbit/s for each camera view. The *Prioritization* mode creates a weighted distribution of the total available transmission rate using the proposed TAMVA scheme.

Then, we encode the three driving video sequences using either the encoding parameters estimated by the *Prioritization* mode or the target bitrate of the *Uniform* mode. For both modes, we measure the quality of each camera view using the PSNR and VMAF. The resulting quality scores are weighted with the normalized user importance ratings as outlined in Table 5.2. This weighting of the quality scores further highlights whether the respective mode achieves quality gains for camera views most important for the current driving situation. Lastly, the overall quality for both modes is calculated as the average of the weighted quality scores. Table 5.5 shows the resulting weighted average quality with PSNR and VMAF scores for both modes, *Uniform* and *Prioritization*.

Table 5.5 Average video quality scores of the three driving scenarios weighted by user ratings for the individual camera views.

Mode	VMAF	PSNR
Uniform	37.42	21.87 dB
Prioritization	**39.28**	**22.28 dB**

The *Prioritization* mode achieves a higher overall quality score of 0.41 dB for PSNR and 1.86 for VMAF. This corresponds to relative improvements of roughly 1.8 % for PSNR and 5 % for VMAF. Since we selected a target bitrate of 6000 kbit/s, which is rather low for streaming six videos with a resolution of 1920×1080, the absolute quality scores can be considered as poor quality [183]. At such low visual conditions, a quality improvement of 5 % supports the operator in perceiving relevant details of the driving scene. For higher target bitrates, the relative quality gains achieved by the *Prioritization* mode will further increase, while the prioritization is in particular relevant in situations with a low transmission rate. These quality gains demonstrate the usability of the proposed TAMVA. The adaptation scheme assigns more bit budget to camera views with a higher priority for the users, resulting in a higher visual quality for these views.

5.7 Summary

In this chapter, we proposed a framework for increasing the video quality and scene understanding of the remote operator in ToD. We proposed considering vehicle parameters such as the steering angle or the velocity to prioritize certain camera views. Based on this prioritization and the total available transmission rate, we estimate a specific bit budget for each camera view. Then, an MDA scheme selects the optimal combination of streaming parameters for each camera view to optimize the resulting video quality. The MDA scheme controls the frame size, the frame rate, and the target bitrate for each video stream of the respective camera view. To further improve the system, we propose a dynamic ROI mask to remove unnecessary parts of the frame before the encoding step.

We implemented all methods as separate modules using the TELECARLA framework introduced in Chapter 3. The usability of the proposed system was evaluated in a user study consisting of two separate experiments. In the first experiment, the participants drove actively in a simulated

environment. In the second experiment, the participants observed driving situations recorded in advance and rated the importance of the individual camera views for the current traffic situation.

We could not observe a clear improvement in the driver's performance for the first experiment. However, the proposed view prioritization module achieves a high correlation between the subjective view priority ratings obtained in the second experiment of the user study. With at least 19.8 % BDR savings, the ROI filter approach significantly reduces the required bitrate compared to streaming the full camera frame. Finally, the proposed TAMVA scheme increases the video quality by roughly 5 %, with an increased VMAF score of 1.86.

Chapter 6

Adaptive Multi-View Live Video Streaming Using a Single Encoder

The traffic-aware multi-view adaptation (TAMVA) scheme presented in Chapter 5 enables the individual adaptation of video streams from multiple camera views in teleoperated driving (ToD). The video streams are controlled based on the current traffic situation to provide the remote operator with the best possible understanding for the remote situation. However, commercial vehicles such as used for ToD are often limited in cost and size, and hence equipped with only a single hardware encoder. A possible solution for transmitting multiple video streams using a single encoder is to combine the individual camera views into a single video, compress this single video, and then decompose this single video on the receiver side [184]. While this composition approach allows to transmit multiple camera views with only a single encoder, it prevents the rate/quality adaptation of the individual camera views such as provided by the proposed TAMVA scheme.

In this chapter, we present a preprocessing concept that allows for the individual rate/quality adaptation of multiple camera views while using only a single encoder. We first introduce the general concept of the proposed preprocessing approach and experimentally validate a proof-of-concept implementation. Based on the insights of these first experiments, we discuss the fundamental aspects required for then developing a comprehensive approach for single-view adaptation. We systematically evaluate the influence of different preprocessing algorithms on the rate-distortion (RD) performance in video encoding to select the most suitable candidate. Based on the results of this evaluation, we identify the spatial Gaussian low-pass filter as the preprocessing algorithm that achieves the best RD performance after encoding. Then, we design both one analytical and one machine learning (ML)-based bitrate model that specifically consider the influence of the Gaussian low-pass filter as well as the influence of the quantization parameter (QP), the frame size, the frame rate, and the group of pictures (GoP) length on the video bitrate. Using the two bitrate models proposed, we present a preprocessor rate control approach that allows for estimating the parameters required to control the respective preprocessing filters, based on the adaptation parameters estimated by the TAMVA scheme to control multiple encoders. This way, the preprocessing filters can be used as a rate control scheme to enable the individual rate/quality adaption of multiple camera views while using only a single encoder. Finally, we evaluate the usability of the proposed preprocessing approach in three representative multi-view driving scenarios compared to using multiple encoders and a single encoder approach without any preprocessing.

Some of the concepts and contributions of this chapter have been published in [21, 24, 25]. A reference implementation of all methods proposed in this chapter is publicly available as open source on GitHub[1].

[1] https://github.com/hofbi/amvs-se

6.1 Preprocessing Concept

In this section, we present a concept that uses image filters in a preprocessing step to enable individual rate/quality adaptation while using a single encoder. The limited encoding hardware available in commercial vehicles requires the combination of multiple camera views into a single frame. We refer to this composition of individual frames as a superframe and to the respective video stream as a superframe video (SFV). A workflow that uses this frame composition of multiple camera views into a single SFV is shown in Figure 6.1.

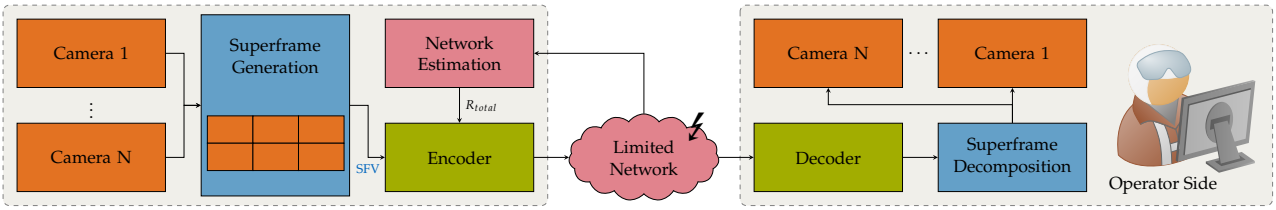


Figure 6.1 Conceptual overview of a multi-view streaming pipeline with limited encoding hardware. N camera view are combined into a superframe video (SFV) that is encoded by a single encoder.

Multiple camera views are combined into a single SFV based on a predefined layout for the superframe. The SFV is then encoded by a single encoder to match the available transmission rate of the mobile network. The resulting bitstream is transmitted over a communication network with resource constraints. On the receiver side, the bitstream is decoded and decomposed back into the individual camera views. These camera views are then consumed by the human operator.

While this approach enables the compression and transmission of multiple camera views simultaneously with only a single encoder, it prevents the rate/quality adaptation of individual camera views. An encoder such as available in embedded systems or industrial applications, ranging from smartphones to vehicles, is usually a closed system with only a limited interface for the parameter settings to control the encoding process. Apart from this interface, the encoder is a black box and does not allow for in-depth rate control that would be required for individual rate/quality adaptation. The only way to adapt the rate/quality of the SFV is to use the rate control application programming interface (API) of the encoder, which results in a uniform adaptation of all camera frames combined in the superframe.

To address this issue, we propose to use preprocessing image filters as an alternative rate/quality adaptation strategy if only a single encoder is available. Next, we present the proposed concept and the individual preprocessing image filters in detail.

6.1.1 Concept

We investigate how to transmit multiple camera views with individual rate/quality adaptation while using a single encoder. To still allow for individual rate/quality adaptation, we propose preprocessing image filters to manipulate the individual camera views before combining them into a single SFV. Due to this preprocessing, the content of the individual video is slightly modified. These changes in the individual camera views affect the bitrate required by the encoder to encode the SFV and the resulting video quality of the individual camera views after decoding and decomposing the SFV. Hence, the rate and quality is controllable for every individual segment in the superframe. Figure 6.2 summarizes the concept of the proposed preprocessing approach and how it allows for streaming multiple camera views using a single encoder.

The video streams of multiple camera views are preprocessed with the proposed image filters. The preprocessed video sequences of the individual views are then composed into a larger SFV. From this point forward, the workflow is identical to the workflow introduced in Figure 6.1. The SFV is encoded by the single encoder and transmitted to the client. On the client side, the bitstream is decoded and

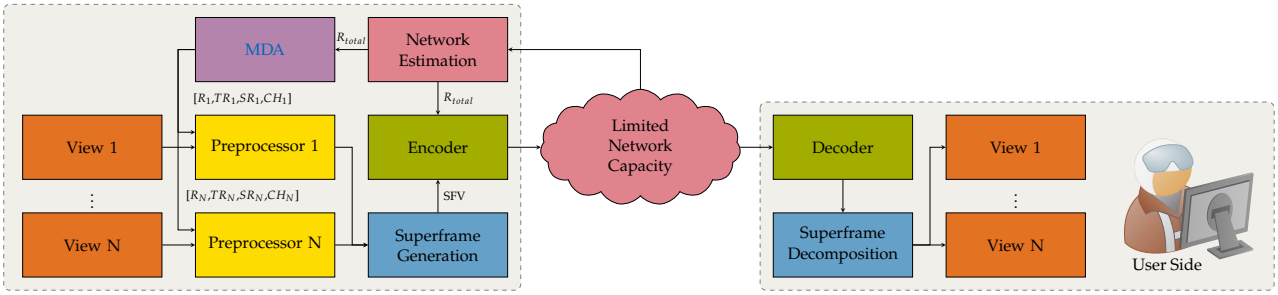


Figure 6.2 Conceptual overview of the proposed preprocessing filter approach for individual rate/quality adaptation within a superframe video (SFV) while using a single encoder. A multi-dimensional adaptation (MDA) model such as used in the proposed traffic-aware multi-view adaptation (TAMVA) scheme estimates the optimal encoding parameters of every camera view based on the total available transmission rate R_{total}. These encoding parameters are the bitrate R, the temporal resolution TR, the spatial resolution SR, and the used color channels CH. The preprocessing filters manipulate the individual video streams to achieve a similar rate/quality after encoding as if they would have been encoded by multiple encoders.

the resulting SFV is decomposed back into the individual video streams, which are finally shown to the operator.

In this scenario, the bitrate required by the encoder and the video quality after decoding is controllable for each individual segment in the superframe due to the proposed preprocessing. Next, we introduce the preprocessing filters used in this approach and discuss how they affect the actual encoding process.

6.1.2 Preprocessing Filters

In the literature, preprocessing image filters are mainly used to improve the image quality of a single frame. Such preprocessing approaches try, for instance, to spend more bit budget on the foreground of the frame compared to its background [167, 165]. However, these approaches are often proposed as extensions of specific video encoders, where detailed information about the bitrate distribution within the frame is available during the encoding step. In embedded systems such as vehicles, the encoder is a closed system where this information is not available. We therefore first discuss how preprocessing filters can be used when treating the encoder as a black box, especially when it comes to the simultaneous transmission of multiple views using a single encoder.

To address the limited access to detailed information during the encoding process, we propose a preprocessing step before the individual views are combined into a single superframe and encoded by a single encoder. In total, we propose four preprocessing filters in this thesis: a temporal filter controlling the frame rate, a spatial filter controlling the frame size, a low-pass filter controlling the quality, and a channel filter controlling individual color channels. Figure 6.3 shows the four preprocessing filters applied on the individual camera views before the superframe composition.

The raw video sequence of the respective camera view is either not filtered at all or preprocessed by one or more of the preprocessing filters. The preprocessor module controlling the individual filters accepts the adaptation parameters provided by an MDA model such as introduced in Chapter 5. The bitrate R_i controls the rate/quality, the temporal resolution TR_i controls the frame rate, the spatial resolution SR_i controls the frame size, and the parameter CH_i specifies which color channels should be used for the respective camera view. This design allows to integrate the proposed preprocessing approach to existing systems such as proposed in Chapter 5. Next, we discuss how the individual preprocessing filters operate in detail and how they affect the encoding process.

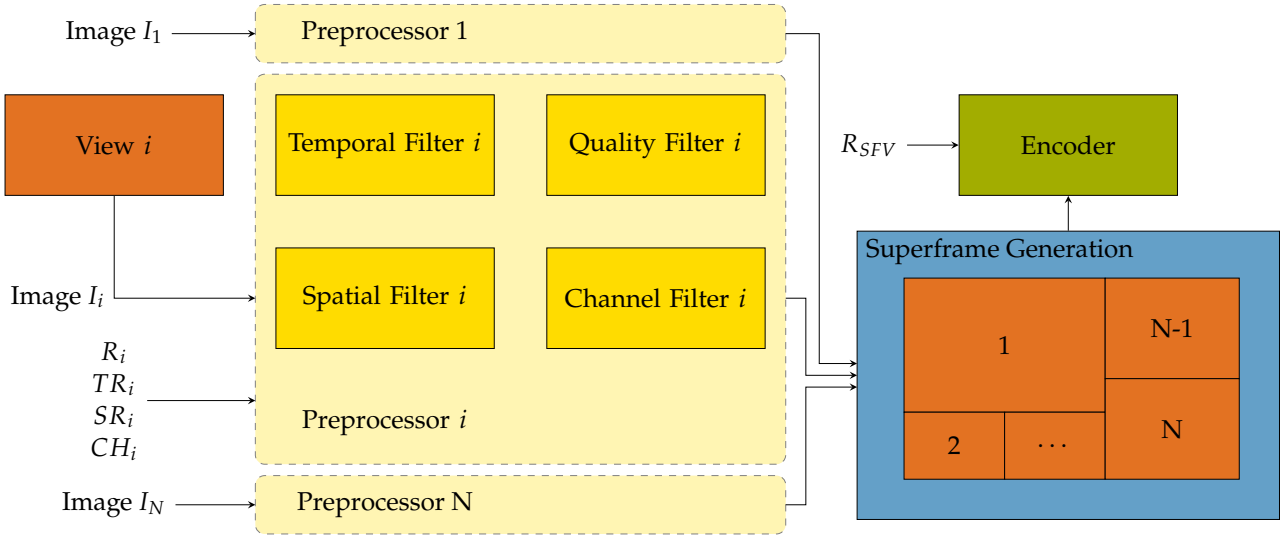


Figure 6.3 The four preprocessing filters applied on the individual camera views before the composition into the superframe.

6.1.2.1 Temporal Filter

The temporal filter is designed to control the temporal resolution TR (frame rate) of individual videos. The single encoder operates at a predefined common frame rate TR_{SFV} when encoding the SFV. If the temporal resolution TR_i of a single video should be reduced due to the current situation not requiring a high frame rate for the respective camera view, the temporal filter drops frames of the individual video. To still maintain the common frame rate TR_{SFV} required by the encoder, the previous frame buffered by the temporal filter will be reused instead of the current one. This way, the individual frame and hence the respective segment in the superframe after the composition is identical to the previous frame. All image blocks such as macro blocks (MBs) or coding tree units (CTUs) related to the individual frame will be treated by the encoder with skip mode. This significantly reduces the bitrate required for encoding the respective segment of the SFV, down to almost zero. As a side effect of the reduced frame rate, the inter-frame difference between two neighboring frames will be slightly increased, which increases the bitrate. However, the additional bitrate required for encoding the larger inter-frame difference is significantly smaller than the bitrate of the frame and can be neglected. As a result, the frame rate TR_i of the individual segment containing novel information is reduced with almost no additional bitrate required, while the common frame rate TR_{SFV} required by the single encoder is still maintained.

6.1.2.2 Spatial Filter

The spatial filter controls the spatial resolution SR of an individual frame. The filter resizes the output image to the requested frame size SR. After the combination of the individual videos into the SFV, different frame sizes result in different amounts of bits required to encode the individual sub-videos.

6.1.2.3 Quality Filter

The quality filter is an adjustable spatial low-pass filter applied to the input image. This filter is used to control the bitrate or the video quality required for encoding each individual segment in the SFV. Depending on the cut-off frequency of this filter configured for the respective input image, a different bitrate will be required for encoding each part of the SFV and hence will affect the resulting video quality.

6.1.2.4 Channel Filter

Lastly, we design a channel filter to control which color channels CH should be used for encoding the respective camera view segment. Based on the current situation and the human vision system (HVS), not all color channels available in the input image are of similar importance for the human operator. For instance, since the HVS is more sensitive to brightness than to color, the luminance channel representing the brightness should be transmitted, whereas the chrominance channels representing the color components can be set to zero in case of severe communication constraints. This reduces the bitrate of the image segment filtered by the channel filter since the encoder has to encode only a gray-scale image for this segment of the SFV.

In particular during the night, the chrominance channels provide little additional information for the operator. Such conditions are a possible application of the channel filter. While existing adaptation models typically do not consider individual color channels for controlling a video stream, the proposed channel filter can be beneficial for highly specific applications such as driving.

6.1.3 Evaluation Setup

In this section, we present the setup used in the following experiments to evaluate the proposed preprocessing approach. We use a representative driving scenario recorded with multiple camera views as the uncompressed video sequences required for the evaluation. The driving scene was recorded in the CARLA driving simulator [16] using a vehicle equipped with six RGB cameras as shown in Figure 6.4.

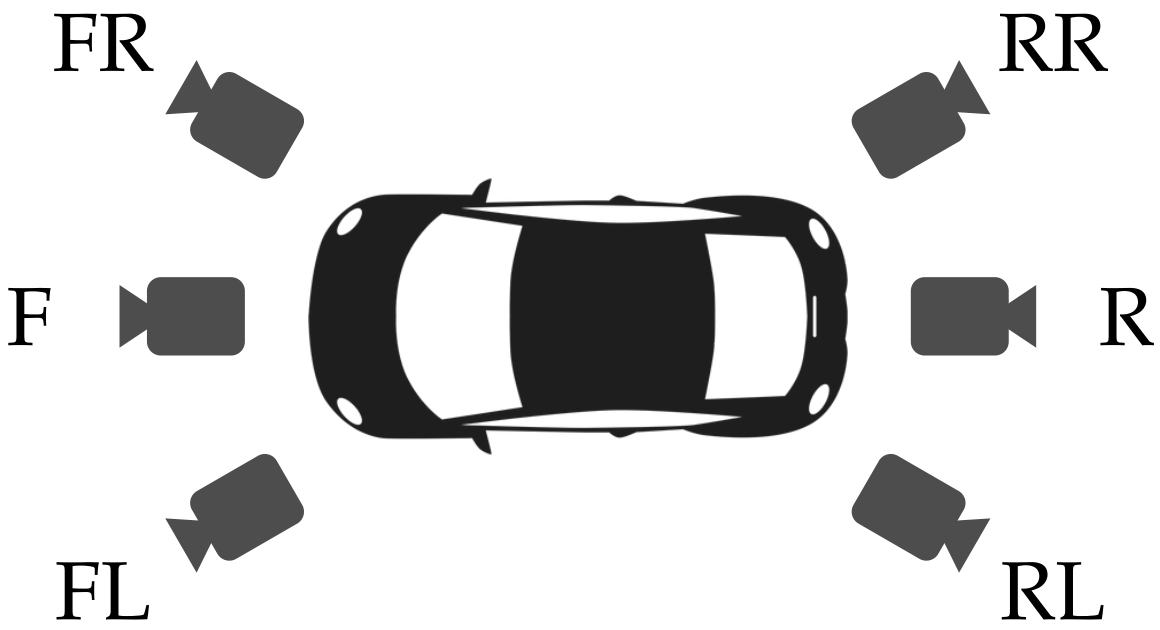
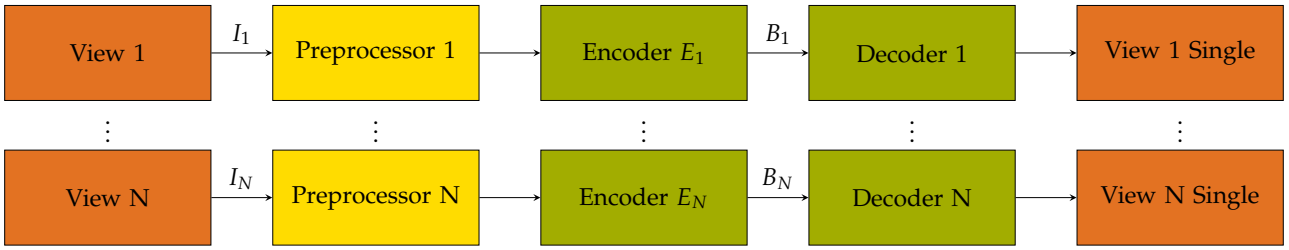


Figure 6.4 Camera setup of a vehicle with six RGB cameras used for recording the evaluation scenario. The camera views are referred to as front (F), front left (FL), front right (FR), rear (R), rear left (RL), and rear right (RR). Adopted from [21] © 2020 IEEE.

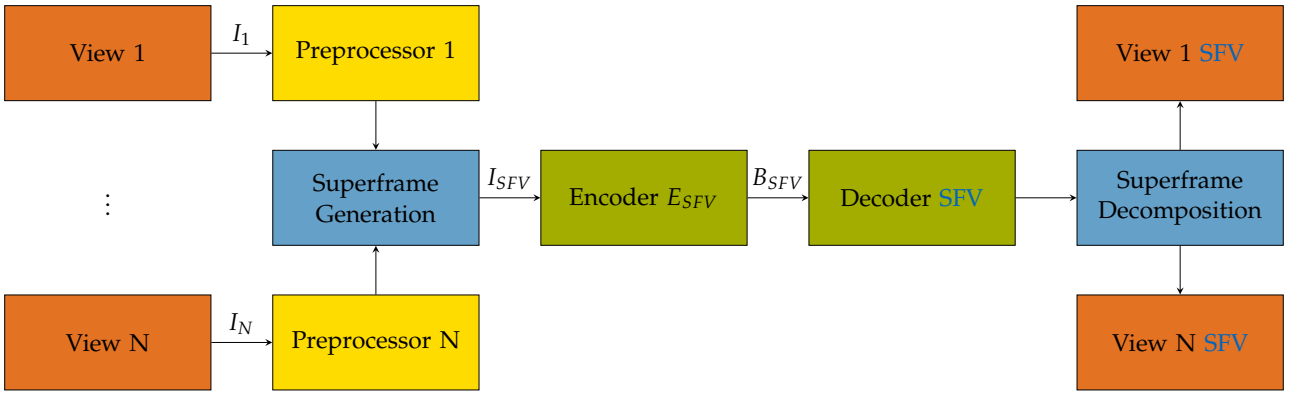
The driving scenario was selected from the MV-ROI dataset introduced in Section 4.1.1. The selected driving scenario *turn-left* includes a left turn at an intersection with oncoming traffic.

We use the open source x264 software video encoder [134] to process the uncompressed video sequences captured from the *turn-left* scenario. To evaluate the effects of the individual preprocessing filters and to avoid the influence of any rate control algorithm, we use the encoder in constant quan-

tization parameter (CQP) mode throughout the experiments. Figure 6.5 presents the two workflows used for the evaluation of the proposed approach.



(a) Individual encoding.



(b) Superframe video (SFV) encoding.

Figure 6.5 Workflow of the evaluation setup and reference software implemented to validate the proposed preprocessing concept.

The uncompressed video sequence I_i recorded from every camera view is either preprocessed by one of the filters introduced in Section 6.1.2 or directly fed into the pipeline without applying any preprocessing. Then, every camera view is encoded and decoded by a separate encoder E_i as outlined in Figure 6.5a. We refer to this video sequence encoded individually as *Single*. In parallel, a copy of every video sequence I_i preprocessed in the same way as for the respective *Single* mode is combined into a single SFV I_{SFV}. Figure 6.5b visualizes this second workflow for the SFV. The resulting SFV is also encoded and decoded by a separate encoder E_{SFV}, and finally divided back into the individual frame segments. These segments decomposed from the SFV are denoted as *SFV*.

This processing pipeline enables us to analyze the bitrate distribution of the encoded SFV bitstream and to compare the frames encoded individually with the SFV segments after the decomposition. For this analysis, we measure the size of the encoded bitstream B from the individual camera views as well as from the individual segments decomposed from the SFV. Additionally, we evaluate the perceptual video quality using three video quality metrics (VQMs). These VQMs are the multi-dimensional video quality metric (MDVQM) [153], the spatio-temporal video quality metric (STVQM) [149], and Video Multi-Method Assessment Fusion (VMAF) [156]. Next, we design two experiments using this setup to evaluate the proposed approach and present the results.

6.1.4 Results

In this section, we present two experiments to evaluate the proposed concept for multi-view live video stream adaptation with a single encoder. The first experiment validates the proposed concept using preprocessing image filters to enable individual rate/quality adaptation of image segments within the superframe while using a single encoder for encoding the SFV. In the second experiment, we evaluate the general usability of the proposed concept in an exemplary multi-view driving scenario.

We compare the proposed method to an approach that encodes the SFV without preprocessing and to a reference approach that encodes the individual video streams using multiple encoders.

6.1.4.1 Proof of Concept

The first experiment evaluates the influence of the four preprocessing filters introduced in Section 6.1.2 on the individual video sequence as well as on the respective segment after the combination into the SFV. For this evaluation, we use the setup and test video sequences introduced in Section 6.1.3.

Two individual video streams of the same camera view are fed into the evaluation pipeline of Figure 6.5. We use the identical video stream of the front view camera (F) twice to exclusively highlight the effects of the individual preprocessing filters compared to the unfiltered segment, without considering the scene content. The first video stream I_1 is not modified by any preprocessing filter. The second video stream I_2 is preprocessed with one of the four preprocessing filters introduced in Section 6.1.2.

The temporal filter reduces the temporal resolution TR by a factor of two. This is achieved by dropping every second frame. The remaining frames are then duplicated to keep the effective frame rate constant. The constant frame rate is required for composing the individual frames into the SFV. The spatial filter scales both the width and the height of the original image down by a factor of two. The resulting height difference existing during the superframe composition between I_1 and I_2, downscaled by the spatial filter, is compensated by filling the remaining area with black pixels. Similar to the region of interest (ROI) masking approach introduced in Chapter 5, these areas do not contribute to the encoded frame size due to the encoder's skip mode. For the quality filter, we select a median filter configured with a kernel size of five to reduce the bitrate required for encoding the respective segment. The channel filter creates a single channel image by dropping both chrominance channels while only keeping the luminance channel. Figure 6.6 shows the size of the encoded bitstream for the two individual streams B_1 (without filter) and B_2 (with filter), the bitstream of the SFV B_{SFV}, and the sum of the individual streams $B_\Sigma = B_1 + B_2$.

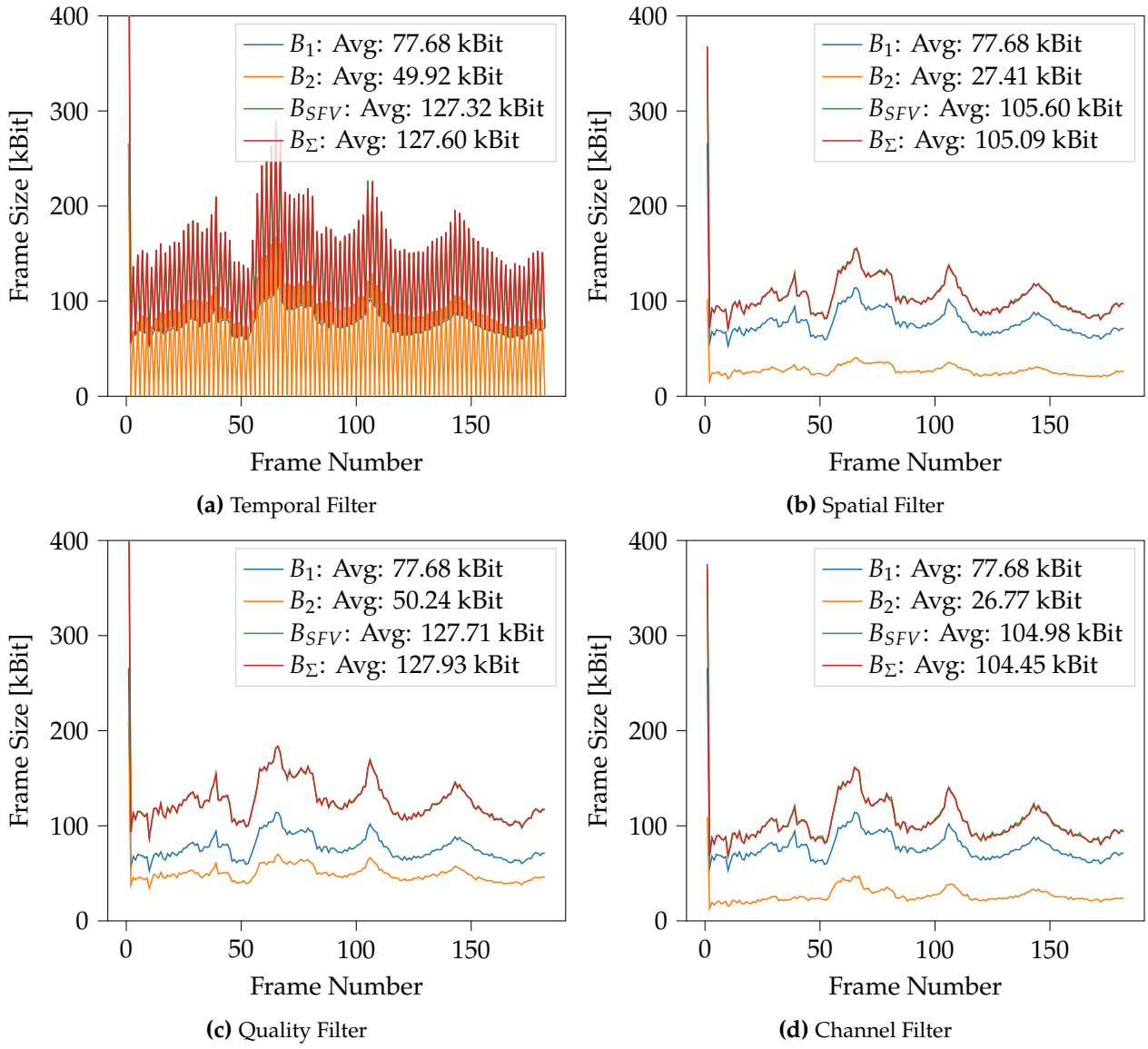
The orange line representing the trend of B_2 visualizes the influence of the corresponding filter on the encoded bitstream. Figure 6.6a presents an encoded frame size of close to zero for every second frame of B_2 such as configured for the temporal filter. Both B_{SFV} and B_Σ follow this behavior. This suggests that the image segments repeated by the temporal filter do not contribute to the SFV bitstream B_{SFV} and are properly processed by the encoder's skip mode. The spatial filter in Figure 6.6b, the quality filter in Figure 6.6c, and the channel filter in Figure 6.6d show a constantly reduced bitstream size for B_2 compared to the size of the unfiltered bitstream B_1. This reduced size demonstrates the influence of the respective preprocessing filter. The size of the encoded bitstream for B_{SFV} and B_Σ is almost identical for every preprocessing filter. This behavior suggests that the preprocessing filters do not influence any of the other segments and exclusively affect the camera view where the filter is applied to. This can be explained by the block-based image processing of the video encoder.

In addition to the bitrate, we evaluate the perceptual video quality to measure the influence of preprocessing filters on the quality of individual segments combined in the SFV. We measure the video quality for the video streams B_1 and B_2 encoded individually (*Single*) and the individual segments decomposed from the SFV (*SFV*) using three VQMs. Then, we calculate the mean absolute difference (MAD) of the VQM scores for *SFV* compared to *Single* as follows:

$$MAD = \frac{1}{N}\sum_{i=1}^{N}|VQM(Single_i) - VQM(SFV_i)|. \tag{6.1}$$

Table 6.1 summarizes the MAD calculated for the three different VQMs MDVQM, STVQM, and VMAF.

According to the value range from 0 to 100 for all VQMs, the MAD is on average about 0.2 %. The negligible difference in video quality confirms the results of the bitrate evaluation and demonstrates that the preprocessing filters do not affect the video quality of any other segment. This is an important



(a) Temporal Filter **(b)** Spatial Filter

(c) Quality Filter **(d)** Channel Filter

Figure 6.6 Size of the encoded bitstream for the two video bitstreams B_1 and B_2 encoded individually using the identical front camera view (F), B_1 without preprocessing filter applied and B_2 with preprocessing, the SFV bitstream B_{SFV}, and the sum of the individual streams $B_\Sigma = B_1 + B_2$. All video sequences are encoded in CQP mode with a QP of $q_p = 25$. One of the four different preprocessing filters is applied on B_2.

Table 6.1 Mean absolute difference (MAD) of the VQMs for the video bitstreams B_1 and B_2 processed individually (*Single*) compared to the VQMs of the segments decomposed from the SFV bitstream B_{SFV} (*SFV*).

VQM	View	Temporal	Spatial	Quality	Channel
MDVQM	B_1	0.121	0.116	0.105	0.071
MDVQM	B_2	0.018	0.270	0.387	1.093
STVQM	B_1	0.046	0.042	0.057	0.051
STVQM	B_2	0.165	0.404	0.269	0.170
VMAF	B_1	0.031	0.033	0.031	0.033
VMAF	B_2	0.209	0.275	0.165	0.000

insight for the proposed approach, since the influence of the preprocessing filters can be considered for every camera view independently. Based on this first proof of concept, we next evaluate the general usability of the proposed preprocessing concept in an exemplary multi-view driving scenario.

6.1.4.2 Usability

In the previous experiment, we validated the concept of the proposed preprocessing filters. The second experiment evaluates the usability of the proposed method for the *turn-left* driving scenario. The scenario consists of video sequences recorded from six camera views as outlined in Figure 6.4.

For this experiment, we use all six camera views available from the multi-view driving scenario. We compare the proposed preprocessing approach with an individual adaptation strategy using multiple encoders and the simple SFV approach without preprocessing. We again use the evaluation setup introduced in Section 6.1.3 with the encoder in CQP mode. To control the video encoders, we assume the following adaptation parameters for the current driving scenario such as outlined in Table 6.2.

Table 6.2 Assumed multi-dimensional adaptation (MDA) model output for the six individual camera views. The output consists of the quantization parameter q_p, the temporal resolution TR relative to the original temporal resolution $TR_0 = 20\,\text{Hz}$, the spatial resolution SR relative to the original spatial resolution $SR_0 = 640 \times 480$ px, and the kernel size k of the median filter.

MDA	F	FL	FR	R	RL	RR
q_p	25	25	26	27	26	25
TR/TR_0	1	1	1	0.5	0.5	0.25
SR/SR_0	1	1	1	1	1	0.5
k (median)	-	-	3	5	3	-

We selected the CQP encoder mode to avoid modifications of the encoded bitrate due to any rate control methods implemented in the encoder. Hence, a quantization parameter q_p is provided to control the quality level for the resulting video stream. Additionally, the MDA model provides a temporal resolution TR and a spatial resolution SR for the individual camera views. The values for TR and SR listed in Table 6.2 are relative to the temporal resolution TR_0 and spatial resolution SR_0 of the original video, respectively. Since there is no model that considers which color channels should be used for the transmission, we do not consider the channel filter in this experiment. In a real driving scenario, the parameters q_p, TR, and SR could have been estimated by the TAMVA scheme proposed in Chapter 5.

The goal of the TAMVA scheme including an MDA model is to match the available transmission resources of the mobile network while providing the operator with the best possible situation awareness for the current traffic situation. This means ensuring a high spatial and temporal resolution as well as a good video quality for the most important views. This can be achieved by assigning a larger bit budget for camera views which are highly relevant for the current driving situation at the cost of reduced quality for the less important camera views. Camera views which are less relevant for the current driving task hence tolerate a reduction of the bit budget as well as spatial and temporal resolution in favor of the more important views. Furthermore, existing operator interfaces such as proposed in [55] show the rear view as a small picture-in-picture overlay similar to a rear-view mirror. This means that some views could be downscaled by default, avoiding an unnecessarily high spatial resolution which has to be transmitted over the network and downscaled on the operator side.

In case of the *turn-left* driving scenario, which is used for this experiment, the parameters estimated by the MDA model favor the front (F) and front left (FL) views since they cover the driving route of the ego vehicle. During the left turn, the front right (FR) view is required to perceive oncoming traffic. Due to the high motion of this view caused by the turn, the q_p for FR is slightly increased to

save bit budget. The rear (R) and rear left (RL) views are less important views and only required for regular check ups. These views are not constantly observed and thus a reduced temporal and spatial resolution is sufficient for this task. The rear right view (RR) is the least important view for the left turn. Hence, SR and TR are reduced the most in favor of the other views.

Based on the parameters in Table 6.2 specified for the *turn-left* driving scenario, we evaluate the bitrate and perceptual video quality for four modes. An individual adaptation strategy using multiple encoders acts as the baseline implementation. We refer to this mode using multiple encoders as *MulEnc*, which uses the q_p, TR, and SR settings from Table 6.2.

Second, the proposed preprocessing approach using a single encoder is referred to as *Ours-25*. The individual video streams are preprocessed based on the respective TR, SR, and k listed in Table 6.2. Restricted to a single encoder, a suitable QP for the superframe encoder is required. Since the preprocessing filters only allow for a rate/quality reduction, we selected the minimum QP of the *MulEnc* mode. This results in a QP of 25 for the SFV of *Ours-25*. To match the bitrates for the individual views of *MulEnc*, we apply the proposed preprocessing filters to the individual views before the combination into the SFV. TR and SR of Table 6.2 are directly used by the preprocessing filters. The median filter is applied to better match the bitrate of the video streams encoded individually that have a different QP than the superframe encoder. The kernel size k of the median filter is shown in the last row of Table 6.2.

Lastly, we evaluate a simple SFV approach without any preprocessing applied for two QPs, 25 and 28. We refer to these modes as *SFV-25* and *SFV-28*. *SFV-25* uses the same QP of 25 as *Ours-25* for all camera views. The QP for *SFV-28* is 28, which matches the total bitrate of *MulEnc*. Table 6.3 shows the resulting bitrate for every camera view as well as the total bitrate of all camera views.

Table 6.3 Resulting bitrate R [kbit/s] using the selected encoding settings for the six camera views.

Mode	Total	F	FL	FR	R	RL	RR
MulEnc	6618	1552	1446	1506	978	864	266
Ours-25	6684	1552	1446	1454	1036	924	266
SFV-25	10 634	1552	1446	1818	2370	1690	1752
SFV-28	6594	1002	956	1178	1290	1062	1102

The mode *SFV-25* reaches similar bitrates as *MulEnc* for the most important front views. However, the total bitrate required for all views is about 60 % larger. Such a large bitrate being required for transmission might be an issue in communication networks with limited transmission resources. On the other hand, both *Ours-25* and *SFV-28* closely match the total bitrate achieved by the baseline *MulEnc*. The bitrate for the individual views of *SFV-28* deviates from the bitrates of *MulEnc*, while *Ours-25* also matches the bitrates for the individual camera views. For *SFV-28*, this means a reduced quality for the important front view due to the lower bitrate and an unnecessarily high quality for the less important rear views. Table 6.4 summarizes the resulting VQM scores for the individual camera views as well as the average VQM scores of all camera views using the MDVQM, STVQM, and VMAF.

The quality measurements show a behavior, which is similar to the bitrates for the different modes. For *Ours-25*, both the average quality and the quality of the individual views have a similar trend as *MulEnc*, while *SFV-25* only matches the important front view. The average quality and the quality for the rear views is even better, at the cost of an increased bitrate required for the transmission.

For the video streams preprocessed with the median filter, the video quality of *Ours-25* shows higher VQM scores than *MulEnc* for almost the same bitrate. The full reference STVQM shows even higher scores than the no-reference MDVQM. A possible explanation for this behavior of STVQM is that the calculation is based on the peak signal-to-noise ratio (PSNR). The median filter smooths the image content, which reduces the noise or tiny details in the image and hence increase the PSNR that

Table 6.4 Resulting perceptual video quality using the encoder settings selected for the six camera views.

Mode	VQM	Avg.	F	FL	FR	R	RL	RR
MulEnc	STVQM	54.5	68.2	71.2	67.4	30.6	35.1	32.5
	MDVQM	63.4	87.2	85.8	88.5	29.4	26.1	18.7
	VMAF	79.6	96.0	95.9	95.5	70.5	73.8	46.4
Ours-25	STVQM	61.9	68.2	71.2	74.6	48.8	46.9	32.5
	MDVQM	66.1	87.2	85.8	91.1	38.2	28.5	18.7
	VMAF	73.7	96.0	95.9	88.2	49.2	66.7	46.4
SFV-25	STVQM	63.8	68.2	71.2	72.5	54.5	53.0	63.7
	MDVQM	81.9	87.2	85.8	91.8	84.1	60.8	81.5
	VMAF	95.8	96.0	96.0	97.3	95.4	94.2	95.9
SFV-28	STVQM	52.7	56.9	60.9	61.5	42.2	42.2	53.1
	MDVQM	64.6	68.6	69.2	82.7	63.3	39.3	69.1
	VMAF	91.3	91.8	92.0	93.4	92.0	90.1	92.3

affects the STVQM score. Since the MDVQM only models the quality on a signal-to-noise ratio (SNR) base, but does not require access to the original frame for comparison, it is less sensitive to this kind of filters.

VMAF shows a lower score for segments with reduced spatial or temporal resolution such as for the rear views of *MulEnc* and *Ours-25*. The remaining views show a very high absolute score with small deviations for the different QPs. These high scores achieved for the QPs used in this experiment can be explained with VMAF's viewing distance assumed by Nexflix when collecting the training data and designing the model. The viewing distance was selected to predict the quality of videos displayed on a 1080p HDTV in a living-room-like environment, which should be three times the display height for 1080 pixels. With an original height of 480 pixels, the viewing distance assumed by the model further increases, which can hide the artifacts introduced by the quantization [185].

In summary, the results highlight that the proposed preprocessing solution matches the individual bitrates and perceptual video quality scores more closely compared to the simple SFV approach. However, the median filter selected in this example affects the visual quality, although in a different way compared to using multiple encoders. Next, we systematically evaluate the influence of different preprocessing algorithms to identify the most suitable low-pass filter for this task.

6.2 Preprocessing Filter Performance Evaluation

In the previous section, we introduced the concept of preprocessing filters to enable individual rate/quality adaptation of multiple segments in a SFV with only a single encoder available. Based on our first experiments, we demonstrated the general usability in a multi-view driving scenario as well as how the preprocessing filters affect the rate/quality of the SFV. None of the preprocessing filters introduced in Section 6.1.2 has any side effects on the neighboring views during encoding if the encoder is used in CQP mode. Additionally, the spatial and temporal filter controlling the spatial and temporal resolution of the individual segments achieve similar results as if using multiple encoders. However, the quality filter used to control the rate/quality of the individual video affects the video quality differently compared to using multiple encoders. Since there are multiple preprocessing algorithms available that can be used for this task, we are interested in determining an algorithm that saves the most bitrate while having the smallest effect on the visual quality. Hence, we systematically evaluate the influence of preprocessing algorithms on the RD performance in video encoding next.

6.2.1 Rate-Distortion Analysis

In this section, we analyze the RD performance of the encoder for input video sequences preprocessed with a quality filter based on the definition of Section 6.1.2. To motivate this evaluation, we first discuss how such preprocessing filters are currently used in the literature.

Grois et al. [167] proposed a complexity-aware adaptive ROI prefiltering scheme for scalable video coding. The authors applied the prefilters dynamically with a transition region between foreground and background to improve visual presentation quality. For the evaluation, they used Gaussian, Wiener, and Wavelet filters. Due to its low computational complexity, the Gaussian filter showed the best performance trade-off [168, 169]. In [167], the authors encoded the background with a fixed QP of 30 and the foreground with a variable QP between 20 and 30. While each of the preprocessing methods achieved rate-quality gains, these gains were only measured for specific QPs. Since their main goal was to improve the visual quality of the ROI, the quality degradations for the background are less important. In our application, we use the preprocessing filters for rate control. Thus, we require performance measures for a larger range of parameters and must consider the quality degradations. To effectively compare the performance of different preprocessing filters, the rate-quality gains have to be evaluated for a larger range of QPs and for different preprocessing parameters.

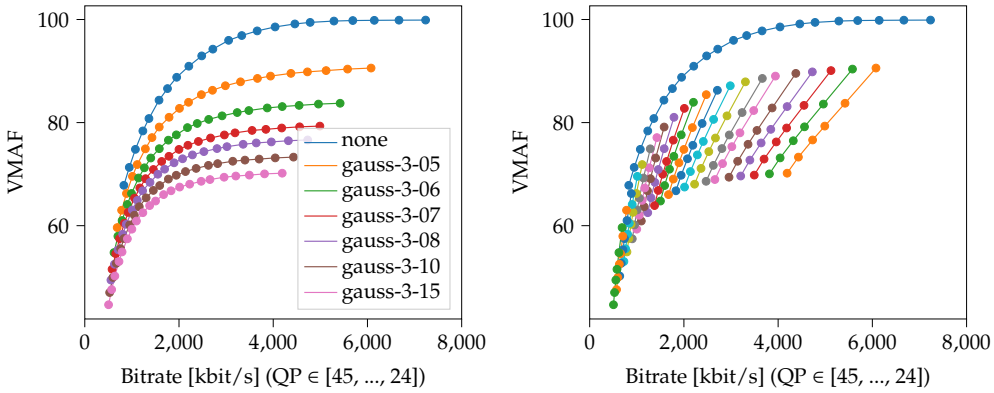
For this, we systematically evaluate the rate-quality gains of different preprocessing filters by analyzing the RD curves of a Gaussian filter, a median filter, and a Joint Photographic Experts Group (JPEG) preprocessor. The median filter and the Gaussian filter are used as usual. The JPEG preprocessor simply encodes and then decodes the image. We select the JPEG preprocessor as an easily available implementation of a discrete cosine transformation (DCT). The values in the quantization table used for the DCT coefficients are chosen to preserve low-frequency information and discard high-frequency information, since humans are less critical to the loss of these high-frequency information. Hence, the DCT of the JPEG preprocessor represents a filter optimized for the HVS. The Gaussian filter has been selected since it is most commonly used in the literature due to its low computational complexity [163, 168, 169, 167]. To enable comparability of this evaluation with the results of Section 6.1.4.2, we use the median filter as a third filter with a low computational complexity.

For the RD analysis, we use the uncompressed *Bus* video sequence available in common intermediate format (CIF) resolution from [186]. We create different video sequences of *Bus* preprocessed with the following preprocessing filters and filter parameters: a Gaussian filter with a kernel size $k = 3$ and standard deviations $\sigma \in \{0.5, 0.6, 0.7, 0.8, 1.0, 1.5\}$, a Gaussian filter with $k = 5$ and the same values of σ, a median filter with kernel sizes $k \in \{3, 5, 7, 9\}$, and a JPEG preprocessor with the quality levels $Q \in \{10, 20, 40, 60\}$.

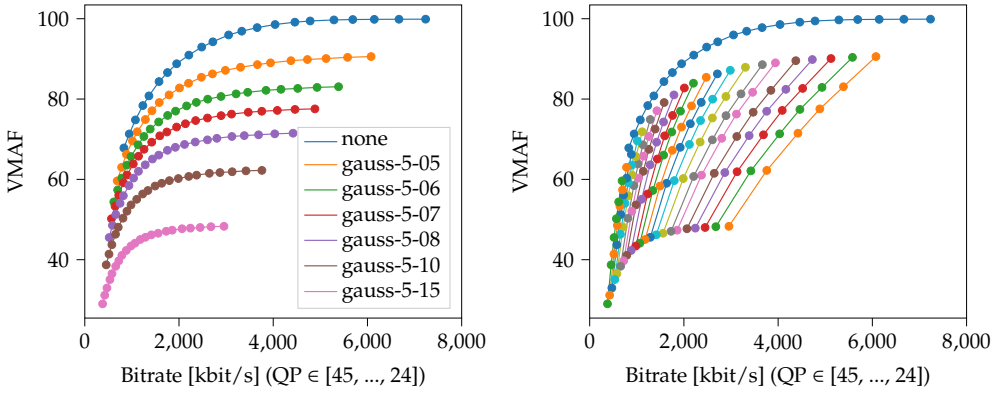
We encode the preprocessed video sequences using the x264 software video encoder [134] with CQP mode. The QPs used for encoding range from 24 to 45 with a step size of one. Further, we select a GoP length of one, meaning I-frames only. We measure the video quality for each encoded video sequence using the state-of-the-art video quality metric VMAF [156, 157]. Figure 6.7 shows the resulting RD curves for the four preprocessing filter configurations used in this evaluation.

Every row in Figure 6.7 represents one preprocessing filter configuration. The left column shows the regular RD curves for all QPs with the specific filter parameters in a separate color. The blue curve *none* represents the baseline without any preprocessing applied. The right column contains the same RD points as the left column, but with the RD curves per QP. Each new color represents an RD curve for the preprocessing parameters with a different QP. In the following, we refer to these kind of curves as rate-distortion per quantization parameter (RD-QP) curves.

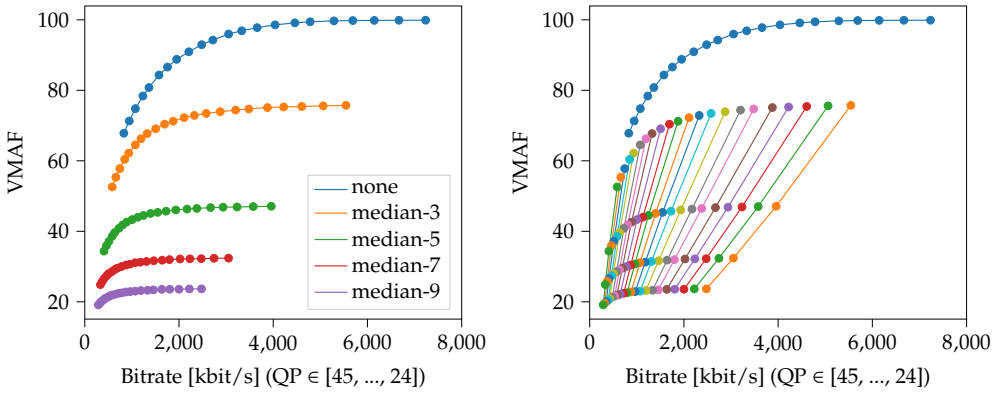
We observe smooth RD curves for the Gaussian filters and the median filter. In contrast, the JPEG preprocessor produces a shaky RD curve. At the same time, the JPEG preprocessor still reaches high VMAF scores of 70 to 80, even for a quality level of 10. For the Gaussian filters, a kernel size of five achieves higher rate savings while causing higher quality costs at the same time. The median filter has a strong influence on the VMAF score already at a kernel size of three.



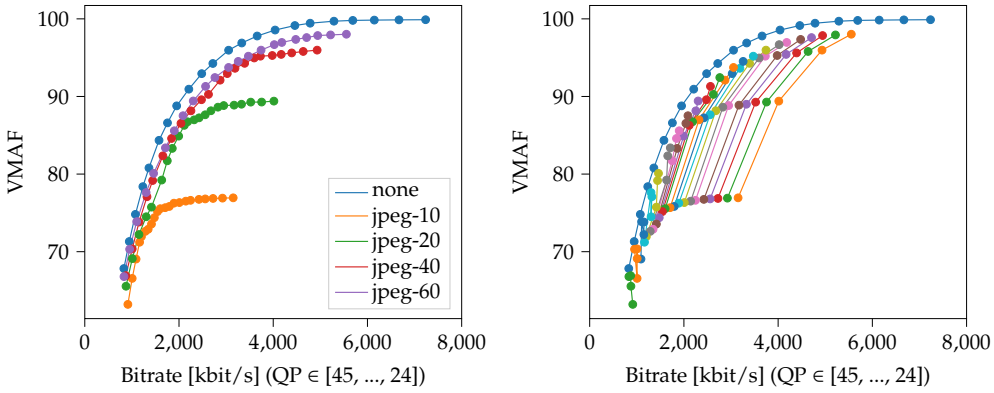
(a) Gaussian Filter $k = 3$



(b) Gaussian Filter $k = 5$



(c) Median Filter



(d) JPEG Filter

Figure 6.7 Rate-distortion (RD) analysis for the preprocessing filters Gaussian, median, and JPEG. Every row contains the RD curves for one of the preprocessing filters. The left column shows the RD curves color-coded by the filter parameter. The right column color-codes the RD curves per QP. Adopted from [25] © 2022 IEEE.

Our first analysis demonstrates the high dimensionality of the problem presented. To effectively compare the performance of different preprocessing filters, we require a new evaluation method which we present next.

6.2.2 Evaluation Methods

In this section, we present our evaluation methods to compare the performance of different preprocessing filters. We first analyze the bitrate savings that can be achieved while considering the costs of certain visual quality degradations. Then, we introduce the Bjøntegaard Delta (BD) curves to compare two RD-QP curves of two different preprocessing filters with the same QP. Finally, we propose a novel evaluation method to effectively compare the RD performance of different preprocessing algorithms and their parameters.

6.2.2.1 Quality-Cost Rate-Saving Curves

When using the preprocessing filters for the purpose of rate control, we are mainly interested in the rate savings that can be achieved and the quality costs required for these savings. Hence, we calculate the quality costs and rate savings for every point of the RD curves relative to the baseline without preprocessing. We define the rate saving $S(q_p)$ for a certain quantization parameter $q_p \in \{24, \dots, 45\}$ as the difference of the baseline bitrate $R_{none}(q_p)$ without any preprocessing applied and the video bitrate $R_{filter}(q_p)$ preprocessed with one of the preprocessing filters:

$$S(q_p) = R_{none}(q_p) - R_{filter}(q_p) \tag{6.2}$$

Similar to the rate savings, we define the quality costs $C(q_p)$ as the quality difference of the baseline distortion $D_{none}(q_p)$ and the filter distortion $D_{filter}(q_p)$ preprocessed with one of the preprocessing filters:

$$C(q_p) = D_{none}(q_p) - D_{filter}(q_p) \tag{6.3}$$

Figure 6.8 visualizes the resulting quality costs $C(q_p)$ over their rate savings $S(q_p)$. From now on, we refer to them as cost-saving (CS) curves.

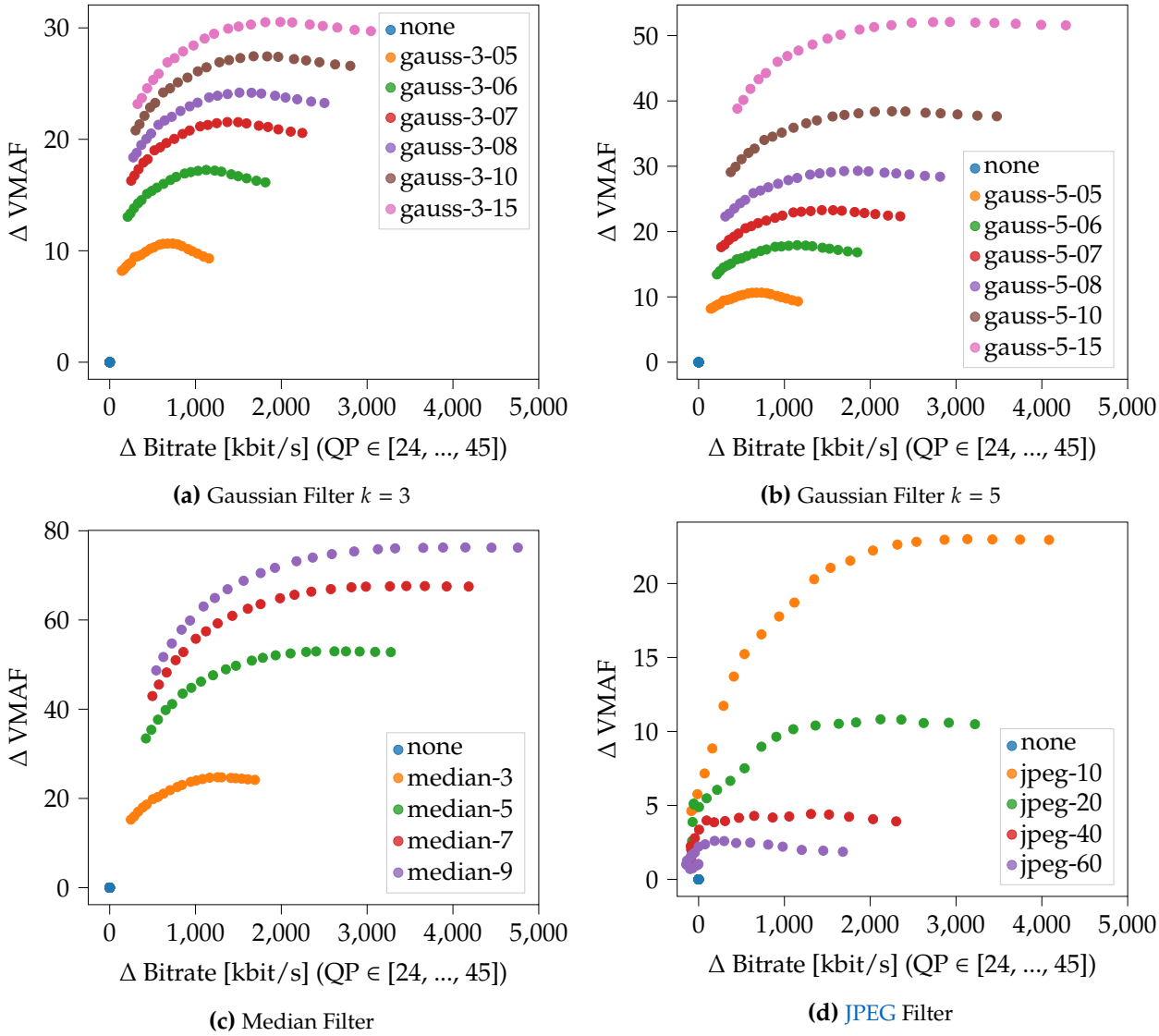
The CS curves describe an inverse trend compared to the RD curves. The higher the rate savings, the higher the quality difference compared to the baseline without any preprocessing applied. For the Gaussian filter with a kernel size of $k = 3$ and a low σ, the quality costs are especially high for QPs in the range of 30 to 40 compared to higher or lower QPs.

6.2.2.2 Bjøntegaard Delta Curves

We use the Bjøntegaard Delta Rate (BDR) [182] and the Bjøntegaard Delta (BD) of the VMAF score (BD-VMAF) to compare two RD-QP curves of two different preprocessing filters with the same QP. We use the Gaussian filter with a kernel size of $k = 3$ as the reference signal. Then, we calculate the BDR and BD-VMAF for every RD-QP curve and preprocessing filter. Figure 6.9 visualizes the resulting BDR and BD-VMAF scores per QP. We refer to these curves presented in Figure 6.9 as Bjøntegaard Delta (BD) curves. Every point on that curve represents either a BDR score in orange or a BD-VMAF score in blue, calculated between two RD-QP curves that contain all filter parameters at a certain QP.

The Gaussian filter with a kernel size of $k = 5$ performs slightly worse than for a kernel size of $k = 3$ in Figure 6.9a. The encoder requires 0.4 % to 1.3 % more bitrate to reach the same quality or reaches 0.7 to 0.2 lower VMAF scores for the same bitrate. Additionally, the BDR savings become smaller for increasing QPs.

For the median filter in Figure 6.9b, the encoder requires up to 22 % more bitrate for low QPs to reach the same visual quality. With increasing QPs, the BDR reduces right up to QP 44 and 45 where the median filter performs better than the Gaussian filter with $k = 3$ used as reference.



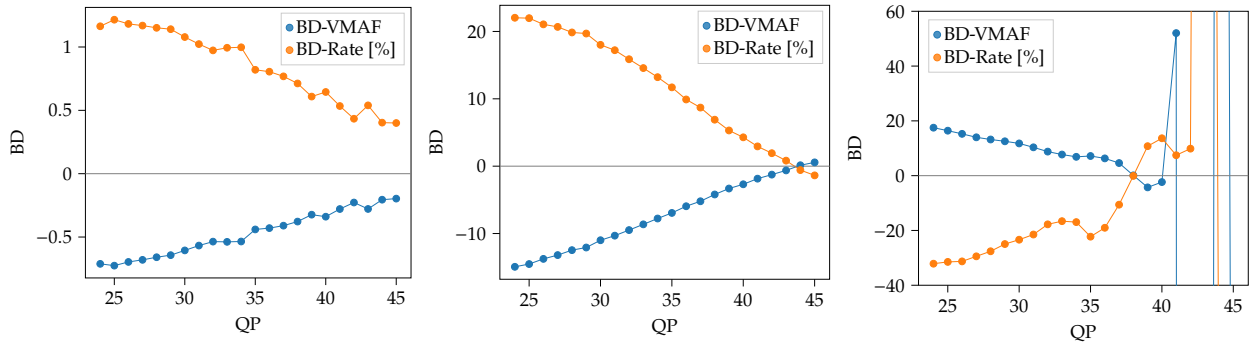
(a) Gaussian Filter $k = 3$

(b) Gaussian Filter $k = 5$

(c) Median Filter

(d) JPEG Filter

Figure 6.8 Quality costs C (ΔVMAF) over bitrate savings S (ΔBitrate) of the individual preprocessing filters compared to the unfiltered baseline. Adopted from [25] © 2022 IEEE.



(a) Gaussian Filter $k = 5$

(b) Median Filter

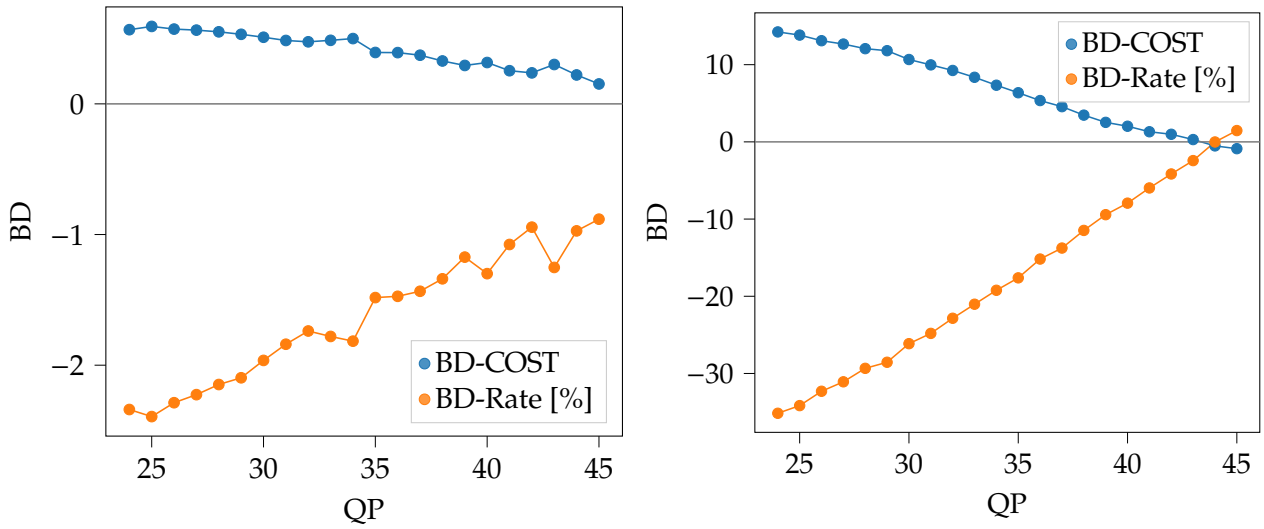
(c) JPEG Filter

Figure 6.9 Bjøntegaard Delta (BD) per QP between RD-QP curves of the Gaussian filter with $k = 3$ and the preprocessing filter introduced. Adopted from [25] © 2022 IEEE.

Lastly, the JPEG filter in Figure 6.9c shows a better performance for QPs lower than 38 compared to the Gaussian filter with $k = 3$. For QPs of 40 and higher, extensive jumps in the BD curves can

be observed. These jumps can be explained by the artifacts introduced by the JPEG preprocessor. Encoding the artifacts introduced by the JPEG preprocessor leads to inconsistent RD curves as shown in Figure 6.7d. Calculating the BDR and BD-VMAF from such inconsistent curves results in noticeably variations.

Similarly to the RD-QP curve comparison, we define the cost-saving per QP (CS-QP) curve as the CS curve for a preprocessing filter per QP. Then, we calculate the BDR and BD-VMAF scores between two CS-QP curves for the same QP and different preprocessing filters. Figure 6.10 shows the resulting BD curves for the Gaussian filter with $k = 5$ and the median filter.



(a) Gaussian Filter $k = 5$

(b) Median Filter

Figure 6.10 Bjøntegaard Delta (BD) per QP between the CS-QP curves of the Gaussian filter with $k = 3$ and the preprocessing filter stated below the figure.

Both BD curves use the Gaussian filter with a kernel size of $k = 3$ as a reference and show a similar trend as calculated from the RD-QP curves. The BD calculation for the JPEG preprocessor is not possible due to the artifacts introduced by the JPEG preprocessor. The coding artifacts introduced by the JPEG preprocessor influence the interpolation step of the BD calculation.

6.2.2.3 Mean Saving-Cost Ratio

This varying performance further highlights the motivation for our extensive analysis. Based on this analysis and the additional dimension introduced by the preprocessing filters, we propose a novel evaluation method named mean saving-cost ratio (MSCR). The MSCR allows to effectively compare the RD performance of different preprocessing algorithms and their parameters. We define the MSCR as the logarithmic mean ratio of maximum bitrate savings $S(q_p)$ over maximum quality cost $C(q_p)$ for all parameters of a preprocessing algorithm:

$$MSCR = \log_{10}\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\max(\{S_i(q_p) : q_p = 24,\ldots,45)\}}{\max(\{C_i(q_p) : q_p = 24,\ldots,45)\}}\right), \tag{6.4}$$

with i as the preprocessing parameter for a certain filter. A high MSCR value means high bitrate savings with low quality costs.

The MSCR together with the BD curve allows for effectively comparing the performance of different preprocessing filters. The MSCR provides a single score describing the trade-off of bitrate savings and quality costs. The BD curves enable us to analyze the BDR and BD-VMAF over certain QPs. These two metrics can be seen as a similar approach to performance curves such as the receiver operating characteristic (ROC) or Precision-Recall (PR) curve and their corresponding area under the

curve (AuC), which are widely used in the field of ML to compare the performance of different ML models [187, 188, 189]. Next, we use the MSCR proposed in this section for an extensive comparison of the different preprocessing filters for multiple video sequences.

6.2.3 Evaluation

In this section, we use the evaluation methods proposed in Section 6.2.2 to analyze the RD performance of different preprocessing filters. So far, we analyzed the *Bus* video sequence available at CIF resolution and encoded it at a GoP length of one while using the x264 video encoder. For an extensive evaluation, we use multiple video sequences showing different video content in terms of spatial and temporal complexity [142, 141]. For CIF resolution, we evaluate with four video sequences [186] for a GoP length of one and 20 using the x264 video encoder. Additionally, we use the NVIDIA HEVC (N-HEVC) [190] hardware video encoder with full high definition (HD) resolution and GoP length of one and 20 on three test video sequences from [191].

For both resolutions and video codecs, we use the mean bitrate and the mean VMAF scores of the respective video sequences. We calculate the MSCR for the two GoP lengths, resolutions, and codecs following the definition in Equation 6.4. Table 6.5 shows the resulting MSCR scores.

Table 6.5 Mean saving-cost ratio (MSCR) for the test video sequences at CIF and HD resolution and GoP lengths of 1 and 20, encoded with x264 or NVIDIA HEVC, respectively.

Filter	x264-CIF-1	x264-CIF-20	HEVC-HD-1	HEVC-HD-20
JPEG	**2.45**	-	**4.00**	1.36
Gauss-3	1.96	**1.41**	3.42	**3.02**
Gauss-5	1.93	1.36	3.37	2.97
Median	1.72	0.94	3.12	2.68

For a GoP length of one, the MSCR is highest for the JPEG preprocessor, followed by the Gaussian filter. We selected the JPEG preprocessor as a readily available implementation of a DCT representing a filter optimized for the HVS. In practice, the JPEG usually would not be considered as a preprocessor due to its higher computational complexity compared to the Gaussian and median filter. Additionally, the JPEG preprocessor shows unpredictable inconsistencies in the BD curve, according to Figure 6.9.

For a GoP length of 20, the JPEG preprocessor performs worst while the Gaussian filter achieves the highest MSCR. The low MSCR values of the JPEG preprocessor with a GoP length of 20 can be explained with the artifacts introduced by the JPEG preprocessor. These artifacts affect the motion estimation of the inter-frame coding enabled with a GoP length larger than one. The Gaussian filter and the median filter only blur the image content, which does not affect the inter-frame coding process. Here, the Gaussian filter with a kernel size of $k = 3$ achieves the highest MSCR and clearly outperforms the median filter. Hence, we select the Gaussian filter as the most suitable preprocessing algorithm for the quality filter introduced in Section 6.1.2. While the Gaussian filter has been a popular choice in the literature before, the proposed evaluation method offers a systematic method for quantifying the benefits of a chosen filter compared to other options.

6.3 Preprocessing Filter Bitrate Model

In the previous section, we systematically evaluated the influence of different preprocessing algorithms on the RD performance. Based on the results of this evaluation, the Gaussian low-pass filter achieved the best RD performance at a low computational complexity. Hence, we select the Gaussian low-pass filter as the quality filter for the framework introduced in Section 6.1.2. To finally use

the proposed preprocessing approach for rate control, we require a bitrate model that considers the influence of a Gaussian low-pass filter on the video bitrate.

In this section, we present two novel bitrate models that specifically incorporate the effect of the Gaussian low-pass filter on the video bitrate. For this, we model the bitrate as a function of the quantization parameter q_p, the video frame rate f, the video resolution r, the GoP length n, and the two Gaussian low-pass filter parameters: kernel size k and standard deviation σ. To the best of our knowledge, no other bitrate model available considers the influence of the Gaussian low-pass filter on the video bitrate.

Inspired by [141] and [142], we use the two content-dependent parameters of spatial activity (SA) and temporal activity (TA) to consider the influence of the video content. Both parameters SA and TA can be extracted from the uncompressed video sources, which is a suitable approach for the video encoding hardware available in vehicles. Before we present the implementation of both the analytical and the ML-based rate models, we summarize the two datasets created for developing the rate models.

6.3.1 Dataset Generation

First, we created two datasets for the training and validation of both bitrate models proposed in this thesis. The datasets represent the relationship of video bitrate and the respective parameters used for encoding the video sequence. We used both the H.264/Advanced Video Coding (AVC) [105] and the H.265/High Efficiency Video Coding (HEVC) [106] video codecs for the dataset generation.

The first dataset was created with the x264 [134] software video encoder on video sequences with CIF resolution of 352×288 pixels. For the second dataset, we used the NVIDIA HEVC [190] hardware video encoder on video sequences with HD resolution of 1920×1080 pixels. To achieve reliable results, we use representative uncompressed video sequences with different content properties for both datasets.

6.3.1.1 Video Content Measures

We quantify the video contents using the temporal activity (TA) and spatial activity (SA) as defined in [192]. Both activity measures are modified versions of the spatial perceptual information (SI) and temporal perceptual information (TI) such as defined by the International Telecommunication Union (ITU) [193]. The SA values indicate the amount of spatial detail in the video sequence. The SA value for a video sequence is defined as

$$SA = mean_{time}\{std_{space}\,[Sobel(F_n)]\}. \tag{6.5}$$

The luminance channel of a frame F_n is processed by the Sobel filter to determine the gradient at each pixel. Then, the standard deviation over all pixels is calculated. Finally, the mean of all standard deviations for all frames is used to calculate a single SA value. Video sequences with a high amount of spatial complexity lead to large SA values.

Similar to SA, the TA values indicate the amount of temporal change in the video sequence. The TA value is calculated by

$$TA = mean_{time}\{std_{space}\,[F_n - F_{n-1}]\}, \tag{6.6}$$

based on the difference of two consecutive frames F_n and F_{n-1}. Similar to SA, we calculate the standard deviation over all pixels of the frame difference. Then, the mean of all standard deviations for all frames is used to calculate a single TA value. A high TA value corresponds to a video sequence with a high amount of motion.

6.3.1.2 x264-CIF

We created the dataset *x264-CIF* from 30 videos in CIF resolution with different spatial and temporal characteristics [186, 191]. The videos of [186] are available in YUV format. The videos from [191] have to be converted from Y4M format into YUV format.

From the 30 video sequences, we use 22 video sequences for training, four for validation, and four for testing. Figure 6.11 visualizes the SA and TA properties of *x264-CIF* dataset as well as the distribution of training, validation, and test set.

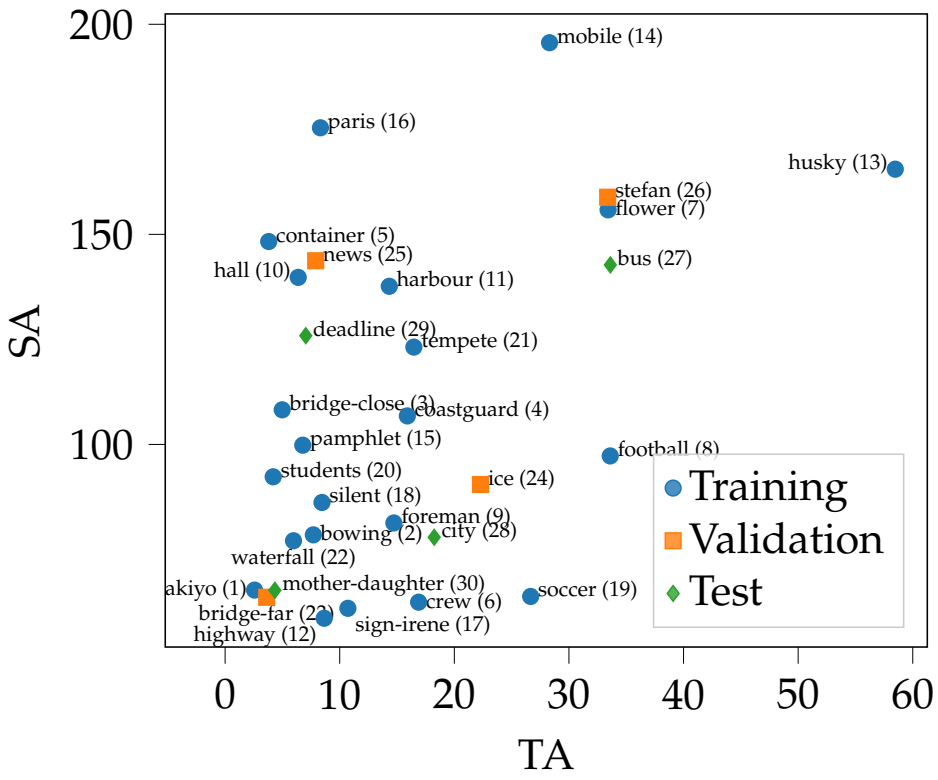


Figure 6.11 SA and TA values of the training set (●), validation set (■), and test set (♦) for *x264-CIF*. Adopted from [24] © 2022 IEEE.

For all videos, we create processed video sequences (PVSs) with QPs ranging from 24 to 45 at a step size of one, five different frame rates 30 fps, 15 fps, 10 fps, 5 fps, and 3 fps, nine different frame sizes ranging from the initial resolution CIF at 352×288 pixels down to quarter common intermediate format (QCIF) at 176×144 pixels with a step size of 22×18 pixels, with different GoP lengths $n \in \{1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60, 120\}$, and the parameters for the Gaussian low-pass filter: kernel size $k \in \{1, 3, 5, 7, 9\}$ and standard deviation $\sigma \in \{0.0, 0.5, 1.0, \ldots, 3.5\}$.

All PVSs are encoded with the H.264/AVC *High* profile using the x264 [134] software video encoder. Using all permutations of encoding parameters results in 554 400 dataset entries per video sequence. Given the 30 videos, the *x264-CIF* dataset consists of 16.6×10^6 samples.

6.3.1.3 NHEVC-HD

Similar to *x264-CIF*, we create the *N-HEVC-HD* dataset from 23 videos [191] in HD resolution. We again convert the video sequences available at [191] from Y4M format into YUV format.

From the 23 videos available, we select 17 videos for training, three for validation, and three for testing. Figure 6.12 presents SA and TA of the *N-HEVC-HD* dataset in the same way as for *x264-CIF*.

For all videos in HD resolution, we create different PVSs with encoding parameters similar to the ones used for *x264-CIF*: $q_p \in \{24, 29, 34, 40, 46\}$, $f \in \{30, 15, 3\}$, $r \in \{1920\times1080, 1600\times900, 1280\times720\}$,

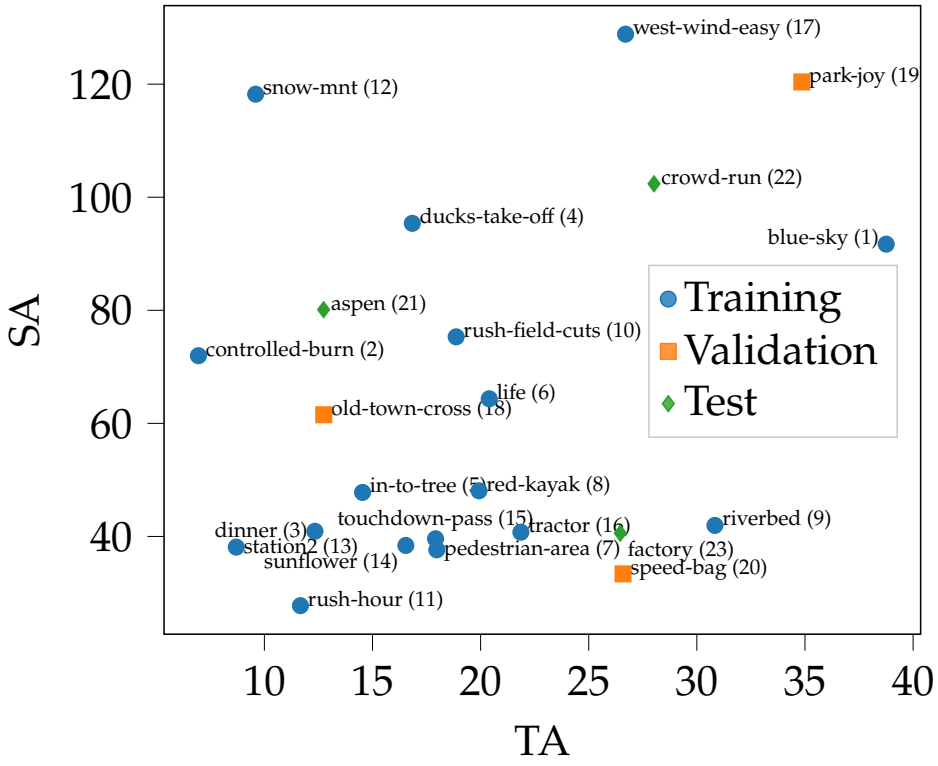


Figure 6.12 SA and TA values of the training set (•), validation set (■), and test set (♦) for *N-HEVC-HD*. Adopted from [24] © 2022 IEEE.

and $n \in \{1, 3, 6, 10, 20, 40, 120\}$. We select fewer parameter combinations compared to *x264-CIF* due to the higher encoding time of the HEVC and the larger HD resolution, which also increases the encoding time. For the Gaussian filter parameters, we use the same number of combinations as for *x264-CIF*, since incorporating the influence of the Gaussian filter is the main purpose of the dataset and has not been investigated before. The smaller *N-HEVC-HD* dataset is still sufficient for demonstrating that the proposed models can work with different video coding standards.

We encode all video sequences with the H.265/HEVC *Main* profile using the NVIDIA HEVC [190]. This results in 9450 dataset entries per video sequence and 217 350 entries in total for the *N-HEVC-HD* dataset.

6.3.2 Analytical Bitrate Model

Using the *x264-CIF* dataset created in Section 6.3.1.2, we develop an analytical model for estimating the video bitrate. The bitrate model design is inspired by the bitrate models of [141] and [142]. We model the video bitrate R as a function of the quantization parameter q_p, the video frame rate f, the video resolution r, the GoP length n, and the kernel size k and standard deviation σ of a Gaussian low-pass filter.

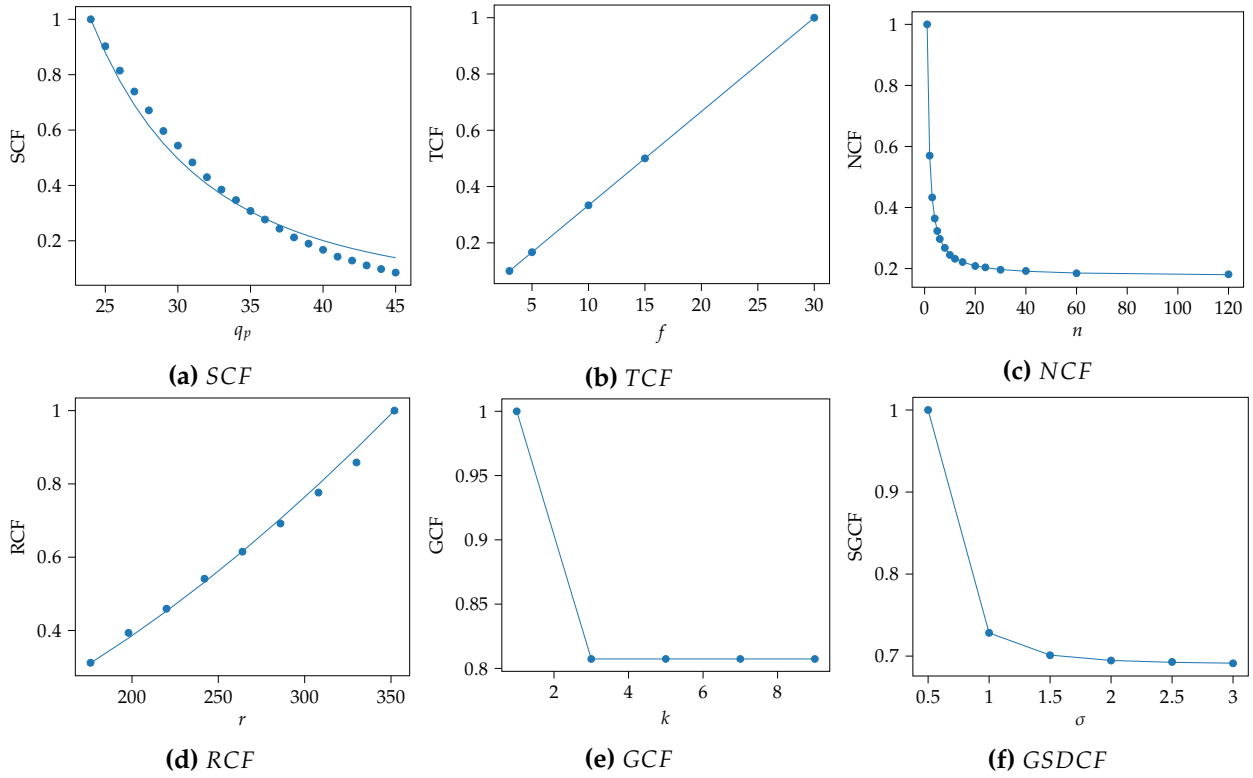
The influence of each parameter on the video bitrate is modeled as a separate correction factor. These correction factors cover the spatial correction factor $SCF(q_p)$, the temporal correction factor $TCF(f)$, the GoP length correction factor $NCF(n)$, the resolution correction factor $RCF(r)$, the Gaussian correction factors $GCF(k)$, and Gaussian standard deviation correction factors $GSDCF(\sigma)$. Additionally, we define $R_{max,I}$ as the maximum bitrate of the encoded video at the maximum frame rate f_{max}, the maximum resolution r_{max}, the minimum quantization parameter $q_{p,min}$, an I-frame only GoP structure with $n = 1$, and without Gaussian image filtering. If no Gaussian image filtering is used, then the kernel size can be considered as $k = 1$, independently of the value for σ. Thus, we

formulate the video bitrate R as the product of maximum bitrate $R_{max,I}$ and the separate correction factors:

$$R = R_{max,I} \cdot SCF(q_p) \cdot TCF(f) \cdot NCF(n) \cdot RCF(r) \cdot GCF(k) \cdot GSDCF(\sigma). \tag{6.7}$$

We follow the idea of [141, 142] and model the individual correction factors in Equation 6.7 using a two-step approach. In the first step, we model the correction factor based on the given PVSs. For this, we select a subset of all PVSs based on the correction factor that should be modeled. The subset selected from all PVSs contains all encoding parameters equal to their extrema such as defined by $R_{I,max}$, except for the dominating parameter of the correction factor that should be modeled. For the dominating parameter, the subset includes the full range of encoding parameters, since the dominating parameter defines the behavior of the respective correction factor. Then, we normalize the subset selected from all PVSs for the dominating extrema factor of the corresponding correction factor. For instance, f is the dominating factor of $TCF(f)$. In this case, we select all PVSs at r_{max} = CIF or HD, $q_{p,min} = 24$, $n = 1$, $k_{min} = 1$, and $\sigma_{min} = 0.5$, while all values of f are available in this subset. Then, we normalize this subset for $TCF(f)$ by f_{max}. Finally, we determine the model parameters using least square non-linear fitting (LSNF).

In the second step, we estimate the model parameters based on the standard video activity measures SA and TA using an iterative generalized linear regression method. Figure 6.13 gives an overview of all six correction factors for the exemplary video sequence *Foreman*.



(a) *SCF* **(b)** *TCF* **(c)** *NCF*

(d) *RCF* **(e)** *GCF* **(f)** *GSDCF*

Figure 6.13 Measured (dots) and estimated (line) correction factors for the exemplary video *Foreman* of the *x264-CIF* training set. Overall, the bitrate of *Foreman* could be modeled with a root mean square error (RMSE) of 66.06 kbit/s, a normalized RMSE of 0.0132, and a Pearson correlation (PC) of 0.9695 according to Equation 6.7. Adopted from [24] © 2022 IEEE.

The dots in Figure 6.13 represent the correction factors measured from PVSs and normalized for the dominating parameter. The line represents the correlation factors estimated by the model based on the two-step approach. We present the modeling process of the individual correction factors in detail for the *x264-CIF* dataset in the following sections.

6.3.2.1 Model Parameter Estimation

Before modeling the individual correction factors, we describe the stepwise feature selection approach as used in [141, 142]. This approach allows for estimating the content-dependent maximum video bitrate and the content-dependent model parameters for the individual correction factors based on the standard video activity measures SA and TA.

Similar to [141, 142], we use the iterative generalized linear regression method (GLM) proposed by McCullagh and Nelder [194] for predicting the content-dependent model parameters. We use the cross-validation error (CVE) as a performance measure for the selection and combination of the most suitable features. Here, the CVE is calculated as the RMSE of the measured and predicted value. For the parameter estimation, we use the individual features SA and TA, the interaction terms of the features $SA \cdot TA$, $\frac{SA}{TA}$, $\frac{TA}{SA}$, and the logarithm of the individual features as well as the interaction terms $\log(SA)$, $\log(TA)$, $\log(SA \cdot TA)$, $\log\left(\frac{SA}{TA}\right)$.

Then, the GLM for modeling a value y using the features $f(x)$ with N different properties can be written as

$$y = \sum_{i=1}^{N} \sum_{j=1}^{N} a_i \cdot f(x_i) + b_{i,j} \cdot f(x_i, x_j) + a_0. \tag{6.8}$$

The weights of both the single features a_i and the interaction terms $b_{i,j}$ are calculated with LSNF. For every iteration, a new feature is added to the GLM and the feature that offers the lowest CVE is selected. This process of adding features is repeated until the CVE does not further improve.

To obtain reliable results, we use the training set of the *x264-CIF* dataset introduced in Section 6.3.1.2 to train the models. The generalization capability of the models is then verified with the test set. Further, we use leave-one-out cross-validation (LOOCV) for training the weights and calculating the CVE. Using LOOCV avoids overfitting on the given training data and allows for developing more robust and generic models.

The iterative GLM introduced previously is a generic method for estimating the content-dependent model parameters based on the video activities SA and TA. In the following sections, we present how we apply the iterative GLM to select the best-matching feature combinations for the PVSs of the training set.

6.3.2.2 Maximum Bitrate Estimation

We start with the feature selection based on SA and TA to model $R_{max,I}$ using the iterative GLM approach introduced in Section 6.3.2.1 on the training set. For every feature, the CVE is calculated and the best feature or combination of features offering the lowest CVE is selected. For the first iteration, the feature SA offers the lowest CVE. We repeat this process two times while always selecting the feature with the lowest CVE. We stop after iteration three, since the CVE cannot be further reduced by adding more features. The resulting SA and TA dependent linear combination $R_{max,I,st}$ for the maximum bitrate $R_{max,I}$ can be written as

$$R_{max,I,st} = \rho_0 \cdot SA + \rho_1 \cdot \frac{SA}{TA} + \rho_2 \cdot SA \cdot TA + \rho_3, \tag{6.9}$$

with the model parameters $\rho_0 = 38.0332\,\text{kbit/s}$, $\rho_1 = -61.2067\,\text{kbit/s}$, $\rho_2 = 0.3884\,\text{kbit/s}$, and $\rho_3 = 1.8015 \times 10^3\,\text{kbit/s}$.

We evaluate the performance of the model by calculating the RMSE and the PC between the measured and expected values for $R_{max,I}$ for the training set. The model achieves an RMSE of 1319.3 kbit/s and a PC of 0.8641 for the training set. The values of $R_{max,I}$ for the *x264-CIF* dataset range from 0.511 kbit/s to 11.7 Mbit/s.

6.3.2.3 Spatial Correction Factor

We model the spatial correction factor SCF to describe the influence of the quantization parameter q_p on the video bitrate. We use the subset of PVSs with a bitrate normalized by $R_{max,I}$ and a varying q_p such as introduced in Section 6.3.2. Figure 6.13a visualizes the SCF over the q_p for the exemplary video sequence *Foreman*. Similar to [141, 142], we model SCF as an inverse power function

$$SCF(q_p, q_{p,min}) = \left(\frac{q_p}{q_{p,min}}\right)^{-a}, \tag{6.10}$$

with the content-dependent model parameter a. The content-dependent model parameter a specifies the decreasing rate of SCF for increasing QPs. We determine a for every video sequence in the training set by LSNF with an RMSE of 0.0397 and a PC of 0.9941 between estimated and measured SCF.

The estimation of a_{st} based on SA and TA follows the same iterative GLM procedure as for $R_{max,I,st}$. The model achieves the lowest CVE of 0.2704 after two iterations. Adding more features to the GLM does not reduce the CVE any further. The resulting model can be written as

$$a_{st} = \alpha_0 \cdot SA + \alpha_1 \cdot SA \cdot TA + \alpha_2, \tag{6.11}$$

with the weights $\alpha_0 = -4.9626 \times 10^{-3}$, $\alpha_1 = -7.0579 \times 10^{-5}$, and $\alpha_2 = 3.5886$.

We again calculate the RMSE and PC between estimated and measured SCF. This time we use the content-dependent parameter a_{st} to estimate the SCF. The results show an RMSE of 0.0482 and a PC of 0.9887.

6.3.2.4 Temporal Correction Factor

We model the influence of the video frame rate f on the bitrate using the temporal correction factor TCF. First, the PVS subset with varying f is normalized by $R_{max,I}$ of the individual video sequences. Figure 6.13b shows the measured and estimated TCF for the exemplary *Foreman* video sequence. We model TCF as a power function, similar to [141, 142]:

$$TCF(f, f_{max}) = \left(\frac{f}{f_{max}}\right)^{b}. \tag{6.12}$$

The content-dependent parameter b indicates the slope of TCF for increasing frame rates. We calculate b between measured and estimated TCF using LSNF with an RMSE of 0.0 and a PC of 1.0. It should be noted that the RMSE and PC for TCF are not perfect matches, but rounded based on the floating point accuracy.

Similar to [142], we set $b_{st} = 1$, since b is almost one for every video. The performance b_{st} could not be further improved by adding more features to the GLM. The performance of TCF_{st} results in an RMSE of 1×10^{-6} and a PC of 1.0.

6.3.2.5 GoP Length Correction Factor

The influence of the GoP length n is modeled with the GoP correction factor NCF. For this, we use the PVS subset with varying n, normalized by $R_{max,I}$. Figure 6.13c shows NCF over n for the *Foreman* video sequence. Similar to [142], we model the factor as an inverse power function

$$NCF(n) = c \cdot \left(\frac{1}{n}\right)^{d} + e, \tag{6.13}$$

with the content-dependent parameters c, d, and e. The model performance for measured and estimated NCF using LSNF is an RMSE of 0.0013 and a PC of 0.9999.

We model the content-dependent parameters c_{st}, d_{st}, and e_{st} based on SA and TA using the iterative GLM on the training set. For c_{st}, we stop after two iterations with the lowest CVE of 0.0660. The resulting model can be written as

$$c_{st} = \gamma_0 \cdot \log(TA) + \gamma_1 \cdot TA + \gamma_2, \tag{6.14}$$

with $\gamma_0 = -0.1824$, $\gamma_1 = -0.0034$, and $\gamma_2 = -0.0034$.

d_{st} shows the lowest CVE 0.0457 after one iteration of the GLM. The model for d_{st} can be written as

$$d_{st} = \delta_0 \cdot \frac{TA}{SA} + \delta_1, \tag{6.15}$$

with $\delta_0 = 0.1011$ and $\delta_1 = 1.0240$.

Finally, e_{st} achieves the lowest CVE of 0.0457 after two iterations. The resulting content-dependent parameter model e_{st} is

$$e_{st} = \epsilon_0 \cdot \log(TA) + \epsilon_1 \cdot TA + \epsilon_2. \tag{6.16}$$

with $\epsilon_0 = 0.1813$, $\epsilon_1 = 0.0034$, and $\epsilon_2 = -0.0328$.

The overall performance of NCF_{st} using the content-dependent parameters c_{st}, d_{st}, and e_{st} leads to an RMSE of 0.0588 and a PC of 0.9641.

Lottermann et al. [142] also model the influence of the GoP structure. This structure includes the distribution of P-frames and B-frames within the GoP. Since the proposed approach is designed for live streaming, we do not consider using B-frames to avoid additional delays. Hence, we do not model the influence of the GoP structure, assuming only I-frames and P-frames in an IPPP structure.

6.3.2.6 Resolution Correction Factor

The correction factor models presented so far were inspired by [141, 142]. The following correction factors are novel models specifically designed for the requirements of the proposed preprocessing filter concept.

The resolution correction factor RCF specifies the influence of the frame resolution r on the video bitrate. The frame resolution is defined by the width and the height of a frame. In the following, we only use the word *width* to refer to the frame resolution, since width and height are reduced in the same way when downscaling a frame to maintain the aspect ratio. We select the PVS subset with variable r and normalize it by $R_{max,I}$. Then, we visualize the RCF for the *Foreman* sequence in Figure 6.13d. The RCF is zero for a resolution of zero and converges to one for the maximum resolution r_{max}. This behavior is similar to the TCF. Therefore, we model the RCF as a power function

$$RCF(r, r_{max}) = \left(\frac{r}{r_{max}}\right)^j, \tag{6.17}$$

with the content-dependent parameter j. Similar to the TCF and the content-dependent parameter a, j indicates how fast the RCF rises for increasing frame resolutions. Using LSNF, the model achieves an RMSE of 0.0146 and a PC of 0.9982.

Next, we model j_{st} based on SA and TA using the iterative GLM. We stop after the first iteration with the lowest CVE of 0.2135. The resulting linear combination for j_{st} can be written as

$$j_{st} = \varphi_0 \cdot SA \cdot TA + \varphi_1. \tag{6.18}$$

The model parameters depending on SA and TA are $\varphi_0 = 3.4594 \times 10^{-5}$ and $\varphi_1 = 1.5879$. These parameters lead to a performance for the model RCF_{st} with an RMSE of 0.0384 and a PC of 0.9848.

6.3.2.7 Gaussian Kernel Correction Factor

To model the influence of the Gaussian low-pass filter, we start with the kernel size k. We use the PVS subset normalized by $R_{max,I}$ and a variable k for creating the Gaussian correction factor GCF. Figure 6.13e presents the GCF over k for the exemplary video *Foreman*. Similar to the NCF, we model the GCF as an inverse power function

$$GCF(k, k_{min}) = l \cdot \left(\frac{k_{min}}{k}\right)^{m} + o, \tag{6.19}$$

with the content-dependent parameter l, m, and o. The parameters describe how fast the GCF is decreasing for an increasing kernel size. We determine the content-dependent parameters by LSNF with an RMSE of 0.0 and a PC of 1.0. The accuracy for RMSE and PC is four decimal points.

Next, we use the GLM to select the features for the content-dependent parameter l_{st} based on SA and TA. We stop the GLM after the first iteration with the lowest CVE of 0.0267. The model obtained for l_{st} can be written as

$$l_{st} = \lambda_0 \cdot SA + \lambda_1. \tag{6.20}$$

The weight factors of l_{st} are $\lambda_0 = -9.2718 \times 10^{-5}$ and $\lambda_1 = 0.1759$.

Similarly to l_{st}, we use the GLM for the feature selection of m_{st} and o_{st}. However, both parameters show the best performance for constant values of $m_{st} = 15.7108$ and $o_{st} = 0.8342$. Thus, we use the constant values for m_{st} and o_{st}, since none of the features added could improve the performance.

We calculate the overall performance for the measured and estimated GCF_{st}. The results show an RMSE of 0.0282 and a PC of 0.9202.

6.3.2.8 Gaussian Standard Deviation Correction Factor

As the second influencing factor of the Gaussian low-pass filter, we use its standard deviation σ. We model the Gaussian standard deviation correction factor $GSDCF$ from the PVS subset normalized by $R_{max,I}$ with a variable σ. The $GSDCF$ over σ for the *Foreman* video sequence is visualized in Figure 6.13f. With a similar behavior as for NCF and GCF, we model the $GSDCF$ as an inverse power function:

$$GSDCF(\sigma, \sigma_{min}) = t \cdot \left(\frac{\sigma_{min}}{\sigma}\right)^{u} + v, \tag{6.21}$$

with the content-dependent parameters t, u, and v. The content-dependent parameters describe the decreasing rate of the $GSDCF$ for an increasing σ. We determine the parameters using LSNF, achieving an RMSE of 0.0006 and a PC of 0.9999 between measured and estimated $GSDCF$.

We use the iterative GLM to model the content-dependent parameters based on SA and TA. We stop the GLM after one iteration, as the performance could not be further improved by adding more features. t_{st} achieves the lowest CVE of 0.0370, u_{st} of 0.5147, and o_{st} of 0.0373. The resulting models for the content-dependent parameters are

$$t_{st} = \tau_0 \cdot \log(SA \cdot TA) + \tau_1, \tag{6.22}$$

$$u_{st} = v_0 \cdot SA \cdot TA + v_1, \tag{6.23}$$

$$v_{st} = \nu_0 \cdot \log(SA \cdot TA) + \nu_1. \tag{6.24}$$

The model performance between measured and estimated $GSDCF_{st}$ using the model parameters estimated for t_{st}, u_{st}, and v_{st} shows an RMSE of 0.0377 and a PC of 0.9444.

With the $GSDCF$ as the last parameter of Equation 6.7, we are now able to estimate the video bitrate R based on given encoding parameters and the video complexity measures SA and TA using an analytical bitrate model. Next, we present an ML-based approach to estimate the video bitrate.

6.3.3 Machine Learning-Based Bitrate Model

In addition to developing an analytical model, we also investigate a data-driven approach for estimating the video bitrate R. We propose an ML-based approach where we approach the bitrate estimation as a regression task. We design a multi-layer perceptron consisting of five fully connected layers (FCLs) in a triangular shape as shown in Figure 6.14.

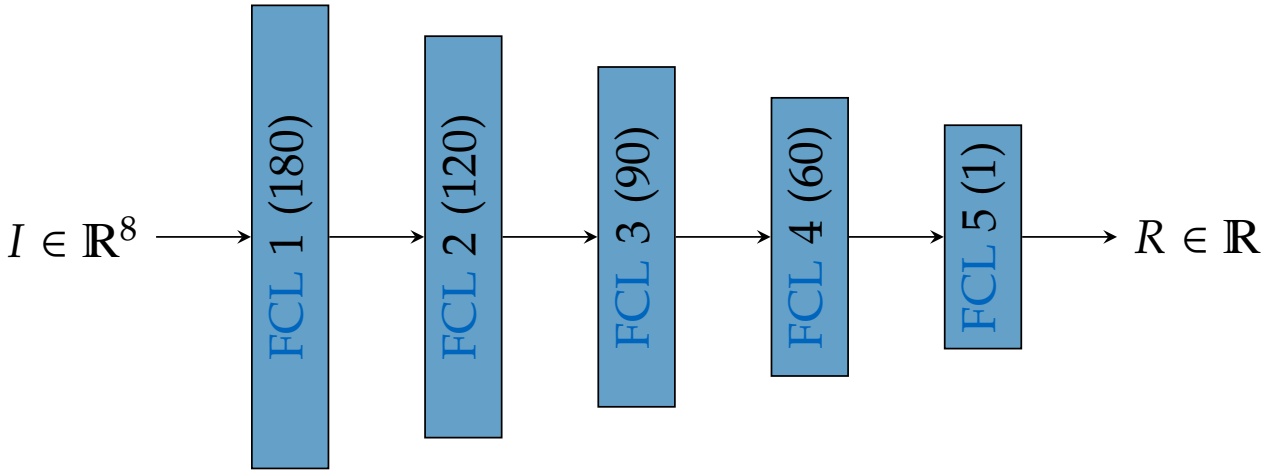


Figure 6.14 The schematic architecture of the proposed bitrate prediction model. The multi-layer perceptron consists of five fully connected layer (FCL) in triangular shape to predict the bitrate R from the input $I = \{SA, TA, q_p, f, n, r, k, \sigma\} \in \mathbb{R}^8$. Adopted from [24] © 2022 IEEE.

The FCLs consist of 180, 120, 90, 60, and 1 neurons. We use Rectified Linear Units (ReLUs) as activation functions, except for the last layer. Adding further layers or increasing the size of the existing layers did not improve the performance. The decreasing number of neurons in the triangular shape reduced the complexity of the model while not affecting the accuracy. For any input I, the model predicts the resulting video bitrate R, with the input

$$I = \{SA, TA, q_p, f, n, r, k, \sigma\} \in \mathbb{R}^8. \tag{6.25}$$

For the training, we use the Adam optimizer [195] with a learning rate of 0.0001. The Adam optimizer has shown superior results in our experiments compared to the stochastic gradient descent (SGD) optimizer. We train with a batch size of 256 for 200 epochs with a reduce-on-plateau strategy. We multiply the learning rate with a factor of 0.5 if the training loss does not change for ten epochs. Additionally, we store the best model after every epoch. The learning rate and batch size values used for the training process were determined empirically using the validation set.

The trained ML-based model as well as the analytical model of Equation 6.7 are two novel solutions for estimating the video bitrate using encoding parameters and video activity measures. Next, we evaluate the performance of both models on the test set and compare them to three state-of-the-art approaches.

6.3.4 Results

In this section, we present the experiments to evaluate the performance of the proposed bitrate models. First, we analyze the performance of the analytical and ML-based model in detail on both datasets. Then, we compare both proposed bitrate models to three state-of-the-art bitrate models. Finally, we discuss the analytical and ML-based model considering deployment aspects such as hardware limitations and the requirements of embedded systems.

6.3.4.1 Bitrate Model Comparison

In Section 6.3.2, we already demonstrated the performance of the individual correction factors and content-dependent parameters on the *x264-CIF* training set. Here, we further analyze the performance of the correction factors and compare the overall performance of both proposed bitrate models, analytical and ML-based, on the test sets of both datasets, *x264-CIF* and *N-HEVC-HD*.

6.3.4.1.1 x264-CIF We start with the performance of the individual correction factors for the analytical model on the *x264-CIF* dataset. Table 6.6 highlights the RMSE and PC of all correction factors for the training set and the test set.

Table 6.6 Performance of the SA- and TA-dependent correction factors SCF_{st}, TCF_{st}, NCF_{st}, RCF_{st}, GCF_{st}, $GSDCF_{st}$ and the maximum bitrate $R_{max,I,st}$ for *x264-CIF*. $R_{max,I}$ ranges from 0.511 kbit/s to 11.7 Mbit/s.

Factor	Training Set		Test Set	
	RMSE	PC	RMSE	PC
$R_{max,I,st}$	1319.3082 kbit/s	0.8641	1106.06 kbit/s	0.8468
SCF_{st}	0.0482	0.9887	0.0460	0.9910
TCF_{st}	1×10^{-6}	1.0000	1×10^{-7}	1.0000
NCF_{st}	0.0588	0.9641	0.0721	0.9799
RCF_{st}	0.0384	0.9848	0.0367	0.9879
GCF_{st}	0.0282	0.9202	0.0180	0.9646
$GSDCF_{st}$	0.0377	0.9444	0.0366	0.9525

All models designed for the individual correction factors achieve a performance on the test set comparable to the performance on the training set. It should be noted that the PC for TCF_{st} is not exactly one, but rounded to four decimal points.

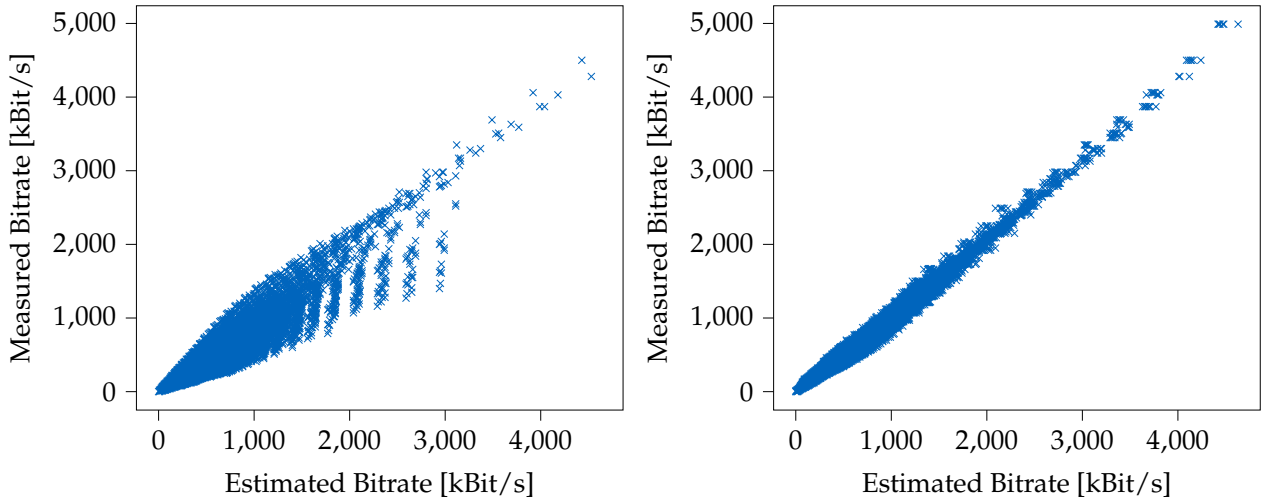
Next, we calculate the overall performance for estimating the video bitrate of both the analytical and ML-based bitrate model. Table 6.7 lists the normalized RMSE and PC between the estimated and measured bitrates on the training set, the test set, and for all videos.

Table 6.7 Performance results of the proposed analytical and ML-based models for *x264-CIF*.

Model	Training Set		Test Set		All Videos	
	RMSE	PC	RMSE	PC	RMSE	PC
Analytical	0.0179	0.9627	0.0186	0.9564	0.0180	0.9617
ML	**0.0111**	**0.9723**	**0.0078**	**0.9813**	**0.0106**	**0.9721**

For the test set, the analytical model shows an average performance with an RMSE of 1.86 % and a PC of 0.9564. The ML model outperforms the analytical model for the RMSE in all cases by a delta of 0.7 % on average. With an RMSE of 0.78 % on the test set, the ML model reduces the error by more than half relative to the analytical model.

To visualize the overall performance of both bitrate models proposed in this chapter, we use both models to estimate the bitrate for all encoding parameters on the *Foreman* video sequence. Figure 6.15 shows the resulting bitrates estimated by the analytical model in Figure 6.15a and the ML-based model in Figure 6.15b.



(a) Analytical Model **(b)** ML-Based Model

Figure 6.15 Performance evaluation of the analytical rate model and the ML-based rate model for the video *Foreman* of *x264-CIF*. The analytical model on the left achieves an RMSE of 66.06 kbit/s and a PC of 0.9695. The ML-based rate model on the right achieves an RMSE of 23.03 kbit/s and a PC of 0.9933. Adopted from [24] © 2022 IEEE.

We visualize the bitrate estimated by the proposed models over the bitrate measured from encoding the video sequences. A close distance to the angle bisector represents a good performance. By inspection, the ML-based model performs superior to the analytical bitrate model.

6.3.4.1.2 N-HEVC-HD Similar to the *x264-CIF* dataset, we present the performance results for both of the bitrate models on the *N-HEVC-HD* dataset. Table 6.8 summarizes the RMSE and PC on the training set, the test set, and for all videos of *N-HEVC-HD*.

Table 6.8 Performance results of the proposed analytical and ML-based models for *N-HEVC-HD*.

Model	Training Set		Test Set		All Videos	
	RMSE	PC	RMSE	PC	RMSE	PC
Analytical-CIF	0.0900	0.9527	0.0935	0.9649	0.0906	0.9545
Analytical-HD	0.0483	0.9536	0.0444	**0.9673**	0.0477	**0.9557**
ML-CIF	0.0512	0.7951	0.0613	0.8985	0.0528	0.8152
ML-HD	**0.0244**	**0.9851**	0.0929	0.8035	0.0424	0.9411
ML-HD-D	0.0415	0.9335	**0.0433**	0.8666	**0.0418**	0.9187

Unlike the evaluation on the *x264-CIF* dataset, we evaluated multiple versions of the analytical and ML-based model on the *N-HEVC-HD* dataset. First, we directly evaluate the analytical model *Analytical-CIF*, which was trained on the *x264-CIF* dataset. *Analytical-CIF* reaches an RMSE of 0.0935 and a PC of 0.9649 on the test set of the *N-HEVC-HD* dataset. Using the training data of the *N-HEVC-HD* dataset, we retrain the analytical model *Analytical-HD*. The *Analytical-HD* model retrained for HD resolution reduces the RMSE to 0.0444. The PC for *Analytical-HD* increases to 0.9673, which is the best result on the *N-HEVC-HD* test set. This demonstrates that the proposed analytical bitrate model generalizes well and allows for a fast adaptation to new training data. Further improvements are possible by repeating the GLM process introduced in Section 6.3.2.1 to model the content-dependent parameters specifically for the new data.

Similar to the analytical model, we start with the ML-based model *ML-CIF*, which was trained on the *x264-CIF* dataset. Then, we evaluate the *ML-CIF* model on the *N-HEVC-HD* dataset. Next, we use the same architecture of *ML-CIF* for training the new model *ML-HD* for 400 epochs on the *N-HEVC-HD* dataset. Since the *N-HEVC-HD* dataset contains significantly less data compared to the *x264-CIF* dataset, we observe a strong overfitting of the model. We address the overfitting on the fewer training data by adding a dropout layer after every FCL. We empirically determine that a large dropout rate of 0.75 reaches the best performance. The resulting model *ML-HD-D* that includes the dropout layers reaches an RMSE of 0.0433 and a PC of 0.8666 on the *N-HEVC-HD* test set. This demonstrates that the ML-based approach is also able to generalize to the new data and still outperforms the analytical model regarding the RMSE.

In general, both approaches perform worse on the *N-HEVC-HD* dataset compared to the *x264-CIF* dataset. A possible explanation for this is the significantly smaller size of the *N-HEVC-HD* dataset compared to the *x264-CIF* dataset. The *N-HEVC-HD* dataset contains 98.7 % fewer data samples compared to the *x264-CIF* dataset, which is in particular critical for the ML-based approach. Since the *N-HEVC-HD* dataset is only used for demonstrating that the proposed models can work with different video coding standards and resolutions, this issue could be easily addressed by creating a larger dataset with more samples. Next, we compare both proposed models to similar state-of-the-art approaches.

6.3.4.2 State-of-the-Art Bitrate Model Comparison

Here, we evaluate the proposed models and three comparable bitrate models that can be considered as state of the art. We compare the spatio-temporal rate model (STRM) [141], which we refer to as *STRM*, and its extension [142], which we refer to as *STRM-GOP*. *STRM* and *STRM-GOP* show state-of-the-art performance for rate estimations based solely on encoding parameters and video characteristics. The estimation based on these parameters makes both models independent of the underlying video codec. Additionally, we compare our models to the bitrate model of Ma et al. [138], which we refer to as *MaRM*. *MaRM* estimates the bitrate based on encoding parameters and three content-dependent features. The content-dependent features are based on motion vector (MV) information of the underlying codec. This dependency on the underlying codec makes the model less suitable for hardware encoders with only limited access to the underlying codec.

Since all models are designed for the x264 [134] software video encoder and video sequences at CIF resolution, we only evaluate for the *x264-CIF* dataset. Additionally, the three state-of-the-art models do not support all encoding parameters introduced in Section 6.3.2. Hence, we compare all models on two subsets with fewer parameters as well as on the full *x264-CIF* dataset. The two subsets are defined by the correction factors which are supported by the state-of-the-art approaches. *MaRM* [138] and *STRM* [141] only consider the spatial correction factor SCF and the temporal correction factor TCF. *STRM-GOP* [142] also considers the GoP length correction factor NCF in addition to these two correction factors. We compare all models on all different dataset configurations to ensure a fair comparison.

MaRM [138], *STRM* [141], and *STRM-GOP* [142] were designed with smaller and less diverse datasets of five to seven videos. Therefore, we evaluate both models twice, in two different versions. First, we use the parameters such as provided in the original paper as a reference. For a second version, we use the parameters from finetuning the models on the larger *x264-CIF* dataset introduced in Section 6.3.1.2. The models finetuned on the *x264-CIF* dataset are referred to as *MaRM**, *STRM** and *STRM-GOP**. Table 6.9 summarizes the results for the proposed models and the state-of-the-art models, first with the parameters as provided in the papers [138, 141, 142] and a second time finetuned on the *x264-CIF* training set.

STRM, *STRM-GOP*, and *MaRM* show a poor performance with RMSEs of 43.1 %, 4724 %, and 35.7 %, respectively. In particular *STRM-GOP* exhibits an RMSE several orders of magnitudes larger than the other methods. This can be explained by the smaller datasets of five to seven videos used for training those models compared to our training dataset with 22 videos. Since the NCF used in

Table 6.9 Performance results of the proposed models in comparison to the state-of-the-art models on the *x264-CIF* test set. The state-of-the-art models were evaluated twice, with and without finetuning on the *x264-CIF* training dataset. The models finetuned on the *x264-CIF* training set are indicated with ⋆.

Features	SCF, TCF		SCF, TCF, NFC		All	
Model	RMSE	PC	RMSE	PC	RMSE	PC
Analytical	0.0565	0.9951	0.0333	0.9788	0.0186	0.9564
ML	**0.0510**	0.9787	**0.0178**	0.9776	**0.0078**	**0.9813**
STRM [141]	0.4311	−0.3404	0.4362	−0.2502	0.4486	−0.2031
STRM⋆ [141]	0.0810	0.9951	0.2351	0.6784	0.2713	0.5870
GOP-STRM [142]	109.9401	−0.1833	47.2404	−0.0259	47.2498	−0.0034
GOP-STRM⋆ [142]	0.0621	0.9955	0.0302	0.9812	0.0760	0.8628
MaRM [138]	0.3570	−0.8584	0.3199	−0.8462	0.2142	−0.7149
MaRM⋆ [138]	0.1073	**0.9976**	0.1454	**0.9855**	0.2602	0.8306

STRM-GOP is an inverse power function, the error grows exponentially for inaccurate parameters. However, after finetuning all models with the *x264-CIF* training set, they achieve a good performance again with an RMSE of 8.1 % for *STRM*⋆, an RMSE of 3.0 % for *STRM-GOP*⋆, and an RMSE of 10.5 % for *MaRM* on their operational design domains. After finetuning, *MaRM* reaches the best PC with 0.9976. This validates the approach of using analytical models, since they can be easily finetuned with additional training data.

For the subsets SCF, TCF and SCF, TCF, NFC, the *MaRM*⋆ reaches comparable or the best results regarding the PC. In terms of RMSE, the proposed analytical and ML models perform best on the subsets SCF, TCF and SCF, TCF, NFC, respectively. A possible explanation for the superior performance of the analytical model compared to the ML model for the subset SCF, TCF is the higher flexibility due to the individual correction factors. The ML model is trained for the entire dataset including all variations of parameters, while the analytical model only considers the contribution of the supported correction factors. The correction factors that do not contribute in the reduced dataset result in a multiplication with one for the analytical model.

However, on the full *x264-CIF* dataset including variable frame sizes and the Gaussian low-pass filter, the ML model performs best with the lowest RMSE of 0.78 % and the highest PC of 0.9813. This demonstrates the potential of using neural networks for this task. While our analytical model still reaches a good performance with an RMSE of 1.86 %, the finetuned state-of-the-art models cannot reach a comparable performance with RMSEs of 7.6 % for *STRM-GOP*⋆, 27.1 % for *STRM*⋆, and 26.0 % for *MaRM*⋆. This can be explained by the fact that these models were not designed and trained for the consideration of these influence factors. Especially the influence of the Gaussian low-pass filter is novel for a bitrate model, with the idea of using the low-pass filter for video adaptation first being proposed in this chapter. Interestingly, the RMSE of *MaRM* became worse after finetuning. A possible explanation for this is that the finetuning only improves the supported correction factors SCF and TCF, which can still reduce the overall performance on the entire dataset.

In summary, both proposed bitrate models outperform the state of the art by achieving an RMSE that is 22 % lower for the subset SCF, TCF and 70 % lower on the subset SCF, TCF, NFC. Further, no existing state-of-the-art model was able to estimate the influence of the Gaussian low-pass filter on the video bitrate. The proposed bitrate models are the first to address this factor.

6.3.4.3 Analytical Model vs. Machine Learning Model

In the previous sections, the ML model clearly outperforms the analytical model in terms of accuracy. Here, we compare both proposed bitrate models in practical streaming systems. For this comparison, we consider the aspects of hardware limitations and the requirements of embedded systems.

While both models show a good accuracy compared to the state of the art, the ML model outperforms the analytical model for the *x264-CIF* dataset by 138 % RMSE on the test set. However, the analytical model requires only a lightweight training and the GLM repeated on the individual content-dependent parameters when changing the dataset or the video encoder. On the other hand, the ML model needs to be retrained from scratch for new data.

The size and the runtime of a specific model are always critical requirements on embedded systems. With only 35 parameters and an evaluation time of 8.75×10^{-6} s per prediction, the analytical model is faster than the ML model and computationally less complex. On the other hand, the ML model can benefit from hardware optimization commonly used for ML applications. Additionally, the ML proposed in this thesis is still a lightweight model with 32 641 parameters and a prediction time of 14.78×10^{-6} s per evaluation. To evaluate the ML model, we used an NVIDIA 1050Ti Graphics Processing Unit (GPU) and almost matched the execution time of the analytical model. This execution time comparable to the analytical model shows that with suitable hardware, the runtime of the ML model is not an issue.

Finally, the analytical model can be solved for any variable in Equation 6.7. Due to the clear definition, the analytical model is more intuitive to understand and easier to debug. The multilayer perceptron used for the ML model is a black box, which makes the identification of potential errors more challenging. Additionally, a separate ML model needs to be trained for predicting different variables.

Based on the previous discussion, both models can be recommended for estimating the video bitrate. If the best possible performance should be achieved, the ML model is preferred. The analytical model should be considered on embedded systems with high computational constraints. Next, we use the two bitrate models presented in this section to estimate the parameters required for controlling the individual preprocessing filters.

6.4 Preprocessor Rate Control

The preprocessing filter concept proposed in Section 6.1 allows for an individual adaptation of temporal and spatial resolution as well as the rate/quality of multiple segments in a SFV with only a single encoder available. Since this approach should be used in the same way as if there were multiple encoders available, the parameters required for the preprocessing filters need to be estimated from the encoding parameters used to control the individual video encoders. In Section 6.1, we already demonstrated that the temporal and spatial filter can directly use the given encoding parameters such as estimated by the TAMVA scheme introduced in Chapter 5. Now, the open challenge is estimating the preprocessing filter parameters of the quality filter. With the Gaussian low-pass filter selected as the quality filter in Section 6.2, we designed two bitrate models that specifically consider the influence of the Gaussian low-pass filter on the video bitrate in the previous Section 6.3.

In this section, we use these bitrate models to design a preprocessor rate control approach that allows for estimating the parameters required to control the respective preprocessing filter. The parameters controlling the preprocessing filters are estimated from the adaptation parameters to control multiple encoders such as provided by the TAMVA scheme. We next introduce the general rate conditions valid for the superframe composition and the CQP mode used for the single encoder. Then, we present the preprocessor model estimating the filter parameters.

6.4.1 Superframe Composition

We compose the individually preprocessed camera view frames into a single superframe before encoding the resulting superframe. One possible solution for the superframe composition is to assign every camera view segment a fixed position and origin within the superframe. For the fixed location in the superframe, the maximum resolution of each camera view segment is used. In case the spatial filter rescales individual camera views, they will be placed at the same origin. The remaining parts of the superframe not covered by downscaled segments remain black or empty. Figure 6.16 visualizes this superframe composition process.

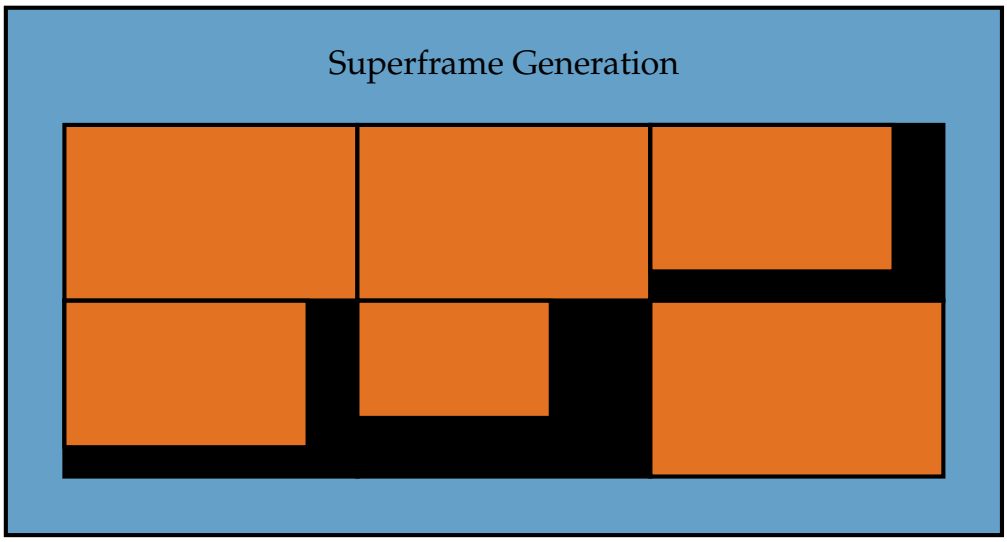


Figure 6.16 Superframe composition with fixed frame positions based on the maximum resolution possible.

These empty parts in the SFV do not contribute to the overall bitrate since the encoder will treat them with skip mode, similar to the ROI masking approach introduced in Chapter 5. We experimentally validated this behavior in Section 6.1.4.1.

The individual camera views in vehicles typically have almost no overlapping regions, which prevents the encoder from exploiting inter-view dependencies. Thus, the superframe bitrate R_{SFV} can be formulated as the sum of individual view bitrates R_i for scenarios with almost no overlap among the cameras:

$$R_{SFV} = \sum_i R_i. \tag{6.26}$$

6.4.2 CQP Mode Conditions

The proposed preprocessing concept enables individual rate/quality adaptations in the SFV. This means N individual segments can be encoded at different temporal, spatial, and quality levels using a single encoder. For this, we use the encoder in CQP mode to avoid modifications of the bitrate due to the rate control of the single encoder. In CQP mode, the QP can be directly used to control the quality and resulting bitrate $R(q_p)$, which depends on the quantization parameter q_p. For N individual segments, this results in N QPs $q_{p,1}, \ldots, q_{p,N}$.

Controlling the single encoder requires a common superframe quantization parameter $q_{p,SFV}$. We define $q_{p,SFV}$ as the minimum of all individual QPs $q_{p,1}, \ldots, q_{p,N}$, since the preprocessing filters introduced in Section 6.1.2 only allow for reducing the rate/quality:

$$q_{p,SFV} = \min(q_{p,1}, \ldots, q_{p,N}). \tag{6.27}$$

Then, the resulting bitrate $R_i(q_{p,i})$ of a frame encoded individually should be equal to the resulting bitrate $R_{SFV,i}(q_{p,SFV})$ of the same frame as part of an encoded superframe. In case no preprocessing is applied on the respective frame, this is only possible for $q_{p,i} = q_{p,SFV}$. To fulfill this condition for all cases $q_{p,i} > q_{p,SFV}$, we apply the quality filter to reduce the bitrate required for encoding the respective frame. Based on the evaluation in Section 6.2, we suggest using a Gaussian low-pass filter with kernel size k and standard deviation σ. Considering the parameters k and σ of the Gaussian low-pass filter allows us to formulate the bitrate equality condition for all individual segments $i \in 1, \ldots, N$ under the condition $q_{p,SFV} \leq q_{p,i}$ and with the assumption of Equation 6.26:

$$R_i(q_{p,i}) = R_{SFV,i}(q_{p,SFV}, k_i, \sigma_i). \tag{6.28}$$

The kernel size k of the Gaussian filter can only take discrete odd values ($k \in 1, 3, \ldots$), where a value of $k = 1$ means no filtering. This prevents perfectly matching Equation 6.28 for all $k > 1$. Next, we discuss how to find the optimal combination of k and σ to minimize the difference of both bitrates in Equation 6.28.

6.4.3 Preprocessor Model

Minimizing the bitrate difference in Equation 6.28 requires a bitrate model for estimating the video bitrate R for a given q_p, k, and σ. For this, we use the two bitrate models designed in Section 6.3. Second, we need a preprocessor model to estimate the bitrate damping after applying a Gaussian low-pass filter configured with the parameters k and σ. The Gaussian filter is applied in a preprocessing step before the individual views are combined into a single superframe and encoded by the single encoder. Figure 6.17 shows the concept of the preprocessing filters used and where the preprocessor model is integrated in the pipeline.

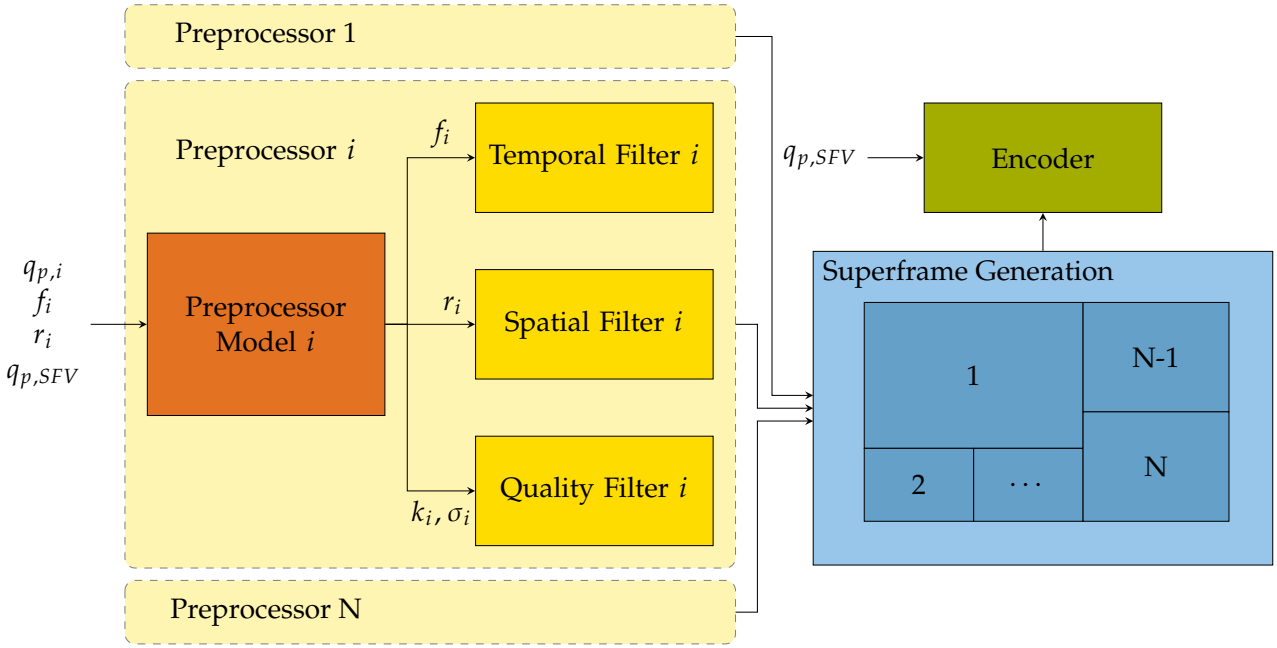


Figure 6.17 The preprocessing filter concept with the proposed model estimating the preprocessing filter parameters.

As summarized in Figure 6.17, the preprocessor accepts four encoding parameters: the quantization parameter $q_{p,i}$, the frame rate f_i, and the frame resolution r_i for an individual camera view i as well as the common quantization parameter $q_{p,SFV}$ of the SFV. These parameters are usually provided for an individual encoder, which makes our approach applicable to existing adaptation models.

For encoders used in embedded systems, there is limited access to the encoding hardware. We therefore require a bitrate model that works independently of the underlying video codec. Lottermann et al. [141] proposed the analytical rate model STRM, considering the QP and video frame rate. The STRM depends on two constant parameters and two content-dependent parameters that can be calculated directly from the source video. The authors further extended their model in [142], studying the influence of GoP characteristics. Both models estimate the bitrate as the product of maximum bitrate and separate correction factors modeling the influence of each parameter. We followed this idea and proposed a bitrate model that additionally includes the influence of the frame size and the Gaussian low-pass filter. As a second approach besides the proposed analytical model, we proposed an ML-based rate model using the same input as the analytical model. Both models were designed in Section 6.3 and can be used by the preprocessor model for minimizing the difference of both bitrates in Equation 6.28 to find the optimal combination of k and σ.

6.4.4 Results

In this section, we demonstrate the performance and usability of the proposed preprocessor rate control. We evaluate a multi-view streaming scenario for the four videos of the *x264-CIF* test set introduced in Section 6.3.1.2. Since the encoding parameters controlling the spatial and temporal resolution of the individual video streams can be used directly with the temporal and spatial preprocessing filter, we only focus on the influence of the quality filter in this experiment. Hence, we evaluate the preprocessor model estimating the filter parameters for the Gaussian low-pass filter.

For the evaluation, we consider the four videos of the *x264-CIF* test set as the four different camera views that should be encoded at different rate/quality levels with a single encoder. We randomly select four QPs within the range of 24 to 45. The maximum distance between two QPs should not exceed 15. Larger differences of two QPs would exceed the maximum reasonable damping of the Gaussian low-pass filter with a kernel size of nine. Larger kernel sizes are possible, but not recommended since such large kernel sizes heavily affect the video quality.

Using the four QPs selected randomly, we compare five different multi-view streaming approaches. As a baseline, we use multiple video encoders as typically used in multi-view streaming. We refer to this solution as *Multi*. We compare the baseline *Multi* to a solution without preprocessing that uses a single encoder, referred to as *Single*, and the proposed preprocessing concept using a single encoder. For the proposed preprocessing concept, we evaluate our preprocessor model twice, referring to the analytical bitrate model as *Analytic* and to the ML-based bitrate model as *ML*. As a reference, we include the best possible solution that can be achieved with the proposed preprocessing concept and a single encoder. We refer to this optimal reference approach as *Oracle*, since it does not estimate the bitrates but uses the ground truth available in the *x264-CIF* dataset. In practical applications, this information is not available, hence *Oracle* serves as a theoretical upper performance bound. Table 6.10 summarizes the results for the five multi-view streaming approaches.

In the *Settings* block, the *Multi* mode shows the QPs selected randomly. k and σ are empty since the preprocessing is not required when multiple encoders are available. The *Single* mode also does not use preprocessing, but only the minimum QP of all QPs for the four videos. The remaining modes contain both the minimum QP selected according to Equation 6.27 and the preprocessing parameter estimated by the preprocessor model. For the *Oracle* mode, the preprocessing parameters are selected from the *x264-CIF* dataset minimizing Equation 6.28.

The *Bitrate* block represents the resulting bitrates, achieved with the specified settings. Finally, we calculate the mean absolute error (MAE) of all views relative to the *Multi* approach and normalized by the bitrate of *Multi*. With an MAE of 0.0102, the *Oracle* approach represents the best possible solution that can be achieved using the proposed preprocessing filter concept. This demonstrates the usability of the preprocessing filter compared to an MAE of 0.2969 when no preprocessing is applied. The *ML* approach cannot reach the optimal performance of the *Oracle* mode, but is able to reach an MAE that is 87 % lower than the *Single* mode. The *ML* approach clearly outperforms the *Analytic* approach. However, the *Analytic* approach still reaches an MAE that is 75 % lower compared to *Single*.

Table 6.10 Comparison of a multi-view streaming scenario using the four videos of the *x264-CIF* test set and with a QP selected randomly for every video.

	Video	Multi	Single	Oracle	Analytic	ML
Settings $\{q_p, k, \sigma\}$	*bus*	{26,-,-}	{26,-,-}	{26,1,0.5}	{26,1,0.5}	{26,1,0.5}
	city	{38,-,-}	{26,-,-}	{26,7,2.5}	{26,9,3.0}	{26,9,3.0}
	deadline	{29,-,-}	{26,-,-}	{26,3,0.5}	{26,3,1.0}	{26,5,0.5}
	mother-daughter	{32,-,-}	{26,-,-}	{26,5,2.0}	{26,9,3.0}	{26,5,2.0}
Bitrate R [kbit/s]	*bus*	6145	6145	6145	6145	6145
	city	1353	5225	1355	2282	1179
	deadline	3696	4746	3919	3919	3919
	mother-daughter	1098	2088	1097	1806	1317
Normalized MAE	⌀	-	0.2969	0.0102	0.0731	0.0377

Table 6.10 summarizes the results for a single multi-view streaming scenario. To demonstrate reliable results, we repeat this experiment 100 times, selecting random QPs for each of the four videos. We calculate the MAE for all four videos and the 100 iterations. The resulting MAE for *Oracle* compared to multiple encoders is 0.0158. *Analytic* and *ML* reach MAEs of 0.0226 and 0.0434, respectively. The *Single* approach results in an MAE of 0.4546, showing a similar trend as the single streaming scenario of Table 6.10.

This demonstrates the usability of the proposed preprocessing filter approach and the performance of both bitrate models used. Neither the *ML* nor the *Analytic* approach are able to estimate the optimal preprocessing parameters and reach *Oracle* performance. These filter rate models are the first of their kind with the capability of estimating such preprocessing parameters. It should be noted that *Oracle* is the best theoretically possible solution that can only be found by performing the actual encoding steps, whereas our approach estimates the rates without performing any encoding.

In this experiment, we focused on matching the bitrate of individual encoders as closely as possible for validating the proposed preprocessor rate control. Next, we evaluate the entire preprocessing approach presented in this chapter in several multi-view driving scenarios, similar to Section 6.1.4.2.

6.5 Preprocessing Filter Application

Throughout this chapter, we developed a preprocessing filter concept that allows for the individual rate/quality adaptation of multiple camera views while using only a single encoder. With the preprocessor rate control introduced in the previous section, all parameters required to control the preprocessing filters can be estimated from encoding parameters that would also be used for controlling multiple encoders.

In this section, we demonstrate the performance and usability of the proposed preprocessing concept and all related components. For this, we compare the framework proposed in this chapter with a solution using multiple encoders in three multi-view driving scenarios such as introduced in Section 5.5. First, we analyze a single scenario in detail and discuss the trade-offs that need to be considered when using the proposed solution. Then, we analyze three different scenarios to demonstrate the usability of the proposed approach for the diverse requirements in ToD.

6.5.1 Single Scenario Results

We evaluate a multi-view teledriving scenario for the example of turning left at a crossing. The scenario evaluation uses a similar six camera setup and procedure as in Section 6.1.3. As defined in Section 5.5, we refer to this scenario as *left-turn*. The scenario was recorded in the CARLA simulator

with the six camera views according to their orientation in Figure 6.4: front (F), front left (FL), front right (FR), rear (R), rear left (RL), and rear right (RR).

The performance in terms of video bitrate and video quality reached by multiple encoders is evaluated as a reference. We compare the proposed preprocessing approach and a regular SFV approach without preprocessing to this reference of using multiple encoders. We select the encoding parameters for the individual views to represent their importance for the current traffic situation, similar to Section 6.1.3. In case of the *left-turn* driving scenario, the resulting encoding parameters favor the front view (F) and the front left view (FL), since they show the driving lane of the ego vehicle and the oncoming traffic. The least important view for the *left-turn* is the rear right view (RR), which results in the largest reduction of bitrate and quality in favor of the other views.

6.5.1.1 Encoding Parameter Definition

We evaluate the bitrate and perceptual quality for *x264-CIF* and *N-HEVC-HD* in five modes, using the parameters listed in Table 6.11. An individual adaptation strategy using multiple encoders defines the reference approach. We refer to this reference as *Multiple Encoders*. The proposed preprocessing approach using the filter parameters estimated by the analytical bitrate model is referred to as *Analytical*. The mode *ML* uses the same approach, but with the parameters estimated by the ML-based bitrate model. Lastly, two simple SFV approaches without preprocessing are evaluated. One simple SFV mode uses the minimum QP of the *Multiple Encoders* solution, named *SFV-27*, since the minimum QP is 27. The second mode matches the total bitrate required by all camera views. For *x264-CIF*, this is referred to as *SFV-33*. For *N-HEVC-HD*, we refer to the second mode as *SFV-32*. Table 6.11 summarizes the encoding parameters chosen for the *left-turn* driving scenario.

Table 6.11 Encoding parameters used for the six individual camera views: the quantization parameter q_p, the frame rate f relative to the maximum frame rate $f_{max} = 30\,\text{Hz}$, the frame resolution r, and the kernel size and standard deviation (k, σ) of the Gaussian low-pass filter. k and σ are estimated from the preprocessor model using the analytical or the ML bitrate model for both encoder x264 and N-HEVC. The individual views are grouped into important and remaining views based on their relevance.

Parameter	F	FL	FR	R	RL	RR
q_p	27	27	29	27	31	27
f / f_{max}	1	1	1	0.5	0.5	0.5
r	CIF	CIF	CIF	286×234	286×234	QCIF
Analytical (x264) (k, σ)	-	-	(3,0.52)	-	(3,0.7)	-
ML (x264) (k, σ)	-	-	(3,0.55)	-	(5,1.0)	-
r	1080p	1080p	1080p	900p	900p	720p
Analytical (HEVC) (k, σ)	-	-	(3,0.48)	-	(3,0.65)	-
ML (HEVC) (k, σ)	-	-	(3,0.87)	-	(5,1.65)	-
Relevance	High	High	Low	Low	Low	Low

The given QPs, frame rates, and frame sizes are the respective encoding parameters that should be used when multiple encoders are available. These parameters are selected to represent the view prioritization of the *left-turn* driving scenario. We provide the parameters for both *x264-CIF* and *N-HEVC-HD*. Before, *x264-CIF* and *N-HEVC-HD* referred to the dataset introduced for the training the bitrate models in Section 6.3. In this section, *x264-CIF* and *N-HEVC-HD* refer to the codec that is used for encoding the video sequences at the respective resolution. We use a GoP length of 20 for the entire scenario. For the single encoder case, we list the parameters for the Gaussian low-pass filter estimated using the proposed bitrate models, *Analytical* and *ML*, for both *x264-CIF* and *N-HEVC-HD*.

Lastly, we group the individual views into two groups, *High* and *Low*, based on their relevance for the current traffic situation. This grouping based on the relevance will be used for the evaluation later. In the following, we refer to the relevance level *High* also as "important" views and to the level *Low* also as the "remaining" views.

6.5.1.2 x264-CIF

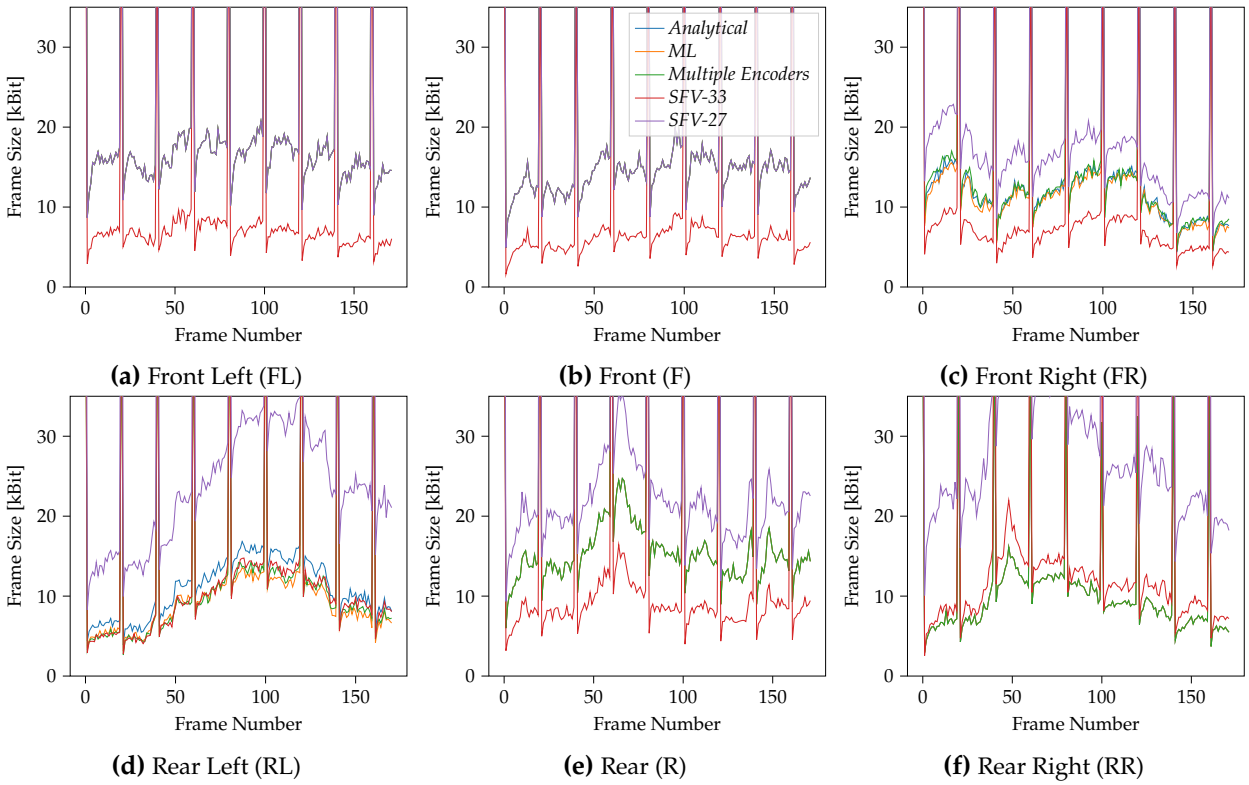
For the first experiment, the video sequences at CIF resolution were encoded with the x264 [134] software video encoder using the parameters listed in Table 6.11. We analyze the resulting video bitrate and the video quality using the PSNR, the Structural Similarity Index (SSIM), and the VMAF [157] as quality metrics. Table 6.12 presents the results for the five modes.

Table 6.12 Video bitrate [kbit/s] and quality results for x264 and CIF resolution. The bold values are the closest match to the reference *Multiple Encoders*.

Mode	Metric	Avg.	Avg. High	Avg. Low	F	FL	FR	R	RL	RR
Multiple Encoders	Bitrate	366.30	575.09	261.90	532.19	617.98	442.59	274.45	172.26	158.30
	PSNR	38.75	39.97	38.14	40.33	39.60	38.89	39.01	37.12	37.52
	SSIM	0.97	0.97	0.97	0.97	0.97	0.96	0.97	0.96	0.97
	VMAF	97.60	98.33	97.24	98.11	98.55	97.27	98.98	94.52	98.17
Analytical	Bitrate	**370.56**	**575.09**	**268.30**	**532.19**	**617.98**	**436.74**	**274.45**	203.72	**158.30**
	PSNR	**38.47**	**39.97**	37.72	**40.33**	**39.60**	36.92	**39.01**	**37.41**	**37.52**
	SSIM	**0.97**	**0.97**	**0.96**	**0.97**	**0.97**	0.95	**0.97**	**0.96**	**0.97**
	VMAF	95.35	**98.33**	93.87	**98.11**	**98.55**	90.77	**98.98**	87.54	**98.17**
ML	Bitrate	361.78	**575.09**	255.13	**532.19**	**617.98**	421.93	**274.45**	**165.84**	**158.30**
	PSNR	37.87	**39.97**	36.82	**40.33**	**39.60**	36.48	**39.01**	34.29	**37.52**
	SSIM	0.96	**0.97**	**0.96**	**0.97**	**0.97**	0.95	**0.97**	0.95	**0.97**
	VMAF	93.40	**98.33**	90.94	**98.11**	**98.55**	89.46	**98.98**	77.14	**98.17**
SFV-27	Bitrate	719.63	**575.09**	654.29	**532.19**	**617.98**	584.65	776.94	843.09	**412.48**
	PSNR	39.59	**39.97**	**38.45**	**40.33**	**39.60**	**40.14**	39.36	39.16	**35.14**
	SSIM	**0.97**	**0.97**	**0.96**	**0.97**	**0.97**	**0.96**	0.97	**0.96**	0.93
	VMAF	**98.55**	**98.33**	**97.27**	**98.11**	**98.55**	**98.42**	99.22	98.20	93.25
SFV-33	Bitrate	318.05	262.63	483.37	244.20	281.06	266.99	339.36	364.21	962.92
	PSNR	35.89	36.32	36.62	36.68	35.95	36.51	35.59	35.44	38.94
	SSIM	0.94	0.95	0.95	0.95	0.94	0.94	0.94	0.94	0.96
	VMAF	92.64	91.94	94.38	91.87	92.01	92.02	95.00	**91.68**	98.80

The row *Multiple Encoders* serves as the baseline reached with multiple encoders available. The values in bold face highlight the closest match to the baseline *Multiple Encoders*. All modes except *SFV-33* perfectly match the bitrate and quality for the most important views (F) and (FL). However, the average bitrate of *SFV-27* is about 96 % larger than the baseline *Multiple Encoders*. *SFV-33*, *Analytical*, and *ML* almost match the average bitrate of the baseline *Multiple Encoders*. For the individual views, *SFV-33* deviates from the individual bitrates, which results in a reduced quality for the important front views (F) and (FL) as well as a higher quality for the less important rear views. It should be noted that for the rear view, the higher quality is not needed.

Overall, the *Analytical* mode reaches the best trade-off of high video quality on the most important views while still matching the average bitrate. We further quantify this by calculating the average quality of the important views (F) and (FL) as well as for the remaining views and compare them to the reference quality of *Multiple Encoders*. At a similar average bitrate, *Analytical* and *ML* reach the same quality as *Multiple Encoders* for the important views. This comes at the cost of reduced average quality on the remaining views of 3.4 VMAF score for *Analytical* and 6.3 for *ML*. While *SFV-33* reduces the VMAF score on the remaining views by only 2.9, it also reduces the quality of the important views by



(a) Front Left (FL) **(b)** Front (F) **(c)** Front Right (FR)

(d) Rear Left (RL) **(e)** Rear (R) **(f)** Rear Right (RR)

Figure 6.18 Video bitrate [kbit/s] trend over the scenario frame numbers.

6.4. For a similar bitrate, *SFV-33* reaches an average VMAF score that is 0.51 higher than for *Analytical*. The higher average score for *SFV-33* highlights the additional quality costs of using the Gaussian filter in *Analytical* to achieve the additional 6.4 VMAF score on the important views. Figure 6.18 shows the size of the encoded frames and Figure 6.19 the VMAF score of every frame for the *left-turn* driving scenario.

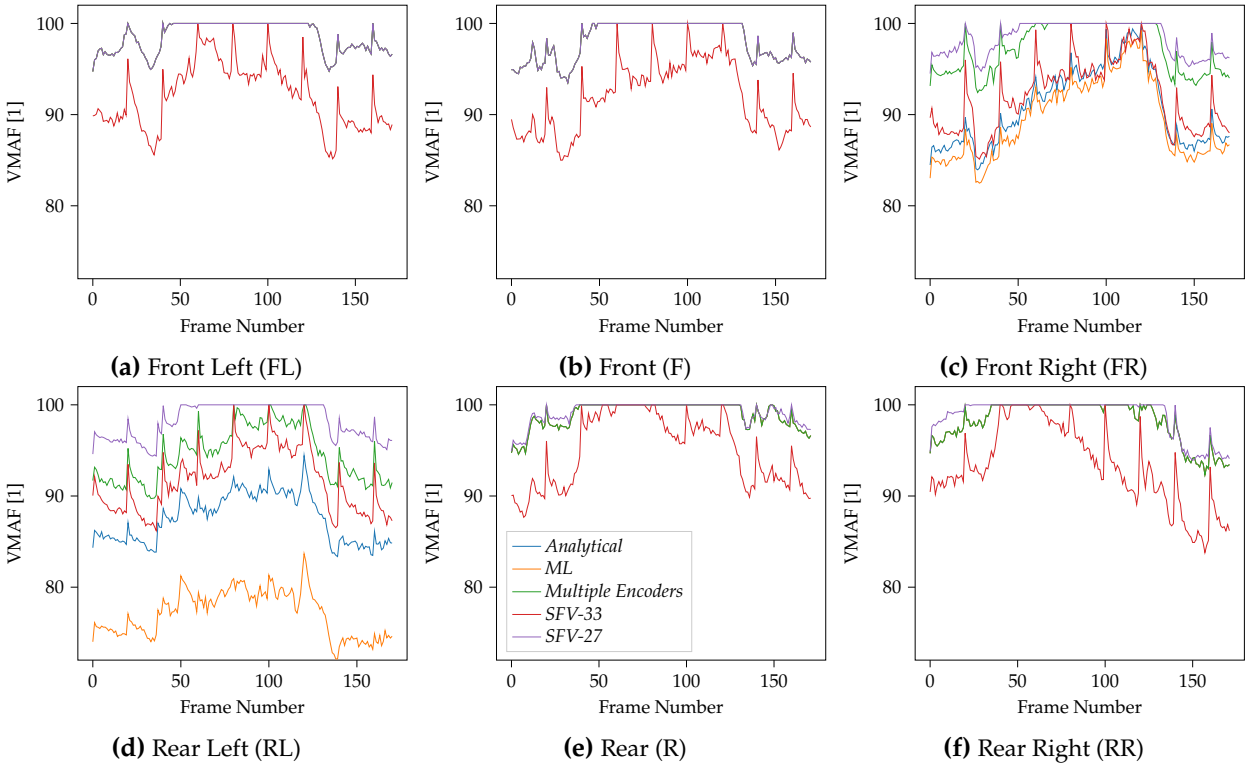
Similar to the results of Table 6.12, *SFV-33* shows higher bitrates and reduced VMAF scores for the front view (F) and front-left view (FL) compared to the other modes. At the same time, *Analytical* and *ML* heavily affect the VMAF score for the rear left view (RL), while *SFV-33* matches the reference of *Multiple Encoders* much more closely. This again highlights the trade-off when shifting the bit-budget and the resulting quality towards the most important views.

6.5.1.3 NHEVC-HD

Similar to *x264-CIF*, we evaluate the same scenario for the video sequences at HD resolution encoded with the NVIDIA HEVC hardware video encoder using the parameters listed in Table 6.11. Table 6.13 presents the results for the five modes.

Here, the *ML* mode reaches the closest bitrate matches. Including both rate and quality, the *Analytical* mode reaches the best trade-off. For a similar average bitrate, both *Analytical* and *ML* reach the same quality as *Multiple Encoders* on the important views. At the same time, the average VMAF score on the remaining views is reduced by 1.3 and 7.7, respectively. *SFV-32* results in a reduced VMAF score of 2.0 on the important views and 1.6 on the remaining views.

The lower video qualities of *ML* compared to *Analytical* can be explained by the different low-pass filter parameters. Especially for the view (RL), the *ML* mode selects a kernel size of five, which causes a reduction in the PSNR and VMAF score. For the given scenario, this view can be assumed to be inspected less frequently by the driver. This loss of quality can be still worthwhile to ensure a high video quality for the important views (F) and (FL).



(a) Front Left (FL) **(b)** Front (F) **(c)** Front Right (FR)

(d) Rear Left (RL) **(e)** Rear (R) **(f)** Rear Right (RR)

Figure 6.19 VMAF score over the scenario frame numbers.

Table 6.13 Video bitrate [kbit/s] and quality results for N-HEVC and HD resolution. The bold values are the closest match to the reference *Multiple Encoders*.

Mode	Metric	Avg.	Avg. High	Avg. Low	F	FL	FR	R	RL	RR
Multiple Encoders	Bitrate	3842.96	5510.37	3009.25	5553.19	5467.54	4567.29	2944.00	1962.35	2563.37
	PSNR	42.47	43.25	42.11	43.15	43.35	42.06	42.99	41.26	42.11
	SSIM	0.97	0.97	0.98	0.97	0.97	0.97	0.98	0.97	0.98
	VMAF	99.09	99.40	98.94	99.08	99.71	98.67	99.88	97.73	99.46
Analytical	Bitrate	4006.78	**5510.37**	**3254.99**	**5553.19**	**5467.54**	**4928.36**	**2944.00**	2584.24	**2563.37**
	PSNR	**42.66**	**43.25**	42.37	**43.15**	**43.35**	**42.03**	**42.99**	**42.35**	**42.11**
	SSIM	0.98	**0.97**	**0.98**	**0.97**	**0.97**	**0.97**	**0.98**	0.98	**0.98**
	VMAF	98.23	**99.40**	97.65	**99.08**	**99.71**	96.91	**99.88**	94.33	**99.46**
ML	Bitrate	**3735.62**	**5510.37**	2848.24	**5553.19**	**5467.54**	4008.99	**2944.00**	**1876.60**	**2563.37**
	PSNR	41.65	**43.25**	40.85	**43.15**	**43.35**	40.25	**42.99**	38.06	**42.11**
	SSIM	**0.97**	**0.97**	0.97	**0.97**	**0.97**	0.96	**0.98**	0.96	**0.98**
	VMAF	93.98	**99.40**	91.28	**99.08**	**99.71**	90.32	**99.88**	75.44	**99.46**
SFV-27	Bitrate	7690.97	**5510.37**	8781.27	**5553.19**	**5467.54**	6221.11	8591.95	9676.33	10 635.70
	PSNR	**42.66**	**43.25**	**42.36**	**43.15**	**43.35**	42.79	42.49	42.46	41.71
	SSIM	**0.97**	**0.97**	0.97	**0.97**	**0.97**	**0.97**	0.97	**0.97**	0.97
	VMAF	**99.48**	**99.40**	**99.52**	**99.08**	**99.71**	**99.29**	99.87	99.39	99.54
SFV-32	Bitrate	3710.79	2584.57	4273.90	2510.12	2659.01	2881.88	4182.16	4869.80	5161.76
	PSNR	40.59	41.38	40.19	41.37	41.38	40.95	40.21	40.10	39.51
	SSIM	0.96	0.97	0.96	0.96	0.97	0.96	0.96	0.96	0.95
	VMAF	97.36	97.44	97.32	96.96	97.91	96.91	98.58	**96.65**	97.15

In summary, the results highlight that the proposed preprocessing solution matches the different video bitrates and perceptual quality scores more closely than the simple SFV approach for both proposed bitrate models, evaluated on both *x264-CIF* and *N-HEVC-HD*. Next, we repeat this evaluation for multiple driving scenarios.

6.5.2 Multi Scenario Results

After the detailed analysis of a single scenario, several key properties required for the evaluation can be determined. The solutions used must match the total bitrate of using multiple encoders, since this can be considered as the maximum transmission rate available in the mobile network. Additionally, the resulting video quality represented by the VMAF score for all views, the important views, and the remaining views, will be analyzed. Hence, we only evaluate the four modes matching the total bitrate in this section. In addition to the *left-turn* scenario evaluated in the previous section, we include the *right-turn* and *straight-driving* introduced in Chapter 5.

6.5.2.1 Encoding Parameter Definition

Similar to the encoding parameter specified for the *left-turn* scenario in Table 6.11, we additionally define the encoding parameters for *right-turn* and *straight-driving* in Table 6.14.

Table 6.14 Encoding parameters used for the six individual camera views of the scenarios *left-turn*, *right-turn*, and *straight-driving*. The individual views are grouped into important views (*High*) and the remaining views (*Low*) based on their relevance.

Scenario	Parameter	F	FL	FR	R	RL	RR
left-turn	q_p	27	27	29	27	31	27
	f/f_{max}	1	1	1	0.5	0.5	0.5
	r	CIF	CIF	CIF	286×234	286×234	QCIF
	Analytical (k,σ)	-	-	(3,0.52)	-	(3,0.7)	-
	ML (k,σ)	-	-	(3,0.55)	-	(5,1.0)	-
	Relevance	High	High	Low	Low	Low	Low
right-turn	q_p	27	29	27	31	31	27
	f/f_{max}	1	1	1	0.5	0.5	1
	r	CIF	286×234	CIF	QCIF	QCIF	CIF
	Analytical (k,σ)	-	(3,0.53)	-	(3,0.85)	(3,0.77)	-
	ML (k,σ)	-	(3,0.66)	-	(5,1.9)	(5,1.2)	-
	Relevance	High	Low	High	Low	Low	High
straight-driving	q_p	27	27	27	30	30	30
	f/f_{max}	1	1	1	0.5	0.5	0.5
	r	CIF	286×234	286×234	QCIF	QCIF	QCIF
	Analytical (k,σ)	-	-	-	(3,0.64)	(3,1.4)	(3,0.5)
	ML (k,σ)	-	-	-	(3,1.31)	(5,1.0)	(3,0.5)
	Relevance	High	Low	Low	Low	Low	Low

For the *right-turn* scenario, the front view (F) and the front right view (FR) are of high relevance for the turning task. Additionally, the rear right view (RR) is favored since pedestrians or cyclists might cross the street and the operator has to verify this before turning into the new street. The remaining views are considered as less important and encoded with a reduced rate and quality. Since front views are generally more important than rear views, the front left view (FL) is encoded with a higher quality than the remaining rear views.

For the *straight-driving* scenario, the straight front view (F) is of highest importance, followed by the front-side views (FL) and (FR). The rear views are the least important views in this scenario. We selected the encoding parameters such that they represent this prioritization, inspired by the user priority ratings of Table 5.2. In this scenario, we consider only the single front view (F) as highly relevant for the driving task.

6.5.2.2 x264-CIF

Similar to the single scenario evaluation, we encode the video sequences at CIF resolution using the x264 [134] software video encoder with the encoding parameters as listed in Table 6.14. We analyze the resulting VMAF scores for the three driving scenarios introduced in this chapter and the four modes *Multiple Encoders*, *Analytical*, *ML*, and *SFV-Rate*. The first three modes are identical to the previous experiments. *SFV-Rate* uses a single encoder without preprocessing and a QP to match the total bitrate of *Multiple Encoders*. Since the QP for the single encoder varies for the three scenarios, we excluded the QP from the solution's name. The QPs of *SFV-Rate* for the individual scenarios are 33 for *straight-driving* and 32 for both turning tasks. Table 6.15 presents the average VMAF scores for all modes and scenarios as well as for the grouping into important views and remaining views.

Table 6.15 Average VMAF scores grouped by relevance for the current driving situation. The bold values are the closest match to the reference *Multiple Encoders*.

Mode	*All*	*High*	*Low*
Multiple Encoders	95.40	97.0	94.60
Analytical	**94.22**	**97.0**	**92.84**
ML	91.07	**97.0**	88.64
SFV-Rate	90.72	91.57	90.30

None of the solutions using a single encoder is able to reach the average VMAF scores of *Multiple Encoders*. However, the proposed preprocessing approach reaches the same VMAF scores for the respective camera views rated with *High* relevance. For the remaining views, the *Analytical* model reaches the highest quality of all solutions using a single encoder. The *ML* preserves the quality level of *Multiple Encoders*, while causing even stronger quality degradations than *SFV-Rate* on the remaining views. This investment in quality can be still worthwhile in situations with a low transmission rate available, allowing to maintain the quality required for driving using the most important views.

The higher quality degradations of *ML* are caused by the larger kernel sizes for the Gaussian filter. Although the ML-based bitrate model estimates the bitrate more precisely than the analytical model, it results in higher kernel sizes estimated by the preprocessor rate control. These higher kernel sizes cause larger quality degradations. A possible improvement for the preprocessor rate control could include additional conditions, such as keeping the kernel size small besides matching the bitrate as closely as possible.

6.6 Summary

In this chapter, we presented a preprocessing concept that allows for the individual rate/quality adaptation of multiple camera views while using only a single encoder. We proposed four preprocessing filters to control the spatial resolution, the temporal resolution, the rate/quality, and the number of color channels for the individual camera views. The individual camera views are preprocessed with the proposed filters and then combined into a single SFV. The SFV is then compressed, transmitted, and decomposed on the receiver side.

We first introduced the general concept of the proposed preprocessing approach and experimentally validated a proof-of-concept implementation. Based on the insights of these first experiments, we demonstrated that the temporal and spatial filter can directly use the given encoding parameters such as estimated by the TAMVA scheme introduced in Chapter 5.

Next, we systematically evaluated the influence of different preprocessing algorithms on the RD performance in video encoding to select the most suitable algorithm for the quality filter. We proposed

an evaluation method that offers a systematic approach for quantifying the benefits of a chosen filter compared to other options. Based on the results of this evaluation, we identified the spatial Gaussian low-pass filter as the preprocessing algorithm that achieves the best RD performance after encoding, which has been a popular choice in the literature before.

With the Gaussian low-pass filter as the most suitable algorithm for the quality filter, we design both one analytical and one ML-based bitrate model that specifically consider the influence of the Gaussian low-pass filter as well as the influence of the QP, the frame size, the frame rate, and the GoP length on the video bitrate. Both proposed bitrate models outperformed the state-of-the-art bitrate models by at least 22 % regarding the overall RMSE. Additionally, our bitrate models are the first of their kind to consider the influence of a Gaussian low-pass filter.

Using the two bitrate models proposed in this chapter, we designed a preprocessor rate control approach that allows for estimating the parameters required to control the respective preprocessing filters. This estimation is based on the adaptation parameters estimated by the TAMVA scheme to control multiple encoders. This way, the preprocessing filters can be used as a rate control scheme to enable the individual rate/quality adaptation of multiple camera views while using only a single encoder.

Finally, we evaluated the usability of the proposed preprocessing approach in three representative multi-view driving scenarios. We compared the proposed approach to a solution using multiple encoders and a SFV approach using a single encoder without any preprocessing. For three representative driving scenarios and a similar total bitrate of all views, the proposed approach achieved the same video quality for the most important views compared to the solution using multiple encoders, while it caused quality reductions of only 1.8 % on the remaining view. In comparison, the SFV approach using a single encoder without preprocessing causes quality degradations of 4.5 % for the remaining views and of 5.6 % on the most important views when using a similar total bitrate of all views.

Chapter 7

Conclusion

Teleoperated driving (ToD) has the potential to become an important solution for handling failures of autonomous vehicles. While many approaches in the literature focus on improving the operator performance by augmenting the visual representation [12, 11] or providing haptic feedback [13], the actual transmission of sensor information has received less attention. The transmission rate available in mobile networks is usually sufficient for transmitting video streams of multiple camera views such as required for ToD. However, this available transmission rate cannot be guaranteed by the network provider at any time, which can cause severe safety issues when controlling a vehicle from remote. In this thesis, we focus on how to distribute the limited transmission rate in a way that the camera views most important for the current driving situation are still transmitted in a quality sufficient for safely controlling the vehicle. This challenge led to developing the ToD framework presented in Figure 1.2 and adaptation strategies to control multiple video streams while considering the hardware limitations of commercial vehicles. We conclude this thesis by summarizing the key contributions, discussing the major limitations, and outlining directions for future work.

7.1 Summary

We considered the entire pipeline of a general ToD system for designing and implementing the ToD framework presented in this thesis. In summary, we proposed four main additions: a system and streaming pipeline for ToD in simulation, a driver situation awareness (SA) assessment system, a traffic-aware multi-view adaptation (TAMVA) scheme, and a video preprocessing approach to cope with the hardware limitations of autonomous vehicles. The four key contributions have been presented in Chapter 3, Chapter 4, Chapter 5, and Chapter 6.

The design and evaluation of adaptation strategies for ToD requires controllable network and driving conditions. For this, we designed the teleoperation framework TELECARLA, which extends the open source driving simulator CARLA [16] for vehicle remote control. This framework includes a customizable user interface, a scenario evaluation module, and an online-configurable low delay video streaming pipeline. The streaming pipeline provides an adaptation interface for controlling the frame rate, frame size, bitrate, and quality of the video stream. The TELECARLA framework builds the basis for the remaining contributions of this thesis.

A low delay video streaming pipeline is essential for providing the operator with the necessary visual information to understand the current situation. Despite the necessary visual information available for the operator, there is no guarantee that the operator actually recognizes all relevant elements correctly. As the second main contribution, we therefore proposed a method for assessing the driver's SA in realtime. Inspired by an approach used in the aviation context [103], we first introduced a concept for assessing the driver's SA. Then, we discussed the challenges when transferring an SA model designed for aviation to the driving context and proposed an SA method for driving. The driver's current SA is measured using eye tracking and compared to the optimal SA. The optimal SA describes all elements the driver should have perceived and is estimated by region of interest (ROI) prediction. As part of this work, we extended a state-of-the-art ROI prediction network [95] from

a single-view design towards supporting multiple camera views. The resulting driver awareness model allows for accurately measuring the driver SA, which has been validated in a user study for eight driving scenarios.

With the capability to control the adaptation parameters as part of the teledriving framework and the ability to measure the driver's SA, we next proposed a traffic-aware multi-view adaptation (TAMVA) scheme to provide the operator with the best possible SA for the current traffic situation. Considering the vehicle's realtime movement in traffic, the adaptation scheme estimates the importance of each camera view for the current traffic situation. This priority score together with the total available transmission rate of the mobile network determines the bit-budget available for a specific camera view. Then, the optimal combination of frame rate, frame size, and target rate/quality is estimated using a quality-of-experience-driven multi-dimensional adaptation (MDA) scheme to control the encoder of each individual video stream. Additionally, we proposed a dynamic ROI masking approach to improve the video quality of individual camera views by filtering less important regions in the respective camera view. We evaluated the proposed adaptation scheme in a user study consisting of two separate experiments, both with a six camera setup. In the first experiment, the participants drove actively in a simulated environment. In the second experiment, the participants observed driving situations recorded in advance and rated the importance of the individual camera views for the current traffic situation. We could not observe a clear improvement in the driver's performance for the first experiment. However, the proposed view prioritization module achieves a high correlation between the subjective view priority ratings obtained in the second experiment of the user study. With at least 19.8 % Bjøntegaard Delta Rate (BDR) savings, the ROI filter approach significantly reduces the required bitrate compared to streaming the full camera frame. Overall, the proposed TAMVA scheme increases the average video quality per camera by 5 % VMAF score compared to a uniform adaptation.

Finally, we considered the hardware limitations of autonomous vehicles, which prevent the direct usage of the proposed TAMVA scheme to control the encoders for individual camera views. Limited by cost and size, commercial vehicles are often restricted to a single hardware encoder. Encoding all camera views at the same time with a single encoder requires the combination of the individual frames into a single superframe video (SFV). While this is a possible solution for streaming multiple camera views, it prevents the adaptation of the individual camera views using the proposed TAMVA scheme. To address this issue, we proposed a video preprocessing concept that allows for individual rate/quality adaptation while using a single encoder. The proposed preprocessing filters control the frame rate, frame size, target rate/quality, and the number of color channels for the individual frames before combining them into a single SFV. We developed a comprehensive solution that enables automatic control of the individual preprocessing filters from the parameters estimated by the TAMVA scheme. A first proof-of-concept experiment demonstrated that due to the block-wise processing of the encoder, the temporal and spatial filters can directly use the frame rate and frame size estimated by the TAMVA scheme for the respective camera view. To select to most suitable algorithm for controlling the rate/quality, we systematically evaluated the influence of different preprocessing algorithms on the rate-distortion (RD) performance in video encoding. The proposed single score metric mean saving-cost ratio (MSCR) as well as the concept of Bjøntegaard Delta (BD) curves offer a systematic method for quantifying the benefits of a chosen filter compared to other options. Based on the results of this evaluation, we selected the spatial Gaussian low-pass filter as the preprocessing algorithm that achieves the best RD performance after encoding. To use the Gaussian low-pass filter for controlling the rate/quality, we designed two novel bitrate models that specifically consider the influence of the Gaussian low-pass filter as well as the influence of the quantization parameter (QP), the frame size, the frame rate, and the group of pictures (GoP) length on the video bitrate. Both bitrate models proposed in this thesis outperformed the state of the art bitrate models by at least 22 % regarding the overall RMSE. Using these bitrate models, we designed a preprocessor rate control approach that estimates the parameters required to control the respective preprocessing filters based on the encoding parameters estimated by the TAMVA scheme. With the ability to estimate the filter

parameters from the encoding parameter used to control multiple encoders, the preprocessing filters can be used as a rate control scheme to enable the individual rate/quality adaption of multiple camera views while using only a single encoder. For three representative driving scenarios and a similar total bitrate of all views, the proposed approach achieves the same video quality for the most important views as when using multiple encoders. Simultaneously, it causes quality reductions of only 1.8 % on the remaining views. In comparison, an approach using a single encoder without preprocessing causes quality degradations of 4.5 % for the remaining views and even 5.6 % on the most important views when using a similar total bitrate of all views. The relative quality degradations on the important views are larger, since the absolute quality score of the important views for multiple encoders is larger compared to the remaining views. The absolute quality scores on the important views are still higher for all modes compared to the remaining views.

7.2 Limitations

All concepts proposed in this thesis were evaluated in user studies or compared to state-of-the-art approaches. The methods proposed in this thesis outperformed existing approaches or were the first of their kind addressing the respective problem. Despite their novelty or superior performance to the state of the art, they still have some limitations and open questions which we discuss next.

The teleoperation framework TELECARLA presented in Chapter 3 extends the CARLA simulator for a ToD component. While the CARLA simulator provides a rich set of features, it comes at the cost of a high computational complexity. In particular, the rendering of multiple camera views such as required for ToD requires high-end hardware and Graphics Processing Units (GPUs) to achieve a sufficient frame rate for ToD.

The driver SA model presented in Chapter 4 already addressed several challenges when transferring the SA model designed for aviation into the driving context. The challenges addressed in this thesis include the dynamic situation elements (SEs) and distraction available in driving compared to the static cockpit SEs in the aviation context. An existing limitation that has not been addressed in our contributions is the occurrence of multiple redundant SEs. In driving, there can be multiple traffic lights that all refer to the same intersection. As soon as a driver perceives one of them, they usually have a sufficient SA regarding the traffic light. So far, the driver SA model proposed in this thesis would only consider the single SE perceived by the driver as comprehended. The other SEs representing the other traffic lights are considered as undetected. This negatively affects the SA score of the driver, even though they might have full SA of the current situation. A similar problem exists for the same road object being visible in multiple overlapping camera views. To address this limitation, the SA model requires more semantic understanding for the current situation.

Regarding the TAMVA scheme proposed in Chapter 5, a limitation is that the prioritization module only considers the movement of the ego vehicle independently of other road users and the spatio-temporal complexity of the respective camera view. This could result in a visual quality that is higher than required for the views considered as most important due to a low spatio-temporal complexity. At the same time, a view with many road users and a high spatio-temporal complexity could result in a visual quality that is too low even for a less important view due to the high amount of information included in this view. To address this limitation, the TAMVA scheme could be extended to also consider influence factors that contain information about the surrounding environment.

The preprocessing concept proposed in Chapter 6 controls the frame rate, frame size, and target rate/quality of the individual views streams before the combination into the SFV. While frame rate and frame size can be directly used for controlling the spatial and temporal filter, the parameters to control the Gaussian low-pass filter need to be estimated. For this we match the bitrate when using an individual encoder with the bitrate using the preprocessing filter and the superframe encoder with a QP different to the individual encoder. We estimate the bitrates using the two bitrate models introduced in Section 6.3. Although the machine learning (ML)-based bitrate model estimates the bitrate more precisely than the analytical model, it results in larger kernel sizes when matching the

bitrates for preprocessor rate control. This causes stronger quality degradations. To address this limitation, the preprocessor rate control should consider additional conditions for estimating the optimal filter parameters.

7.3 Future Work

In this thesis, we investigated the topic of adaptive video streaming for ToD. The goal was to develop a system that automatically controls multiple video streams in a way that the operator has the best possible SA for the current driving situation, while matching the available transmission resources of the network. Each of the proposed methods was thoroughly evaluated and achieved state-of-the-art performance. While the framework developed in this thesis can already be used to automatically control multiple video streams, there is still potential for further improvements towards immersive ToD. Next, we discuss ideas for addressing the known limitations and suggest additional ideas for future work.

With the SA model proposed in Chapter 4, we focused on a first validation of the proposed driver awareness approach. The user study validated the proposed approach in a monitoring task where the participants observed driving scenarios recorded in the CARLA simulator. For future work, this pure observation task could be compared to an observation-and-acting task. This comparison could give further insights on the influence of additional workload on the driver's SA. As possible results, either the additional workload caused by the active driving tasks affects the SA in a positive way due to a higher focus on the driving task or in a negative sense due to the higher workload caused by the driving task.

Another direction for future work on the driver SA model is the improvement of the ROI prediction used to determine the optimal SA. Determining the optimal SA with all SEs the driver should have perceived requires a highly accurate ROI prediction. To further improve the ROI prediction for multiple camera views, the redundancies in overlapping camera regions could be exploited. If an ROI is predicted in one camera view, then this prediction could be used for other views with overlapping regions, since the orientation and camera intrinsic parameters are known. First experiments already demonstrated partial improvements for the prediction, but also revealed that a second verification step is required. By only calculating the ROIs for neighboring views without verification, it is possible that an ROI is actually occluded in a neighboring camera view. For future work, the geometric dependencies between neighboring camera views should be exploited. As a recommendation, the geometric dependencies should be used for the verification of the ROI predictions or to provide region proposals used by the ROI prediction.

The TAMVA scheme proposed in Chapter 5 considers the vehicle's realtime movement in traffic to distinguish the most important camera views. For future work, additional input factors such as the number of ROIs in a certain camera view or the spatio-temporal complexity of the respective frame could be considered by the view prioritization model. Another source of information for identifying the relevant camera views is the operator's gaze. Considering these additional inputs would not only rely on the vehicle's movement for determining important views, but also include influence factors of the environment and the actual traffic situation as well as the direct focus of the human operator. For instance, the same turning maneuver in a crowded urban situation might require a different view prioritization compared to a turning maneuver on an empty street. Although the vehicle's movement is identical in both situations, the information provided by the environment such as the higher complexity in the urban scene can be used to further improve the prioritization model specifically for the current traffic environment. While the results presented in Section 5.6 showed clear improvements regarding the visual quality, we could not observe any significant improvements regarding the driver's performance. To this end, we first recommend to conduct a more extensive user study evaluating the view prioritization. This evaluation could be used to identify the key influencing factors that result in a clear improvement of the driver's performance as well as the visual quality.

A possible direction for future work on the preprocessing concept introduced in Chapter 6 could be the improvement of the preprocessor rate control approach. The preprocessor rate control introduced in Section 6.4 estimates the parameters used to control the Gaussian low-pass filter. These parameters are determined by matching the bitrate when using the parameters such as estimated by the TAMVA scheme with the bitrate when using the preprocessing filter with the QP of the superframe encoder. While reaching a similar bitrate is still required for matching the available transmission rate of the network, a different bitrate for the individual segments compared to the individual views when using multiple encoders can improve the overall quality. A possible improvement for the preprocessor rate control could be to include additional conditions, such as keeping the kernel size and standard deviation small in addition to matching the bitrate as closely as possible. Larger kernel sizes and standard deviation for the Gaussian low-pass filter cause stronger quality degradations for the respective image segment. Another possible option for improving the preprocessor rate control would be the development of a video quality model that specifically considers the influence of the Gaussian low-pass filter on the video quality. This way, the quality estimated by the model could be considered in addition to the bitrate to find the optimal solution for all camera segments. Similar to regular video coding, this solution can be approached by implementing a rate-distortion optimization (RDO) for the preprocessing filters. Alternatively, an end-to-end ML approach could be designed to directly estimate the optimal filter parameters from the parameters estimated by the TAMVA scheme. Implementing these recommendations could further improve the SA of a human operator when controlling a vehicle from remote.

Appendix

Appendix A

TELECARLA Framework and Ecosystem

TELECARLA is an open source framework for teleoperated driving (ToD) research that extends the Car Learning to Act (CARLA) simulator. It has been implemented alongside this dissertation and acts as the basic framework for the evaluations and user studies conducted in this work. All implementations of this dissertation have been realized on top of the basic TELECARLA framework and have been integrated to build a comprehensive framework for ToD research. Figure A.1 shows an overview of the proposed TELECARLA framework and the additional modules integrated in blue.

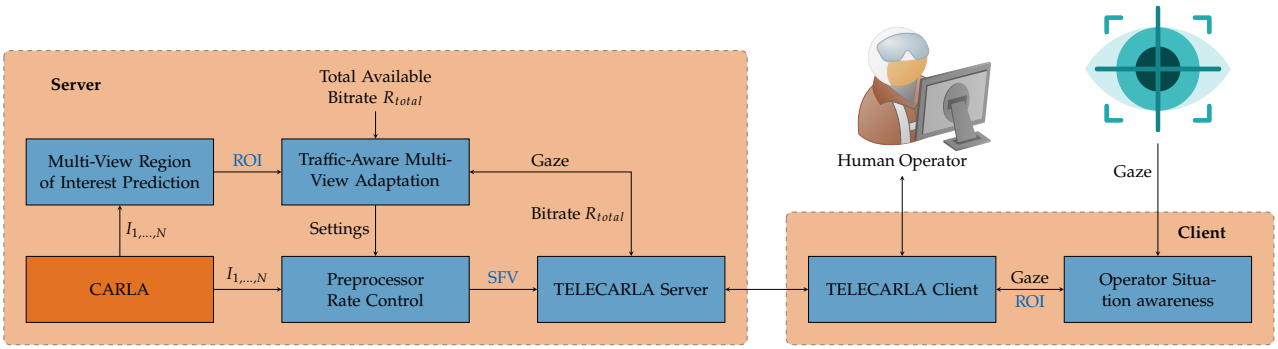


Figure A.1 Overview of the TELECARLA framework and extension modules.

The modules include the basic framework and streaming architecture proposed in Chapter 3, a multi-view region of interest (MV-ROI) framework as well as well as a model for assessing the driver's situation awareness (SA) proposed in Chapter 4, a traffic-aware multi-view adaptation (TAMVA) scheme proposed in Chapter 5, and a preprocessor-based rate control system proposed in Chapter 6. In the following sections, we summarize implementation details for the graphical user interface (GUI), the video streaming pipeline, the TAMVA scheme, and the driver SA model.

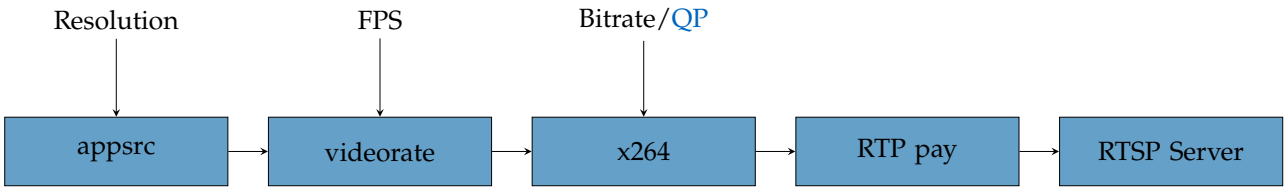
A.1 TELECARLA User Interface

The GUI uses the Simple DirectMedia Layer 2 (SDL2) [196] for rendering and displaying the sensor information. SDL2 is used to read the operator control commands entered by a keyboard, steering wheel, and pedals. It is also used to apply haptic feedback on the steering wheel. If an autonomous agent is in control of the vehicle, the steering wheel moves according to the vehicle's steering angle. This allows for the evaluation of takeover scenarios from autonomous driving mode to manual driving mode.

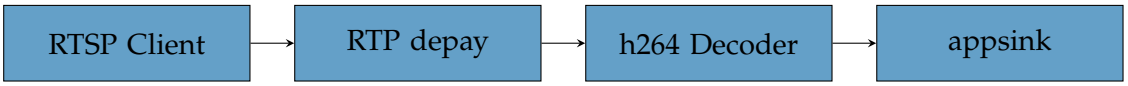
A.2 Adaptive Video Streaming Pipeline

The streaming pipeline itself is based on GStreamer [172], a multi-platform, plugin-based multimedia framework. GStreamer is a powerful and flexible multimedia framework due to is modular structure.

However, finding the right configuration for a specific task can be time consuming [197]. The core streaming workflow of our system is defined by two GStreamer pipelines for server and client side. Figure A.2 shows both GStreamer pipelines for the server (Figure A.2a) and client (Figure A.2b) conceptually.



(a) Server pipeline



(b) Client pipeline

Figure A.2 GStreamer server and client pipelines.

The server node reads the input data from the Robot Operating System (ROS) and feeds them into the GStreamer pipeline using the *appsrc* element. The *appsrc* element parses the frame resolution as well as the color format from the input image and downscales the frame if specified by the adaptation interface. This enables the streaming system to accept various input formats. We configured the *appsrc* plugin for *live-mode* to avoid artificial delays introduced by buffering multiple arriving frames. The *videorate* plugin [198] updates the temporal resolution (frame rate) of the video stream according to the target frame rate specified by the adaptation interface. Then, the video encoder plugin [199] uses the x264 [134] video codec, configured for low delay video streaming [130], to encode the video stream. The modular pipeline concept provided by GStreamer allows for using other video encoders as well. The target rate/quality for the video encoder is again specified by the adaptation interface. The encoded bitstream is then forwarded into the Real-Time Streaming Protocol (RTSP) server element [200] using the RTSP protocol for the data transmission. RTSP uses either User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) as the underlying transport protocol. By default, the RTSP server uses UDP.

On the receiver side, the RTSP client feeds the bitstream into the client pipeline where it is decoded and buffered into the *appsink* element. The *appsink* element publishes the decoded image data as a ROS image topic into the client system.

Both core pipelines are wrapped into a ROS node to provide a flexible interface and process scalability with respect to the multi-view streaming system. Further, we use the ROS package *dynamic_reconfigure* [201] to implement the adaptation interface controlling the frame size, frame rate, and target rate/quality.

A.3 Traffic-Aware Multi-View Video Stream Adaptation

The proposed TAMVA scheme is implemented as several individual ROS nodes that extend the proposed TELECARLA framework. In total, we designed three ROS nodes that interact in the way presented in Figure 5.1.

A single instance of the *View Prioritization* node considers the available transmission rate of the network as well as the vehicle's status information to estimate the view priority and the respective bit budget of every camera view. Then, for every camera view, a node running a multi-dimensional adaptation (MDA) adaptation model controls the video stream. The MDA node uses the bit budget available for the respective camera view as well as the current frame of this camera view to estimate the optimal combination of frame rate and frame size for the given bit budget. With the bit budget

determining the target bitrate and the optimal combination of frame size and frame rate estimated by the MDA model, the MDA node uses the adaptation interface of the *GStreaming* node to control the individual video stream. The *GStreaming* node, which is already part of the basic TELECARLA framework, encodes and transmits the video stream of a camera view from a server to a client.

To further improve the resulting visual quality, we designed an region of interest (ROI) filter as a separate ROS node. This node can then be used as a preprocessing step by integrating it between the node that outputs the camera image and the *GStreamer* node that encodes and transmits the video stream. Further, the ROI filter node can be configured to use the block-based mode for a given block size to further improve the resulting image quality after encoding. For our setup consisting of six camera views, we use the ROI filter only for the rear views.

A.4 Driver Awareness Model

The driver SA model is implemented as several individual ROS nodes that can be used as part of the TELECARLA framework. Figure A.3 shows the architecture of the proposed driver awareness module.

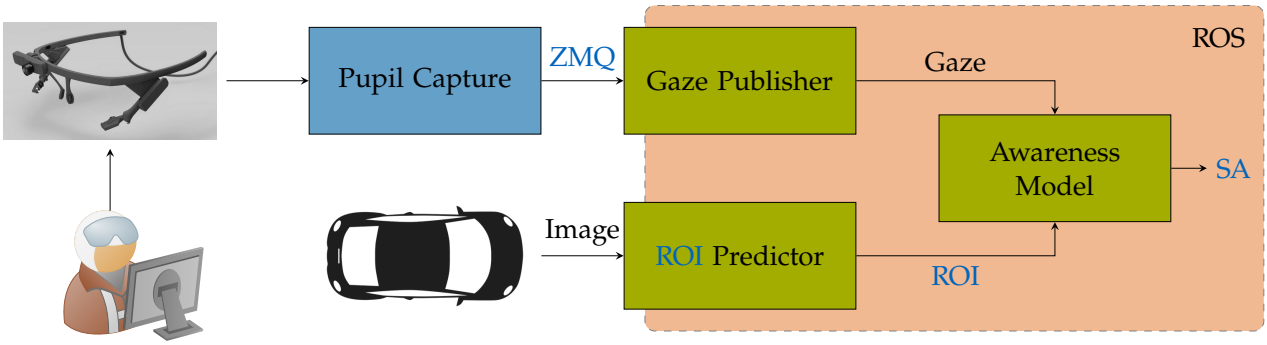


Figure A.3 Architecture of the proposed driver situation awareness approach. ROI prediction defines the optimal SA which is compared to the driver's actual SA captured by the eye-tracking device. Adopted from [22] © 2020 IEEE.

We use the *Pupil Core* wearable eye-tracking device for capturing the driver's gaze [178]. The *Pupil Core* ships with a standalone software and Python application programming interface (API) for communicating with the eye-tracking device. The communication between the Pupil software and the Python API is done via zero message queue (ZMQ). The ZMQ-based connection between Python API and Pupil software allows for running the Pupil software on a separate machine. Running Pupil on a second machine is particularly useful if less powerful hardware is available, allowing to maintain the maximum gaze frequency of 120 Hz. The gaze publisher module provides an interface for the eye-tracking device implemented using the Pupil Python API. The interface provided by the gaze publisher allows for a straightforward integration of other eye-tracking devices as well.

List of Figures

List of Tables

List of Publications

Journal Articles

[24] Markus Hofbauer, Christopher B. Kuhn, Goran Petrovic, and Eckehard Steinbach. "Preprocessor Rate Control for Adaptive Multi-View Live Video Streaming Using a Single Encoder". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2022), pp. 1–16. DOI: 10.1109/TCSVT.2022.3142403.

[188] Christopher B Kuhn, Markus Hofbauer, Goran Petrovic, and Eckehard Steinbach. "Introspective Failure Prediction for Autonomous Driving Using Late Fusion of State and Camera Information". In: *IEEE Transactions on Intelligent Transportation Systems* (2020), pp. 1–15. DOI: 10.1109/TITS.2020.3044813.

[203] Markus Hofbauer, Christoph Bachhuber, Christopher Kuhn, Sebastian Schwarz, Bart Kroon, and Eckehard Steinbach. "Large-Scale Collaborative Writing: Technical Challenges and Recommendations". In: *Proceedings of the ACM on Human-Computer Interaction* (2023), Submitted for review.

[205] Furkan Kaynar, Markus Hofbauer, Asa MacWilliams, Joseph Newman, Andreas Hutter, and Eckehard Steinbach. "AR in VR: Omnistereoscopic Telepresence with Holograms for Remote Maintenance and Collaboration". In: *IADIS International Journal on WWW/Internet* 20.2 (2022), pp. 99–116. ISSN: 1645-7641.

Conference Proceedings

[17] Simon Hoffmann, Felix Willert, Markus Hofbauer, Andreas Schimpe, and Frank Diermeyer. "Quantifying the Influence of Image Quality on Operator Reaction Times for Teleoperated Road Vehicles". In: *13th International Conference on Applied Human Factors and Ergonomics (AHFE 2022)*. New York, USA, July 2022. DOI: 10.54941/ahfe1002322.

[19] Markus Hofbauer, Christopher Kuhn, Goran Petrovic, and Eckehard Steinbach. "TELECARLA: An Open Source Extension of the CARLA Simulator for Teleoperated Driving Research Using Off-the-Shelf Components". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. Las Vegas, NV, USA, 2020, pp. 335–340. DOI: 10.1109/IV47402.2020.9304676.

[20] Markus Hofbauer, Christopher B. Kuhn, Jiaming Meng, Goran Petrovic, and Eckehard Steinbach. "Multi-View Region of Interest Prediction for Autonomous Driving Using Semi-Supervised Labeling". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. Rhodes, Greece, Sept. 2020, pp. 1–6. DOI: 10.1109/ITSC45102.2020.9294387.

[21] Markus Hofbauer, Christopher Kuhn, Goran Petrovic, and Eckehard Steinbach. "Adaptive Multi-View Live Video Streaming for Teledriving Using a Single Hardware Encoder". In: *2020 IEEE International Symposium on Multimedia (ISM)*. Naples, Italy, 2020, pp. 9–16. DOI: 10.1109/ISM.2020.00008.

[22] Markus Hofbauer, Christopher Kuhn, Lukas Püttner, Goran Petrovic, and Eckehard Steinbach. "Measuring Driver Situation Awareness Using Region-of-Interest Prediction and Eye Tracking". In: *2020 IEEE International Symposium on Multimedia (ISM)*. Naples, Italy, 2020, pp. 91–95. DOI: 10.1109/ISM.2020.00022.

[23] Markus Hofbauer, Christopher Kuhn, Mariem Khlifi, Goran Petrovic, and Eckehard Steinbach. "Traffic-Aware Multi-View Video Stream Adaptation for Teleoperated Driving". In: *2022 IEEE 95th Vehicular Technology Conference: VTC2022-Spring*. Helsinki, Finland, June 2022, pp. 1–8. DOI: 10.1109/VTC2022-Spring54318.2022.9860513.

[25] Markus Hofbauer, Christopher Kuhn, Goran Petrovic, and Eckehard Steinbach. "Measuring the Influence of Image Preprocessing on the Encoder Rate-Distortion Performance". In: *24th IEEE International Symposium on Multimedia (ISM)*. Naples, Italy, 2022.

[187] Christopher B Kuhn, Markus Hofbauer, Goran Petrovic, and Eckehard Steinbach. "Introspective Black Box Failure Prediction for Autonomous Driving". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. Las Vegas, NV, USA, 2020, pp. 1907–1913. DOI: 10.1109/IV47402.2020.9304844.

[189] Christopher B. Kuhn, Markus Hofbauer, Goran Petrovic, and Eckehard Steinbach. "Trajectory-Based Failure Prediction for Autonomous Driving". In: *2021 IEEE Intelligent Vehicles Symposium (IV)*. Nagoya, Japan, 2021, pp. 980–986. DOI: 10.1109/IV48863.2021.9575937.

[202] Markus Hofbauer, Christoph Bachhuber, Christopher Kuhn, and Eckehard Steinbach. "Teaching Software Engineering As Programming Over Time". In: *IEEE/ACM 4th International Workshop on Software Engineering Education for the Next Generation*. Pittsburgh, PA, USA: Association for Computing Machinery, May 2022, pp. 1–8. ISBN: 978-1-4503-9336-2/22/05. DOI: 10.1145/3528231.3528353. URL: https://doi.org/10.1145/3528231.3528353.

[204] Furkan Kaynar, Markus Hofbauer, Asa MacWilliams, Joseph Newman, Andreas Hutter, and Eckehard Steinbach. "AR in VR: Augmented Reality Cues in 360 Degree Stereoscopic Telepresence for Remote Collaboration and Maintenance". In: *16th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing 2022 (CGVCVIP 2022)*. Lisbon, Portugal, July 2022.

[206] Andreas Noll, Markus Hofbauer, Evelyn Muschter, Shu-Chen Li, and Eckehard Steinbach. "Automated Quality Assessment for Compressed Vibrotactile Signals Using Multi-Method Assessment Fusion". In: *IEEE Haptics Symposium 2022*. Santa Barabara, California, USA, 2022. DOI: 10.1109/HAPTICS52432.2022.9765599.

[207] Kristian Fischer, Markus Hofbauer, Christopher Kuhn, Eckehard Steinbach, and Andre Kaup. "Evaluation of Video Coding for Machines without Ground Truth". In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 1616–1620. DOI: 10.1109/ICASSP43922.2022.9747633.

[208] Christopher B Kuhn, Hofbauer Markus, Sungkyu Lee, Goran Petrovic, and Eckehard Steinbach. "Introspective Failure Prediction for Semantic Image Segmentation". In: Rhodes, Greece, Sept. 2020, pp. 1–6. DOI: 10.1109/ITSC45102.2020.9294308.

[209] Christopher B Kuhn, Markus Hofbauer, Goran Petrovic, and Eckehard Steinbach. "Better Look Twice - Improving Visual Scene Perception Using a Two-Stage Approach". In: *2020 IEEE International Symposium on Multimedia (ISM)*. Naples, Italy, Dec. 2020, pp. 33–40. DOI: 10.1109/ISM.2020.00013.

[210] Christopher B. Kuhn, Markus Hofbauer, Ziqin Xu, Goran Petrovic, and Eckehard Steinbach. "Pixel-Wise Failure Prediction For Semantic Video Segmentation". In: *2021 IEEE International Conference on Image Processing (ICIP)*. Anchorage, AK, USA, Sept. 2021, pp. 614–618. DOI: 10.1109/ICIP42928.2021.9506552.

[211] Christopher B Kuhn, Markus Hofbauer, Bowen Ma, Goran Petrovic, and Eckehard Steinbach. "Improving Multimodal Object Detection with Individual Sensor Monitoring". In: *24th IEEE International Symposium on Multimedia (ISM)*. Naples, Italy, 2022.

[212] Christopher B. Kuhn, Markus Hofbauer, Goran Petrovic, and Eckehard Steinbach. "Reverse Error Modeling for Improved Semantic Segmentation". In: *2022 IEEE International Conference on Image Processing (ICIP)*. 2022, pp. 106–110. DOI: 10.1109/ICIP46576.2022.9897331.

[213] Lukas Habermayr, Markus Hofbauer, Joao-Vitor Zacchi, and Christopher B. Kuhn. "Situation-Aware Model Refinement for Semantic Image Segmentation". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. Indianapolis, IN, USA, 2021, pp. 2696–2702. DOI: 10.1109/ITSC48978.2021.9564549.

[214] Driton Salihu, Adam Misik, Markus Hofbauer, and Eckehard Steinbach. "Multi-Method Assessment Fusion for Scan-to-CAD Methods". In: *24th IEEE International Symposium on Multimedia*. Dec. 2022.

Bibliography

[1] Lei Kang, Wei Zhao, Bozhao Qi, and Suman Banerjee. "Augmenting self-driving with remote control: Challenges and directions". In: *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*. 2018, pp. 19–24.

[2] Marco Galvani. "History and future of driver assistance". In: *IEEE Instrumentation & Measurement Magazine* 22.1 (2019), pp. 11–16.

[3] Junyao Guo, Unmesh Kurup, and Mohak Shah. "Is It Safe to Drive? An Overview of Factors, Metrics, and Datasets for Driveability Assessment in Autonomous Driving". In: *IEEE Transactions on Intelligent Transportation Systems* (2019).

[4] Natasha Merat, A. Hamish Jamson, Frank C. H. Lai, Michael Daly, and Oliver M. J. Carsten. "Transition to manual: Driver behaviour when resuming control from a highly automated vehicle". In: *Transportation Research Part F: Traffic Psychology and Behaviour*. Vehicle Automation and Driver Behaviour 27 (Nov. 2014), pp. 274–282. ISSN: 1369-8478. DOI: 10.1016/j.trf.2014.09.005. URL: http://www.sciencedirect.com/science/article/pii/S1369847814001284 (visited on 05/02/2019).

[5] Alex Davis. "Nissan's Self-Driving Car Solution Relies on Human-Operated Call Centers | WIRED". In: (Jan. 2017). URL: https://www.wired.com/2017/01/nissans-self-driving-teleoperation/ (visited on 05/19/2020).

[6] Alex Davis. "The War to Remotely Control Self-Driving Cars Heats Up". en. In: *Wired* (Mar. 2019). ISSN: 1059-1028. URL: https://www.wired.com/story/designated-driver-teleoperations-self-driving-cars/ (visited on 05/19/2020).

[7] Sebastian Gnatzig., Frederic Chucholowski., Tito Tang., and Markus Lienkamp. "A System Design for Teleoperated Road Vehicles". In: *Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO,* INSTICC. SciTePress, 2013, pp. 231–238. ISBN: 978-989-8565-71-6. DOI: 10.5220/0004475802310238.

[8] Jean-Michael Georg, Johannes Feiler, Frank Diermeyer, and Markus Lienkamp. "Teleoperated Driving, a Key Technology for Automated Driving? Comparison of Actual Test Drives with a Head Mounted Display and Conventional Monitors*". en. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI: IEEE, Nov. 2018, pp. 3403–3408. ISBN: 978-1-72810-321-1 978-1-72810-323-5. DOI: 10.1109/ITSC.2018.8569408. URL: https://ieeexplore.ieee.org/document/8569408/ (visited on 07/17/2019).

[9] Amin Hosseini and Markus Lienkamp. "Predictive safety based on track-before-detect for teleoperated driving through communication time delay". en. In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. Gotenburg, Sweden: IEEE, June 2016, pp. 165–172. ISBN: 978-1-5090-1821-5. DOI: 10.1109/IVS.2016.7535381. URL: http://ieeexplore.ieee.org/document/7535381/ (visited on 05/04/2019).

[10] Stefan Neumeier, Philipp Wintersberger, Anna-Katharina Frison, Armin Becher, Christian Facchi, and Andreas Riener. "Teleoperation: The Holy Grail to Solve Problems of Automated Driving? Sure, but Latency Matters". en. In: *Proceedings of the 11th International Conference on Automotive User Interfaces and Interactive Vehicular Applications - AutomotiveUI '19*. Utrecht, Netherlands: ACM Press, 2019, pp. 186–197. ISBN: 978-1-4503-6884-1. DOI: 10.1145/3342197.3344534. URL: http://dl.acm.org/citation.cfm?doid=3342197.3344534 (visited on 05/11/2020).

[11] Amin Hosseini and Markus Lienkamp. "Enhancing telepresence during the teleoperation of road vehicles using HMD-based mixed reality". en. In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. Gotenburg, Sweden: IEEE, June 2016, pp. 1366–1373. ISBN: 978-1-5090-1821-5. DOI: 10.1109/IVS.2016.7535568. URL: http://ieeexplore.ieee.org/document/7535568/ (visited on 05/04/2019).

[12] James Davis, Christopher Smyth, and Kaleb McDowell. "The Effects of Time Lag on Driving Performance and a Possible Mitigation". en. In: *IEEE Transactions on Robotics* 26.3 (June 2010), pp. 590–593. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2010.2046695. URL: http://ieeexplore.ieee.org/document/5460896/ (visited on 08/07/2019).

[13] Amin Hosseini, Florian Richthammer, and Markus Lienkamp. "Predictive Haptic Feedback for Safe Lateral Control of Teleoperated Road Vehicles in Urban Areas". en. In: *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*. Nanjing, China: IEEE, May 2016, pp. 1–7. ISBN: 978-1-5090-1698-3. DOI: 10.1109/VTCSpring.2016.7504430. URL: http://ieeexplore.ieee.org/document/7504430/ (visited on 07/29/2019).

[14] S. Neumeier, E. A. Walelgne, V. Bajpai, J. Ott, and C. Facchi. "Measuring the Feasibility of Teleoperated Driving in Mobile Networks". In: (2019), pp. 113–120. DOI: 10.23919/TMA.2019.8784466.

[15] Ayman Gaber, Waleed Nassar, Ahmed Maher Mohamed, and Moataz Kamel Mansour. "Feasibility Study of Teleoperated Vehicles Using Multi-Operator LTE Connection". en. In: *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*. Aswan, Egypt: IEEE, Feb. 2020, pp. 191–195. ISBN: 978-1-72814-801-4. DOI: 10.1109/ITCE48509.2020.9047764. URL: https://ieeexplore.ieee.org/document/9047764/ (visited on 04/28/2020).

[16] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. "CARLA: An Open Urban Driving Simulator". en. In: *arXiv:1711.03938 [cs]* (Nov. 2017). URL: http://arxiv.org/abs/1711.03938 (visited on 05/03/2019).

[18] Stefan Neumeier, Simon Stapf, and Christian Facchi. "The Visual Quality of Teleoperated Driving Scenarios How good is good enough?" en. In: *2020 International Symposium on Networks, Computers and Communications (ISNCC)*. Montreal, QC, Canada: IEEE, Oct. 2020, pp. 1–8. ISBN: 978-1-72815-628-6. DOI: 10.1109/ISNCC49221.2020.9297343. URL: https://ieeexplore.ieee.org/document/9297343/ (visited on 03/24/2021).

[26] Marvin Minsky. "Telepresence". In: (1980).

[27] Terrence Fong and Charles Thorpe. "Vehicle teleoperation interfaces". In: *Autonomous robots* 11.1 (2001), pp. 9–18.

[28] Bill Ross, John Bares, David Stager, Larry Jackel, and Mike Perschbacher. "An Advanced Teleoperation Testbed". en. In: vol. 42. Springer, July 2007. URL: https://hal.inria.fr/inria-00272962/document (visited on 01/23/2019).

[29] Zeashan Hameed Khan, Jean-Marc Thiriet, and Denis Genon-Catalot. "Drive-by-Wireless Teleoperation with Network QoS Adaptation". en. In: (2011), p. 11.

[30] Xiaotong Shen et al. "Teleoperation of On-Road Vehicles via Immersive Telepresence Using Off-the-shelf Components". en. In: *Intelligent Autonomous Systems 13*. Ed. by Emanuele Menegatti, Nathan Michael, Karsten Berns, and Hiroaki Yamaguchi. Vol. 302. Cham: Springer International Publishing, 2016, pp. 1419–1433. ISBN: 978-3-319-08337-7 978-3-319-08338-4. DOI: 10.1007/978-3-319-08338-4_102. URL: http://link.springer.com/10.1007/978-3-319-08338-4_102 (visited on 01/29/2019).

[31] On-Road Automated Driving (ORAD) committee. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Sept. 2016. DOI: https://doi.org/10.4271/J3016_201609. URL: https://doi.org/10.4271/J3016_201609.

[32] 5GAA Automotive Association. *Tele-operated Driving (ToD): Use Cases and Technical Requirements*. Technical Report. 2020. URL: https://5gaa.org/wp-content/uploads/2021/08/ToD_D1.1-Use-Cases-and-Technical-Requirements.pdf (visited on 12/13/2021).

[33] WIVW GmbH. *Driving Simulation and SILAB*. URL: https://wivw.de/en/silab (visited on 10/29/2019).

[34] Aitor Ruano. *DeepGTAV v2*. Accessed on: 2021-11-27. URL: https://github.com/aitorzip/DeepGTAV.

[35] Craig Quiter. *Deepdrive*. en. 2017. URL: https://deepdrive.io/ (visited on 10/29/2019).

[36] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles". In: (2018). Ed. by Marco Hutter and Roland Siegwart, pp. 621–635.

[37] Josephine Mertens. "Generating Data to Train a Deep Neural Network End-To-End within a Simulated Environment". en. In: (2018), p. 48.

[38] Guodong Rong et al. "LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving". In: (2020), pp. 1–6. DOI: 10.1109/ITSC45102.2020.9294422.

[39] *Baidu Apollo team (2017), Apollo: Open Source Autonomous Driving*. https://github.com/ApolloAuto/apollo. Accessed: 2011-11-30.

[40] Shinpei Kato et al. "Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems". In: *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. 2018, pp. 287–296. DOI: 10.1109/ICCPS.2018.00035.

[41] Epic Games. *Unreal Engine*. en. URL: https://www.unrealengine.com/en-US/ (visited on 12/12/2021).

[42] Stevan Stevic, Momcilo Krunic, Marko Dragojevic, and Nives Kaprocki. "Development and Validation of ADAS Perception Application in ROS Environment Integrated with CARLA Simulator". en. In: *2019 27th Telecommunications Forum (TELFOR)*. Belgrade, Serbia: IEEE, Nov. 2019, pp. 1–4. ISBN: 978-1-72814-790-1. DOI: 10.1109/TELFOR48224.2019.8971063. URL: https://ieeexplore.ieee.org/document/8971063/ (visited on 05/11/2020).

[43] CARLA Team. *CARLA Autonomous Driving Leaderboard*. en. URL: https://leaderboard.carla.org/ (visited on 12/13/2021).

[44] Oscar Bodell and Erik Gullikson. *Teleoperation of Autonomous Vehicle With 360° Camera Feedback*. Tech. rep. 2016. URL: http://publications.lib.chalmers.se/records/fulltext/244866/244866.pdf (visited on 01/29/2019).

[45] Morgan Quigley et al. "ROS: an open-source Robot Operating System". en. In: (2009), p. 6.

[46] N. Koenig and A. Howard. "Design and use paradigms for Gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[47] Andre-Marcel Hellmund, Sascha Wirges, Omer Sahin Tas, Claudio Bandera, and Niels Ole Salscheider. "Robot operating system: A modular software framework for automated driving". en. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. Rio de Janeiro, Brazil: IEEE, Nov. 2016, pp. 1564–1570. ISBN: 978-1-5090-1889-5. DOI: 10.1109/ITSC.2016.7795766. URL: http://ieeexplore.ieee.org/document/7795766/ (visited on 12/11/2019).

[48] Stefan Neumeier, Michael Hopp, and Christian Facchi. "Yet Another Driving Simulator Open-ROUTS3D: The Driving Simulator for Teleoperated Driving". en. In: *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. Graz, Austria: IEEE, Nov. 2019, pp. 1–6. ISBN: 978-1-72810-142-2. DOI: 10.1109/ICCVE45908.2019.8965037. URL: https://ieeexplore.ieee.org/document/8965037/ (visited on 05/11/2020).

[49] Unity Technologies. *Unity*. en. URL: https://unity.com/ (visited on 12/12/2021).

[50] Daniel Labonte, Patrick Boissy, and François Michaud. "Comparative Analysis of 3-D Robot Teleoperation Interfaces With Novice Users". en. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 40.5 (Oct. 2010), pp. 1331–1342. ISSN: 1083-4419, 1941-0492. DOI: 10.1109/TSMCB.2009.2038357. URL: http://ieeexplore.ieee.org/document/5398937/ (visited on 03/06/2019).

[51] Terrence Fong, Charles Thorpe, and Charles Baur. "Advanced Interfaces for Vehicle Teleoperation: Collaborative Control, Sensor Fusion Displays, and Remote Driving Tools". en. In: (2001), p. 9.

[52] Tito Tang, Jan Kurkowski, and Markus Lienkamp. "Teleoperated Road Vehicles: A Novel Study on the Effect of Blur on Speed Perception". en. In: *International Journal of Advanced Robotic Systems* 10.9 (Sept. 2013), p. 333. ISSN: 1729-8814, 1729-8814. DOI: 10.5772/56735. URL: http://journals.sagepub.com/doi/10.5772/56735 (visited on 12/14/2021).

[53] Frederic Chucholowski, Stefan Buchner, Johannes Reicheneder, and Markus Lienkamp. "Prediction Methods for Teleoperated Road Vehicles". en. In: (2013), p. 7.

[54] Frederic Chucholowski. "Evaluation of Display Methods for Teleoperation of Road Vehicles". In: *Journal of Unmanned System Technology* 3 (Feb. 2016), pp. 80–85. DOI: 10.21535/just.v3i3.38.

[55] Jean-Michael Georg and Frank Diermeyer. "An Adaptable and Immersive Real Time Interface for Resolving System Limitations of Automated Vehicles with Teleoperation". en. In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. Bari, Italy: IEEE, Oct. 2019, pp. 2659–2664. ISBN: 978-1-72814-569-3. DOI: 10.1109/SMC.2019.8914306. URL: https://ieeexplore.ieee.org/document/8914306/ (visited on 03/21/2020).

[56] Jean-Michael Georg, Elena Putz, and Frank Diermeyer. "Longtime Effects of Videoquality, Videocanvases and Displays on Situation Awareness during Teleoperation of Automated Vehicles *". en. In: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Toronto, ON, Canada: IEEE, Oct. 2020, pp. 248–255. ISBN: 978-1-72818-526-2. DOI: 10.1109/SMC42975.2020.9283364. URL: https://ieeexplore.ieee.org/document/9283364/ (visited on 07/15/2021).

[57] Parham M. Kebria, Abbas Khosravi, Saeid Nahavandi, Peng Shi, and Roohallah Alizadehsani. "Robust Adaptive Control Scheme for Teleoperation Systems With Delay and Uncertainties". en. In: *IEEE Transactions on Cybernetics* (2019), pp. 1–11. ISSN: 2168-2267, 2168-2275. DOI: 10.1109/TCYB.2019.2891656. URL: https://ieeexplore.ieee.org/document/8624418/ (visited on 05/02/2019).

[58] Parham M. Kebria, Hamid Abdi, Mohsen Moradi Dalvand, Abbas Khosravi, and Saeid Nahavandi. "Control Methods for Internet-Based Teleoperation Systems: A Review". en. In: *IEEE Transactions on Human-Machine Systems* 49.1 (Feb. 2019), pp. 32–46. ISSN: 2168-2291, 2168-2305. DOI: 10.1109/THMS.2018.2878815. URL: https://ieeexplore.ieee.org/document/8536440/ (visited on 05/02/2019).

[59] Tito Tang, Pascal Vetter, Simon Finkl, Korbinian Figel, and Markus Lienkamp. "Teleoperated Road Vehicles – The "Free Corridor" as a Safety Strategy Approach". en. In: *Applied Mechanics and Materials* 490-491 (Jan. 2014), pp. 1399–1409. ISSN: 1662-7482. DOI: 10.4028/www.scientific.net/AMM.490-491.1399. URL: https://www.scientific.net/AMM.490-491.1399 (visited on 12/14/2021).

[60] Ali Gohar, Ali Raza, and Sanghwan Lee. "A Distributed Remote Driver Selection for Cost Efficient and Safe Driver Handover". en. In: *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. Jeju: IEEE, Oct. 2018, pp. 801–804. ISBN: 978-1-5386-5041-7. DOI: 10.1109/ICTC.2018.8539583. URL: https://ieeexplore.ieee.org/document/8539583/ (visited on 07/17/2019).

[61] Pedro d'Orey, Amin Hosseini, Jose Azevedo, Frank Diermeyer, Michel Ferreira, and Markus Lienkamp. "Hail-a-Drone: Enabling teleoperated taxi fleets". In: June 2016. DOI: 10.1109/IVS.2016.7535475.

[62] Itai Dror, Raz Birman, and Ofer Hadar. "Content adaptive video compression for autonomous vehicle remote driving". en. In: *Applications of Digital Image Processing XLIV*. Ed. by Andrew G. Tescher and Touradj Ebrahimi. San Diego, United States: SPIE, Aug. 2021, p. 29. DOI: 10.1117/12.2595863. URL: https://doi.org/10.1117/12.2595863 (visited on 10/17/2021).

[63] Yucheng Yang, Martin Götze, Annika Laqua, Giancarlo Caccia Dominioni, Kyosuke Kawabe, and Klaus Bengler. "A method to improve driver's situation awareness in automated driving". en. In: (2017), p. 19.

[64] Joonwoo Son and Myoungouk Park. "Situation Awareness and Transitions in Highly Automated Driving: A Framework and Mini Review". en. In: *Journal of Ergonomics* 07.05 (2017). ISSN: 21657556. DOI: 10.4172/2165-7556.1000212.

[65] Monika Lohani, Brennan R. Payne, and David L. Strayer. "A Review of Psychophysiological Measures to Assess Cognitive States in Real-World Driving". en. In: *Frontiers in Human Neuroscience* 13 (Mar. 2019), p. 57. ISSN: 1662-5161. DOI: 10.3389/fnhum.2019.00057. URL: https://www.frontiersin.org/article/10.3389/fnhum.2019.00057/full (visited on 05/15/2020).

[66] Arie P. van den Beukel and Mascha C. van der Voort. "The influence of time-criticality on Situation Awareness when retrieving human control after automated driving". en. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. The Hague, Netherlands: IEEE, Oct. 2013, pp. 2000–2005. ISBN: 978-1-4799-2914-6. DOI: 10.1109/ITSC.2013.6728523. URL: http://ieeexplore.ieee.org/document/6728523/ (visited on 06/24/2020).

[67] Jennifer M. Riley, David B. Kaber, and John V. Draper. "Situation awareness and attention allocation measures for quantifying telepresence experiences in teleoperation". en. In: *Human Factors and Ergonomics in Manufacturing* 14.1 (2004), pp. 51–67. ISSN: 1090-8471, 1520-6564. DOI: 10.1002/hfm.10050. URL: http://doi.wiley.com/10.1002/hfm.10050 (visited on 05/08/2019).

[68] M. R. Endsley. "Situation awareness global assessment technique (SAGAT)". In: *Proceedings of the IEEE 1988 National Aerospace and Electronics Conference*. 1988, 789–795 vol.3.

[69] Mica Endsley. "Endsley, M.R.: Toward a Theory of Situation Awareness in Dynamic Systems. Human Factors Journal 37(1), 32-64". In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 37 (Mar. 1995), pp. 32–64. DOI: 10.1518/001872095779049543.

[70] Paul Salmon, Neville Stanton, Guy Walker, and Damian Green. "Situation awareness measurement: A review of applicability for C4i environments". en. In: *Applied Ergonomics* 37.2 (Mar. 2006), pp. 225–238. ISSN: 0003-6870. DOI: 10.1016/j.apergo.2005.02.001. URL: http://www.sciencedirect.com/science/article/pii/S0003687005000554 (visited on 06/23/2020).

[71] Thanh Nguyen, Chee Peng Lim, Ngoc Duy Nguyen, Lee Gordon-Brown, and Saeid Nahavandi. "A Review of Situation Awareness Assessment Approaches in Aviation Environments". In: *IEEE Systems Journal* 13.3 (2019), pp. 3590–3603. DOI: 10.1109/JSYST.2019.2918283.

[72] DAVID HOGG, KNUT FOLLES, Frode Volden, and BELÉN TORRALBA. "Development of a situation awareness measure to evaluate advanced alarm systems in nuclear power plant control rooms". In: *Ergonomics* 38.11 (1995), pp. 2394–2413. DOI: 10.1080/00140139508925275. eprint: https://doi.org/10.1080/00140139508925275. URL: https://doi.org/10.1080/00140139508925275.

[73] Yorck Hauss and Klaus Eyferth. "Securing future ATM-concepts' safety by measuring situation awareness in ATC". In: *Aerospace Science and Technology* 7.6 (2003), pp. 417–427. ISSN: 1270-9638. DOI: https://doi.org/10.1016/S1270-9638(02)00011-1. URL: https://www.sciencedirect.com/science/article/pii/S1270963802000111.

[74] Francis T. Durso, Carla A. Hackworth, Todd R. Truitt, Jerry Crutchfield, Danko Nikolic, and Carol A. Manning. "Situation Awareness as a Predictor of Performance for En Route Air Traffic Controllers". In: *Air Traffic Control Quarterly* 6.1 (1998), pp. 1–20. DOI: 10.2514/atcq.6.1.1. eprint: https://doi.org/10.2514/atcq.6.1.1. URL: https://doi.org/10.2514/atcq.6.1.1.

[75] E. Jeannot, D. Thompson, and C. Kelly. "The Development of Situation Awareness Measures in ATM Systems". In: 2003.

[76] Richard M Taylor. "Situational awareness rating technique (SART): The development of a tool for aircrew systems design". In: *Situational awareness*. Routledge, 2017, pp. 111–128.

[77] Wayne L Waag and Michael R Houck. "Tools for assessing situational awareness in an operational fighter environment." In: *Aviation, space, and environmental medicine* (1994).

[78] K Dennehy. *Cranfield situation awareness scale*. Tech. rep. 1997.

[79] Michael D Matthews, Robert J Pleban, Mica R Endsley, and Laura D Strater. "Measures of infantry situation awareness for a virtual MOUT environment". In: *Proceedings of the Human Performance, Situation Awareness and Automation: User Centred Design for the New Millennium Conference*. 2000.

[80] Michael D Matthews and Scott A Beal. *Assessing situation awareness in field training exercises*. Tech. rep. Military Academy West Point NY Office of Military Psychology and Leadership, 2002.

[81] Barry McGuinness. *Quantitative analysis of situational awareness (QUASA): Applying signal detection theory to true/false probes and self-ratings*. Tech. rep. BAE SYSTEMS BRISTOL (UNITED KINGDOM) ADVANCED TECHNOLOGY CENTRE, 2004.

[82] MW Smolensky. "Toward the physiological measurement of situation awareness: The case for eye movement measurements". In: *Proceedings of the Human Factors and Ergonomics Society 37th annual meeting*. Vol. 41. Human Factors and Ergonomics Society Santa Monica. 1993.

[83] Nikolas Martelaro, David Sirkin, and Wendy Ju. "DAZE: a real-time situation awareness measurement tool for driving". en. In: *Adjunct Proceedings of the 7th International Conference on Automotive User Interfaces and Interactive Vehicular Applications - AutomotiveUI '15*. Nottingham, United Kingdom: ACM Press, 2015, pp. 158–163. ISBN: 978-1-4503-3858-5. DOI: 10.1145/2809730.2809753. URL: http://dl.acm.org/citation.cfm?doid=2809730.2809753 (visited on 03/06/2019).

[84] David Sirkin, Nikolas Martelaro, Mishel Johns, and Wendy Ju. "Toward Measurement of Situation Awareness in Autonomous Vehicles". en. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. Denver, Colorado, USA: ACM Press, 2017, pp. 405–415. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025822. URL: http://dl.acm.org/citation.cfm?doid=3025453.3025822 (visited on 03/06/2019).

[85] Arnaud Desvergez, Arnaud Winer, Jean-Bernard Gouyon, and Médéric Descoins. "An observational study using eye tracking to assess resident and senior anesthetists' situation awareness and visual perception in postpartum hemorrhage high fidelity simulation". en. In: *PLOS ONE* 14.8 (Aug. 2019). Ed. by Andrea Cortegiani, e0221515. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0221515. URL: https://dx.plos.org/10.1371/journal.pone.0221515 (visited on 05/30/2020).

[86] J. C. F. de Winter, Y. B. Eisma, C. D. D. Cabrall, P. A. Hancock, and N. A. Stanton. "Situation awareness based on eye movements in relation to the task environment". en. In: *Cognition, Technology & Work* 21.1 (Feb. 2019), pp. 99–111. ISSN: 1435-5566. DOI: 10.1007/s10111-018-0527-6. URL: https://doi.org/10.1007/s10111-018-0527-6 (visited on 05/30/2020).

[87] Hasanzadeh Sogand, Esmaeili Behzad, and Dodd Michael D. "Measuring Construction Workers? Real-Time Situation Awareness Using Mobile Eye-Tracking". In: *Construction Research Congress 2016*. Proceedings (2016), pp. 2894–2904. DOI: 10.1061/9780784479827.288. URL: https://ascelibrary.org/doi/10.1061/9780784479827.288 (visited on 05/30/2020).

[88] Koen van de Merwe, Henk Dijk, and Rolf Zon. "Eye Movements as an Indicator of Situation Awareness in a Flight Simulator Experiment". In: *The International Journal of Aviation Psychology* 22 (Jan. 2012), pp. 78–95. DOI: 10.1080/10508414.2012.635129.

[89] Leo Gugerty. "Situation Awareness in Driving". en. In: *Handbook of Driving Simulation for Engineering, Medicine, and Psychology*. Ed. by John Lee. CRC Press, Apr. 2011. ISBN: 978-1-4200-6100-0 978-1-4200-6101-7. DOI: 10.1201/b10836-20. URL: http://www.crcnetbase.com/doi/10.1201/b10836-20 (visited on 03/06/2019).

[90] Mariusz Bojarski et al. "VisualBackProp: Efficient Visualization of CNNs for Autonomous Driving". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 4701–4708. DOI: 10.1109/ICRA.2018.8461053.

[91] Kelvin Xu et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 2048–2057.

[92] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. "Recurrent Models of Visual Attention". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, 2014, pp. 2204–2212.

[93] Marcella Cornia, Lorenzo Baraldi, Giuseppe Serra, and Rita Cucchiara. "Predicting human eye fixations via an lstm-based saliency attentive model". In: *IEEE Transactions on Image Processing* 27.10 (2018), pp. 5142–5154.

[94] A. Palazzi, D. Abati, S. Calderara, F. Solera, and R. Cucchiara. "Predicting the Driver's Focus of Attention: The DR(eye)VE Project". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 41.07 (July 2019), pp. 1720–1733. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2018.2845370.

[95] Ye Xia, Danqing Zhang, Jinkyu Kim, Ken Nakayama, Karl Zipser, and David Whitney. "Predicting driver attention in critical situations". In: *Asian Conference on Computer Vision*. Springer. 2018, pp. 658–674.

[96] Anwesan Pal, Sayan Mondal, and Henrik I. Christensen. ""Looking at the Right Stuff" - Guided Semantic-Gaze for Autonomous Driving". In: (2020), pp. 11880–11889. DOI: 10.1109/CVPR42600.2020.01190.

[97] Pierluigi Vito Amadori, Tobias Fischer, Ruohan Wang, and Yiannis Demiris. "Decision Anticipation for Driving Assistance Systems". In: June 2020.

[98] Erik Murphy-Chutorian and Mohan Manubhai Trivedi. "Head Pose Estimation and Augmented Reality Tracking: An Integrated System and Evaluation for Monitoring Driver Awareness". en. In: *IEEE Transactions on Intelligent Transportation Systems* 11.2 (June 2010), pp. 300–311. ISSN: 1524-9050, 1558-0016. DOI: 10.1109/TITS.2010.2044241. URL: http://ieeexplore.ieee.org/document/5443483/ (visited on 06/24/2020).

[99] A. Tawari, S. Sivaraman, M. M. Trivedi, T. Shannon, and M. Tippelhofer. "Looking-in and looking-out vision for Urban Intelligent Assistance: Estimation of driver attentive state and dynamic surround for safe merging and braking". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. 2014, pp. 115–120.

[100] L. Fletcher, L. Petersson, N. Barnes, D. Austin, and A. Zelinsky. "A Sign Reading Driver Assistance System Using Eye Gaze". en. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. Barcelona, Spain: IEEE, 2005, pp. 4655–4660. ISBN: 978-0-7803-8914-4. DOI: 10.1109/ROBOT.2005.1570838. URL: http://ieeexplore.ieee.org/document/1570838/ (visited on 06/23/2020).

[101] Masataka Mori et al. "Measuring driver awareness based on correlation between gaze behavior and risks of surrounding vehicles". en. In: *2012 15th International IEEE Conference on Intelligent Transportation Systems*. Anchorage, AK, USA: IEEE, Sept. 2012, pp. 644–647. ISBN: 978-1-4673-3063-3 978-1-4673-3064-0 978-1-4673-3062-6. DOI: 10.1109/ITSC.2012.6338802. URL: http://ieeexplore.ieee.org/document/6338802/ (visited on 06/02/2020).

[102] Tobias Langner, Daniel Seifert, Bennet Fischer, Daniel Goehring, Tinosch Ganjineh, and Raul Rojas. "Traffic awareness driver assistance based on stereovision, eye-tracking, and head-up display". en. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm: IEEE, May 2016, pp. 3167–3173. ISBN: 978-1-4673-8026-3. DOI: 10.1109/ICRA.2016.7487485. URL: http://ieeexplore.ieee.org/document/7487485/ (visited on 01/29/2020).

[103] Becky L. Hooey et al. "Modeling Pilot Situation Awareness". In: *Human Modelling in Assisted Transportation*. Ed. by P. Carlo Cacciabue, Magnus Hjälmdahl, Andreas Luedtke, and Costanza Riccioli. Milano: Springer Milan, 2011, pp. 207–213. ISBN: 978-88-470-1821-1.

[104] Liu Shuang, Wanyan Xiaoru, and Zhuang Damin. "A Quantitative Situational Awareness Model of Pilot". en. In: *Proceedings of the International Symposium on Human Factors and Ergonomics in Health Care* 3.1 (June 2014), pp. 117–122. ISSN: 2327-8595, 2327-8595. DOI: 10.1177/2327857914031019. URL: http://journals.sagepub.com/doi/10.1177/2327857914031019 (visited on 05/02/2019).

[105] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. "Overview of the H.264/AVC Video Coding Standard". In: *IEEE Trans. Cir. and Sys. for Video Technol.* 13.7 (July 2003), pp. 560–576. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2003.815165. URL: https://doi.org/10.1109/TCSVT.2003.815165.

[106] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand. "Overview of the High Efficiency Video Coding (HEVC) Standard". In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (Dec. 2012), pp. 1649–1668. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2012.2221191.

[107] Benjamin Bross et al. "Overview of the Versatile Video Coding (VVC) Standard and its Applications". In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.10 (2021), pp. 3736–3764. DOI: 10.1109/TCSVT.2021.3101953.

[108] Ralf Schreier, A. Tushar Iqbal Rahman, Ganesh Krishnamurthy, and Albrecht Rothermel. "Architecture Analysis for Low-Delay Video Coding". en. In: *2006 IEEE International Conference on Multimedia and Expo*. Toronto, ON, Canada: IEEE, July 2006, pp. 2053–2056. ISBN: 978-1-4244-0367-7. DOI: 10.1109/ICME.2006.262618. URL: http://ieeexplore.ieee.org/document/4037034/ (visited on 01/18/2019).

[109] Christoph Bachhuber. "Low Delay Video Communication". PhD thesis. München: Technische Universität München, 2019.

[110] Fan Zhang and Eckehard Steinbach. "Improved Rho-domain rate control with accurate header size estimation". en. In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Prague, Czech Republic: IEEE, May 2011, pp. 813–816. ISBN: 978-1-4577-0538-0. DOI: 10.1109/ICASSP.2011.5946528. URL: http://ieeexplore.ieee.org/document/5946528/ (visited on 01/15/2019).

[111] Anthony Vetro, Thomas Wiegand, and Gary Sullivan. "Overview of the Stereo and Multiview Video Coding Extensions of the H.264/MPEG-4 AVC Standard". In: *Proceedings of the IEEE* 99 (May 2011), pp. 626–642. DOI: 10.1109/JPROC.2010.2098830.

[112] Gerhard Tech, Ying Chen, Karsten Muller, Jens-Rainer Ohm, Anthony Vetro, and Ye-Kui Wang. "Overview of the Multiview and 3D Extensions of High Efficiency Video Coding". en. In: *IEEE Transactions on Circuits and Systems for Video Technology* 26.1 (Jan. 2016), pp. 35–49. ISSN: 1051-8215, 1558-2205. DOI: 10.1109/TCSVT.2015.2477935. URL: http://ieeexplore.ieee.org/document/7258339/ (visited on 04/09/2020).

[113] Miska M. Hannuksela, Ye Yan, Xuehui Huang, and Houqiang Li. "Overview of the multiview high efficiency video coding (MV-HEVC) standard". en. In: *2015 IEEE International Conference on Image Processing (ICIP)*. Quebec City, QC, Canada: IEEE, Sept. 2015, pp. 2154–2158. ISBN: 978-1-4799-8339-1. DOI: 10.1109/ICIP.2015.7351182. URL: http://ieeexplore.ieee.org/document/7351182/ (visited on 04/09/2020).

[114] John Apostolopoulos. "Selecting bit rates for encoding multiple data streams". US20080025413A1. Jan. 2008. URL: https://patents.google.com/patent/US20080025413A1/en (visited on 10/23/2019).

[115] Agnes Y. Ngai and Edward F. Westermann. "Distributed control strategy for dynamically encoding multiple streams of video data in parallel for multiplexing onto a constant bit rate channel". en. US7085322B2. Aug. 2006. URL: https://patents.google.com/patent/US7085322B2/en (visited on 10/23/2019).

[116] Hang Zhang, Jiahao Li, Bin Li, and Yan Lu. "A Deep Reinforcement Learning Approach to Multiple Streams' Joint Bitrate Allocation". en. In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.6 (June 2021), pp. 2415–2426. ISSN: 1051-8215, 1558-2205. DOI: 10.1109/TCSVT.2020.3021489. URL: https://ieeexplore.ieee.org/document/9186037/ (visited on 07/19/2021).

[117] Thomas Stockhammer. "Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles". In: *Proceedings of the Second Annual ACM Conference on Multimedia Systems*. MMSys '11. New York, NY, USA: ACM, 2011, pp. 133–144. ISBN: 978-1-4503-0518-1. DOI: 10.1145/1943552.1943572. URL: http://doi.acm.org/10.1145/1943552.1943572 (visited on 01/22/2019).

[118] Abdelhak Bentaleb, Bayan Taani, Ali C. Begen, Christian Timmerer, and Roger Zimmermann. "A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP". en. In: *IEEE Communications Surveys & Tutorials* 21.1 (2019), pp. 562–585. ISSN: 1553-877X, 2373-745X. DOI: 10.1109/COMST.2018.2862938. URL: https://ieeexplore.ieee.org/document/8424813/ (visited on 07/08/2019).

[119] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. "BOLA: Near-optimal bitrate adaptation for online videos". en. In: *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. San Francisco, CA, USA: IEEE, Apr. 2016, pp. 1–9. ISBN: 978-1-4673-9953-1. DOI: 10.1109/INFOCOM.2016.7524428. URL: http://ieeexplore.ieee.org/document/7524428/ (visited on 01/23/2019).

[120] Luca De Cicco, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. "ELASTIC: A Client-Side Controller for Dynamic Adaptive Streaming over HTTP (DASH)". en. In: *2013 20th International Packet Video Workshop*. San Jose, CA: IEEE, Dec. 2013, pp. 1–8. ISBN: 978-1-4799-2172-0. DOI: 10.1109/PV.2013.6691442. URL: http://ieeexplore.ieee.org/document/6691442/ (visited on 01/15/2019).

[121] Cong Wang, Amr Rizk, and Michael Zink. "SQUAD: A Spectrum-based Quality Adaptation for Dynamic Adaptive Streaming over HTTP". In: *Proceedings of the 7th International Conference on Multimedia Systems*. MMSys '16. New York, NY, USA: ACM, 2016, 1:1–1:12. ISBN: 978-1-4503-4297-1. DOI: 10.1145/2910017.2910593. URL: http://doi.acm.org/10.1145/2910017.2910593 (visited on 01/23/2019).

[122] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. "Neural Adaptive Video Streaming with Pensieve". In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 197–210. ISBN: 978-1-4503-4653-5. DOI: 10.1145/3098822.3098843. URL: http://doi.acm.org/10.1145/3098822.3098843 (visited on 01/23/2019).

[123] Christian Lottermann, Serhan Gul, Damien Schroeder, and Eckehard Steinbach. "Network-aware video level encoding for uplink adaptive HTTP streaming". en. In: *2015 IEEE International Conference on Communications (ICC)*. London: IEEE, June 2015, pp. 6861–6866. ISBN: 978-1-4673-6432-4. DOI: 10.1109/ICC.2015.7249419. URL: http://ieeexplore.ieee.org/document/7249419/ (visited on 01/15/2019).

[124] Alexey Vinel, Evgeny Belyaev, Boris Bellalta, and Honglin Hu. "Live Video Streaming in Vehicular Networks". en. In: *Communication Technologies for Vehicles*. Ed. by David Hutchison et al. Vol. 8435. Cham: Springer International Publishing, 2014, pp. 156–162. ISBN: 978-3-319-06643-1 978-3-319-06644-8. DOI: 10.1007/978-3-319-06644-8_14. URL: http://link.springer.com/10.1007/978-3-319-06644-8_14 (visited on 01/18/2019).

[125] Mohamed Aymen Labiod, Mohamed Gharbi, François-Xavier Coudoux, Patrick Corlay, and Noureddine Doghmane. "Enhanced adaptive cross-layer scheme for low latency HEVC streaming over Vehicular Ad-hoc Networks (VANETs)". en. In: *Vehicular Communications* 15 (Jan. 2019), pp. 28–39. ISSN: 22142096. DOI: 10.1016/j.vehcom.2018.11.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S2214209618301864 (visited on 01/15/2019).

[126] Chenguang Yu, Yang Xu, Bo Liu, and Yong Liu. "“Can you SEE me now?” A measurement study of mobile video calls". en. In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. Toronto, ON, Canada: IEEE, Apr. 2014, pp. 1456–1464. ISBN: 978-1-4799-3360-0. DOI: 10.1109/INFOCOM.2014.6848080. URL: http://ieeexplore.ieee.org/document/6848080/ (visited on 01/15/2019).

[127] Leonardo Favario, Matti Siekkinen, and Enrico Masala. "Mobile live streaming: Insights from the periscope service". en. In: *2016 IEEE 18th International Workshop on Multimedia Signal Processing (MMSP)*. Montreal, QC, Canada: IEEE, Sept. 2016, pp. 1–6. ISBN: 978-1-5090-3724-7. DOI: 10.1109/MMSP.2016.7813395. URL: http://ieeexplore.ieee.org/document/7813395/ (visited on 02/18/2019).

[128] Jonathan Kua, Grenville Armitage, and Philip Branch. "A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming Over HTTP". en. In: *IEEE Communications Surveys & Tutorials* 19.3 (2017), pp. 1842–1866. ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2685630. URL: http://ieeexplore.ieee.org/document/7884970/ (visited on 01/15/2019).

[129] Burak Cizmeci, Xiao Xu, Rahul Chaudhari, Christoph Bachhuber, Nicolas Alt, and Eckehard Steinbach. "A Multiplexing Scheme for Multimodal Teleoperation". en. In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 13.2 (Apr. 2017), pp. 1–28. ISSN: 15516857. DOI: 10.1145/3063594. URL: http://dl.acm.org/citation.cfm?doid=3058792.3063594 (visited on 01/21/2019).

[130] Christoph Bachhuber, Eckehard Steinbach, Martin Freundl, and Martin Reisslein. "On the Minimization of Glass-to-Glass and Glass-to-Algorithm Delay in Video Communication". en. In: *IEEE Transactions on Multimedia* 20.1 (Jan. 2018), pp. 238–252. ISSN: 1520-9210, 1941-0077. DOI: 10.1109/TMM.2017.2726189. URL: http://ieeexplore.ieee.org/document/7976319/ (visited on 12/04/2019).

[131] Christoph Bachhuber and Eckehard Steinbach. "A System for High Precision Glass-to-Glass Delay Measurements in Video Communication". In: *IEEE International Conference on Image Processing (ICIP)*. 2016, pp. 2132–2136.

[132] Lester C. Loschky and Gary S. Wolverton. "How Late Can You Update Gaze-contingent Multiresolutional Displays Without Detection?" In: *ACM Trans. Multimedia Comput. Commun. Appl.* 3.4 (Dec. 2007), 7:1–7:10. ISSN: 1551-6857. DOI: 10.1145/1314303.1314310. URL: http://doi.acm.org/10.1145/1314303.1314310 (visited on 01/25/2019).

[133] Nabajeet Barman and Maria G. Martini. "H.264/MPEG-AVC, H.265/MPEG-HEVC and VP9 codec comparison for live gaming video streaming". en. In: *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. Erfurt, Germany: IEEE, May 2017, pp. 1–6. ISBN: 978-1-5386-4024-1. DOI: 10.1109/QoMEX.2017.7965686. URL: http://ieeexplore.ieee.org/document/7965686/ (visited on 06/19/2019).

[134] VideoLAN. *x264*. en. URL: https://www.videolan.org/developers/x264.html (visited on 12/12/2021).

[135] Zhihai He and S.K. Mitra. "A linear source model and a unified rate control algorithm for DCT video coding". In: *IEEE Transactions on Circuits and Systems for Video Technology* 12.11 (2002), pp. 970–982. DOI: 10.1109/TCSVT.2002.805511.

[136] Min Gao, Burak Cizmeci, Michael Eiler, Eckehard Steinbach, Debin Zhao, and Wen Gao. "Macroblock level rate control for low delay H.264/AVC based video communication". en. In: *2015 Picture Coding Symposium (PCS)*. Cairns, Australia: IEEE, May 2015, pp. 210–215. ISBN: 978-1-4799-7783-3. DOI: 10.1109/PCS.2015.7170077. URL: http://ieeexplore.ieee.org/document/7170077/ (visited on 01/15/2019).

[137] Yao Wang, Zhan Ma, and Yen-Fu Ou. "Modeling rate and perceptual quality of scalable video as functions of quantization and frame rate and its application in scalable video adaptation". en. In: *2009 17th International Packet Video Workshop*. Seattle, WA, USA: IEEE, May 2009, pp. 1–9. ISBN: 978-1-4244-4651-3. DOI: 10.1109/PACKET.2009.5152161. URL: http://ieeexplore.ieee.org/document/5152161/ (visited on 02/05/2021).

[138] Zhan Ma, Meng Xu, Yen-Fu Ou, and Yao Wang. "Modeling of Rate and Perceptual Quality of Compressed Video as Functions of Frame Rate and Quantization Stepsize and Its Applications". en. In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.5 (May 2012), pp. 671–682. ISSN: 1051-8215, 1558-2205. DOI: 10.1109/TCSVT.2011.2177143. URL: http://ieeexplore.ieee.org/document/6086602/ (visited on 02/05/2021).

[139] Zhan Ma, Felix C. A. Fernandes, and Yao Wang. "Analytical rate model for compressed video considering impacts of spatial, temporal and amplitude resolutions". en. In: *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. San Jose, CA, USA: IEEE, July 2013, pp. 1–6. ISBN: 978-1-4799-1604-7. DOI: 10.1109/ICMEW.2013.6618414. URL: http://ieeexplore.ieee.org/document/6618414/ (visited on 12/10/2020).

[140] Yen-Fu Ou, Zhan Ma, Tao Liu, and Yao Wang. "Perceptual Quality Assessment of Video Considering Both Frame Rate and Quantization Artifacts". en. In: *IEEE Transactions on Circuits and Systems for Video Technology* 21.3 (Mar. 2011), pp. 286–298. ISSN: 1051-8215, 1558-2205. DOI: 10.1109/TCSVT.2010.2087833. URL: http://ieeexplore.ieee.org/document/5604671/ (visited on 07/19/2021).

[141] Christian Lottermann, Alexander Machado, Damien Schroeder, Yang Peng, and Eckehard Steinbach. "Bit rate estimation for H.264/AVC video encoding based on temporal and spatial activities". en. In: *2014 IEEE International Conference on Image Processing (ICIP)*. Paris, France: IEEE, Oct. 2014, pp. 3195–3199. ISBN: 978-1-4799-5751-4. DOI: 10.1109/ICIP.2014.7025646. URL: http://ieeexplore.ieee.org/document/7025646/ (visited on 01/15/2019).

[142] Christian Lottermann and Eckehard Steinbach. "Modeling the bit rate of H.264/AVC video encoding as a function of quantization parameter, frame rate and GoP characteristics". en. In: *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. Chengdu, China: IEEE, July 2014, pp. 1–6. ISBN: 978-1-4799-4717-1. DOI: 10.1109/ICMEW.2014.6890567. URL: http://ieeexplore.ieee.org/document/6890567/ (visited on 07/14/2020).

[143] Michele Covell. "Optimizing Transcoder Quality Targets Using a Neural Network with an Embedded Bitrate Model". en. In: (2016), p. 7.

[144] Yangfan Sun, Mouqing Jin, Li Li, and Zhu Li. "A Machine Learning Approach to Accurate Sequence-Level Rate Control Scheme for Video Coding". en. In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. Athens: IEEE, Oct. 2018, pp. 1013–1017. ISBN: 978-1-4799-7061-2. DOI: 10.1109/ICIP.2018.8451080. URL: https://ieeexplore.ieee.org/document/8451080/ (visited on 02/10/2021).

[145] Fan Zhang. "Quality of Experience-driven Multi-Dimensional Video Adaptation". PhD thesis. Technischen Universität München, Mar. 2014. URL: https://mediatum.ub.tum.de/doc/1173314/1173314.pdf (visited on 01/18/2019).

[146] M. Vranjes, S. Rimac-Drlje, and D. Zagar. "Objective video quality metrics". en. In: *ELMAR 2007*. Zadar, Croatia: IEEE, Sept. 2007, pp. 45–49. ISBN: 978-953-7044-05-3. DOI: 10.1109/ELMAR.2007.4418797. URL: http://ieeexplore.ieee.org/document/4418797/ (visited on 01/18/2019).

[147] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[148] Yue Wang, Tingting Jiang, Siwei Ma, and Wen Gao. "Spatio-temporal ssim index for video quality assessment". In: *2012 Visual Communications and Image Processing*. 2012, pp. 1–6. DOI: 10.1109/VCIP.2012.6410779.

[149] Yang Peng and Eckehard Steinbach. "A novel full-reference video quality metric and its application to wireless video transmission". In: *2011 18th IEEE International Conference on Image Processing*. 2011, pp. 2517–2520. DOI: 10.1109/ICIP.2011.6116174.

[150] Yen-Fu Ou, Yuanyi Xue, and Yao Wang. "Q-STAR: A Perceptual Video Quality Model Considering Impact of Spatial, Temporal, and Amplitude Resolutions". en. In: *IEEE Transactions on Image Processing* 23.6 (June 2014), pp. 2473–2486. ISSN: 1057-7149, 1941-0042. DOI: 10.1109/TIP.2014.2303636. URL: http://ieeexplore.ieee.org/document/6728690/ (visited on 09/23/2020).

[151] Matthias Wichtlhuber, Gregor Wicklein, Stefan Wilk, Wolfgang Effelsberg, and David Hausheer. "RT-VQM: Real-time Video Quality Assessment for Adaptive Video Streaming Using GPUs". In: *Proceedings of the 7th International Conference on Multimedia Systems*. MMSys '16. event-place: Klagenfurt, Austria. New York, NY, USA: ACM, 2016, 20:1–20:11. ISBN: 978-1-4503-4297-1. DOI: 10.1145/2910017.2910600. URL: http://doi.acm.org/10.1145/2910017.2910600 (visited on 08/09/2019).

[152] M.H. Pinson and S. Wolf. "A new standardized method for objectively measuring video quality". In: *IEEE Transactions on Broadcasting* 50.3 (2004), pp. 312–322. DOI: 10.1109/TBC.2004.834028.

[153] Fan Zhang, Eckehard Steinbach, and Peng Zhang. "MDVQM: A Novel Multidimensional No-Reference Video Quality Metric for Video Transcoding". In: *J. Vis. Comun. Image Represent.* 25.3 (Apr. 2014), pp. 542–554. ISSN: 1047-3203. DOI: 10.1016/j.jvcir.2013.11.011. URL: https://doi.org/10.1016/j.jvcir.2013.11.011.

[154] Joe Yuchieh Lin, Tsung-Jung Liu, Eddy Chi-Hao Wu, and C.-C Jay Kuo. "A fusion-based video quality assessment (fvqa) index". In: *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*. 2014, pp. 1–5. DOI: 10.1109/APSIPA.2014.7041705.

[155] Joe Yuchieh Lin, Chi-Hao Wu, Ioannis Katsavounidis, Zhi Li, Anne Aaron, and C.-C. Jay Kuo. "EVQA: An ensemble-learning-based video quality assessment index". In: *2015 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*. 2015, pp. 1–6. DOI: 10.1109/ICMEW.2015.7169760.

[156] Netflix Technology Blog. *Toward A Practical Perceptual Video Quality Metric*. en. Apr. 2017. URL: https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652 (visited on 02/26/2020).

[157] Netflix Technology Blog. *VMAF: The Journey Continues*. en. Oct. 2018. URL: https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12 (visited on 02/26/2020).

[158] E.C. Reed and J.S. Lim. "Optimal multidimensional bit-rate control for video communication". en. In: *IEEE Transactions on Image Processing* 11.8 (Aug. 2002), pp. 873–885. ISSN: 1057-7149. DOI: 10.1109/TIP.2002.801122. URL: http://ieeexplore.ieee.org/document/1025162/ (visited on 01/29/2020).

[159] Mariana Afonso, Fan Zhang, and David R. Bull. "Video Compression Based on Spatio-Temporal Resolution Adaptation". en. In: *IEEE Transactions on Circuits and Systems for Video Technology* 29.1 (Jan. 2019), pp. 275–280. ISSN: 1051-8215, 1558-2205. DOI: 10.1109/TCSVT.2018.2878952. URL: https://ieeexplore.ieee.org/document/8517114/ (visited on 07/27/2021).

[160] Timothy Popkin, Andrea Cavallaro, and David Hands. "Accurate and Efficient Method for Smoothly Space-Variant Gaussian Blurring". en. In: *IEEE Transactions on Image Processing* 19.5 (May 2010), pp. 1362–1370. ISSN: 1057-7149, 1941-0042. DOI: 10.1109/TIP.2010.2041400. URL: http://ieeexplore.ieee.org/document/5398926/ (visited on 07/14/2020).

[161] Shijun Sun, Cheng Chang, and Stacey Spears. "Filtering and dithering as pre-processing before encoding". en. US8750390B2. June 2014. URL: https://patents.google.com/patent/US8750390B2/en (visited on 10/23/2019).

[162] Brian Astle. "Image signal encoding with variable low-pass filter". US6026190A. Feb. 2000. URL: https://patents.google.com/patent/US6026190A/en (visited on 10/18/2019).

[163] L.S. Karlsson and M. Sjostrom. "Improved ROI video coding using variable Gaussian pre-filters and variance in intensity". en. In: *IEEE International Conference on Image Processing 2005*. Genova, Italy: IEEE, 2005, pp. II–313. ISBN: 978-0-7803-9134-5. DOI: 10.1109/ICIP.2005.1530054. URL: http://ieeexplore.ieee.org/document/1530054/ (visited on 10/18/2019).

[164] Hong-jie Huang, Xing-ming Zhang, and Zhi-wei Xu. "Semantic Video Adaptation using a Preprocessing Method for Mobile Environment". en. In: *2010 10th IEEE International Conference on Computer and Information Technology*. Bradford, United Kingdom: IEEE, June 2010, pp. 2806–2810. ISBN: 978-1-4244-7547-6. DOI: 10.1109/CIT.2010.468. URL: http://ieeexplore.ieee.org/document/5578553/ (visited on 10/18/2019).

[165] Linda S Karlsson, Mårten Sjöström, and Roger Olsson. "Spatio-temporal filter for ROI video coding". en. In: (2006), p. 5.

[166] Madhukar Budagavi, John Furton, Guoxin Jin, Ankur Saxena, Jeffrey Wilkinson, and Andrew Dickerson. "360 degrees video coding using region adaptive smoothing". en. In: *2015 IEEE International Conference on Image Processing (ICIP)*. Quebec City, QC, Canada: IEEE, Sept. 2015, pp. 750–754. ISBN: 978-1-4799-8339-1. DOI: 10.1109/ICIP.2015.7350899. URL: http://ieeexplore.ieee.org/document/7350899/ (visited on 02/12/2021).

[167] Dan Grois and Ofer Hadar. "Complexity-Aware Adaptive Preprocessing Scheme for Region-of-Interest Spatial Scalable Video Coding". en. In: *IEEE Transactions on Circuits and Systems for Video Technology* 24.6 (June 2014), pp. 1025–1039. ISSN: 1051-8215, 1558-2205. DOI: 10.1109/TCSVT.2014.2302557. URL: http://ieeexplore.ieee.org/document/6727577/ (visited on 04/19/2020).

[168] Dan Grois and Ofer Hadar. "Efficient adaptive bit-rate control for Scalable Video Coding by using Computational Complexity-Rate-Distortion analysis". en. In: *2011 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. Metropolitan Area Nuremberg, Germany: IEEE, June 2011, pp. 1–6. ISBN: 978-1-61284-121-2. DOI: 10.1109/BMSB.2011.5954877. URL: http://ieeexplore.ieee.org/document/5954877/ (visited on 04/20/2020).

[169] Dan Grois and Ofer Hadar. "Efficient Region-of-Interest Scalable Video Coding with Adaptive Bit-Rate Control". en. In: *Advances in Multimedia* 2013 (2013), pp. 1–17. ISSN: 1687-5680, 1687-5699. DOI: 10.1155/2013/281593. URL: http://www.hindawi.com/journals/am/2013/281593/ (visited on 10/11/2021).

[170] CARLA-Team. *CARLA Plugins*. Accessed on: 2022-01-11. URL: https://github.com/carla-simulator/carla-plugins.

[171] CARLA-Team. *CARLA ROS Bridge*. Accessed on: 2021-11-27. URL: https://github.com/carla-simulator/ros-bridge.

[172] GStreamer-Team. *GStreamer*. Accessed on: 2021-11-27. URL: https://gstreamer.freedesktop.org/.

[173] Chair of Communication Networks. *tcgui*. Accessed on: 2021-11-27. URL: https://github.com/tum-lkn/tcgui.

[174] CARLA-Team. *CARLA Scenario Runner*. Accessed on: 2021-11-29. URL: https://github.com/carla-simulator/scenario_runner.

[175] Fisher Yu et al. "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning". In: (2020), pp. 2633–2642. DOI: 10.1109/CVPR42600.2020.00271.

[176] Kentaro Wada. *Labelme: Image Polygonal Annotation with Python*. DOI: 10.5281/zenodo.5711226. URL: https://github.com/wkentaro/labelme.

[177] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[178] Moritz Kassner, William Patera, and Andreas Bulling. "Pupil: an open source platform for pervasive eye tracking and mobile gaze-based interaction". In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. UbiComp '14 Adjunct. Seattle, Washington: Association for Computing Machinery, Sept. 2014, pp. 1151–1160. ISBN: 978-1-4503-3047-3. DOI: 10.1145/2638728.2641695. URL: http://doi.org/10.1145/2638728.2641695 (visited on 07/06/2020).

[179] Mr Kai Gao, Mr Jingxuan Zhang, Y Richard Yang, and Jun Bi. "Prophet: Fast Accurate Model-based Throughput Prediction for Reactive Flow in DC Networks". en. In: *IEEE INFOCOM* (2018), p. 9.

[180] Vadim Furman, Andrew Hughes Chatham, Abhijit Ogale, and Dmitri Dolgov. "Image and video compression for remote vehicle assistance". en. US9767369B2. Sept. 2017. URL: https://patents.google.com/patent/US9767369/en (visited on 01/23/2019).

[181] Guilherme Correa, Pedro Assuncao, Luciano Agostini, and Luis A da Silva Cruz. *Complexity-Aware High Efficiency Video Coding*. Springer, 2016.

[182] Gisle Bjontegaard. "Calculation of average PSNR differences between RD-Curves". In: *ITU-T VCEG and ISO/IEC MPEG document VCEG-MM33* (Apr. 2001).

[183] P ITU. "800.1 : Mean opinion score (MOS) terminology". In: *International telecommunications union telecommunication sector* (2003).

[184] Tomi Heinonen. "Multi-camera solution for electronic devices". US20060139463A1. June 2006. URL: https://patents.google.com/patent/US20060139463A1/en (visited on 10/23/2019).

[185] Netflix. *Q: When computing VMAF on low-resolution videos (480 height, for example), why the scores look so high, even when there are visible artifacts?* en. URL: https://github.com/Netflix/vmaf/blob/master/FAQ.md (visited on 03/29/2022).

[186] *YUV video sequences*. en. http://trace.eas.asu.edu/yuv/. URL: http://trace.eas.asu.edu/yuv/ (visited on 12/21/2020).

[190] Nvidia Corporation. *NVIDIA Video Codec SDK*. Accessed on: 2021-04-16. 2021. URL: https://developer.nvidia.com/nvidia-video-codec-sdk.

[191] *Y4M video sequences*. en. https://media.xiph.org/video/derf/. URL: https://media.xiph.org/video/derf/ (visited on 12/21/2020).

[192] Yang Peng and Eckehard Steinbach. "A novel full-reference video quality metric and its application to wireless video transmission". de. In: *IEEE International Conference on Image Processing (ICIP)*. Sept. 2011.

[193] P ITU. "910. Subjective video quality assessment methods for multimedia applications". In: *International telecommunications union telecommunication sector* (1999).

[194] P McCullagh and John A Nelder. *Generalized Linear Models*. Vol. 37. CRC Press, 1989.

[195] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[196] Sam Lantinga. *Simple DirectMedia Layer (SDL)*. Accessed on: 2021-11-27. URL: https://www.libsdl.org/.

[197] Sundari Govindarajan, T Bernatin, and Pratik Somani. "H. 264 encoder using Gstreamer". In: Mar. 2015, pp. 1–4. DOI: 10.1109/ICCPCT.2015.7159511.

[198] GStreamer-Team. *videorate*. Accessed on: 2021-11-27. URL: https://gstreamer.freedesktop.org/documentation/videorate/index.html?gi-language=c.

[199] GStreamer-Team. *x264enc*. Accessed on: 2021-11-27. URL: https://gstreamer.freedesktop.org/documentation/x264/index.html?gi-language=c.

[200] GStreamer-Team. *rtsp server*. Accessed on: 2021-11-27. URL: https://gstreamer.freedesktop.org/documentation/gst-rtsp-server/rtsp-server.html?gi-language=c.

[201] ROS-Team. *Dynamic Reconfigure*. Accessed on: 2021-11-27. URL: http://wiki.ros.org/dynamic_reconfigure.