Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik
Lehrstuhl für Medientechnik

Low Delay Video Communication

Christoph Bachhuber, M. Sc.

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:		Prof. Dr.-Ing. Bernhard U. Seeber
Prüfer der Dissertation:	1.	Prof. Dr.-Ing. Eckehard Steinbach
	2.	Prof. Martin Reisslein, Ph.D.

Die Dissertation wurde am 13.03.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 04.06.2019 angenommen.

Abstract

Low delay video communication facilitates a multitude of new technologies and applications. For humans, low delay video communication enables teleoperation in novel scenarios which are impossible to realize using a video communication setup with high delay. These scenarios can be remote control of robots in dynamic environments, such as quadrocopters in search and rescue operations or telesurgery over long distances. For machines, low delay video communication generally renders many new vision-based networked control systems feasible. These systems can be used in networked autonomous driving, general dynamic navigation in unexplored environments, or vision-based state retrieval of robots, for instance the position of a robot arm. Modern digital video communication solutions are unable to provide the low latencies required for such tasks. Consequently, to make the novel fields described in this paragraph possible, this thesis researches the reduction of video communication delay.

Before any steps to reduce video communication delay can be undertaken, an analysis of delay contributors must identify the elements of a video communication setup that contribute the greatest latencies. These elements exhibit the highest delay reduction potential. A detailed theoretical analysis of the delay contributors in video communication shows that in particular the temporal sampling rates of camera and display should be increased to reduce latency. This thesis confirms the theoretical analysis through a survey of the delays of modern video communication solutions, performed with a newly developed highly precise delay measurement system. The survey additionally shows that the majority of the state of the art in video communication exhibits a large delay that prevents usage in applications such as dynamic teleoperation.

Low delay video requires high temporal sampling rates that may exceed human perception limits. In consequence, not all video images recorded by the camera need to be further processed and shown. Skipping irrelevant images yields variable frame rate video systems. These systems should operate at sampling rates at which the temporal discreteness of the video is imperceptible to humans. Through a series of perception experiments, a model for the human perception of temporal sampling in visual signals is created. Using this model, sampling rate adaptation takes place outside of the human visual perceptual window.

The previously described analysis and perception model are used for the proposed video delay reduction methods. All methods employ a camera with high temporal sampling rate. From the resulting video stream, only a subset of images is further processed while the remaining frames are skipped. This approach reduces data rates and computational require-

ments. The first method is greedy, content-based frame skipping. Greedy frame skipping achieves a significant latency reduction of a factor five to ten compared to modern video communication solutions. However, the frame rate adaptation of this method is perceivable. This is why the second method integrates the perceptual model into the greedy frame skipping algorithm such that the benefits of the first method are retained, but frame skipping is imperceptible.

Kurzfassung

Videokommunikation mit geringer Verzögerung ermöglicht eine Vielzahl neuer Technologien und Anwendungen. Niedriglatente Videokommunikation befähigt Menschen zu Teleoperation in neuartigen Szenarien, was mit bisherigen hochlatenten Videoübertragungssystemen unmöglich war. Solche Szenarien können die Fernsteuerung von Robotern in dynamischen Umgebungen, zum Beispiel von Quadrokoptern in Rettungsmissionen oder Telemedizin über lange Distanzen sein. Für Maschinen macht Videokommunikation mit geringer Latenz neuartige kamerabasierte Netzwerkkontrollsysteme möglich. Derartige Systeme können im vernetzten autonomen Fahren, allgemeiner dynamischer Navigation in unbekannten Umgebungen, oder in der kamerabasierten Zustandserfassung von Robotern, beispielsweise der Position eines Roboterarmes, eingesetzt werden. Moderne digitale Videokommunikationslösungen genügen nicht den Latenzanforderungen solcher Anwendungen. Um die beschriebenen Anwendungen zu ermöglichen, beschäftigt sich diese Arbeit mit der Reduktion von Verzögerung in der Videokommunikation.

Bevor Schritte zur Reduktion von Verzögerung in Videokommunikationssystemen unternommen werden, muss eine Analyse die Elemente identifizieren, welche die größten Verzögerungen beitragen. Diese Elemente weisen gleichzeitig das höchste Potential zur Verzögerungsreduktion auf. Eine detaillierte theoretische Analyse der Elemente, die zur Verzögerung in der Videokommunikation beitragen, zeigt, dass zur Verzögerungsreduktion die zeitlichen Abtastraten von Kamera und Bildschirm erhöht werden müssen. Diese Arbeit bestätigt die theoretische Analyse durch eine Studie von Verzögerungen in modernen Videokommunikationslösungen, welche mit einem neu entwickelten System zur präzisen Messung von Videoverzögerung durchgeführt wird. Die Studie zeigt zusätzlich, dass der Großteil der modernen Videokommunikationstechnik zu große Verzögerungen für eine Verwendung in dynamischer Teleoperation oder ähnlichen Gebieten aufweist.

Die hohen zeitlichen Abtastraten können über den menschlichen Wahrnehmungsgrenzen liegen. Folglich müssen nicht alle von der Kamera aufgenommenen Videobilder verarbeitet und angezeigt werden. Das Auslassen von irrelevanten Bildern ergibt Videosysteme mit variabler Abtastrate. Diese Systeme sollen bei Abtastraten, bei denen die zeitliche Diskretheit des Videos für Menschen nicht wahrnehmbar ist, arbeiten. Durch eine Reihe von Wahrnehmungsexperimenten wird ein Modell für die menschliche Wahrnehmung von zeitlich abgetasteten visuellen Signalen erstellt. Mit diesem Wahrnehmungsmodell findet jegliche Abtastfrequenzanpassung außerhalb des menschlich wahrnehmbaren Bereiches statt.

Die oben beschriebene Analyse und das Wahrnehmungsmodell werden für die Metho-

den zur Verzögerungsreduktion genutzt. Alle Methoden verwenden eine Kamera mit hoher zeitlicher Abtastfrequenz. Von dem sich daraus ergebenden Video wird nur eine Teilmenge der Bilder weiter verarbeitet, die übrigen Bilder werden verworfen. Diese Herangehensweise reduziert Datenraten und Anforderungen an die Rechenleistung. Als erste Methode schlägt diese Arbeit das Auslassen von unwichtigen Videobildern vor, worin Bilder basierend auf Inhaltsunterschieden weitergegeben oder verworfen werden. Die Methode erreicht eine deutliche Verzögerungsreduktion von einem Faktor von fünf bis zehn verglichen mit dem Stand der Technik. Allerdings ist die adaptive Bildratenreduktion wahrnehmbar. Deswegen integriert die zweite Methode das Wahrnehmungsmodell in das Auslassen der Bilder, so dass die Vorteile der ersten Methode noch vorhanden sind, aber die Unterabtastung der Bilder nicht mehr wahrnehmbar ist.

Acknowledgements

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. Eckehard Steinbach for providing me with the opportunity to conduct research in his group. With his trust and scientific expertise, he managed to push me in the right direction while giving me ample space to develop my own ideas. I am particularly grateful for his commitment to my project. Despite his busy schedule, he took his time for our discussions and provided excellent feedback.

Furthermore, I would like to thank Prof. Martin Reisslein for agreeing to become the second examiner and Prof. Bernhard Seeber for chairing the thesis committee.

I feel lucky to have met my former and current colleagues at LMT. Not only have they helped me professionally, but also made my time at LMT special and enjoyable on a personal level. I am particularly grateful to Dr. Rahul Chaudhari and Dr. Burak Çizmeci for kick-starting my research with their experience. My thanks also go to Prof. Martin Reisslein from ASU and Dr. Amit Bhardwaj from POSTECH for their expert contributions to our joint projects. Furthermore, I would like to extend special thanks to our industry partners, Dr. Rastin Pries and Dr. Marco Hoffmann for their support during our interesting and fruitful collaborations. I highly value all my colleagues at LMT, I feel however obliged to highlight a few people. I thank Dr. Nicolas Alt, Tamay Aykut, Dmytro Bobkov, Mojtaba Karimi, Andreas Noll, Martin Oelsch, Dominik van Opdenbosch, Dr. Damien Schroeder, Dr. Clemens Schuwerk, Matti Strese, and Jingyi Xu for interesting discussions about virtually anything from rather trivial to highly complex matters, and for brightening my days. I am also grateful to all my students, especially Simon Conrady, Martin Freundl, and Michael Schütz, who contributed to our publications.

I appreciate the quick and reliable administrative support provided by Marta Giunta, Ingrid Jamrath, Simon Krapf, Dr. Martin Maier, Martina Schmid, and Brigitte Vrochte.

Finally I would like to thank my family, in particular parents, my brother and my sister who believed in my abilities, and my wonderful girlfriend Melanie who supported me throughout my time at LMT.

Christoph Bachhuber, Munich, October 29, 2018

Contents

Notation

Abbreviations

Abbreviation	Description	Definition
4K	Video with a spatial resolution of 3840 ($\approx$ 4000) times 2160 pixels	page 32
ANOVA	Analysis of Variance	page 81
API	Application Programming Interface	page 32
ASF	Adaptive Sampling Frequency	page 81
ASIC	Application-Specific Integrated Circuit	page 31
AVC	Advanced Video Coding	page 14
CSF	Constant Sampling Frequency	page 81
CU	Coding Unit	page 21
E2E	End-to-End	page 2
FHD	Full High-Definition (spatial resolution of 1920 times 1080 pixels)	page 39
FoV	Field of View	page 9
FPGA	Field-Programmable Gate Array	page 18
G2A	Glass-to-Algorithm	page 2
G2G	Glass-to-Glass	page 3
G2X	Glass-to-Anything	page 58
GOP	Group-of-Pictures	page 15
GPU	Graphics Processing Unit	page 36
HDMI	High Definition Multimedia Interface	page 14
HEVC	High Efficiency Video Coding	page 14
HM	HEVC Test Model	page 33
HMD	Head-Mounted Display	page 3
LCD	Liquid-Crystal Display	page 38
LED	Light-Emitting Diode	page 4
MAD	Mean Absolute Difference	page 86
MTU	Maximum Transmission Unit	page 34
NCS	Networked Control System	page 2
OLED	Organic Light-Emitting Diode	page 38
PDF	Probability Density Function	page 40
PT	Photo Transistor	page 43
PWM	Pulse-Width Modulation	page 46
QoE	Quality of Experience	page 3
RANOVA	Repeated Measures Analysis of Variance	page 81
RD	Rate-Distortion	page 17
RDC	Rate-Distortion-Complexity	page 21
RGB	Red-Green-Blue Color Space	page 14
RGBA	Red-Green-Blue-Alpha Color Space	page 36
QR	Quick Response	page 20
SSIM	Structural Similarity Index	page 86

Abbreviation	Description	Definition
TU	Transform Unit	page 21
UDP	User Datagram Protocol	page 89
USB	Universal Serial Bus	page 14
VSync	Vertical Synchronization	page 37
VoV	Volume of Visibility	page 60
WoV	Window of Visibility	page 10

Subscripts and superscripts

$\bar{x}$ mean value of x
$\hat{x}$ quantized value of x

Symbols

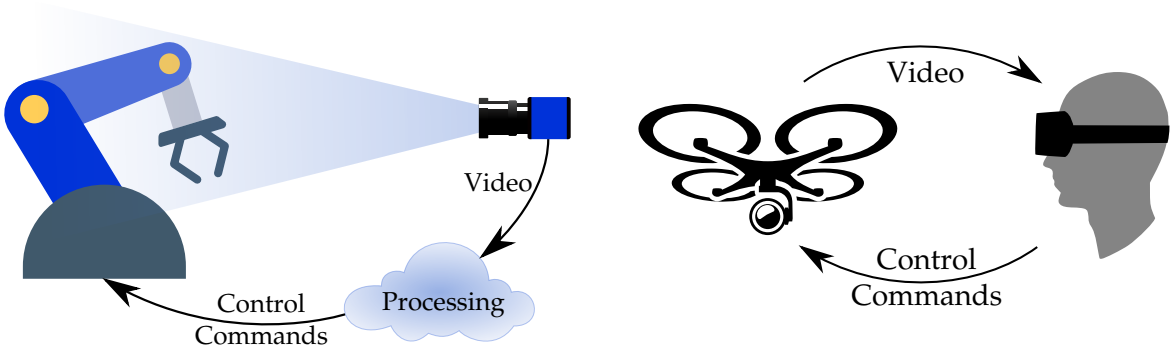
c	Speed of light
C	Available bit rate of a network channel
f	Frame rate or sampling frequency
T	Frame or sampling period
t_{exp}	Exposure time
t_{post}	Post-exposure time
t_{CTS}	Delay from camera temporal sampling
t_{CP}	Delay from image processing in the camera
t_{FS}	Delay from frame skipping
t_{CSC}	Delay from color space conversion
t_{Enc}	Delay from encoding
t_{EB}	Delay from encoder buffer
t_{Netw}	Delay from network: transmission and propagation
t_{DB}	Delay from decoder buffer
t_{Dec}	Delay from decoding
t_{DTS}	Delay from display temporal sampling
t_{DP}	Delay from image processing in the display
t_{DPR}	Delay from pixel response time in the display
t_{G2A}	Glass-to-Algorithm delay
t_{G2G}	Glass-to-Glass delay

Chapter 1

Introduction

Video communication is the process of transporting visual information in real time from one point to another. Nowadays, this task is performed using cameras and electronic circuits for acquiring, processing, transporting, and reproducing the visual information, i.e. the video. This thesis distinguishes between two application fields of video communication, shown in Figure 1.1. The first field is video communication for machine vision, and the second field comprises video communication for human interaction. Both fields are explained in the following. In particular, the following paragraphs describe how unavoidable processing latencies in video communication constrain possible applications and deteriorate task performance.

Cameras have the potential to replace many sensors for process control [9], [10] through machine vision. There are several advantages of using cameras in combination with machine vision algorithms over dedicated sensors. Video cameras are comparably low cost and universally usable. In conjunction with visual tracking techniques, they can, for instance, replace conventional sensors such as radar sensors [11]. Alternatively, cameras can replace



(a) A robot arm is observed by a camera. The video stream is analyzed to extract state information of the robot arm, such as joint angles. This is used to control the robot arm [8].

(b) Remote control of a quadrocopter. The quadrocopter streams the video from its camera to a user, who views it through a head-mounted display. Based on the video information, the user controls the quadrocopter.

Figure 1.1: Examples from the two main application fields of low delay video communication: video communication in machine vision/process control, and video communication for human interaction. Note that lower delay in both control loops facilitates usage in more dynamic environments.

angle sensors in the joints of robot arms by visually tracking the arm's state [8], see Figure 1.1a. The camera observes the robot arm and sends the video to a processing unit. The processing unit applies computer vision algorithms to identify the robot arm and its state. This information can be used to control the robot arm while it is working on its tasks. An advantage of video camera sensing is that a single camera can replace multiple sensors, such as the camera observing a robot arm with multiple joints, substituting multiple angle sensors [12]. Also, compared to low-level hardware sensors, video cameras are more future-proof through improvements in computer vision software. Already today, there are tasks that can best be performed by cameras, for example the inspection of granule or powder that passes by on a conveyor belt.

In general vision-based control, a traditional sensor is replaced by a unit consisting of camera, video compression, network, and image processing. This unit should provide a sensor-to-controller latency, sampling rate, and data precision comparable to the original sensor. If the vision sensor unit performs substantially worse than the original sensor in one of these dimensions, control quality can suffer severely, to the point of making control impossible [13]. This imposes requirements on camera, video compression, network, and image processing. As shown in Section 3.4, latency and sampling rate are closely related. This section focuses on latency, because latency additionally describes further parts of a video transmission chain. Data precision is mainly a question of the specific image processing algorithm and therefore not discussed here.

Latency is the main challenge to overcome before cameras can be widely used as sensors not only for relatively slow-paced inspection tasks, but as part of fast feedback loops for control applications. In networked control systems (NCSs), growing sensor-to-controller latency [14] deteriorates the system stability [15], and when the latency exceeds an allowable limit, leads to instability [5], [16]. Researchers have thoroughly investigated the effect of latency on NCS stability [13], and proposed many algorithms to compensate delay [17]–[20] to a limited extent. Still, creating a low-delay sensor-to-controller transmission is advantageous [21] for stabilizing a system and enables control of more dynamic applications.

For machines that are controlled using visual information from a camera, the challenge corresponding to the low sensor-to-controller latency is the Glass-to-Algorithm (G2A) delay. The G2A delay characterizes the time difference between a visible event taking place (conveyed through its photons passing through the camera lens glass), and the first image of the event being available for an image processing algorithm. If a control loop includes a video transmission chain with low G2A delay, the dead time of the chain is low, enabling better control compared to a transmission chain with longer delay. State-of-the-art video transmission systems achieve G2A delays of 80–150 ms. In contrast, the end-to-end (E2E) delays of applications envisioned for 5G and the "Tactile Internet" should be very low, in general smaller than 10 ms. In extreme cases, an E2E delay of 1 ms or less can be required [22], [23]. In addition, E2E delay in a control context includes all delays from the sensor that captures an event to processing, transmission, and finally the actuator delay. Thus, the G2A delay is only part of the entire E2E delay in control applications which further emphasizes the very low delay requirements for video transmission solutions in NCSs.

If the video is presented to a human observer, the relevant delay is the Glass-to-Glass (G2G) delay, which typically has less restrictive delay requirements. The G2G delay is the time difference between a visible event taking place and the event being displayed on a screen. Depending on the application, humans can visually perceive latencies as low as 6 ms for inking on a touchscreen [24], or around 50 ms for interactive applications, such as gaming or graphics tools [25]. In general, the detection threshold for visual delay differentiation by trained observers lies between 8 ms and 17 ms [26]. Kawamura *et al.* [27] showed that the one-foot balancing performance of test subjects wearing a head-mounted display (HMD) increases monotonically when decreasing the delay of the virtual scene presented in the HMD from 66 ms down to 23 ms. In addition, the authors found that a 1 ms delay setup (realized via pose prediction) gave superior task performance compared to the 23 ms delay case. Therefore, it is safe to conclude that humans can not only perceive latencies below 66 ms, but latency reductions down to the delay detection threshold also benefit task performance. For video-based teleoperation [1], [2] as displayed in Figure 1.1b, depending on the usage scenario and the display, a G2G latency between 50 ms and 100 ms is required for a good experience.

In contrast, modern digital video communication implementations typically achieve G2G delays between 100 ms and 200 ms [6]. Thus, most of the applications described in the previous paragraph will not be possible using video communication, or only with a severe quality of experience (QoE) degradation. Together with the many vision-based NCS applications that are only possible with low G2A delay, there is a clear need to reduce latency in video communication. Accordingly, this thesis comprehensively analyzes delay in video communication, and proposes methods to reduce this delay.

1.1 Main Contributions

The first three contributions investigate video communication latency, latency measurement, and the perception of temporal sampling in videos. They lay the basis for the fourth contribution, which proposes various methods for delay reduction. The main contributions are as follows.

1. **Delay analysis and delay model:** Before applying any optimization for delay reduction, it is necessary to investigate which parts of a video communication chain contribute how much delay. The results of this analysis enabled building a detailed E2E model for video communication setups. This model divides G2G delay into atomic steps such as processing of the raw image in the camera, color conversion, encoding, and network transmission. The theoretical delay models for each block were unified into a comprehensive theoretical G2G delay model. This model was confirmed through measurements, see contribution 2. Analyzing the main contributors of the G2G delay model with parameters of state-of-the-art high-end video communication solutions motivated G2G delay reduction methods that focus on increasing the temporal sampling frequencies of cameras and displays.

2. **Delay measurement and delay survey:** To assess G2G delay and individual block delays in video communication setups and to confirm the delay model, precise G2G delay measurements are needed. Existing measurement systems proved to be insufficient, which is why this contribution develops a novel system for accurate G2G delay measurement. The system creates a visual event using a light-emitting diode (LED) in the camera's field of view. Once the visual event is visible on the screen it is detected by a photoresistive element. The measurement circuitry computes the time difference between the original visual event, and when it is shown on the display. The system achieves an accuracy of 0.5 ms. Extending the system to perform G2A measurements enables a further analysis of delays in the video communication chain and measurements to any point in the chain, starting from the camera. Using the measurement system, the G2G delay in state-of-the-art video communication systems was surveyed. Most applications showed G2G delays of 100 ms to 200 ms, which is too large for the low latency applications targeted in this work.

3. **Model for the perception of temporal sampling:** The delay analysis led to the conclusion that increasing temporal sampling rates reduces latency. However, increased sampling frequencies will also inflate data rates and stress processing and transmission units. This is why this contribution proposes adaptive frame skipping to discard irrelevant video frames. Wrongly parametrized, overly aggressive frame skipping will be perceivable and deteriorate subjective video quality. Therefore, this part investigates from a temporal perspective how humans perceive visual signals. Visual signals from the real world are continuous in the temporal domain, whereas digital visual signals (videos) are temporally sampled and thus discrete in time. The discrete sampling instances are images, called video frames. If the temporal sampling frequency is low, humans can, with little effort, distinguish a video from the corresponding continuous visual signal. However, at high sampling rates, humans are not able to perceive temporal sampling. To make frame skipping imperceptible, this contribution determines the threshold at which humans can or can just not perceive temporal sampling. In the theoretical investigations and in the psychophysical experiments conducted to find the threshold, it became evident that the sampling rate threshold depends on the exposure time of the camera and the velocity of the objects visible in the video. These results allowed us to create a model which expresses the sampling frequency threshold as a function of exposure time and object velocity.

4. **Delay reduction methods:** In all proposed latency reduction approaches, a camera with a high sampling frequency is used. To avoid data rate inflation and overburdening of processing or transmission units, the following methods employ frame rate reduction by skipping irrelevant frames.

    a) **Greedy frame skipping:** A greedy approach determines the content difference of a new frame to the previously transmitted frame on a pixel level. If the content difference exceeds a threshold, the new frame carries a visual event and will hence be relevant for G2G latency. Such a frame is forwarded, other frames are skipped.

Our implementation showed a data rate reduction of a factor 40 compared to the high sampling rate video, with an average G2G latency of 21.2 ms, and an average G2A latency of 11.5 ms. Greedy frame skipping is, however, perceivable and deteriorates subjective video quality.

b) **Perceptual frame skipping:** To counteract this deterioration, this contribution exploits the model for the perception of temporal sampling from contribution 3. The contribution comprises an algorithm that enables the application of the model in real video sequences, meaning that the algorithm extracts the velocity of an object in the scene and quantizes the resulting temporal sampling frequency threshold to available frequencies. This result is then used to maintain a sampling frequency that is always at least as high as the threshold for perceiving temporal sampling, irrespective of any frames that are transported because of content differences.

c) **Merged perceptual-greedy frame skipping:** This algorithm is created as a combination of the previous two algorithms. It transmits frames because of content differences (greedy frame skipping), and simultaneously maintains a minimum sampling rate (perceptual frame skipping). This combination achieved a data rate reduction of on average 63 % compared to a high sampling rate video. This is while delay is kept small and there is no perceptual difference between the frame skipped and the full frame rate video.

d) **Preemption:** Finally, the frames forwarded because of content differences are at times stalled because frames forwarded to maintain a minimum sampling rate still need to be processed or are queued. This increases latency because the frames with great content difference carry visual events. This contribution thus proposes a preemption algorithm which discards frames that are blocking visual event frames. In experiments without rate control in video compression, the preemption algorithm reduced G2G latency by a factor five.

1.2 Thesis Organization

This thesis is structured as follows: Chapter 2 provides background information and discusses related work with respect to human visual perception, video communication, and latency in video communication. Chapter 3 gives a detailed analysis of delay in video communication and presents a theoretical delay model. Measurement of delay in video communication is described in Chapter 4, along with a survey of G2G delay in modern video communication systems. Chapter 5 analyzes how humans perceive temporal sampling in visual signals and creates a model for the perception limits of temporal sampling. The insights from the previous chapters are used in the creation and evaluation of G2G delay reduction methods in Chapter 6. Finally, Chapter 7 concludes this thesis and gives possible directions for future research.

Parts of the work presented in this thesis have been published in [3], [4], [6], and [7]. The digital object identifiers (DOIs) of those publications are given in the bibliography.

Chapter 2

Background and Related Work

To enable an understanding of the topics discussed in this thesis, this chapter gives an introduction to the relevant human visual perception limits, fundamentals of video communication, and the latency exhibited by video communication implementations. Simultaneously, this chapter provides an overview over related work.

2.1 Relevant Human Visual Perception Limits

Four aspects of visual perception are relevant for delay reduction in video communication. First, the perception of latency itself sets the target for latency reduction approaches. Second, as we will see in Section 3.7, frame rates and frame skipping play a significant role in the reduction of latency. Therefore, the human visual perception of temporal resolution (frame rate or sampling rate) is relevant. Third, for the subjective tests in Section 5.3, the perception of spatial resolution (pixel density) plays a role. Fourth, the human perception of visual impressions of moving objects is discussed.

2.1.1 Latency

One can distinguish various circumstances for visual latency perception. Let's first consider the situation where a visual reference is given, enabling the subject to compare the delay between the reference and a reaction to the reference in the same field of view. An example for this is touchscreen interaction: a finger or pen serves as visual reference, and the display content under the touch surface is updated as a reaction to the touchscreen input. In applications such as inking on a touchscreen [24], humans can perceive latencies as low as 6 milliseconds (ms). When dragging virtual objects using a touchscreen, Ng *et al.* [28] showed that subjects could perceive latencies down to 1 ms, and that noticeable improvements are achieved when reducing latency well below 10 ms.

Latency constraints are more relaxed for gaming or graphical user interface interaction on a conventional display without a visual reference. For these cross-modal[1] applications,

[1] Cross-modal perception comprises interactions between two or more perception modalities. In the conventional gaming example, the user's visual modality is stimulated by the computer display, while the user also receives haptic feedback from his or her input into mechanical devices such as a keyboard. The user simultaneously processes stimuli from these two different input modalities, which is called cross-modal perception.

the latency perception boundary is approximately 50 ms [25].

For HMDs, the constraints are stricter, even when testing cross-modal latency perception. Kawamura *et al.* [27] found that the balancing performance on one foot increases monotonically when decreasing the delay of the virtual scene presented in an HMD worn by the test subjects from 66 ms down to 23 ms. Additionally, the authors [27] showed that a 1 ms delay setup, realized using the prediction of the subjects' head pose, yielded improved balancing performance compared to the 23 ms delay case. Similarly, Mania *et al.* [26] proved that trained observers can perceive a delay of approximately 15 ms between moving their head and the image with correct perspective being shown in the HMD.

The most fundamental approach to defining what visual latency humans can notice is to determine the perception of the temporal order of two visual stimuli [29]–[31]. Rutschmann [29] showed that test subjects can classify the order of light flashes mostly correctly if the delay between flashes is 20 ms to 70 ms, largely depending on the test subject. These results agree with the investigations of Hirsh and Sherrick [30], whose test subjects could perceive the order of visual events correctly if the delay between these visual stimuli is approximately 20 ms or greater. The authors [30] performed their tests for monocular perception, in which both visual events are projected onto the same retina. If each visual stimulus is presented to a separate eye, humans can perceive stimulus offsets of less than 5 ms [31].

In summary, the ability to perceive latencies depends on the specific scenario under consideration. In presence of a visual reference, latencies as small as 1 ms are required. For HMD-applications that utilize the head pose to adjust the on-screen perspective, delays of approximately 15 ms are the perception limit. For applications on conventional monitors without a visual reference, the latency limit is approximately 50 ms.

2.1.2 Spatial Resolution

The angular resolution of the human eye without any defects is approximately 0.02 degrees [32]. In terms of display resolution, this can be expressed as 50 pixels per degree. Whether a subject can distinguish pixels in a display panel is thus a question of the perspective: the pixels in an old, low pixel-density display might be distinguishable by looking at the panel from a few centimeters, but not if the observer is positioned a couple of meters from the display. An overview of typical viewing conditions in modern display classes is provided in Table 2.1.

2.1.3 Temporal Resolution

Technical video processing systems work with a temporal sequence of discrete images. The human eye in contrast does not perform this temporal discretization, there is infinite exposure time in the human eye [34]. Consequently, information processing in the retina and visual cortex is continuous. Due to the low-pass properties of their visual system, humans can only perceive the time-discreteness of video sequences up to a certain frame frequency, theoretically derived by Watson *et al.* [35]. The authors [35] found that with their experimental setup, two persons can perceive stationary flicker of a light signal up to a frequency

Class	Width [m]	User Dist. [m]	FoV [deg]	Hor. Pix.	Pix. Dens. ρ [1/deg]
TV set	0.8 - 1.3	2 - 5	9 - 36	1280 - 4096	35 - 455
Smartphone	0.06	0.3 [33]	11	720 - 1440	65 - 130
Laptop	0.24 - 0.38	0.5	27 - 42	1440 - 4096	71 - 151
Computer	0.44 - 0.66	0.6 - 0.8	31 - 58	1440 - 4096	25 - 132
HMD	0.09	0.07	110	1080 - 1600	10 - 15

Table 2.1: Typical display and viewing parameters for representative display classes (adapted from [3]). In HMDs, a lens between display and eye widens the covered Field of View (FoV).

of approximately 30 Hz. They note that this is not statistically reliable evidence. Hecht *et al.* [36] showed that the flicker perception threshold depends on the strength of the retinal illumination and is at most approximately 46 Hz.

2.1.4 Motion in Discrete Visual Signals

Videos are spatially and temporally sampled visual sequences. So far, we have investigated the perception in both dimensions separately. Most of the spatiotemporal visual signals we perceive are, however, not separable, in particular moving visual content. It is consequently necessary to investigate where the human spatiotemporal perception limits are. A spatiotemporal perception limit is the point where humans can just distinguish a continuous visual signal from a spatiotemporally sampled visual signal.

As an example, imagine the visual signal of someone waving a hand in front of a static background. The real-world continuous signal is the reference signal, and a recorded video is the spatiotemporally discrete version of the reference signal. In the spatial dimension, an observer could perceive the spatial sampling if the motion speed of the hand is slow enough and if the pixels of the screen are big enough. However, the spatial resolution of modern displays is usually higher than the perceivable spatial resolution of even temporally constant visual signals, see Table 2.1 and Section 2.1.2, so we focus on temporal resolution in the following.

In the temporal dimension, an observer might perceive temporal sampling if the motion velocity of the hand is fast enough and the temporal sampling frequency is low enough. The visual signal will give the impression of jitter or jerkiness. This is intuitively understandable considering that at higher object speed, subsequent images of the time discretized visual signal will exhibit a greater visual difference. If the temporal resolution of the video is very high, an observer will not be able to distinguish the real-world reference signal from the discrete signal. The discrete signal will look perfectly fluid or smooth. Somewhere in between these two extremes is the temporal sampling perception threshold, at which humans are able to tell that the continuous signal is different from the discrete signal 50 % of the time.

Finding the threshold at which humans begin to be unable to categorize a visual signal into the continuous or discrete domain is relevant when applying the frame skipping algo-

rithms proposed in Chapter 6. Those algorithms reduce the frame rate of videos, aiming to make the process of frame skipping imperceptible to the video consumer. The remainder of this section discusses related work in the field of motion perception.

Researchers in the field of psychophysics have thoroughly investigated how the human visual system creates the impression of an object in motion. However, the community has not yet agreed on a theoretical model for the actual neural processes. This renders a perceptually motivated theoretical model of the temporal sampling threshold impossible.

Johansson [34] discusses a major difference between the human eye and camera systems: the human eye does not have a shutter, and accordingly has unlimited exposure time. Nevertheless, humans perceive their moving environment not blurry, but perfectly clear. The author states that neural algorithms which process the continuous exposure data from the photoreceptors on the eye's retina are responsible for this impression.

Derrington *et al.* [37] provide a review of the state of the art in motion perception research. The authors find evidence only to support the assumption of a first-order motion detector on the retina, and propose that there is no evidence for processing steps that are more complex. First order motion detection means determining motion based on the spatiotemporal correlation of brightness or color changes. They review three candidate models for this detection. The paper does not contain a discussion of perception limits.

The results of Derrington *et al.* [37] are in contrast to the findings of Ledgeway and Smith [38], who found hints to support the theory that there are different pathways for processing first-order and second-order motion. Second-order motion is based on spatiotemporal variations such as depth, contrast and relative motion which do not yield a systematic motion in the Fourier domain. It is thus also named "texture-based" motion perception. Examples of such signals are given by Chubb and Sperling [39].

Seiffert and Cavanagh [40] conducted experiments to separately analyze first-order as well as second-order motion. They found that for first-order motion detection, the velocity of an object, and not the spatial displacement determines a lower bound for the perception of motion. Second-order motion perception is performed by tracking object features over time, and its lower thresholds, in contrast to first-order perception, are not affected by the speed of the object to be detected, but by a minimal position change.

Adelson and Bergen [41] suggest energy functions for modeling motion perception. The authors conceptualize two dimensional motion as three dimensional patterns in the x-y-t space. Therefore, motion perception is analogous to identifying an orientation in the x-y-t space. They apply quadrature pair filters to extract spatiotemporal energy which is then further processed to determine perceived motion.

The papers by Watson *et al.* [35], [42] are the most relevant references for this thesis. They are thus discussed in greater detail in Section 5.2.1, and only a short overview is given here. Watson *et al.* [35], [42] conceptualize the spatiotemporal frequency spectra of simple visual signals and show how temporal sampling changes these signals. The Window of Visibility (WoV), see Figure 5.1a, approximates which spatiotemporal signals humans can perceive, and which ones they cannot. The authors [35], [42] utilize both the WoV and the spatiotemporal signal resulting from temporal sampling to theoretically determine a tem-

poral frequency threshold at which the effects of temporal sampling become imperceptible. The frequency threshold is a function of the human perception limits, the motion speed of the visible object, the maximum spatial frequency of the object, the exposure time of the camera, and other parameters such as display luminance and contrast.

The model in [35], [42] does not express the frequency threshold as a function of exposure time. This is what this thesis performs, in addition to evaluating the influence of object velocity on the frequency threshold. Answering the question of the required temporal sampling rate at which humans are unable to distinguish the temporally sampled signal from a continuous signal for a given object velocity and exposure time is hence one contribution of this thesis, presented in Chapter 5.

2.2 Latency Limits in Machine Vision

If a video from a camera is presented to machine vision algorithms instead of humans, different constraints apply. Temporal and spatial resolution are highly application dependent, so for each setup, an individual trade-off optimization between these two parameters has to be performed. Such an optimization is not subject of this thesis, which is why temporal and spatial resolution for machine vision are not further discussed. Latency, however, is highly relevant, which is why related work is presented in the following.

If cameras and machine vision algorithms are used in closed control loops, the video communication latency contributes to the dead time of the control loop. When latency exceeds an application-specific, acceptable limit [14]–[16], the application will lose stability. This is why researchers propose algorithms to compensate for delay [17]–[20] and reduce latency in video processing [5], [43] as well as in the machine vision part [44], [45]. Generally, in machine vision applications, a lower video communication latency is almost always better, even for small latency values.

2.3 Fundamentals of Video Communication

2.3.1 Sampling: From the Continuous to the Digital Domain

With each eye, humans perceive their environment in the following way: the photons, which were reflected from objects, pass through the eye's lens. The lens ensures that photons that originate from the same location in the environment fall onto the same location on the eye's retina. In the retina, the photons are absorbed and further processed in the visual cortex, which creates a moving two-dimensional depiction of the environment.

Videos and cameras mimic how humans perceive the world, see Figure 2.1: camera lenses replace the eye's lens, apertures represent pupils and nowadays, image sensors act as electronic retinas. In the human visual system, each video processing step takes place in continuous temporal and value domains, while the spatial domain is discretized by the photoreceptor cells of the retina. The majority of state-of-the-art electronic information processing systems are digital, meaning that they can only process discrete signals. It is consequently necessary for electronic video processing systems to discretize the visual signal in all three

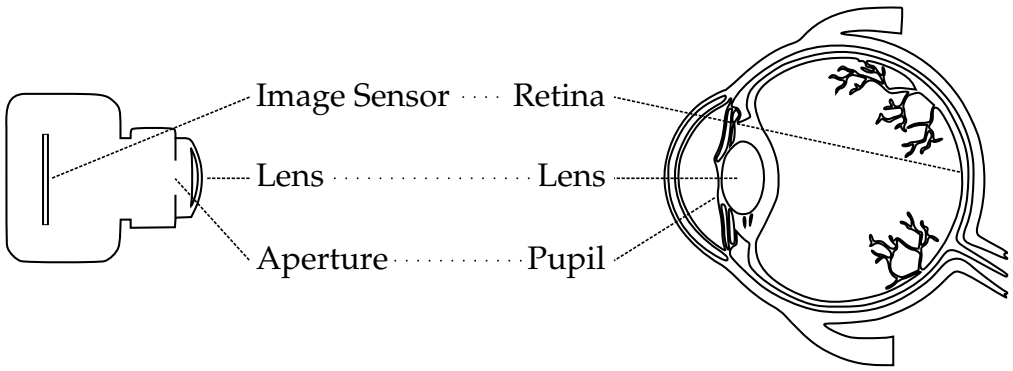


Figure 2.1: Fundamental building blocks of a camera (left), an eye (right), and their correspondences.

domains: in space, in time, and in value. Value discretization is implemented through quantizers in the image sensor. Spatial and temporal discretization are realized by sampling. Both sampling processes are relevant for the remainder of this thesis, and are thus discussed in greater detail in the following sections.

2.3.1.1 Spatial Sampling

Similar to the irregular array of discrete photoreceptors on the eye's retina [46], an image sensor in a camera contains a usually rectangular two-dimensional matrix of picture cells, or short pixels. Each pixel detects the light intensity during the exposure process (detailed in Section 2.3.1.2). The resulting matrix of light intensities yields the corresponding image from the camera and can be processed by digital systems. Consumption by humans is enabled through showing this matrix on a display, constructed by a rectangular array of active pixel elements that emit light with an intensity corresponding to the value from the image matrix.

Whether humans can or cannot perceive spatial sampling in the image depends not on the camera's angular resolution, but on the display's pixel density and the position of the user relative to the display. Table 2.1 therefore gives an overview [3] of representative display classes and the corresponding typical horizontal pixel densities at typical viewing conditions. The pixels in almost all displays are quadratic, hence the vertical pixel density equals the horizontal pixel density.

We see that in the majority of viewing conditions, many display classes exceed the human perception threshold of 50 pixels per degree (see Section 2.1.2).

2.3.1.2 Temporal Sampling

For the temporal discretization of visual sequences, cameras employ temporal sampling. The following describes temporal sampling for one pixel in the image sensor. In state-of-the-art cameras, uniform sampling is employed. Hence, the temporal sampling period (or frame period) will always be the same. One frame period and its neighbors are depicted in Figure 2.2. We can see that one frame period has three phases:

1. **Pre-Exposure:** during this time period, the sensor pixel gets ready for exposure. All previously recorded light intensities (electronic charges) are deleted.

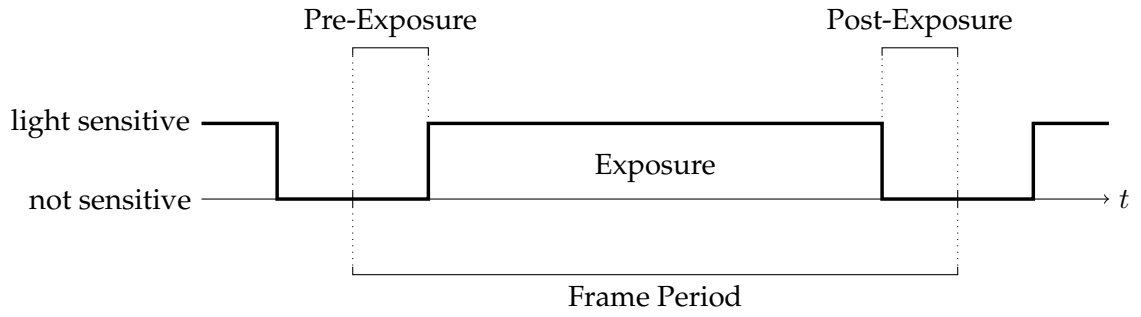


Figure 2.2: The three recording phases of an image sensor during one frame period: preparation of the sensor during pre-exposure, light integration during exposure, and data readout and conversion during post-exposure.

2. **Exposure:** the photosensitive part of the pixel is switched to be sensitive to light. It converts incoming photons to electric charge and stores the charge. This charge is stored in the pixel, thereby the pixel integrates light intensity during exposure time.

3. **Post-Exposure:** the charge is read out of the pixel and converted to a digital representation during quantization in the analog-digital converter. Now the pixel data is ready for further processing steps such as noise removal and white balancing, which are not part of the readout process in the pixel.

Understanding this process allows us to gain two important insights: first, exposure is the integration of light intensity. The light intensity value put out by the pixel will be the average light intensity value observed during exposure. For constant light intensity signals, this average will exactly equal the light intensity at all times during exposure. However, if the light intensity varies during exposure, the final intensity value reported by the pixel will not be equal to the incoming light intensity at all times, it will only be an approximation. Imagine a perfectly sharp black-white boundary of a moving object crossing the pixel during exposure time. After exposure, the pixel will have recorded a light intensity value equal to the ratio of how long it saw the white and the black part of the image. Since exposure time in cameras is usually in the range of 1 ms to 40 ms, this phenomenon is only notable for fast light intensity changes. We encounter this effect most often if an object moves at high velocity through the camera's field of view and exhibits a blurry appearance in the recorded video. Consequently, this is often called motion blur.

The second insight is that exposure time is shorter than the frame period. Consequently, the theoretical derivations in Chapter 5 distinguish exposure time from frame period and respect the fact that the frame period provides an upper bound for exposure time. For consistency with signal processing terminology, the term camera sampling frequency is used as synonym for the frame rate of a camera. Sampling frequency f and sampling period T are reciprocal to each other, meaning

$$T = \frac{1}{f}. \tag{2.1}$$

Displays also have a frame rate for temporal sampling. At the frame rate, the display samples the contents of the graphics card's buffer. It then uses the retrieved data to update (refresh) the image shown on the display panel. For consistency with the camera, and because of the sampling of the graphics buffer, the term sampling frequency will also be used for the rate of the temporal sampling in the display.

2.3.2 Video Compression

Once the sequence of images of the camera reaches a computer, the question of storage or transmission of the video arises. An example illustrates the problem. Assume we have a color image with red, green and blue (RGB) color components for each pixel, and $2^8 = 256$ discrete values per color component. To store this information, we need 3 bytes per pixel. Assume furthermore that the video is recorded at 3840 times 2160 pixels (4K) at a sampling rate of 60 images per second. The data rate of this video will be

$$3 \cdot 3840 \cdot 2160 \cdot 60 \frac{\text{Byte}}{s} = 1.39 \frac{\text{GByte}}{s},$$

which is only available in few high speed interfaces such as universal serial bus (USB) 3.1 Gen 2 and newer, Intel Thunderbolt, or recent display connector standards such as DisplayPort 1.4 or high definition multimedia interface (HDMI) 2.0. For other interfaces, in particular wireless, this data rate exceeds the achievable rate in many cases by orders of magnitude. In addition, storage of videos with such a data rate is not practical in most cases.

The analog problem existed when the first digital video cameras were invented. This was the origin of the research of video compression. Video compression tries to minimize the data rate of a video while affecting the perceived visual quality of the video as little as possible. The video compression community has agreed on video codec standards such as H.264/Advanced Video Coding (AVC) [47] and its successor H.265/High-Efficiency Video Coding (HEVC) [48]. Today, all established video compression standards rely on the same set of fundamental techniques for compressing (encoding) and decompressing (decoding) a video. These techniques shall be briefly explained in the following.

2.3.2.1 Block Structure

Video codecs divide each image of a video into smaller blocks with a size of, for instance, 16x16, 8x8 or 4x4 pixels [47]. This simplifies procedures such as intra prediction and its alternative, inter prediction, which can be done on a per-block basis, and constrains the complexity of the applied transform to enable real-time applicability. The blocks of an image are usually encoded and decoded in row-major order starting at the top left block of the image. The block structure of H.265 is more flexible and thus more complex [49] than that of H.264.

2.3.2.2 Intra Prediction

Intra Prediction utilizes spatial redundancies within one image. Researchers observed that the color value correlation between neighboring pixels and neighboring areas in an image is rather high. This is due to the fact that natural images are often dominated by areas that

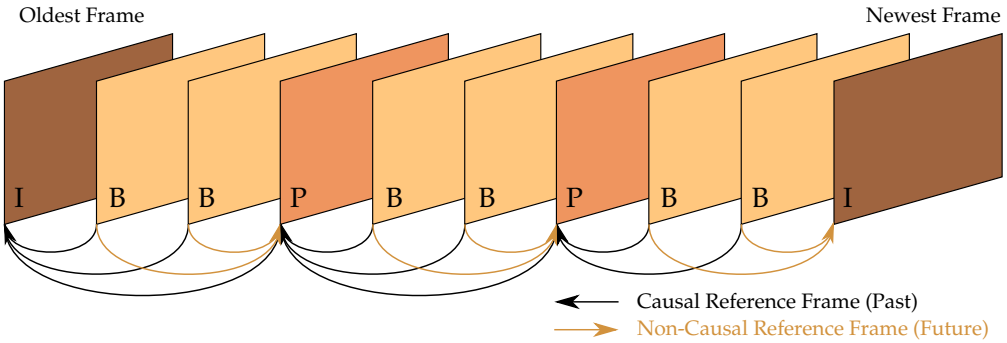


Figure 2.3: Example of the Group-of-Pictures (GOP) structure in state-of-the-art video codecs. Arrows point from a frame to its reference frame(s). Note that the GOP dimensions can change: the number of B frames between two P frames is variable, as well as the number of P frames between two I frames.

exhibit uniform or slowly changing color values. Video codecs use this, and approximate the values of the current block as a function of the surrounding blocks. For intra prediction, only blocks that have already been processed and contain possible compression artifacts may be used, since they are what the decoder can use for reconstruction of the video. Blocks are processed in row-major order from top left to bottom right, so from the surrounding blocks, the three blocks in the row above the current block (two touching the block corners, one touching the top side of the current block) and the block to the left are used for intra prediction. For simple video content (uniform areas), the approximation works better than for complex patterns. The difference (residual image) between the approximation and the true block is further processed, as described in Section 2.3.2.5. Small values in the residual image generally require less bits for representation after the operations from Section 2.3.2.5.

2.3.2.3 Inter Prediction

In analogy to the spatial dimension, there are temporal redundancies in typical videos. Imagine, for example, a block that is part of the background of a video recorded with a static camera. This block's contents will only change if an object from the foreground passes through. Inter prediction tries to approximate this block by creating a weighted sum of neighboring or overlapping blocks from temporally adjacent frames. Similar to intra prediction, for small and simple movements of objects in the video, a better approximation, and therefore a smaller residual and a representation with fewer bits, can be achieved. The search for suitable neighboring blocks is one of the computationally most complex tasks in an encoder [50].

2.3.2.4 The Group of Pictures Structure and Latency

Depending on which kind of prediction is taking place, the images (frames) are named differently, see Figure 2.3:

- **I frame:** in this frame type, only intra prediction is taking place. It is consequently independent of other frames.

- **P frame:** as can be seen in Figure 2.3, these frame types use only past I or P frames for their inter prediction.
- **B frame:** for the two-way inter prediction of this frame type, past and future I and P frames are used, see Figure 2.3. The inter prediction of this frame type is accordingly not causal: the encoder has to stall encoding of all B frames until the next neighboring I or P frame arrives.

Giving an encoder the possibility to perform inter prediction based on past and future frames in some cases enables improved compression efficiency compared with inter prediction that is constrained to past frames for prediction (P frames) [51], which is in turn mostly superior to intra prediction (I frames). This is why in typical video compression scenarios, B frames are the majority of frames, and blocks of B frames can for example be five frames long. Also, the number of I frames should be as small as possible for the highest compression efficiency. There are two major constraints: first, if there is a scene change (entirely new picture), an independent I frame is needed. Second, errors or compression artifacts propagate through inter prediction [52]. Assume that there has been an error during decoding of the leftmost P frame in Figure 2.3. This error will be visible in all frames that use this P frame as reference. Thus, the next I frame will be the first frame without that error. Accordingly, there is a GOP length trade-off between compression efficiency and error suppression.

When using inter prediction in low latency real-time video communication applications, B frames are usually not an option. The latency T_{GOP} added by B frames equals the number of successive B frames n_{bf} between two P or I frames times the frame period T

$$T_{\mathrm{GOP}} = n_{\mathrm{bf}} \cdot T. \tag{2.2}$$

In state-of-the-art video setups with frame periods of $T = 16.\overline{6}\,\mathrm{ms}$ or greater, the contribution of B frame delay T_{GOP} quickly exceeds even the weakest delay constraint from Section 2.1.1, 50 ms for gaming or graphics tools (for $n_{\mathrm{bf}} > 3$). Hence, in low latency video communication, usually an IPPP... structure without B frames is employed. Only using causal references may cause a loss in compression efficiency, but in this case the GOP structure does not contribute any delay.

2.3.2.5 Transform, Quantization and Entropy Coding

The residual from the inter or intra prediction is then transformed, for instance according to the computationally efficient separable integer transform [47]. The transformation enables further processing in the frequency domain, in which quantization can be applied in much less perceivable ways than directly at the pixel level. The coefficients of the transformed image are then quantized and finally compressed using entropy encoding. The quantization coarseness is variable: heavily quantized coefficients will on one hand cause visible artifacts (higher distortion) in the resulting video. On the other hand, because of the fewer quantization levels, the resulting compressed video has a smaller data rate. Conversely, many quantization levels cause a high data rate and yield little distortion.

2.3.2.6 Rate-Distortion Optimization

Section 2.3.2.5 shows that there is a trade-off between the two parameters distortion and data rate. This is why modern encoders employ rate-distortion (RD) optimization. This process finds, depending on a given Lagrangian weight, the optimal point at which we have acceptable distortion at sufficiently low bit rate.

2.3.2.7 Rate control

The temporal (motion) and spatial (patterns) complexity of video content varies over time. Since complex content can be predicted with less accuracy than simple content, the corresponding residuals will be larger, and therefore the data rate will be higher. Thus, data rate will vary with varying content complexity. For low latency video communication over limited data rate channels, this is not a desirable property.

Assume for example that the interface over which we want to transmit the encoded video, can provide a constant data rate of 1 Mbit/s. If the data rate of the compressed video is 2 Mbit/s for one second, we have to temporarily store 1 Mbit in a buffer just before the interface. If the video later has a data rate smaller than 1 Mbit/s, the buffer can be drained. Using a buffer causes additional delay, as data has to reside inside the buffer before it can be transmitted.

To keep buffers as small as possible, mechanisms named rate control keep the video data rate as constant as possible [53]–[59]. This is in many instances done by predicting the complexity of the video content, and adjusting parameters, for instance for the quantization. Related work regarding rate control is presented in detail in Section 2.4.4.

2.3.2.8 Latency Reduction in Video Compression

The creators of AVC and HEVC were aware of the need for low latency video compression, which is why in the common test conditions [60] for encoder-related research, four out of eight test modes have parameters that enable low delay encoding. For instance, these modes define parameters such as the maximum number of contiguous B frames, impose limitations on block size, and constrain the search range and precision for inter prediction. Compared with having fewer or less strict constraints in place, these settings show a degraded compression efficiency, but also reduced encoding and decoding delays.

Similarly, the widely used encoder implementations x264 [61] of AVC and x265 [62] of HEVC define presets and tunings for encoding complexity. These presets are parameter collections which define, in various levels, encoding and decoding settings from low latency and low compression efficiency to high latency and high compression efficiency.

In addition to changing parameters, AVC supports parallel processing [47] for latency reduction on multi-core processing units. The video is divided into non-overlapping slices, which are compressed simultaneously and independently of each other. This approach causes a loss in compression efficiency as correlations that traverse slice boundaries can not be utilized for prediction. Wavefront parallel processing [48] overcomes this issue in HEVC: encoding of a block row of a video frame can be started once at least two blocks of the above

row have been processed. This way, the above blocks can be used for compression and each row can be encoded in a separate thread.

2.4 Latency in Video Communication

The data processing and transmission steps in camera, coders, network, and display require, depending on the implementation and parameters, considerable time (latency). The scientific community has modeled this delay and proposed measurement techniques and methods for reducing this latency. These are discussed in the following.

2.4.1 Existing Delay Analyses and Low Latency Implementations

The first class of papers models the delay contributors of a video communication chain. Schreier *et al.* [63], [64] formalized the latency of video processing blocks such as encoder and decoder relative to the frame period. In their analysis, they do not include delays contributed by camera or display. This is the same in the paper by Vinel *et al.* [65], who considered five contributors for video latency: encoder, encoder buffer, channel, decoder buffer and decoder. Song *et al.* [66] provided a similar model which they use to show the benefit of their intra refresh scheme. None of the previous papers models the camera or display. Baldi *et al.* [67] included the camera in their model, but did not give details on how the camera's sampling process contributes to delay. Also, the paper did not give details about delay contributors after the decoder; all delays were pooled into one contributor, namely processing delay.

Researchers have additionally presented low latency video transmission systems, most notably Holub *et al.* [68]. Their system achieved an E2E latency of 2.5 to 5 frames. With the utilized frame rate of 30 Hz, this corresponds to 83 to 166 ms. The authors did not detail how they measured this latency, or from which points in the video communication chain. The authors could for example have defined E2E latency as delay between the raw image being available for encoding until the image is decoded, and ready for displaying. Such a definition would exclude delays of both camera and display.

Encoding is computationally complex, which is why there has been work on low latency encoders: Inatsuki *et al.* [69] as well as Khan *et al.* [70] proposed hardware implementations of the H.264 standard. They implemented their algorithms on field-programmable gate arrays (FPGAs) and demonstrated low latency encoding. These works disregarded the delay of any other video communication processing blocks.

2.4.2 Latency Measurement in Video Communication

When developing low delay video communication solutions, it is necessary to evaluate them. To do so, we need a system to measure glass-to-glass (G2G) delay. G2G delay is the delay from a visual event taking place in the FoV of the camera, until this event is first shown on the corresponding display. G2G delay is defined in a more detailed manner in Section 3.3.1.

In previous work, there have been approaches to measure G2G delay or a partial delay of video transmission. An overview of these approaches in comparison to the newly devel-

Author	Automatic	Non-Intrusive	Decorrelated	Cost	Precision
Hill [71]	no	yes	no	medium	low
MacCormick [72]	no	yes	no	medium	low
Jacobs [73]	no	yes	no	medium	high
Sielhorst [74]	yes	yes	no	medium	medium
Boyaci [75]	yes	no	no	none	low
Jansen [76]	yes	yes	no	high	low
New method [7]	yes	yes	yes	low	high

Table 2.2: Comparison of previous delay measurement methods with the proposed system (adapted from [7]). The newly developed method [7] is presented in Chapter 4.

oped system is given in Table 2.2. The characteristics depicted in Table 2.2 are detailed in the following.

- **Automatic:** G2G delay is not constant (see Section 3.6.2), it is therefore necessary to gather multiple measurements and analyze the delay distribution. The column titled 'Automatic' describes whether a system acquires multiple delay samples automatically, or if manual assistance is required.
- **Non-Intrusive:** column 'Non-Intrusive' shows whether the video communication system under test needs to be modified and whether other parts than the camera's FoV and the surface of the display need to be accessed.
- **Decorrelated:** taking measurement samples at constant time intervals causes correlation of delay samples, see Section 4.1.3.3 and Figure 4.3. This increases the pairwise mutual information between samples, thereby decreasing the entropy (amount of information) of the samples. It is consequently beneficial for a measurement system to pairwise decorrelate delay measurement samples. For an implementation of decorrelation, see Section 4.1.3.3.
- **Cost:** the cumulative price of the measurement system or its hardware parts.
- **Precision:** most importantly, the measurement error has to be small compared to the absolute delay. The last column of Table 2.2 classifies the precision of the measurement systems.

The entries of Table 2.2 are justified by describing the previous measurement systems in more detail in the following. The systems of Hill *et al.* [71] and MacCormick [72] put a running clock in the FoV of the camera of the video communication system under test. Using another camera which is part of the measurement system, they record both the real clock and the clock shown on the display of the system under test. By computing the difference of the times apparent on the clocks, one can derive the G2G delay of the system under test. This system is not automated; it requires a human operator to read the clock states. In addition, the achievable precision depends on the update frequency of the clock and of the sampling rate of the camera which is part of the measurement system. If we assume for example 60 Hz

in both camera and clock, each would contribute an imprecision of 16.7 ms. Hence, the system would have a low precision of approximately 33.3 ms.

The system proposed by Jacobs *et al.* [73] achieves a significantly higher precision. The system requires an LED to be placed in the camera's FoV, and a photosensitive element, for example a photoresistor, to be put on the display at the position where the LED is shown. Both the LED and the photoresistor are connected to an oscilloscope. The LED is enabled at an arbitrary point in time, which triggers the oscilloscope. From the oscilloscope, one can read the time difference between when the LED was enabled, and when the photoresistor's resistance changed as consequence of the bright LED shown on the screen. This time difference equals the G2G delay, its precision mainly depends on the oscilloscope and is usually smaller than 0.1 ms. This system also requires a human operator to manually evaluate the readings on the oscilloscope.

Sielhorst *et al.* [74] take an approach similar to Hill *et al.* [71] and MacCormick [72], but they replace the clock in the camera's FoV by moving LEDs while the measurement camera filming the entire setup stays still. From the positional differences of the LEDs, an automated algorithm computes the G2G delay. The camera filming the setup runs with at most 200 Hz, contributing an imprecision of 5 ms.

An entirely hardware-less approach is done by Boyaci *et al.* [75]: in software, they embed timestamps as bar codes, which are decoded on the receiver side and used to compute the delay between encoder and decoder. This system does not include delays from camera or display and is intrusive. The necessary modifications in the system under test render the system's application to general latency measurement impossible.

Finally, Jansen *et al.* [76] propose an automated version of the clock-systems by Hill *et al.* [71] and MacCormick [72]. Instead of a clock, they show a changing Quick Response (QR) code with an embedded timestamp to the camera under test. The measurement camera records both the most recent real QR code, and the old one shown on the display. From these, the system automatically computes the G2G delay. However, this system suffers under the same imprecision as the works [71] and [72] because it uses a conventional display for showing the QR codes, and a conventional measurement camera.

All the above systems have at least one drawback that makes usage for extensive evaluation of low latency video transmission systems impossible. Consequently, Chapter 4 proposes a new G2G delay measurement system which combines the advantages of previous work while eradicating disadvantages.

2.4.3 Trading off Compression Complexity Against Compression Efficiency

The number of operations required to (de-)compress one image is termed coder complexity. The complexity of encoders and decoders is proportional to the delay exhibited by them. Therefore, the papers investigating complexity are relevant to the latency of a video communication system.

The video encoder is typically more complex than the decoder because it has to perform the searches for optimal prediction modes, RD optimization, and other computations which the decoder does not have to handle. Consequently, most of the scientific work focuses on

encoder complexity and its minimization. Still, van der Schaar *et al.* [77] created a model for the decoder complexity. This model can be used to formalize decoder complexity in optimization problems.

Many researchers have investigated the complexity and complexity reduction of HEVC. Bossen *et al.* [50] provide a general overview of HEVC complexity, and detail which encoding steps require which percentage of the total encoding time. As a result, they split the total encoding time into intra prediction, inter prediction, entropy coding and others.

Choi *et al.* [78], [79] proposed early transform unit (TU) and fast coding unit (CU) decision. In general, the blocks in HEVC can be split into smaller blocks. Small blocks are suitable for complex patterns or lots of motion, while bigger blocks provide improved compression efficiency for simple patterns or little motion. The encoder has to determine which splitting is optimal for the respective part of a video. Generally, this decision process can be shortened by either exploiting experience gained from many videos, or by analyzing statistics of the present video. In both papers by Choi *et al.* [78], [79], the authors use prior experience to constrain the set of split possibilities by removing rare splits. This process significantly reduces complexity while deteriorating compression efficiency very little. This is a general pattern: by constraining encoder options, researchers accept a small loss in compression efficiency, while trying to achieve a considerable complexity reduction.

Chen *et al.* [80] and Jiang *et al.* [81] use pixel gradient statistics of the current frame to constrain and thereby speed up intra prediction. Khan *et al.* [82] propose an encoder with adaptive complexity and fast PU decision for intra prediction.

The papers presented in this section so far tried to minimize encoder complexity with the least possible impact on compression efficiency. Compression efficiency is equal to RD performance. Hence, the above works perform a simple, non-formal rate-distortion-complexity (RDC) optimization. The following summarizes the literature on formal RDC optimization.

Støttrup-Andersen *et al.* [83] avoid the three-parameter RDC optimization. They convert the RDC optimization problem into a two-parameter optimization by eliminating small differences of either rate or distortion. They use the resulting parameters to constrain the options for inter prediction. For the H.264 codec, they report a speed up factor of 4, while the required bit rate is increased by 1 % at constant quality. Hu *et al.* [84] perform a joint three-parameter RDC optimization. They also use the result to speed up inter prediction.

Finally, Vanne *et al.* [85], [86] created RDC models for both AVC and HEVC. These can be used for RDC optimization.

2.4.4 The Influence of Rate Control on Video Transmission Latency

In addition to encoding and decoding delay, buffer latency can significantly contribute to G2G delay. When only a low data rate video transmission channel is available, the video's variable data rate has to be adapted as close as possible to the available rate, such that on one hand, the channel is fully used and best video quality is achieved, but on the other hand, the queue length in the buffers is as short as possible.

When trying to achieve a predefined video bit rate, encoders face a chicken egg problem: they know the exact amount of data for a compressed frame only after encoding it with

given parameters. However, in low delay video communication, performing multiple encoding probes to find the optimal parameters introduces prohibitively high delay. This is why rate control predicts the bits that are required for the compressed frame depending on parameters such as the quantization coarseness and video characteristics. For the prediction, researchers feed the image analysis data and a target data rate into custom models to retrieve the encoding parameters. Precise and simple models and expressive analysis data are of foremost importance for accurate, low latency rate control.

Ribas-Corbera *et al.* [57] laid the basis for many of the following rate control papers by formalizing and implementing many of the above concepts for the first time in the H.263 [87] video coding standard. Navakitkanok *et al.* [56] later extended these concepts to the H.264 [47] standard. Chang *et al.* [53] use frame data after the transformation (see Section 2.3.2.5) for a rate-quantization model. With their implementation, they can meet a predefined buffer delay time. Gao *et al.* [54] apply their rate-quantization model not at the frame- but at the block-level. This allows extremely precise rate control for lowest latency applications, but introduces noticeable degradation of image quality. Lin *et al.* [55] propose a low complexity rate control scheme for I frame only video compression in very low latency video communication applications. Sanz-Rodriguez *et al.* [58] improved existing schemes for applicability in multi-threaded processors and for high resolution videos.

Zhang *et al.* [59] took another perspective on rate control: the compressed video data consists of two kinds of data, compressed image information and header information. The compressed image information is the entropy coded data from the image itself, while header data contains encoding parameters such as the quantization parameter, prediction mode, reference frames, and more. In particular in modern video codecs, the size of the header is significant in the overall bit stream. Consequently, the authors [59] propose a model for the estimation of header size and combine it with traditional rate control schemes to achieve an accurate rate control.

2.5 Chapter Summary

Latency and latency reduction in video communication have been researched with a focus on encoding and decoding complexity, utilization of parallel computation architectures, and rate control. However, when performing low latency video communication with sufficient computational and network resources, these processing steps are not significant latency contributors, as we will see in Chapter 3. Additionally, no previous project created a comprehensive model of G2G delay and analyzed all delay contributors in detail. In consequence, Chapter 3 proposes a G2G delay model that includes all delay contributors of a video communication setup and identifies elements that can be modified to achieve a substantial G2G latency reduction.

Chapter 3

Delay Contributors in Video Communication

3.1 Block Model of Delay Contributors

The block model of a generic video communication system is depicted in Figure 3.1. On the most abstract level, such a video communication setup consists of three parts: first, the sender, which records the video with a camera and compresses it using an encoder. Second, the compressed video data is sent over a communication network, for example a local ethernet or wireless connection, or a global connection using the internet. Third, once the data arrives at the receiver, it is decoded, and finally shown on a display.

The time required for all these block operations is the G2G delay, defined in Section 3.3.1. If the video is not displayed to a human observer, but to a machine vision algorithm, the display part is omitted, and the corresponding delay is called Glass-to-Algorithm (G2A) delay, see Figure 3.1.

Operation modes between the blocks, delay definitions, block analyses with respect to latency, a theoretical G2G delay model, and an analysis of the delay reduction potential are presented in the remainder of this chapter.

Some of the ideas and contributions of this chapter have been published in [4].

3.2 Cut-Through and Store-and-Forward Operation

Data transmission time between two blocks in Figure 3.1 is the time from the first data bit of a frame being sent out by a block until the last bit of the same frame is being sent out. It is hence the time it takes to transmit one frame over the attached communication interface. For slower communication interfaces (for example USB 3.0), this data transmission can require a considerable amount of time. If transmission time is significant, the mode of operation of the involved blocks becomes relevant. Depending on the mode of operation, a block can, for example, start sending data before it has finished processing a frame, or buffer data until it has entirely processed a frame and then send it out to the next block. The following paragraphs introduce the two operation mode alternatives by example, using blocks from Figure 3.1.

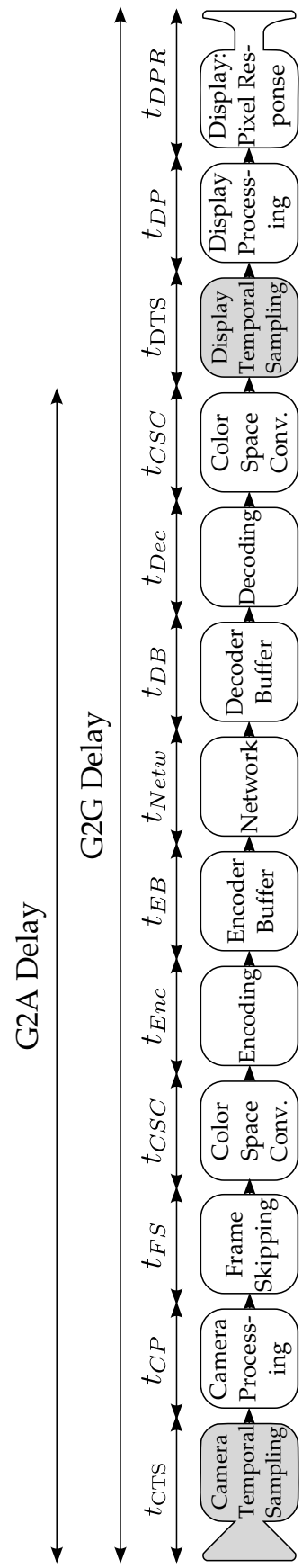


Figure 3.1: Chain model of a video communication system. Blocks with gray background represent delay contributions caused by temporal sampling operations. White blocks represent delay contributions caused by processing (adapted from [4], © 2018 IEEE).

The leftmost block of Figure 3.1, 'Camera Temporal Sampling', represents temporal sampling of the image sensor. Once exposure is finished, data is read out of the sensor. During readout, the data is directly transferred to the next block, 'Camera Processing', which stands for the processing unit on the camera which applies operations such as white balance and debayering [88]. The second block typically works in cut-through (CT) operation mode [89]: while the image sensor is still streaming data, the processing unit already starts to apply its algorithms to the already received data. If all algorithms have been applied to the beginning of the data before the entire frame data was received, the processing unit might already stream it to the next block.

Alternatively, a block can operate in store-and-forward (SF) mode [90]: None of the three phases data reception, data processing, and data sending run simultaneously. Instead, the three phases are handled one after the other. An example for a unit in SF operation is a software encoder: it first needs to receive all data of one frame, then compresses the data using a video codec, and finally sends the compressed data to the next block.

The advantage of CT operations is lower delay. In comparison to SF, it can avoid a waiting time equal to the frame data transmission time over the connection between two blocks. However, it is usually non-trivial to pause processing of CT blocks. In consequence, CT blocks need to be fed with constant data rate. This makes CT operations suitable for intra-device communication of data, but often not suitable for inter-device communication, for example in packet-switched best effort networks.

3.3 Delay Definitions

Delays are among the core variables of interest of this thesis. For humans, the time between a visual event taking place in the camera's FoV and this event being visible on a screen (G2G delay) is relevant. For computer vision algorithms, the time between the visual event and the corresponding image being decoded and ready for further processing (G2A delay) is relevant. Both G2G and G2A delay [4], as shown in Figure 3.1, shall be precisely defined in the following.

3.3.1 Glass-to-Glass Delay

This thesis coins the term G2G delay, as in the context of video communication, E2E delay has been used inconsistently, for instance to describe the time from when an image is available to the encoder until the encoded, transmitted and decoded image is available for display [75]. Such a definition would exclude camera and display latencies. As delay is the temporal difference between two points in time, these points are defined for G2G delay in the following, and naturally, the definition of G2G delay follows.

The first point in time is when a visual event is taking place in front of the camera. This can, for example, be the lighting of an LED or initiation of a linear or rotational movement. This thesis assumes that the time point when the event occurs is equal to the time point when the photons indicating the event pass through the lens of the camera, which this thesis further assumes to be equal to the time point when these photons hit the image sensor (see

Section 2.3.1.2). This approximation is valid because in almost all use cases, the high speed of light, $c \approx 3 \cdot 10^8$ meters per second in earth's atmosphere [91], renders the light propagation delays negligible compared to other delays from video communication (see the measurement results in Section 4.2). For example, assume that the visual event is taking place at a spatial distance of $d = 300$ meters (m) from the camera lens: The photon's propagation time t_{prop} from the event to the lens equals

$$t_{\text{prop}} = \frac{d}{c} = 0.001\text{ms}. \tag{3.1}$$

The resulting t_{prop} is approximately 4 orders of magnitude smaller than the fastest video processing setup examined in this thesis, see Section 6.1.4, which exhibits an average G2G delay of 19.67 ms. The propagation delay will be smaller for less distant events, as well as for the photon propagation from the lens to the image sensor. The photon propagation delay is therefore negligible in standard video applications, and we assume with negligible error that the time points of the visual event and the corresponding photons reaching the image sensor are equal. In specialized applications such as astronomy, we assume that the user is aware of the significant light propagation delay over great distances. To summarize: the first relevant point in time is when the visual event takes place, which is also when the photons of this event pass through the camera lens and are absorbed in the image sensor. The camera lens constitutes the first "glass" in the term glass-to-glass delay.

The second relevant time point is after the visual event has been recorded by the camera, the corresponding image compressed by the encoder, transmitted over a network, decompressed by the decoder, and transmitted to the display. At this point in time, the event is first shown on the display. Analog to the first time point, we utilize the speed of light to approximate three points in time by the same time point: emitted by the display's pixels, the photons indicating the visual event pass through the glass or plastic covering the display panel, and reach the observer's retina. In all known viewing conditions, this approximation produces an error smaller than one microsecond (μs), compare Equation (3.1). The display glass constitutes the second "glass" in glass-to-glass delay.

In conclusion, G2G delay is the time difference, or latency, between when the photons of a visual event are passing through the camera's lens and when the photons of the event first shown on the display are passing through the display glass. This delay includes all processing steps of a video communication system.

3.3.2 Glass-to-Algorithm Delay

G2A delay, as indicated in Figure 3.1, shares the same starting point with G2G delay. The first point in time in G2A delay is also when the visual event occurs in the camera's FoV. However, the second time point differs from G2G delay. For G2A delay, the second time point is when the first image that contains the visual event, has been decoded and is ready to be further processed, see Figure 3.1. This processing is usually performed by a machine vision algorithm such as face detection [92] or object tracking [93]. The delay hence spans the time from the photons passing through the camera's lens glass until the correspond-

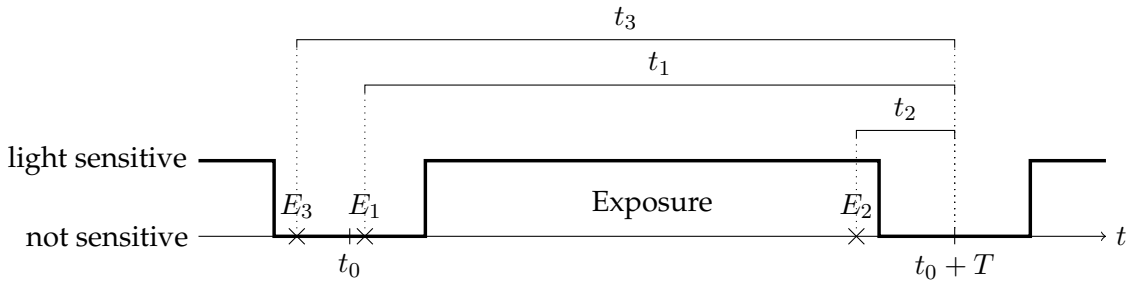


Figure 3.2: Delays t_i inflicted by the temporal sampling of the camera. The delays t_i depend on when during the frame period the event onset E_i occurs.

ing image is ready for the machine vision algorithm. This thesis therefore names this delay Glass-to-Algorithm (G2A) delay.

The processing delays of machine vision algorithms and the actuator delays are out of the scope of this work. Processing speed analysis of machine vision algorithms such as object tracking is a field on its own [93], [94].

3.4 Delay Contributors

The blocks of Figure 3.1 and their theoretical delay models are detailed in this section, and summarized in Section 3.6.

3.4.1 Camera

Any video transmission system starts with image acquisition, performed by the camera.

3.4.1.1 Camera Temporal Sampling

As explained in Section 2.3.1.2, cameras perform temporal sampling. The phase t_0 of this sampling process is generally not synchronized to any of the visual events the camera is recording. Therefore, an event can begin to take place with equal probability at any time during one frame period [4].

The three phases during one frame period, pre-exposure, exposure, and post-exposure (see Figure 3.2), influence the delay contributed by the temporal sampling process of the camera. If an extremely short event would by chance take place entirely outside the exposure phase, it would not be seen at all by the camera. However, such short events are difficult to perceive and process by human or machine observers and therefore do not play a significant role in videos. Additionally, the great majority of real-world events such as motions or general state changes last multiple frame periods. Finally, this thesis can not take any algorithmic or systematic measures to ensure that such short events are recorded. This is why short events outside the exposure period are disregarded in the following.

If an event is initiated during pre-exposure or during exposure, it will be transmitted to the next block, Camera Processing, after the post-exposure, at the end of the frame period. The delay contributed by temporal sampling will therefore be the time difference between

the initiation of the event and the end of the frame period. For example, assume a camera is sampling with 50 Hz, which causes a frame period of $T = 20\,\text{ms}$, see Equation (2.1). Let us assume post-exposure time $t_{\text{post}} = 1\,\text{ms}$. One extreme would be that event onset E_1 in Figure 3.2 happens just after the pre-exposure phase of the camera starts. In that case, almost the entire frame period $T = 20\,\text{ms} \approx t_1$ would pass until the image data containing the event onset is forwarded to the next block. The other extreme is if event onset E_2 happens towards the end of the exposure, in which case there is a much smaller delay $t_2 \approx t_{\text{post}}$.

Event onset E_3 of Figure 3.2 ensues during post-exposure, it will therefore not be forwarded to the next block at the end of the corresponding frame period. Instead, event onset E_3 will be captured during the exposure of the following frame period and further processed at the end of that frame period. The corresponding delay t_3 will therefore be the time difference between the beginning of the event and the end of that frame period, plus the entire $T = 20\,\text{ms}$ of the subsequent frame period.

The above insights allow us to mathematically formalize the delay contributed by the temporal sampling in the camera. The minimum delay occurs if the event comes to pass just before the end of exposure, in which case the delay will equal the post-exposure time t_{post}. The other extreme is if the event onset happens just after the end of exposure, at the beginning of the post-exposure period. In this case, the delay would be the post-exposure processing time plus one frame period $t_{\text{post}} + T$. Temporal sampling in the camera can therefore contribute a delay t_{CTS} in the range $[t_{\text{post}}, t_{\text{post}} + T]$. Within that range, each delay value has equal probability, which is why t_{CTS} follows the uniform distribution

$$t_{\text{CTS}} \sim \mathcal{U}\left(t_{\text{post}}, t_{\text{post}} + T\right). \tag{3.2}$$

In state-of-the-art cameras, t_{post} is small ($t_{\text{post}} < 1\,\text{ms}$). Therefore, the camera frame rate, determining the frame period T (see Equation (2.1)), has major influence on the distribution and the maximum value of t_{CTS}. In a 30 Hz camera, t_{CTS} is at most $33,\overline{3}\,\text{ms} + t_{\text{post}}$, in a 200 Hz camera the maximum is significantly smaller at $5\,\text{ms} + t_{\text{post}}$.

As a sampling process defines the initiation of data readout, it is not relevant to CT or SF operation mode. Distinguishing these operation modes is relevant if the amount of data processed in a block leads to significant processing time, see Camera Processing, Section 3.4.1.2.

In the preceding discussion, we ignored the spatial position of the visible event in the camera's field of view. For the delay contributed by spatial position of an event, the shutter methodology (see Figure 3.3) is relevant: on one hand, global shutter cameras expose each pixel row for the same time interval, subsequently reading each line out and sending it to the next block. Rolling shutter cameras, on the other hand, start exposing the top row of an image, then the second row, and so on. The exposure intervals overlap, but the exposure of each row is started with a small offset after the previous row, so the exposure intervals are not identical. The recorded video is then processed and finally shown on the display, as depicted in Figure 3.3. All conventional displays create the image, similar to a rolling shutter camera, line by line from the top pixel row to the bottom[1]. In Figure 3.3, the temporal extent

[1] https://www.blurbusters.com/understanding-display-scanout-lag-with-high-speed-video/, last visited 31.01.2019

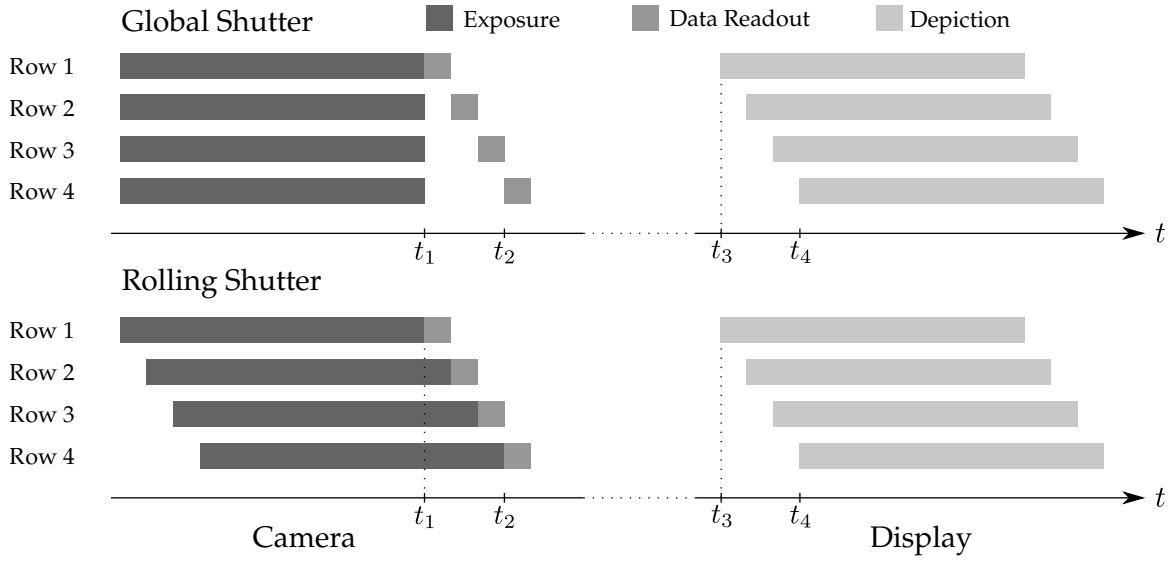


Figure 3.3: Exposure interval arrangement for global and rolling shutter cameras. This image shows only four pixel rows to exemplify the shutter process. Global shutter cameras expose all lines for the same interval to light, rolling shutter cameras and displays record and show lines starting with the top row, respectively. The processing steps between camera and display are irrelevant and thus omitted for clarity. Note that this figure depicts only one exposure and display period for camera and display, respectively.

of each display pixel row represents the time that the pixels emit constant light intensity and color.

Let us first investigate how the shutter procedures influence G2A delay: in the case of global shutter, events in the bottom part of the image are read out later than top parts, so their readout is delayed, equal to the time difference $t_2 - t_1$ in Figure 3.3. The video image containing all this data is, however, processed and finally decoded and presented to the image processing algorithm as a whole data block in memory. In this case, the vertical position of the event in the video does not influence G2A delay, as all pixel rows have to wait for the readout process and all subsequent processing steps to finish: for instance, in Figure 3.3 in the top half, row 1 has to wait for the same time as row 4 until the entire image is available in the image processing algorithm's memory. For a rolling shutter, this is different: there is no delay between the end of the exposure period and the data readout. But since in the end, all lines are presented to the image processing algorithm at the same time, the first row of the video image has to wait until the last image row has been entirely processed. Consequently, from the perspective of the image processing algorithm, the bottom row is the most recent one with lowest delay. The top row, recorded first, carries a delay equal to the exposure interval offset between the top and bottom row ($t_2 - t_1$ in Figure 3.3).

A global shutter would have a negative influence on G2A delay if the video sensor did not support overlapped readout[2]. In overlapped readout, each pixel has an additional local storage for the electric charge recorded during exposure. Using that, the recording proce-

[2] http://hamamatsu.magnet.fsu.edu/articles/readoutandframerates.html, last visited 31.01.2019

dure is the following: every pixel is exposed to light to create an electric charge. At the end of the exposure, each charge is transferred to the additional local storage of the respective pixel, which is a simple and hence fast operation. Afterwards, the next exposure is started. While the next exposure is being performed, the charge is read out of the additional local storage of all pixels. Using overlapped readout, a video sensor can, with an extremely small post-exposure period t_{post}, perform exposures almost back to back and read out data from the previous exposure while the new video image is being recorded. In older video sensors, overlapped readout was not possible, and a global shutter camera would have to wait for all pixel lines to be read out before starting the next exposure. This is why in older sensors, rolling shutter cameras enabled higher frame rates at the same exposure time, and consequently a lower delay.

The delay influence of global and rolling shutter changes if the image is not presented as a whole at one time instance to the image processing algorithm (G2A delay), but shown on a display (G2G delay). As shown in Figure 3.3, a display creates a video image line by line from top to bottom, similar to a rolling shutter. The top row is recorded first and displayed first. This is why for a rolling shutter camera, event location does not influence G2G delay if the temporal offsets between pixel lines in camera and display are equal. If the pixel lines in the camera are exposed in quicker succession than the display's lines are drawn ($t_2 - t_1 < t_4 - t_3$), bottom rows will undergo a greater latency than top rows before being shown on the display. In global shutter cameras, the event position influences G2G delay in a straightforward way: all pixel lines are recorded at the same time, but lower rows have to wait longer before being shown on the display. The bottom row has to wait an additional time period equal to $t_4 - t_3$ compared with the top row. The additional delays of pixel rows in between can be interpolated linearly. In conventional displays, drawing the pixel rows takes the entire frame period, so the bottom row exhibits a G2G delay that is almost one display frame period longer than the G2G delay of the top row.

In the remainder of this thesis, global shutter cameras are considered. We do not further investigate the influence of event location on delays, as it has been discussed in detail here. To retrieve consistent delay measurements in Chapters 4 and 6, visual events are created at the top of the video images. By combining the measurements with the insights of this section, it is possible to compute the G2G and G2A delays for arbitrary spatial event positions.

3.4.1.2 Camera Processing

After exposure, the retrieved per-pixel charges need to undergo multiple processing steps before they yield a color picture. In the first step, the analog charges need to be converted to a digital representation, see Section 2.3.1.2. Second, the pixels in an image sensor suffer from multiple sources of noise, for example thermal or electronic circuit noise, defective pixels, or quantization noise [95]. Modern cameras employ increasingly complex algorithms to suppress these kinds of noise. To create a color image, two dominant approaches are known: in one implementation, a camera can contain three image sensors, each one receiving either red, green or blue light from a prism that splits light from the original scene. The pixel values from the three sensors are combined to RGB pixel values. Alternatively, in a single-sensor

camera, each color pixel in the image is composed of four monochrome pixels with different color filters in the image sensor, arranged in the perceptually optimized Bayer pattern [88]. The process of retrieving one color pixel from the four monochrome pixels is called debayering. To the color image, an arbitrary number of additional post-processing steps such as white balance, ensuring that white in the filmed scene corresponds to white in the image, can be applied.

Within cameras, these algorithms are usually implemented in FPGAs or application-specific integrated circuits (ASICs) for reduced power consumption and higher processing speed compared with a general purpose processor. In an Allied Vision Guppy Pro[3] camera, for instance, all the previously described processing steps require $t_{\mathrm{proc}} = 710 \pm 62.5\,\mu$s. Information about this processing time can typically be retrieved from the camera manufacturer[4].

As the interface between the image sensor and the FPGA or ASIC usually provides a reliable bit rate, camera manufacturers employ the CT concept for the image processing to achieve lowest latency. Also, CT operations avoid costly image buffers.

The definition of t_{CP} also includes the data transmission delay between the camera and the next block. Usually, the camera is attached to a PC or custom board using a USB, FireWire, or Ethernet interface. The raw image size S and the available interface bit rate r determine the corresponding transmission delay

$$t_{\mathrm{interface}} = \frac{S}{r}. \tag{3.3}$$

Keeping the example of the Allied Vision Guppy Pro (on USB 3.0: $C = 5\,\mathrm{Gbit/s}$ data rate) at a resolution of 640x480 pixels with 24 bits per pixel ($S = 640 \cdot 480 \cdot 24\,\mathrm{bit}$), the camera processing delay

$$t_{\mathrm{CP}} = t_{\mathrm{proc}} + t_{\mathrm{interface}} = 710 \pm 62.5 + \frac{640 \cdot 480 \cdot 24}{5000}\,\mu\mathrm{s} \approx 2.2\,\mathrm{ms} \tag{3.4}$$

is smaller than the delay introduced by the temporal sampling of the camera (Section 3.4.1.1). Also, variations in t_{CP} are typically negligibly small because of the hardware implementations, which is why it can be assumed to be constant.

3.4.2 Encoder

In the proposed video communication system, the encoder consists of four parts: first is frame skipping. This block implements the proposed algorithms from Chapter 6, and is optional. Second is color space conversion. This is also optional, and required only if the output color space of the camera does not coincide with the color space required by the encoder. Third is video encoding. The encoder compresses the video frames to reduce the amount of bits required for transmission. Fourth is buffering. The compressed image may have to be buffered prior to transmission onto the network.

[3] www.alliedvision.com, last visited 25.09.2018

[4] In this case the camera processing delay was retrieved from the Allied Vision Guppy Pro Technical Manual V4.1.3, Figure 69 on page 141.

3.4.2.1 Frame Skipping

The G2G latency reduction methods proposed in Chapter 6 rely on using a high frame rate in the camera (small t_{CTS}) while processing only a subset of all frames. For each new frame, this block decides whether the frame should be further processed or skipped. This prevents the data rate from increasing significantly, while reducing latency. The process of selecting a subset of frames is implemented in this block, further details are provided in Chapter 6.

The prototypes proposed in Chapter 6 of this thesis implement this block in SF operation mode. Despite the constant data rate with which images are transferred from the camera, it is not possible to implement the frame skipping in CT mode. The application programming interfaces (APIs) provided by camera manufacturers such as Allied Vision (Vimba[5]), Basler (Pylon[6]), and Ximea (Ximea Software Package[7]) do not signal the initiation of image transfer. Instead, these APIs provide image retrieval functions that return the raw image once it is entirely transferred to the computer's memory, allowing only for SF operation.

If, by using other interfaces or APIs, CT operation would be technically feasible, it would be suboptimal from a logical perspective. Raw images nowadays are stored and transmitted in row-major scan order, starting with the top left pixel. This means that the top left pixel is transmitted first, then the pixel in the same row next to it, until all pixels of the first row are transmitted. This procedure is then repeated for all rows of the image. Assume that 10 % of the image data have been transmitted to the frame skipping block. This means that the top 10 % of the pixels have been transmitted, but the bottom 90 % of the image content are unknown. Based on this information, it is difficult, or might even be impossible to make an informed frame skipping decision.

In the proposed prototype, the delay for frame skipping is small and almost constant at an average of $t_{\mathrm{FS}} = 0.25\,\mathrm{ms}$ for a 640x480 pixel image, so the latency impact of the SF operation in the frame skipping block is not severe.

3.4.2.2 Color Space Conversion

A color perceived by humans is the result of a mixture of photons with various wavelengths hitting the eye's retina. In the retina, cones are responsible for color perception. The three different types of cones have different responsivities as functions of photon wavelength (light color). Each cone type has peak responsivity for photons with a certain wavelength. These three distinct wavelengths represent red, green or blue colored light. Thus, a combination of the three color components red, green and blue of the RGB color space are enough to cover almost all colors that humans can perceive, see Grassmann's laws on color perception [96].

Still, there are more efficient ways of storing color. The human eye has low spatial sensitivity (resolution) to color, in contrast to a high spatial sensitivity to brightness. This is why the YUV color space dissects color into one brightness (luminance, Y) component, and two color (chrominance, UV) components. Further, the two color components are often spatially

[5] https://www.alliedvision.com/en/products/software.html, last visited 25.09.2018
[6] https://www.baslerweb.com/en/products/software/basler-pylon-camera-software-suite/, last visited 25.09.2018
[7] https://www.ximea.com/support/documents/4, last visited 25.09.2018

subsampled to reduce the amount of data required to store an image. Because of this efficiency gain over RGB, and because of its widespread use in older video devices, the YUV color space is used in almost all video compression units. For cameras, on the other hand, the bayer conversion is typically easiest to RGB. The color space conversion block transforms an image from one color space to the other.

The conversion between RGB and YUV is, for each component, a weighted sum of the components of the respective other color space. The color space conversion is applied to single pixels, independent of their surrounding pixels. Hence, these conversions can be done massively parallel. This fact explains the high speed of the conversion: in a prototype that was implemented for this thesis, an average of $t_{\mathrm{CSC}} = 0.32\,\mathrm{ms}$ was observed. For color space conversion, CT operation would be feasible, but since t_{CSC} is small, the corresponding delay reduction would also be small.

3.4.2.3 Encoding

After the raw frame has been converted to the correct color space, it can be compressed in the encoder, according to the description provided in Section 2.3.2. Depending on the used video coding standard and its specific implementation, the encoding delays can differ considerably. The fastest FPGA-based hardware encoders exhibit an encoding delay of as little as $t_{\mathrm{Enc}} = 250\,\mu\mathrm{s}$[8], while the HEVC reference encoder, called the HEVC test model (HM) requires 70 to 80 seconds to encode one frame [50], [60]. Due to the manifold of RD optimization outcomes for encoding one frame in modern video codecs, even within one encoder, per-frame encoding times can show large differences.

However, for low latency, we have to refrain from the most complex compression options, such as two-way inter prediction with sub-pixel accuracy. Within a low latency video communication application, a (manual) RDC optimization (see Section 2.4.3) with high importance of low delay and low complexity can result in almost constant delay with little variation. The prototype discussed in Chapter 6 uses the x264 implementation [61] of the H.264/AVC standard [47]. For a description of the hardware and coding parameters, see Section 6.1.3. In the prototype, x264 encoding of one frame has an average delay of $t_{\mathrm{Enc}} = 0.88\,\mathrm{ms}$ with a standard deviation of 0.05 ms, see Table 6.1. Hence, we assume t_{Enc} to be constant.

Encoders implemented in an FPGA or ASIC usually support CT operation. This is why in many instances, the advertised encoding times are not the time it takes for an entire frame to be compressed, but the propagation delay of one pixel through the encoder. For example, the pixel propagation time through such an encoder might be 1 ms, but the highest frame rate at which the encoder can process incoming videos, could be 60 Hz. Software encoders such as x264 [61], an AVC implementation in the C and assembly programming languages, and x265 [62], an HEVC implementation in the C++ programming language, do not provide any CT functionality. Rewriting a codec to enable CT functionality is out of scope of this thesis, as this would require fundamental changes to the encoding process. Typical codec implemen-

[8] System-on-Chip (SOC) Technologies H.264 AVC Encoder IP Core Datasheet V.4.3, page 5

tations, however, are large software projects: x264 for example consists of approximately 140,000 lines of heavily optimized C and assembly source code.

3.4.2.4 Encoder Buffer

As introduced in Section 2.3.2.7, the data rate of the compressed video produced by the encoder does not necessarily match the available data rate in the network. To deal with this mismatch, a buffer is employed between encoder and network. A highly filled encoder buffer causes a large delay t_{EB}.

Buffers and their fill status distribution have been studied in previous work [56], [57], [59]. Furthermore, with proper rate control (see Section 2.3.2.7), the average buffer queue length can be negligibly short. This is why this section does not provide any further buffer delay analysis here. For the prototype described in Chapter 6, we assume $t_{\mathrm{EB}} = 0\,\mathrm{ms}$, as the available data rate in the ethernet network ($C = 1\,\mathrm{Gbit/s}$) is much greater than the average video data rate of approximately $16\,\mathrm{Mbit/s}$.

Depending on the implementation, CT operation is possible. In software implementations, the usual procedure is SF: wait until one packet containing compressed image data is entirely available before transmitting it to the network. The maximum transmission unit (MTU) , which is an upper limit for packet size, is 1500 bytes for ethernet. With the given small packet size, waiting times for entire packets are small, and thus a lack of CT operation has negligible impact.

3.4.3 Network

Once the compressed image is available in the encoder buffer, it is transmitted to the network, where the data propagates towards the decoder. The two steps of transmission and propagation are detailed in the following.

3.4.3.1 Transmission

Data transmission describes the process of converting the bits stored in the circuit of the encoder buffer to bits represented as, for instance, electrical pulses in a network cable, photons in an optical cable, or electromagnetic waves for wireless transmission. The rate of this conversion is constrained by the physical limits of the wired or wireless channel. In case of Gigabit Ethernet, the bit conversion (transmission) rate equals $C = 1\,\mathrm{Gbit/s}$. In particular for shared packet-switched best-effort networks and wireless networks, the transmission or data rate can vary markedly.

A packet with the size of the MTU (1500 bytes) will require $12\,\mu\mathrm{s}$ for transmission over Gigabit Ethernet. Since the process of transmission is not related to any packet, or does not wait for entire packets to arrive, it is by design a CT operation.

3.4.3.2 Propagation

For the propagation of data through space, this thesis distinguishes three cases: wired, wireless, and multi-hop propagation. In case of wired communication, information propagates

through a copper or fiber optic cable. Propagation speed is usually quantified relative to speed of light, and, depending on the propagation medium, speed ranges from $0.66 \cdot c$ for copper to almost light speed in vacuum $(1 - \varepsilon) \cdot c$, $\varepsilon > 0$ for fiber optics [97]. Wireless information transmission in air achieves a propagation speed close to c, too.

So far, we have only considered a single-line connection. In the more general use case, sender and receiver are connected over a multi-hop network such as the internet. In such packet-switched networks, data packets incur additional queueing, processing, and transmission latencies because they pass through intermediate switches. The delays caused by these mechanisms are highly application specific and can range from less than 1 ms to hundreds of milliseconds.

All three propagation cases are upper bounded by the speed of light c. If, for example, sender and receiver are located at a spatial distance d of 300 km, information propagation delay will be at least 1 ms. For international or intercontinental video communication use cases, information propagation delay plays a significant role. In contrast, in localized applications in which information has to propagate over $d < 1$ km, propagation delay is negligible. Low propagation delay is actually the main motivation for edge computing in 5G networks [98].

Bit propagation is a purely physical process, which is not packet based. It is consequently a CT operation. For multi-hop transmission, the intermediate switches can operate in CT or SF mode.

3.4.4 Decoder

Once the compressed image data is available at the receiver, it is first buffered, then decoded, and finally the raw image's color space is converted to a color space suitable for transfer to the attached display or for processing by a machine vision algorithm.

3.4.4.1 Decoder Buffer

Analog to the encoder buffer (Section 3.4.2.4), the decoder buffer handles disparities between the video data rate and the channel's transmission rate. In the receiver, the buffer must be big enough to avoid a buffer underflow. A buffer underflow can, for example, happen if the data rate in the channel is reduced by severe cross-traffic in a wired network, by interferences in a wireless channel, or by a temporarily large video data rate. In the case of buffer underflow, the decoder is ready to decode a new image, but does not receive any data. If this takes too long, the decoder will miss one or more display sampling periods, causing an interruption of the video playback.

In particular in presence of fluctuating data rates in the network, a large enough decoder buffer is compulsory to avoid buffer underflows. This is why in low latency video communication, a permanently available constant data rate in the network is a prerequisite. With constant data rates, the decoder buffer can be kept small, as well as the decoder buffer latency t_{DB}. In the proposed implementation, we have a constant transmission data rate, and therefore we assume $t_{\mathrm{DB}} = 0$. The fill status of the receiving decoder buffer is closely related to variations in network data rate, which has been thoroughly studied [99]–[103].

CT operation is only useful with non-fluctuating data rates, so it should only be considered when the network can provide a constant data rate. If this is the case, the same CT principles as in the encoder buffer (Section 3.4.2.4) apply: single packets can be forwarded before they are entirely received, but as the maximum packet size (MTU) is small, the gain from CT over SF operation is small.

3.4.4.2 Decoding

The decoding step inverts the encoding process and computes an approximation (if lossy coding, for instance via quantization, was used) of the original, raw image from the compressed image. Decoding is typically significantly faster than encoding as no RD optimization needs to be performed, because the compression options and parameters have already been determined by the encoder. There are heavily optimized implementations such as FFmpeg's[9] H.264 decoder, which requires on average $t_{\mathrm{Dec}} = 272\,\mu$s to decompress one video frame when used in the prototype described in Section 6.1.3. The CT discussion is analogous as for the encoder, see Section 3.4.2.3.

3.4.4.3 Color Space Conversion

The raw image produced by the decoder customarily uses the YUV color space. As opposed to this, modern graphics processing units (GPUs) usually store images in the red-green-blue-alpha (RGBA) color space. The additional alpha component defines transparency of a pixel, which is often needed in applications such as graphical user interfaces or three dimensional virtual environments. As a consequence a color space conversion has to be performed after decoding.

The remaining characteristics, such as color space conversion delay and CT applicability, are the same as for the color space conversion before encoding, see Section 3.4.2.2.

3.4.5 Display

The display with which the video frame is presented to the user is the final part of the video communication system. If the video is instead utilized in a machine vision application, there is no display, and the video communication system ends with the color space conversion after the decoder.

3.4.5.1 Display Temporal Sampling

Just as the camera samples the incoming light with a uniform sampling frequency, displays sample the contents of the graphics buffer in the GPU with the display refresh rate, or update frequency. For consistency, this thesis names the display refresh process *display temporal sampling* in the following. The temporal sampling frequency is limited by display processing such as the raw image data transfer and the panel update process, see Section 3.4.5.2.

[9] https://ffmpeg.org/, last visited 27.09.2018

In traditional displays, the temporal sampling of the display is not synchronized to when the decoder decodes an image. Consequently, the temporal sampling delay t_{DTS} of an unsynchronized display with frame period T is, analogous to the camera temporal sampling, uniformly distributed

$$t_{\mathrm{DTS,nosync}} \sim \mathcal{U}(0, T) . \tag{3.5}$$

An unsynchronized display encounters the effect of image tearing. Image tearing occurs when the display samples the graphics buffer while the decoder is writing a newly decoded image into it. In that case, the display will show two image parts: the top part contains the old image which was in the graphics buffer until the decoder started writing the newly decoded image into the buffer. The bottom part of the display contains the newly decoded image, so the visible image appears to be torn apart.

A countermeasure to tearing is vertical synchronization (VSync). VSync avoids updating of the contents of the graphics buffer while data is being read out by the display. This is implemented with multiple buffers: the decoder writes image data into one buffer while the display reads out the previous image from another buffer. After the data writing and readout processes are finished, buffers are switched. However, VSync introduces additional delay as image data has to wait in a buffer before it is being read out. In the best case, if the decoder is just ahead of the display, and marks a buffer ready just before the display is ready to read the raw image data, the additional delay is one decoder delay t_{Dec} (see Section 3.4.4.2). In the worst case, the display starts reading the old buffer just before the decoder finishes writing data into the new buffer. Thus, the image data in the new buffer would have to wait almost an entire display sampling period T, equal to the setup without VSync. In summary, with enabled VSync, the temporal sampling delay has an increased minimum delay compared to an unsynchronized display, and equals

$$t_{\mathrm{DTS,VSync}} \sim \mathcal{U}(t_{\mathrm{Dec}}, T) . \tag{3.6}$$

Furthermore, there are displays with adaptive sampling, such as NVidia Gsync and AMD FreeSync. Displays using these technologies sample the contents of the graphics buffer only when the GPU signals to the display that the raw image data is ready for readout. Therefore, these displays are not constrained to uniform sampling with a fixed sampling period. Still, such monitors have a minimum sampling period because they also need a minimum amount of time to retrieve data from the graphics buffer and show it on the display panel. In state-of-the-art consumer displays, the maximum temporal sampling frequency is $f_{\mathrm{max}} = 240\,\mathrm{Hz}$[10]. If these displays are operated at a sampling frequency lower than their maximum f_{max}, they cause no temporal sampling delay $t_{\mathrm{DTS,Sync}} = 0$. If fully synchronized displays are used to show images at a rate f equal to or higher than their maximum temporal sampling frequency f_{max}, synchronization will have no effect, as the display always runs at its maximum sampling frequency. In that case, the delay caused by temporal sampling equals the delay $t_{\mathrm{DTS,nosync}}$ from an unsynchronized display. In summary, this thesis distinguishes two cases for the delay $t_{\mathrm{DTS,sync}}$:

[10] for example Acer Predator XB272, Asus ROG PG258Q

$$t_{\mathrm{DTS,sync}} \sim \begin{cases} \delta(0) & \forall f < f_{\max} \\ \mathcal{U}(0,T) & \forall f \geq f_{\max}. \end{cases} \tag{3.7}$$

From these three alternative display temporal sampling synchronization modes, any one can be inserted into the model from Section 3.6.2, depending on the display system that is to be modeled. Analog to the camera temporal sampling, CT or SF is not applicable to a sampling process, as sampling only defines the beginning of a process, not the process itself.

3.4.5.2 Display Processing

Display processing comprises all steps from the start of reading out the graphics buffer until the first pixel in the display panel starts to change its color to the value requested by the data in the graphics buffer. The display processing delay hence comprises the delays of the involved processing steps. There are two main processing steps: data transfer and display signal processing.

Data transfer is analogous to the raw image transfer from the camera, see Section 3.4.1.2. Uncompressed images comprise large amounts of data, and transmission of these images from the graphics buffer to the display requires significant time, even over high speed interfaces such as DisplayPort 1.3, which provides 32.4 Gb/s[11]. The corresponding data transfer delay can be computed using Equation (3.3).

Once image data arrives at the display, the signal processing unit in the display uses the data in its panel driver to control the pixels of the display panel. Some monitors apply extended data processing, for example color correction, or they artificially increase the temporal sampling frequency by interpolating video images. The optional presence of such steps explains the large display processing delay value range observed for monitors. For television displays with heavy processing, values of $t_{\mathrm{DP}} > 100\,\mathrm{ms}$[12] have been observed, while for gaming monitors such as the ACER XB270H, $t_{\mathrm{DP}} = 5.69\,\mathrm{ms}$ were measured, see Table 6.1.

Both steps in display processing are CT operations, the raw image data is not entirely buffered between any of these steps.

3.4.5.3 Display Pixel Response

The display processing unit applies a voltage to the pixels, corresponding to the light intensity that a pixel should emit. The pixels of a monitor panel, however, do not immediately respond to a changing voltage, they require the pixel response time of t_{DPR} to reach the new light intensity state. The model-specific value of t_{DPR} depends on the underlying pixel technique. Panels that utilize active, light-emitting pixels such as organic LEDs (OLEDs) or plasma cells exhibit a pixel response time $t_{\mathrm{DPR}} < 0.1\,\mathrm{ms}$. Competing display technologies such as liquid-crystal displays (LCDs) employ a white backlight for the entire panel. Each pixel partially blocks light such that the image is generated. Using such light filters as pix-

[11] https://vesa.org/featured-articles/vesa-publishes-displayport-standard-version-1-4/, last visited 01.10.2018
[12] https://displaylag.com/display-database/, last visited 01.10.2018

els is slower than light-emitting pixels, and gives delays between $t_{\mathrm{DPR}} = 1\,\mathrm{ms}$ for the Acer XB270H [4] and up to $t_{\mathrm{DPR}} = 25\,\mathrm{ms}$ in older panels [104].

3.5 The Influence of Spatial Image Resolution on G2G and G2A Delay

Many of the delays presented in Section 3.4 are proportional to the amount of data that the corresponding block needs to process. The amount of data that an uncompressed image contains, in turn is proportional to its spatial resolution. While each uncompressed image of the 4K video from Section 2.3.2 contains $3840 \cdot 2160 \cdot 3 \approx 24\,\mathrm{MB}$, a Full High-Definition (FHD) video is spatially sub-sampled to half the number of pixels in both vertical and horizontal direction. Therefore, each raw image contains $1920 \cdot 1080 \cdot 3 \approx 6\,\mathrm{MB}$, one fourth of the data. The delays of all blocks which process the raw images are directly proportional to the amount of data. These are camera processing, frame skipping, color space conversions, and display processing.

Higher resolution images and videos commonly exhibit greater correlation between spatially neighboring pixels. This is due to the fact that natural signals such as images are dominated by low spatial frequency components [105], which is why a higher spatial sampling frequency will only yield a small increase in the entropy of an image. Consequently, the denser, neighboring pixels of a high resolution image are more correlated than the less dense pixels of its low resolution equivalent. This higher correlation is utilized by encoders, and therefore higher resolution images and videos provide a higher compression ratio; dividing the data size of the raw image by the data size of the compressed image yields a greater factor for high resolution images than for low resolution images. Therefore, delays of blocks that process compressed images are not directly proportional to the spatial image resolution, but their delay grows slower than linear with linearly growing data size of the raw images. The blocks that process compressed images are the encoder and decoder buffers, and network transmission. Encoding and decoding relate approximately linear to image resolution, as they have to process the raw video images. Due to the discussed nonlinear growth of entropy, delay growth of the encoder might be a little less than linear in relation to image size because low entropy images are simple to compress and may not require the encoder to evaluate all compression options to find an optimal RD solution. Similarly, less complex techniques used for encoding may lead to faster decoding of an image, resulting in a slightly less than linear growth of decoder delay relative to image resolution.

3.6 Modelling Glass-to-Glass Delay

Until now, separate models for the G2G and G2A delay contributions have been discussed. This section gives an overview of the block models and unifies them into theoretical models for G2G and G2A delay.

3.6.1 Summary of the Delay Models for Individual Blocks

Table 3.1 provides an overview of the blocks discussed in Section 3.4. In particular, Table 3.1 shows the delay distribution, typical delay values, relevant parameters, and the use and applicability of CT operation for each block. Note that some delay distributions are assumed to be deterministic as their real-world delay variations are negligibly small. Frame skipping is one of the core contributions of this thesis (see Chapter 6), and not yet used in state-of-the-art video communication implementations.

Table 3.1 also shows typical block delays for low-end as well as for high-end video communication systems. For low-end systems, we assume a 25 Hz camera and display, slow coders, large buffers, and a great geographical distance between sender and receiver, yielding a high propagation delay. For high-end systems, we assume 60 Hz camera and display sampling frequencies, and high-end components in the remaining system.

Sampling rates in camera f_{cam} and display f_{dis} are two of the few parameters that we can influence with reasonable effort. In addition, we can adjust the network data rate C, and in some settings the distance d between sender and receiver. We can also adjust the weights of the RD optimization in the encoder, which also influences the decoding delay. Finally, the video's spatial resolution can be adapted.

Changing other parameters or algorithms such as camera or display electronics can bring a significant delay reduction. However, developing alternative implementations for such complex products entails a great implementation effort and is accordingly out of scope of this thesis. This thesis will focus on how G2G and G2A delay can be reduced with feasible implementation effort. The G2G and G2A delay reduction potential of video communication is further discussed in Section 3.7.

3.6.2 Theoretical Glass-to-Glass Delay Model

All block delays can be unified into theoretical models for G2A and G2G delay. As we are now interested in delay distributions, this section first defines the probability density function (PDF) of the delay of a block. The PDF p_{block} denotes the distribution of the random variable t_{block} which represents the block's delay.

G2A delay is the sum of all delay contributors, it is therefore defined as

$$t_{\mathrm{G2A}} = t_{\mathrm{CTS}} + t_{\mathrm{CP}} + t_{\mathrm{FS}} + 2t_{\mathrm{CSC}} + t_{\mathrm{Enc}} + t_{\mathrm{EB}} + t_{\mathrm{Netw}} + t_{\mathrm{DB}} + t_{\mathrm{Dec}}. \tag{3.8}$$

For G2G delay, we add the three display delay blocks (see Section 3.4.5) to the G2A delay

$$t_{\mathrm{G2G}} = t_{\mathrm{G2A}} + t_{\mathrm{DTS}} + t_{\mathrm{DP}} + t_{\mathrm{DPR}}. \tag{3.9}$$

Both equations (3.8) and (3.9) describe a random variable which is the sum of other random variables. If all random variables involved in the sum are mutually independent, the PDF of the sum of random variables can be computed by convolving the PDFs of the summands [106]. Mutual independence is not given for the block delays of a video communication setup. Consider, for example, an image with spatially complex patterns. In the compressed version, the image will exhibit a greater size than images with simple spatial

Block Name	Variable	Delay Distribution	Maximum Delay for Low-End Solution	Maximum Delay for High-End Solution	Parameters	CT Operation in High-End
Camera Temporal Sampling	t_{CTS}	uniform	40 ms	$16.\overline{6}$ ms	f_{cam}	impossible
Camera Processing	t_{CP}	deterministic	20 ms	2 ms	-	yes
Frame Skipping	t_{FS}	deterministic	-	-	-	-
Color Space Conversion	t_{CSC}	deterministic	$< 2 \cdot 10$ ms	$2 \cdot 1$ ms	-	yes
Encoding	t_{Enc}	general (Section 6.5)	< 10 ms	1 ms	RDC weights	yes
Encoder Buffer	t_{EB}	general [56], [57], [59]	> 100 ms	< 1 ms	-	yes
Network	t_{Netw}	general	> 100 ms	< 1 ms	C, d	yes
Decoder Buffer	t_{DB}	general [99]–[103]	> 100 ms	< 1 ms	-	yes
Decoding	t_{Dec}	general (Section 6.5)	< 10 ms	1 ms	-	yes
Display Temporal Sampling	t_{DTS}	uniform	40 ms	$16.\overline{6}$ ms	f_{dis}	impossible
Display Processing	t_{DP}	deterministic	> 100 ms	< 10 ms	-	yes
Display Pixel Response	t_{DPR}	deterministic	25 ms	1 ms	-	impossible

Table 3.1: Overview of delays in state-of-the-art low-end and high-end video communication solutions. We assume that the video resolution is fixed. The spatial resolution of a video has additional influence on many of the delay contributors, see Section 3.5. Note that there are two color space conversions in Figure 3.1. Color space conversion is mentioned only once in this table as the conversion has the same characteristics both before the encoder and after the decoder.

patterns. Compressed image size affects, for example, t_{EB} and t_{Netw}. Therefore, these delays are not pairwise independent, which excludes mutual independence [106].

Still, the measurements in Table 6.1 and Section 6.5 show that the error introduced by the false assumption of mutual independence is insignificant. Hence, we assume mutual independence to simplify the following derivations. With this assumption, the PDF of G2A delay t_{G2A} equals the convolution

$$t_{\mathrm{G2A}} \sim p_{\mathrm{G2A}}(t) = (p_{\mathrm{CTS}} * p_{\mathrm{CP}} * p_{\mathrm{FS}} * p_{\mathrm{CSC}} * p_{\mathrm{Enc}} * p_{\mathrm{EB}} * p_{\mathrm{Netw}} * p_{\mathrm{DB}} * p_{\mathrm{Dec}} * p_{\mathrm{CSC}})(t) \quad (3.10)$$

of all blocks from Figure 3.1 and Table 3.1 except the three display blocks. As a consequence of Equation (3.9), the display blocks are included in the PDF for G2G delay

$$t_{\mathrm{G2G}} \sim p_{\mathrm{G2G}}(t) = (p_{\mathrm{G2A}} * p_{\mathrm{DTS}} * p_{\mathrm{DP}} * p_{\mathrm{DPR}})(t). \quad (3.11)$$

The models for the delay distribution of t_{G2A} and t_{G2G} are verified in Section 6.5.

3.7 Analysis of Delay Reduction Potential

In Table 3.1, the column showing delay values for high-end video communication systems gives us a good starting point to identify the blocks with the largest delay reduction potential. We can see that all delays except t_{CTS}, t_{DTS}, and t_{DP} are smaller than 2 ms. The display processing delay t_{DP} requires fundamental changes to the display electronics, which is an extensive topic by itself [107] and out of the scope of this thesis.

Increasing the frame rate of the camera and display, however, is much more viable. Similar approaches were taken by Ishii *et al.* [108], who employ a 2000 Hz camera in a real-time vision system, and by Watanabe *et al.* [109] for a 955 Hz real-time shape measurement system. However, the significantly increased temporal sampling frequency of these systems largely increases the data rate of the compressed video. Another approach tries to overcome temporal sampling using event-based dynamic vision sensors developed by Lichtsteiner *et al.* [43]. In these sensors, each pixel is permanently exposed to light and triggers when the light intensity change exceeds a predefined threshold. They achieve delays as low as $t_{\mathrm{CTS}} + t_{\mathrm{CP}} = 15\,\mu$s. However, such sensors do not provide conventional images, and require fundamental changes to the entire video processing system, including compression and display.

In summary, adapting display electronics or dynamic vision sensors is out of scope of this thesis. For video resolution, the parameter recommendation is trivial: use the lowest possible resolution acceptable in the application, as this yields lowest latency, see Section 3.5. The two remaining parameters are camera and display sampling frequency, for which this thesis recommends the highest possible values, given constraints in energy consumption, computational resources, and circuit properties of camera and displays. To overcome the increase in data rate of the compressed video, Section 6 proposes frame skipping techniques.

Chapter 4

Delay Measurement

When developing novel low-latency video communication solutions, we need a system to precisely measure G2A and G2G delay. As Section 2.4.2 shows, existing systems are insufficient, which is why this thesis proposes novel G2A and G2G measurement systems. These systems are detailed in this chapter. Besides the main target of retrieving accurate delay measurements, additional goals were to develop a simple and inexpensive measurement system. The system is therefore easy to recreate using the building instructions and source code[1].

Parts of this chapter have been published in [4], [6], and [7].

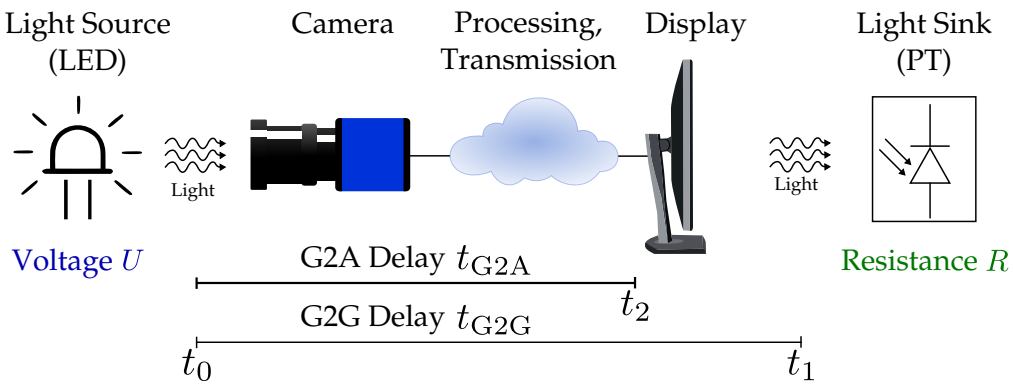
4.1 Glass-to-Glass Delay Measurement

4.1.1 Measurement Principle

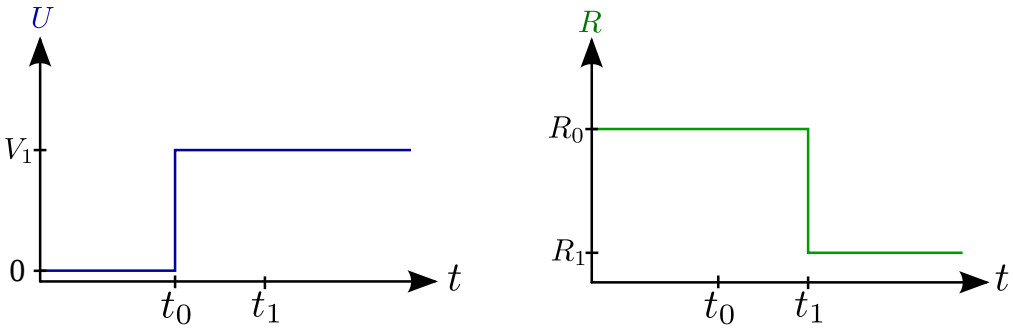
G2G delay comprises all delays from the glass of the camera lens to the photons of the visible event passing through the glass covering the display panel. A video communication setup can be interpreted as delaying the propagation of light, as shown in Figure 4.1a. This is utilized by the proposed measurement system. We trigger a light source in the camera's FoV, and measure the time until the trigger event is shown on the display. This time difference equals the G2G delay.

An LED is used as light source. It is triggered at time point $t = t_0$, meaning that the applied voltage instantaneously rises from $U = 0$ Volts to $U = V_1$, see Figure 4.1b. The LED starts emitting light at time point t_0. The photons from the LED are captured by the camera, and the corresponding video is processed as detailed in Figure 3.1. After a certain time, equaling the G2G delay, the trigger event of the LED is shown on the display. This means that the display's pixels depicting the LED transition from a state in which they emit little light (showing the dark LED) to a state in which they emit much more light (displaying the bright LED). This state transition is captured by a photo transistor (PT) that is placed on the display where the LED is shown. The resistance of a PT decreases with increasing light intensity sensed by it. Thus, when the lighting up of the LED is visible on the display at time point $t = t_1$, the PT's resistance will decrease, for example from $R = R_0$ to $R = R_1$, see Figure 4.1b.

[1] https://github.com/cbachhuber/G2GDelay, last visited 30.11.2018



(a) In the camera's FoV, we trigger a visual event (LED) at $t = t_0$. The propagation of light is delayed by the video communication chain. When the light reaches the light sink at $t = t_1$, the G2G delay $t_{\mathrm{G2G}} = t_1 - t_0$ has passed. For G2A delay measurement, the time point t_2 is determined in software. Details regarding the 'Processing, Transmission' part are given in Section 3.4.



(b) Voltage in the light source (LED) and resistance in the light sink (PT). The time difference in the value changes of the two parameters corresponds to the G2G delay.

Figure 4.1: G2G and G2A delay measurement principle (adapted from [4]).

The corresponding G2G delay t_{G2G} can be computed as the time difference between triggering the LED at $t = t_0$ and the resistance change at the PT at $t = t_1$. The time difference

$$t_{\mathrm{G2G}} = t_1 - t_0 \tag{4.1}$$

assumes a perfect measurement system without any inherent delays and without any noise in voltages, light emissions, or resistance. However, in real implementations, all these imperfections are present. The following sections 4.1.2 and 4.1.3 detail how the proposed solution counteracts such real-world influences.

4.1.2 Hardware System Description

An Arduino Mega 2560 controls the LED. When the Arduino turns the LED on, it simultaneously records time point t_0, see Figure 4.1. In addition, the Arduino monitors the resistance of the PT, allowing it to compute time point t_1 from Figure 4.1. The Arduino can only read voltages, which is why a voltage divider converts the resistance of the PT into a voltage measurable by the Arduino. The voltage divider is designed such that when the resistance of the

PT decreases, the voltage output of the voltage divider increases. This way, the voltage from the voltage divider is proportional to the brightness that the PT observes.

The LED, PT, and Arduino all exhibit their own delays. For the LED, the relevant delays are the turn on delay and the optical rise time. Turn on delay defines the delay between the start of an electrical current through the LED, and emission of the first photons. Optical rise time is, at full current, the time difference between the LED reaching 10 % of its maximum light intensity and reaching 90 % light intensity. In modern LEDs, the sum of turn on delay and optical rise time yields a delay < 10 nanoseconds [110].

PTs exhibit a rise time which is defined as, given an instantaneously enabled light source, the time it takes to change the resistance from 90 % to 10 % of the maximum resistance range. Modern PTs show rise times in the range $[5, 50]$ μs [111].

Complex or clocked electronic circuits such as the Arduino require a non-zero time from a command in software to change or read a voltage until this command is physically executed. There is no relevant literature on this topic, the experiments in Section 4.1.4 found that such delays equal approximately 0.1 ms.

The three presented delays add to the G2G delay reported by the measurement system, wrongfully increasing it. This is due to the fact that the initial software signal in the Arduino makes one round trip through the LED (triggering the LED), through the video communication chain (as first image of the bright led), through the PT (changing its resistance), and back to software. The measured G2G delays lay in the interval $[9, 400]$ ms, see Section 4.2. Consequently, the three delays inherent to the measurement system are negligible for systems with high G2G delay, but become relevant for very low latency systems. The delays inherent to the measurement system are computed by calibrating it and later subtracted, see Section 4.1.4.

4.1.3 Signal Processing

The process of retrieving one G2G delay sample is called one delay measurement. The measurement system repeatedly conducts G2G delay measurements until it has obtained a vector $\boldsymbol{t}_{\mathrm{G2G}}$ of G2G delay samples. This vector can be used to retrieve statistics such as minimum, mean and maximum G2G delay for a video communication system. The procedure of repeated measurements is shown in Algorithm 1 and explained in the following.

One delay measurement starts by enabling the LED, and implicitly recording time stamp t_0 by starting to log the resistance of the PT simultaneously. The system measures the PT's resistance by sampling the voltage of the voltage divider which the PT is part of. The voltage sampling frequency is $f_{\mathrm{msmt}} = 2$ kHz in the proposed implementation, but can be increased if greater temporal accuracy is required. The voltage is quantized to 10 bit, yielding 1024 voltage levels. After turning on the LED, the Arduino records a predefined number of voltage samples, for example $N = 1000$ samples, over a time period of 0.5 s and saves them to the voltage sample vector $\boldsymbol{a}$. Next, the Arduino turns off the LED and applies post-processing to the voltage samples $\boldsymbol{a}$ as described in Section 4.1.3.1 to suppress noise and backlight flicker. This gives the filtered voltage sample vector $\boldsymbol{b}$. The system utilizes Algorithm 2 from Section 4.1.3.2 to determine t_1. As part of Algorithm 2, the Arduino system computes the G2G

Algorithm 1: Procedure for Repeated G2G Delay Measurements

Data: Length K of desired G2G delay sample vector, PT resistance sample count N
Result: G2G delay sample vector $\boldsymbol{t}_{\mathrm{G2G}}$

```
begin
    for i ← 1 to K do
        Enable LED, start sampling PT resistance
        Record N PT resistance samples, store in vector a
        Disable LED, apply post-processing filter (4.3) from Section 4.1.3.1 to a,
          yielding filtered vector b
        Compute G2G delay sample t_G2G,i by applying Algorithm 2 from
          Section 4.1.3.2 to vector b
        Store t_G2G,i in the corresponding position in vector t_G2G
        Wait for a random time, as detailed in Section 4.1.3.3
```

delay sample

$$t_{\mathrm{G2G,i}} = t_1 - t_0 - t_{\mathrm{inh}}, \tag{4.2}$$

where t_{inh} are the delays inherent to the measurement system presented in Section 4.1.2. The value $t_{\mathrm{G2G,i}}$ is finally reported as G2G delay sample and stored in the vector $\boldsymbol{t}_{\mathrm{G2G}}$ of G2G delay samples. Before starting the next measurement, the Arduino waits for a random, short time period, as detailed in Section 4.1.3.3 to avoid correlation of subsequent measurements. After enough, for instance K, G2G delay samples have been recorded, they can be further analyzed in order to extract delay statistics of the system under test.

4.1.3.1 Backlight Flicker Suppression

Modern LCDs utilize LEDs to provide backlight for the panel. To adjust the backlight luminance, these displays employ pulse-width modulation (PWM). The voltage applied to the LEDs is not constant, but sampled at a high frequency of, for example, 400 Hz to ensure that flicker is imperceptible. At 100 % brightness, the LEDs of the backlight are always turned on during one backlight sampling period, but for 50 % brightness, the LEDs can be turned on for 50 % of one backlight sampling period, and turned off for the other 50 % of the time. Manufacturers use PWM instead of adapting the voltage of LEDs because LEDs slightly change their emitted light color with changing current [112], PWM is easy to implement in digital systems, and PWM allows for a great luminance range.

For G2G delay measurement with a PT which samples the display's luminance with a frequency of $f_{\mathrm{msmt}} = 2\,\mathrm{kHz}$, backlight PWM leads to a suboptimal signal, see the unfiltered signal $\boldsymbol{a}$ in Figure 4.2. Similar patterns are observed when measuring on display panels such as plasma, in which an electric discharge creates an extremely short burst of light at an imperceptibly high frequency. To be able to detect a consistent brightness increase in such a display, we need to detect a consistent rise of voltage at the voltage divider. To do so, we first filter the voltage samples $\boldsymbol{a}$. The filtered signal $\boldsymbol{b}$ should have two properties: first, it should be smooth, such that the voltage variations caused by PWM are not visible anymore. Second, the voltage values of the filtered signal $\boldsymbol{b}$ should increase as soon as there is a voltage

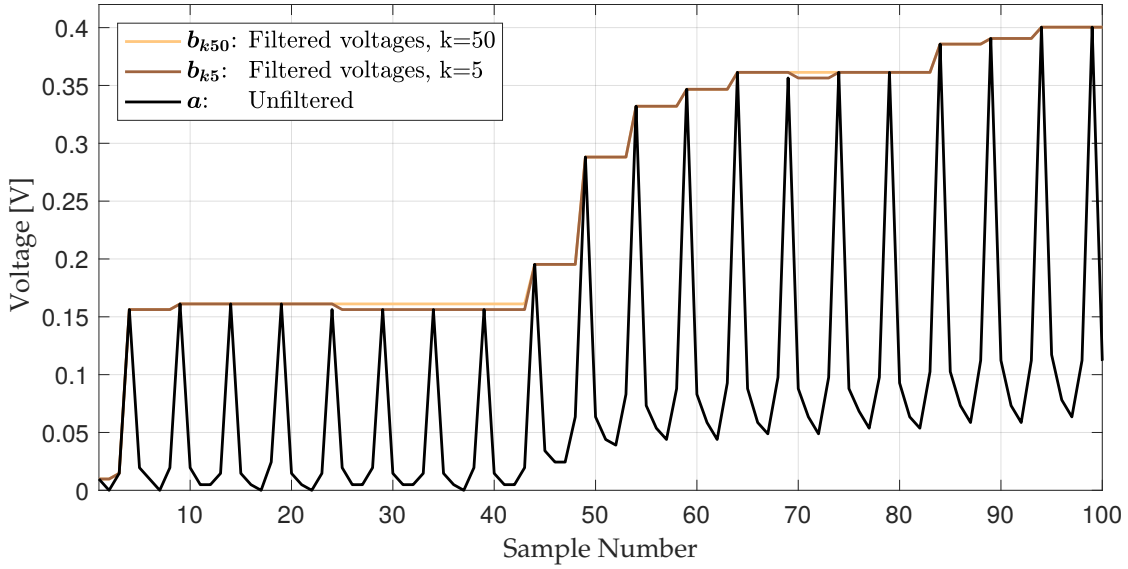


Figure 4.2: Voltage samples measured with a frequency of $f_{\text{msmt}} = 2\,\text{kHz}$ at the voltage divider. Backlight PWM with 400 Hz can be observed in the unfiltered signal $\boldsymbol{a}$, as each PWM period lasts five voltage samples. A significant rise in the voltages starts at sample 44, this is when the luminance observed by the PT consistently increases. The filtered signals $\boldsymbol{b}_{k50}$ and $\boldsymbol{b}_{k5}$ for the two different filter lengths $k = 50$ and $k = 5$ coincide for the majority of samples.

increase in the unfiltered signal $\boldsymbol{a}$, such that there is no delay introduced by the filter. A filter that satisfies these requirements is a maximum filter. Each filtered sample b_i of sample vector $\boldsymbol{b}$ is computed

$$b_i = \max_{\max(0,i-k)\leq j\leq i}(a_j) \tag{4.3}$$

as the maximum value of the current and the past k unfiltered voltage values a_j. Figure 4.2 shows an original signal $\boldsymbol{a}$ with two possible filtered versions for $k = 5$ and $k = 50$. The filter length k has to be parametrized such that the filter covers at least one backlight PWM period. A filter length k longer than the backlight PWM period does not alter the position of a true rising sample in this application, see Figure 4.2. This is why k can be set to large values. A simple rising edge detection, detailed in the following, can be applied to the filtered voltage vector $\boldsymbol{b}$.

4.1.3.2 Rising Edge Detection and G2G Delay Computation

Given vector $\boldsymbol{b}$, the measurement system needs to find the sample index i where the voltage is starting to increase consistently (rising edge). The corresponding algorithm is summarized in Algorithm 2 and detailed in the following.

As can be seen in Figure 4.2, there can be a rising edge in the beginning of the filtered voltage samples $\boldsymbol{b}$, which is caused by PWM and not by a consistently increasing brightness of the display panel. To avoid a wrongful rising edge detection, we start the detection after a predefined number of j samples, such that j samples cover at least one backlight sampling period. The start sample j has to be chosen to be smaller than the expected smallest position

Algorithm 2: Rising Edge Detection and G2G Delay Computation

Data: Vector $\boldsymbol{b}$ of filtered voltage samples, measurement sampling frequency f_{msmt}, inherent delay t_{inh}, thresholds n_{thr} and m_{thr}, start sample j
Result: G2G delay t_{G2G}

1 **begin**
2 $s_{\text{prev}} \leftarrow 0$, $s_{\text{curr}} \leftarrow 0$, $n \leftarrow 0$, $m \leftarrow 0$
3 $N = \text{length}(\boldsymbol{b})$
4 **for** $i \leftarrow j$ **to** N **do**
5 $s_{\text{prev}} \leftarrow s_{\text{curr}}$ `// Previous slope`
6 $s_{\text{curr}} \leftarrow b_i - b_{i-1}$ `// Current slope`
7 **if** $s_{\text{curr}} \geq 0$ *and* $s_{\text{prev}} \geq 0$ **then**
8 $n \leftarrow n + 1$ `// Increment number of subsequent ascents`
9 $m \leftarrow m + s_{\text{curr}}$ `// Update cumulative slope`
10 **if** $s_{\text{curr}} < 0$ **then**
11 $n \leftarrow 0$ `// Reset`
12 $m \leftarrow 0$ `// Reset`
13 **if** $n > n_{\text{thr}}$ *and* $m > m_{\text{thr}}$ **then**
14 **break** `// Found rising edge at current sample` i
15 $t_{\text{G2G}} \leftarrow \frac{i}{f_{\text{msmt}}} - t_{\text{inh}}$

index i of the rising edge. In modern video communication systems, this is not an issue as the periods of backlight sampling are much smaller than G2G delays.

Next, the algorithm loops over the voltage sample vector $\boldsymbol{b}$ by incrementing index i. For each i, the system computes previous (s_{prev}) and current slopes (s_{curr}) as the differences of neighboring samples. If both slopes are greater than or equal to zero, Algorithm 2 increases the number of subsequent non-negative slopes n, and adds the current slope s_{curr} to the cumulative subsequent slopes m. Only if the current slope is negative, the system resets the two variables m and n. This way, Algorithm 2 does not discard positive slopes in the case in which samples b_i are plateauing. This happens for monitors with sampling frequencies lower than the measurement system's sampling frequency f_{msmt}, see voltage samples 45 to 48 in Figure 4.2.

In the final step during each loop, in line 13 of Algorithm 2, the algorithm checks whether the tracking variables m and n have exceeded their thresholds m_{thr} and n_{thr}. This would indicate the presence of a consistently rising edge at the current voltage sample with index i. In this case, we break from the loop (line 14) and retain the value of i, which is where the rising edge starts.

Finally, in line 15 of Algorithm 2, the system computes the G2G delay t_{G2G} according to Equation (4.2). Line 15 corresponds to equation (4.2) as the system starts sampling voltage values when the LED is triggered, therefore at t_0, $i = 0$. When a rising edge is detected, we break from the loop, which happens when the brightness at the monitor is increasing, at t_1. Thus, the time difference $t_1 - t_0$ is, except errors from sampling in the measurement system, equal to the rising edge index i multiplied with the measurement sampling period $1/f_{\text{msmt}}$.

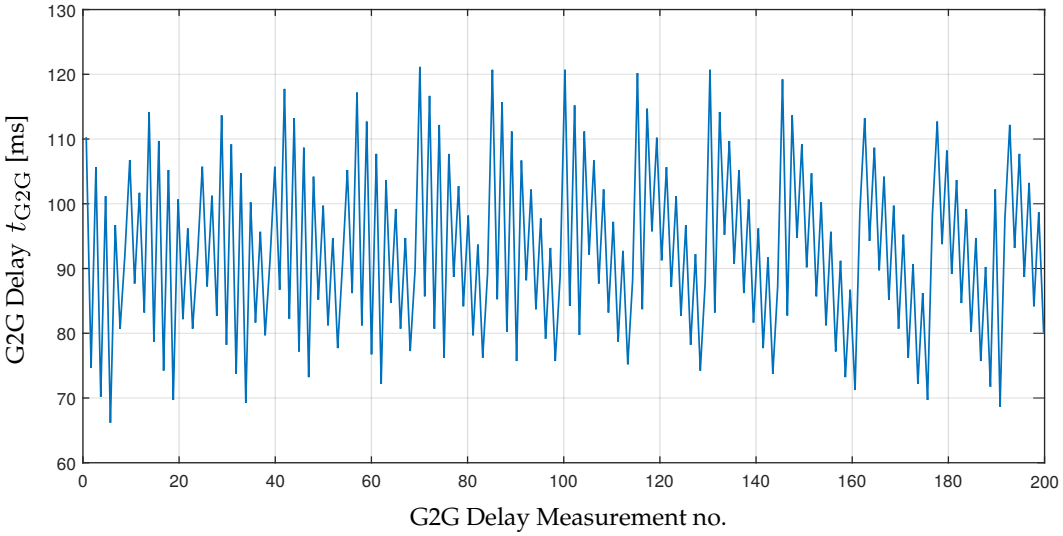


Figure 4.3: Without measures to decorrelate G2G delay measurements, considerable correlations can be observed.

The most relevant parameters of Algorithm 2 are m_{thr} and n_{thr}, defining after how many voltage increases (n_{thr}) and after which cumulative voltage increase (m_{thr}) the system detects a rising edge. In the empirical experiments, it was shown that values $n_{\mathrm{thr}} = 2$ and $m_{\mathrm{thr}} = 34\,\mathrm{mV}$ enable a robust and reliable rising edge detection. These parameter values largely depend on the specific implementation; the proposed implementation, for example, uses an OSRAM LPT80A as PT, and an 11 kΩ resistor in the voltage divider.

4.1.3.3 G2G Delay Sample Decorrelation

As motivated in Section 2.4.2, if no countermeasures are taken, groups of subsequent G2G delay measurements will be correlated. This is caused by the fact that both the video communication system under test and the measurement system are sampling systems which record data (a video image or a set of voltage values $\boldsymbol{b}$) at constant sampling periods. Note that in this context, the measurement system's delay measurement frequency (one loop in Algorithm 1) is debated, not its voltage sampling frequency f_{msmt}. The delay measurement frequency can, for instance, be one G2G delay measurement per second. Consequently, the delay measurement system will observe the other system only at a few different phase shifts. This concept becomes understandable by the use of an example: assume that both sampling processes in the video communication systems run at a frequency of exactly 60 Hz, all other delays are constant, and we take G2G delay measurements at a frequency of precisely 1 Hz. In this case, the measurement system will always report the same t_{G2G}. Measurements of a USB camera to PC setup with an approximately constant measurement period are shown in Figure 4.3. We can observe that the system reports only a few delay values, according to the phase shifts between the sampling processes. Also, given past delay measurements, it is possible to predict the following measurement to some extent.

This scenario should be avoided because of two reasons: first, it does not capture or only partially captures the effect of the uniformly distributed sampling delays t_{CTS} and t_{DTS}.

Usually, events in the camera's FoV are not synchronized to the sampling process, which should be represented by the measurement system. Second, the information gained by a new measurement is very little because of the large correlation.

Decorrelation of the measurements is achieved by waiting for a random time after each G2G delay measurement has been completed. Using this technique, each new G2G delay measurement has no correlation to previous measurements and is distributed according to Equation (3.11), which incorporates both uniform sampling delays t_{CTS} and t_{DTS}.

4.1.4 System Evaluation

The G2G delay measurement system is first evaluated in terms of correctness of the rising edge detection in Algorithm 2. An oscilloscope is connected to both the LED and the voltage divider including the PT to observe the LED's voltage and the PT's resistance. The LED voltage is used to trigger a measurement in the oscilloscope. The oscilloscope records PT resistance data comparable to Figure 4.2, from which the rising edge and the corresponding G2G delay t_{G2G} can be manually extracted. The manually computed G2G delay value is then compared to the value reported by the measurement system. This confirms that Algorithm 2 is working correctly.

In addition, the analysis from Section 4.1.2 did not result in a precise value for the delay t_{inh} inherent to the measurement system. As a consequence, we need to calibrate the system to find t_{inh}. Calibration is performed by removing the video communication system from Figure 4.1a. In this case, LED and PT are spatially co-located, and $t_1 = t_0$. This way, Equation (4.2) simplifies to $t_{\mathrm{G2G}} = t_{\mathrm{inh}}$, so the measurement system will report the inherent system delay t_{inh} as G2G delay t_{G2G}. This approach found an inherent delay of $t_{\mathrm{inh}} \approx 0.255\,\mathrm{ms}$ for the measurement system.

Given the fully calibrated system, the only inaccuracy in reported G2G delay values occurs because of the temporal sampling of the PT resistance at $f_{\mathrm{msmt}} = 2\,\mathrm{kHz}$. The corresponding sampling period $T_{\mathrm{msmt}} = 0.5\,\mathrm{ms}$ defines the accuracy of the measurement system. For the following experiments, an accuracy of 0.5 ms proved to be sufficient. Still, if required, the system's accuracy can be increased by increasing the voltage sampling frequency.

4.2 Glass-to-Glass Delay in State-of-the-Art Video Communication Systems

The G2G delay measurement system enables a survey of the state-of-the-art G2G delay in video communication [6]. This survey should analyze how much delay reduction necessity and potential are given in modern video communication applications.

4.2.1 Video Communication Systems under Test

Four categories of video communication systems were examined: video conferencing applications, video transmission for teleoperation, the camera applications of smartphones and low delay video communication prototypes. These are detailed in the following.

4.2.1.1 Video Conferencing Applications

As representatives for video conferencing applications, three of the most popular video chat services were chosen: *Microsoft Skype*, *Apple FaceTime* and *Google Hangouts*. In addition, the open source platform *Jitsi Meet* was tested. All services were tested with two *Apple MacBook Air* as sender and receiver. The laptops were sharing the same wired network in the same room. Jitsi Meet was hosted by the *Leibniz-Rechenzentrum* in Munich[2].

4.2.1.2 Video Transmission for Teleoperation

With increasing network capabilities and decreasing latencies, video-supported teleoperation is becoming more and more commonplace. Therefore, we look into different teleoperation systems: first, a *DJI Phantom 2 Vision* + quadrocopter drone connected via wireless LAN to a *Samsung Galaxy S4* for displaying the live video.

Second, a first person view (FPV) system offered by *FatShark*. An analog (FatShark Analog) camera, the *600TVL Sony Super Had II CCD*, is mounted on a quadrocopter drone and records a video with 50Hz. The video is transmitted using analog wireless communication and finally displayed with again 50Hz in the *FatShark Dominator 2* FPV goggles worn by the drone pilot.

Third, the analog camera and transmission system is replaced by a digital system (FatShark Digital), the *Connex Prosight HD*, which transmits the 30Hz video digitally, but still uncompressed. The FatShark goggles are again used as displaying device, but now with 60Hz.

Fourth, an *Oculus Rift 2 Development kit* HMD is being tested. The Oculus display is attached to the low delay video communication prototype described in Section 4.2.1.4. It is employed in Extended Display Mode[3], such that the HMD is handled as an additional display and the video can be directly shown on it without the need to perform any rendering for virtual reality.

4.2.1.3 Smartphones

To demonstrate the G2G delay of just the hardware in mobile devices, the G2G latency of the built-in camera applications is measured. This is the time from an event taking place in the smartphone camera's FoV until this image has been processed in the smartphone (for example color correction) and is shown in the live preview of the camera application on the smartphone's display. Five smartphones are investigated: the *LG Nexus 4, the Huawei Nexus 6P, Huawei Mate 20 Pro*, the *Apple iPhone 6*, and the *Apple iPhone XR*.

4.2.1.4 Low Delay Video Communication Prototypes

Finally, it should be shown that video transmission applications can be tuned for very low G2G delay. To this end, this thesis proposes two low delay prototypes. Both employ a Ximea

[2] https://meet.lrz.de/, last visited 14.02.2019
[3] https://developer.oculus.com/documentation/pcsdk/0.4/concepts/dg-monitor-setup/, last visited 13.02.2019

MQ022CG-CM USB 3.0 industrial camera and an Acer XB270H monitor. The first prototype (Raw Prototype) directly feeds the raw images of the camera running at 200 Hz to the display, which is employing a sampling frequency of 144 Hz.

The second prototype (Coding Prototype) takes the uncompressed video from the camera at 240 Hz, encodes it using the x264 [61] software encoder with very low delay settings and transmits the encoded video stream over a network with a channel rate of $C = 1\,\text{Gbit/s}$ to a second PC. The second PC decodes the video using the libav decoder[4] and displays it on the Acer XB270H display which is running at 144 Hz. The Coding Prototype represents the general point to point video communication chain from Figure 3.1.

4.2.2 Discussion of Results

For every system, the survey took $K = 500$ G2G delay measurements. Table 4.1 shows the measurement results for all systems and the corresponding box plot is shown in Figure 4.4. The results provide several interesting insights: first, one can clearly see how conversational applications such as Skype, FaceTime, Hangouts and Jitsi Meet have high G2G delays of over 100 ms up to almost 400 ms in the case of Google Hangouts. It is obvious that all video conferencing applications satisfy the recommendation of the ITU to have a communication delay of at most 400ms [113]. They do not try to achieve lower latencies, as an average latency of 250 ms is acceptable in conversational video communication. Instead, video conferencing applications minimize the number of playback interruptions given possibly unreliable channel rates by using large encoder and decoder buffers. In addition, these applications allow a high encoder and decoder complexity and enable B-Frames (see Section 2.3.2.4) for an optimal compression efficiency.

Jitsi Meet experienced a short time with high G2G delays, as can be seen in Figure 4.4. After that, it returned to the G2G delays seen previously. This might have been caused by high server occupancy or cross-traffic. Ping tests between north American and European servers yielded a mean round-trip time of 141 ms. Between Europe and Australia, the round-trip time was 410 ms. By adding half of these delays to the measured videoconferencing delays in Table 4.1, the local video conferencing measurements can be extended to intercontinental measurements. In case of the largest additional delays ($410/2 = 205\,\text{ms}$), the average G2G delays of video conferencing applications just exceed the ITU recommendation of 400 ms. Respecting the point-to-point connections with the longest data propagation time might have been a design constraint for these applications.

Second, we see that the DJI Phantom system has a mean G2G delay of 255 ms similar to the video conferencing software. In case of the DJI Phantom, this is acceptable because DJI quadrocopter drones are designed for recording cinematic videos and not for more dynamic scenarios such as the FatShark systems. Fatshark's analog video transmission exhibits a significantly lower G2G delay with an observed mean of 28.33 ms, the digital variant has a mean delay of 57.35 ms. Note that for both FatShark systems, no video compression is taking place.

[4] www.libav.org, last visited 09.10.2018

	Minimum [ms]	Mean [ms]	Maximum [ms]	Std. deviation [ms]
Skype	121.15	233.73	374.14	47.52
FaceTime	180.41	222.47	274.75	18.31
Hangouts	131.58	267.69	385.98	65.71
Jitsi Meet	182.53	236.17	372.80	29.89
DJI Phantom	210.81	254.66	330.43	20.68
FatShark Analog	19.97	28.33	38.98	5.26
FatShark Digital	42.69	57.35	79.81	7.59
Oculus Rift	30.08	46.23	57.22	5.59
Nexus 4	66.56	102.42	176.25	18.07
Nexus 6P	63.11	92.98	127.49	13.41
Nexus 6P Video	53.82	84.10	109.95	11.93
Mate 20 Pro	56.96	80.71	107.01	11.04
iPhone 6	54.20	76.06	101.05	10.31
iPhone XR	85.95	106.78	129.91	10.31
Raw Prototype	9.28	14.67	20.29	2.44
Coding Prototype	13.83	19.18	33.47	2.56

Table 4.1: G2G latency statistics of the delay survey of state-of-the-art video communication systems. For each system, the statistics are based on 500 G2G delay samples (adapted from [6]).

Third, we see a comparably large G2G delay in the Oculus Rift. The same system as in the Raw Prototype described in Section 4.2.1.4 is used, the only difference is the output device. This is either the Oculus Rift or the Acer XB270H. The G2G difference is on average 31.56 ms. Oculus Chief Scientist Michael Abrash recommends a G2G delay of 15 ms or even 7 ms[5] for augmented reality applications. The company hence has to drastically improve the delay characteristics of the device. We could not perform comparable measurements with the more recent *Oculus Rift Consumer Version 1* HMD, as this product does not support the Extended Display Mode. The Consumer Version 1 HMD can only be utilized by applications which are based on Oculus' virtual reality libraries and drivers. Hence, it would be necessary to render the video within a three-dimensional virtual reality scene supported by Oculus, and then let Oculus' drivers transfer the image to the HMD. This adds two unknown delays: the rendering delay and the delay contributed by Oculus' HMD driver. In conclusion, measuring the G2G delay of a video transmission setup that uses the Consumer Version 1 HMD would in addition to delays inherent to the display (see Section 3.4.5) include the rendering delay and the unknown delay of Oculus' HMD driver. This would be neither a fair nor an insightful comparison to the other displays in this paragraph, thus the *Oculus Rift Consumer Version 1*

[5] http://blogs.valvesoftware.com/abrash/latency-the-sine-qua-non-of-ar-and-vr/, last visited 10.10.2018

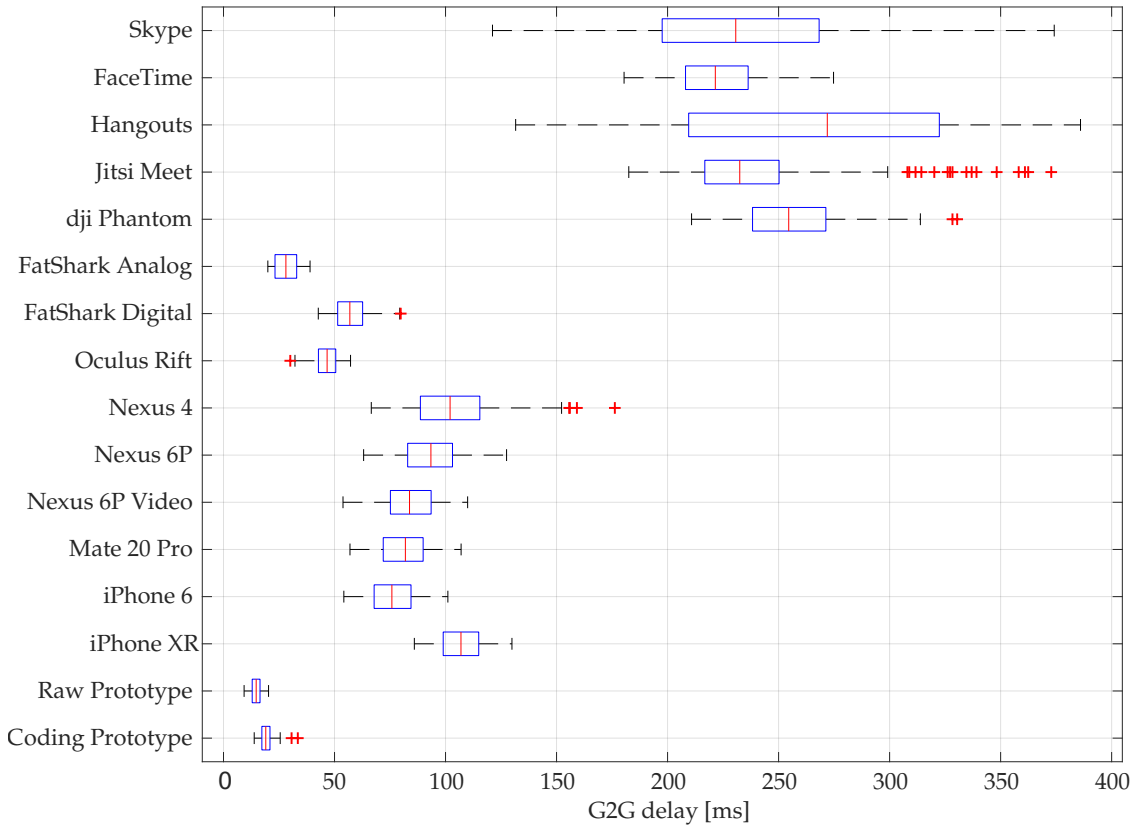


Figure 4.4: Box plots of the G2G delay of the evaluated systems. The box contains 50% of the measurement samples surrounding the median, which is represented by the red line. The whiskers end at the sample most distant from the median, at the same time the whisker's length is at most 1.5 times the box width. Samples more distant from the median are denoted with red pluses (adapted from [6]).

is not included.

Fourth, for the Nexus 6P, we compare the G2G delay when the camera application was in the photo mode (Nexus 6P) versus the video mode (Nexus 6P Video). There is roughly a 10 ms difference between the modes, indicating that additional processing or buffering is performed before the video stream is displayed in the photo mode. For other smartphones, we only report the photo modes, but we saw similar latency differences when switching to video modes and even larger differences when switching to slow-motion modes with high frame rates and short camera sampling periods. The rather high G2G delays in smartphones show that additional processing is taking place, leading to a considerable G2G delay even though no video transmission is involved. Smartphone manufacturers seem to be satisfied with the G2G latencies of the camera applications of their devices, as there is no clear trend towards lower latencies when comparing the delays of older (for example Nexus 4, released in 2012) and more recent devices (for example iPhone XR, released in 2018) in Figure 4.4. For camera applications, customers probably prefer additional functionality such as face detection and complex image enhancements over ultra low latencies.

Fifth, to demonstrate that video systems can meet the low latency requirements of 5G [22] and the Tactile Internet [23], let us next discuss measurements for the delay-optimized prototypes. The Raw Prototype performs as expected, the Coding Prototype has two outliers

at 30.66 ms and 33.47 ms, as shown in Figure 4.4. This is due to the fact that the involved computers are not running a real-time operating system and interrupts and scheduling of the operating system cause delays. Without those outliers, the system achieves a maximum delay of 24.54 ms and a mean delay of 19.13 ms. With real-time kernels, a video transmission prototype that can guarantee such a low worst-case delay could be developed. When comparing the average G2G delays of the raw prototype (14.67 ms) and the Coding Prototype (19.18 ms), we see the difference is less than 5 ms. Hence, the Coding Prototype performs all steps related to encoding, network transmission and decoding in less than 5 ms.

Overall, many of the studied video applications have large G2G delays. As investigated in [7], the camera and display delay contribute approximately 30 ms delay in a state-of-the-art video communication setup. The remaining significant G2G delay contributors are data processing such as color conversion and image compression, which is usually in the one-digit millisecond domain, buffering, and propagation. The last two can vary widely for different applications. The proposed prototypes show, that with off-the-shelf technology, one can achieve very low G2G delays. With adapted hardware and algorithms novel applications in collaborative gaming, education, teleoperation and automated visual process control become possible.

4.2.3 Comparison to Related Work

Other authors previously measured the delay of video communication applications. The remaining paragraph performs a comparison and evaluates the technological progress in the field. For Skype, this thesis found an average delay of 234 ms, which is close to Boyaci's measurement of 233 ms [75] and Jansen's [76] 274.5 ms one-way delay, but quite different from Xu's results with 156 ms [114]. Influences from network and server occupation can explain the difference. For the FaceTime predecessor iChat, Xu *et al.* [114] measured 220 ms average delay, which compares very well to the recorded measurement of 222 ms. The difference is larger for Google Talk, with a measured delay of 99ms by Boyaci *et al.* [75], comparing to the successor Hangouts with a mean G2G delay of 267.69 ms, as measured in the experiments of this thesis. Xu *et al.* [114] measured a mean delay of 180 ms in the Hangouts predecessor Google+. Again, server and network occupation are most probably responsible for the difference. Considering that the delays measured in related work in the years 2009 to 2013 show almost no difference to the new results from 2017, there was no progress related to latency in the field of video conferencing.

The comparisons become more interesting for augmented reality systems. In 1993, Mine [115] measured 70 ms from the computer until the corresponding image is displayed, Sielhorst *et al.* [74] measured 50 ms to 230 ms G2G delay in 2007, and this thesis found on average 50 ms for the Oculus Rift. In HMDs for augmented reality, a clear progress is identifiable. The reason for this development is that in contrast to conversational video, augmented reality applications are required to strive for an extremely low delay video transmission for a good user experience [27].

4.2.4 Accordance of Measurement Results with the Theoretical Delay Model

The theoretical G2G delay model (3.11) is a convolution of many delay contributors. From these, most are deterministic, or their standard deviations are small. As analyzed in Section 3.7, the temporal sampling processes in both camera and display contribute significant delay and significant delay variation. The corresponding delays are both uniformly distributed, see Table 3.1. Thus, Equation (3.11) can be simplified to a convolution of a shifted dirac function representing the delays of all processing blocks except the two temporal sampling processes, and two uniform distributions, representing the two sampling processes. Note that the two uniform distributions are not necessarily identical, they depend on the temporal sampling frequencies chosen for camera and display.

It is straightforward to show that the convolution of two identical uniform distributions gives a triangle distribution, with a width equal to two times the width of the uniform distributions. When the uniform distributions provide different supports (widths), the result of their convolution will be an isosceles trapezoid. Imagine uniform distribution A with a support [0, 16] ms and uniform distribution B with a support of [0, 50] ms. When shifting the flipped version of A over B, the convolution result C will contain an ascending ramp for $t \in [0, 16]$ ms, and be constant for $t \in [16, 50]$ ms, just to end with a descending ramp in $t \in [50, 66]$ ms. Distribution C has the shape of an isosceles trapezoid with a support width equal to the sum of the support widths of A and B. The ramp widths equal the width of the distribution with smaller support, in this example distribution A.

Convolving distribution C with a dirac distribution representing a constant delay would shift the distribution by the constant delay to the right. From the resulting distribution D, the minimum or leftmost point of the support thus indicates the constant delay in a video communication solution.

The previous insights can be used to analyze the delay contributors, given a G2G delay distribution as in Figure 4.5 for a Huawei Nexus 6P. In Figure 4.5, the shifted isosceles trapezoid can be seen. The minimum G2G delay is equal to 63.11 ms, see also Table 4.1. This is the constant delay contributed by elements such as camera and display processing. The width of the distribution in Figure 4.5 is equal to $127.49 - 63.11 = 64.38$ ms. Additionally, the display of the Nexus 6P has a sampling rate of 60 Hz. These two facts can be used to determine the sampling rate of the video: the support of the display sampling delay distribution is $16.\overline{6}$ ms wide, see Equation (3.5). In addition, the widths of the ascending and descending ramps equals approximately 16 ms. Consequently, the support of the distribution of camera temporal sampling has to equal $64.38 - 16.\overline{6} \approx 48$ ms. Hence, the camera temporal sampling rate is approximately 20 Hz. This low temporal sampling rate of the live video preview of the camera application can be perceived when using the camera application.

In summary, the measurements are in accordance with the theoretical delay model (3.11). A more detailed comparison of measurements to the theoretical model, including parameters for all delay contributors, is given in Section 6.5.

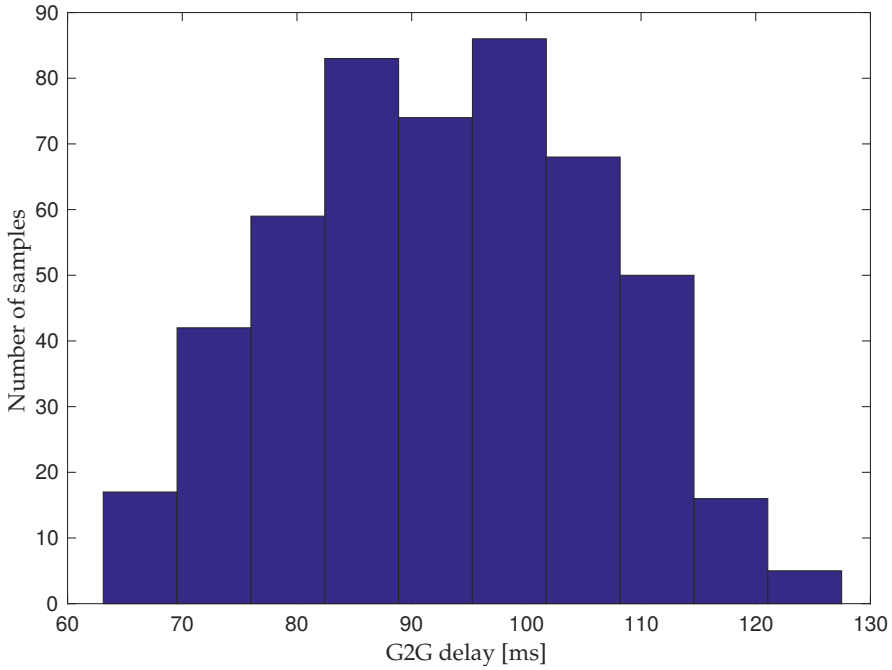


Figure 4.5: G2G delay distribution of 500 delay measurement samples of the camera application of a Nexus 6P smartphone (adapted from [6], © 2017 IEEE).

4.3 Glass-to-Algorithm and other Delay Measurements

Until now, the measurement system can only perform G2G delay measurements. However, we have analyzed the video communication pipeline (Figure 3.1) in great detail and want to confirm the theoretical analysis through practical measurements. Consequently, we need to measure delays between other points of the video communication pipeline. To do so, this thesis proposes a G2A delay measurement system in this section. The system can be used to measure G2A and other delays along the video communication chain.

4.3.1 Measurement Principle

For a G2A delay measurement, we still need a visual event taking place at time point t_0, as in Figure 4.1a. Consequently, we trigger an LED at t_0 in the camera's FoV. Next, the system shall determine when the visual event is first visible in an image after the color conversion after the decoder. Once the computer containing the decoder and color converter detects this event (at time point t_2 in Figure 4.1a), it sends a signal to the measurement system. Finally, the measurement system computes G2A delay

$$t_{\mathrm{G2A}} = t_2 - t_0 - t_{\mathrm{inh},2}, \tag{4.4}$$

as the time difference between the two points and subtracts the delay inherent to the G2A measurement system $t_{\mathrm{inh},2}$.

4.3.2 Hardware System Description

An Arduino Mega 2560 still controls the LED and determines t_{G2A}. For the required time point t_2, an electronic signal is sent from the computer processing the decoded image to the Arduino system. To realize this with lowest latency, the measurement system is connected to the receiver (computer) using an RS-232 serial port, if available. A USB connection can also be utilized, but introduces higher inherent latency because of the polling interval of at least 0.125 ms[6].

4.3.3 Signal Processing

In terms of signal processing, the G2A delay measurement system is identical to the G2G delay measurement system with the exception of the determination of the second time point t_2. The G2A delay measurement system only needs to record the time stamp when the computer signals that it detected the visual event in a frame. Therefore, the more complex part is detection of the visual event in the video.

Detecting the turning on of the LED in software is simplified by two measures in the proposed implementation: the lighting of the scene around the LED is constant and the position of the LED is in the top left corner of the video image. With these constraints simple pixel thresholding can be used, meaning that if the brightness of the top left pixel increases more than a predefined threshold, the LED is just turning on, the visual event is taking place (t_2). This information is then forwarded to the Arduino measurement system for the computation of t_{G2A}.

4.3.4 System Evaluation

The inherent delay $t_{\mathrm{inh},2}$ of the G2A measurement system equals the sum of the LED turn-on time, the pixel detection time, and the processing delay of the RS-232 serial port. These delays are small ($\ll 1$ ms) as they are, except the LED, low-level processes. Compared to the G2A values measured in Section 6.1.4, they are negligible.

Compared to the G2G delay measurement system, the advantage of non-intrusiveness from Table 2.2 is not given anymore since we need to adapt the video communication system. Now, the advantage is that the system can perform Glass-to-Anything (G2X) delay measurements. In G2X, G is the camera glass and X is any point in the video transmission pipeline at which the uncompressed image is available. Point X can, for example, be where the image from the camera is available in the connected computer for the first color conversion. This would define the delay between the visual event taking place in the real world and the corresponding recorded image being ready for color conversion and encoding, see Figure 3.1.

[6] USB Specification Revision 2.0, Table 9-13 on page 271

Chapter 5

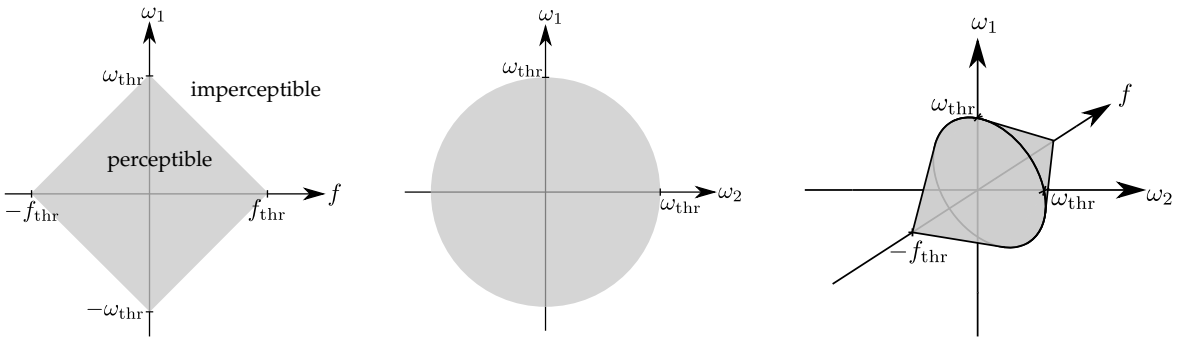
Perception of Temporal Sampling

5.1 Relevance to Glass-to-Glass Delay Reduction

Section 3.7 showed that increasing the temporal sampling frequencies of both the camera and the display offers great potential for G2G delay reduction. The algorithms proposed in Chapter 6 therefore utilize high sampling rates. To avoid an inflation of data that needs to be processed and transmitted, this thesis also proposes frame skipping algorithms. These algorithms reduce the video's frame rate while ensuring that visual events are immediately transmitted. In this context, uniform and irregular sampling become relevant: uniform sampling is a sampling process in which new data is sampled at constant time periods, meaning that the time period between two subsequent data samples is equal to the time period between any other pair of subsequent samples. Conventional cameras record videos using uniform sampling and yield images (data samples) with a time period of, for example, 40 ms sampling period between each pair of subsequent video images. Irregular sampling breaks with this constraint and allows arbitrary time periods between subsequent images. Skipping frames from a uniformly sampled video yields an irregularly sampled video with a variable update rate.

Humans can perceive the temporal sampling process and distinguish the video from a visual sequence that is continuous in time if the update rate of the video images is low enough. In consequence, uniformly as well as irregularly sampled videos have to maintain a minimum update rate to avoid perception of temporal sampling. This minimum update rate should be when humans are just not able to perceive that the video they are watching is temporally discrete, and not continuous. Employing higher update rates would require additional resources while not improving the perceived video quality. Thus, at the minimum update rate, the video requires the least processing power and transmission bandwidth with temporal sampling being imperceptible (the video is perceived smooth or fluid). The minimum update rate for frame skipping is investigated in this chapter, and used in Chapter 6 to render the process of frame skipping imperceptible to humans.

Some of the ideas and contributions presented in this chapter have been published in [3].



(a) Window of Visibility in one spatial and the temporal frequency domain, as in Watson [42].

(b) Window of Visibility in two spatial frequency dimensions, represented by ω_1 and ω_2, respectively. For all spatial directions, the same threshold ω_{thr} holds.

(c) Volume of Visibility in two spatial and the temporal frequency domain.

Figure 5.1: Depiction of the Volume of Visibility, consisting of two cones. Every signal inside gray areas or volumes is perceptible, any signal outside is not.

5.2 Theoretical Background

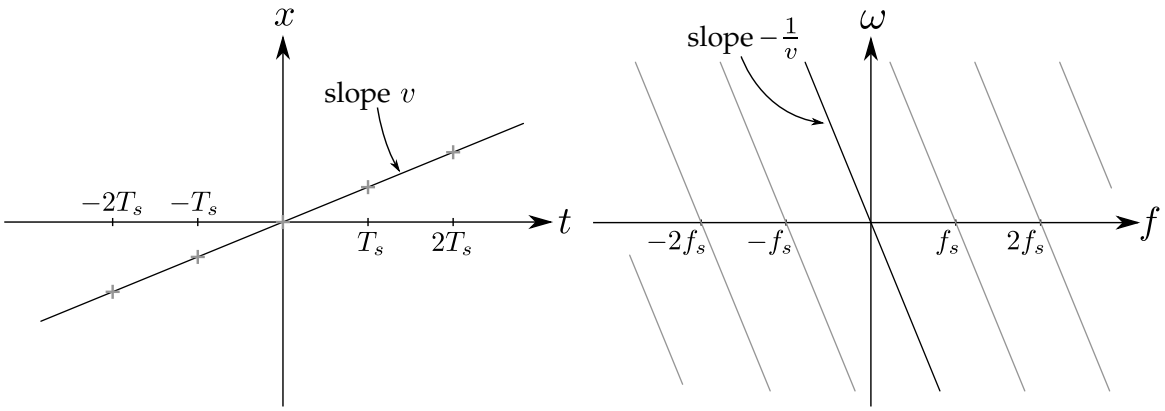
The theoretical background of this chapter is grounded on the work by Watson *et al.* [35] and Watson [42], briefly introduced in Section 2.1.4. This section provides an in-depth introduction of the relevant work by Watson *et al.* [35], [42] and introduces a further analysis created upon that work.

5.2.1 Window of Visibility

Watson *et al.* [35] defined the *Window of Visibility* (WoV), which was revised by Watson [42] based on perception data obtained by Robson [116]. The revised WoV is depicted as gray parallelogram in Figure 5.1a. The window shows the perception thresholds in one spatial frequency domain (ω [cycles/degree]) and the temporal frequency domain (f [Hz]). Visual information within the window is more or less visible to the user, everything outside is not. Thus, the WoV can be seen as a low-pass filter. The shape of the window indicates that the perception thresholds of temporal and spatial frequency are dependent [42], and hence, the WoV is not a separable filter.

For simplicity, the WoV covers only one spatial dimension, but can be extended in a straightforward manner to two spatial dimensions. A second spatial dimension is given in Figure 5.1b. The spatial frequency dimensions are named ω_1 and ω_2, respectively. In the plane spanned by the two spatial dimensions, the shape of the WoV is a disk because, as Section 5.4.4 shows, only motion magnitude, not motion direction is relevant for the perception limit. Together with the diamond-shaped WoVs in both spatiotemporal planes, the disk in the spatial plane yields the *Volume of Visibility* (VoV) in the three dimensional space. The volume consisting of two cones in two spatial and the temporal dimension is depicted in Figure 5.1c.

In paper [35], the human temporal flicker frequency threshold f_{thr} was found to be $f_{\text{thr},1} = 30\,\text{Hz}$ and $f_{thr,2} = 33\,\text{Hz}$ for two observers. Stationary temporal contrast fluctuations



(a) Motion of a point through a one dimensional-space, at constant velocity v. The gray pluses denote sampling instances.

(b) Spectrum of the moving point, assuming that the point exhibits infinitely high spatial frequency. Grey lines are replications at integer multiples of $f_s = 1/T_s$, caused by sampling.

Figure 5.2: Derivation of the frequency spectrum of a temporally sampled point moving with constant velocity through a one-dimensional space.

with a frequency higher than f_{thr} can not be seen. For example, the 50 Hz flicker of ceiling lights is imperceptible to most humans. Analogously, ω_{thr} represents the human spatial sensitivity threshold, where the cycles per degree refer to the point of view of the human. In their experiments, Watson et al. [35] find that the thresholds are at $\omega_{thr,1} = 6$ [cycles/degree] and $\omega_{thr,2} = 13$ [cycles/degree] for two observers. These values are considerably lower than the 50 cycles per degree suggested by Yanoff *et al.* [32]. Also, with such a low spatial resolution of the human eye, display technology would have long exceeded our spatial resolution limit, see Table 2.1. The authors in [35] presume that the low contrast of the display they utilized causes the rather low threshold values seen in their experiments. Experiments with more test subjects would lead to more generally valid numbers, the preceding values are only examples.

5.2.2 Critical Sampling Frequency Based on the Window of Visibility

Let us in the following briefly revisit the derivation of the perception and sampling limits imposed by the WoV, originally derived by Watson *et al.* [35], [42]. As a representative for spatial signals, we assume a point which is moving through a one-dimensional space at constant, non-zero velocity v. The point, which is in the beginning of the following derivations assumed to exhibit infinitely high spatial frequency, is shown in the *t*-*x* plane in Figure 5.2a. In that plane, the motion equation is

$$p(x,t) = \delta(x - vt), \tag{5.1}$$

where $\delta(\cdot)$ is the Dirac function, which equals one if its argument equals zero, and equals zero otherwise. The Fourier transform of $p(x,t)$ both in time and in space can be derived as

$$P(\omega, f) = \mathcal{F}_t\{\mathcal{F}_x\{l(x,t)\}\} \tag{5.2}$$
$$= \mathcal{F}_t\{\mathcal{F}_x\{\delta(x - vt)\}\} \tag{5.3}$$
$$= \mathcal{F}_t\{\exp(-jvt\omega)\} \tag{5.4}$$
$$= 2\pi\delta(f + \omega v) \tag{5.5}$$
$$= 2\pi\delta(\omega + \frac{f}{v}), \tag{5.6}$$

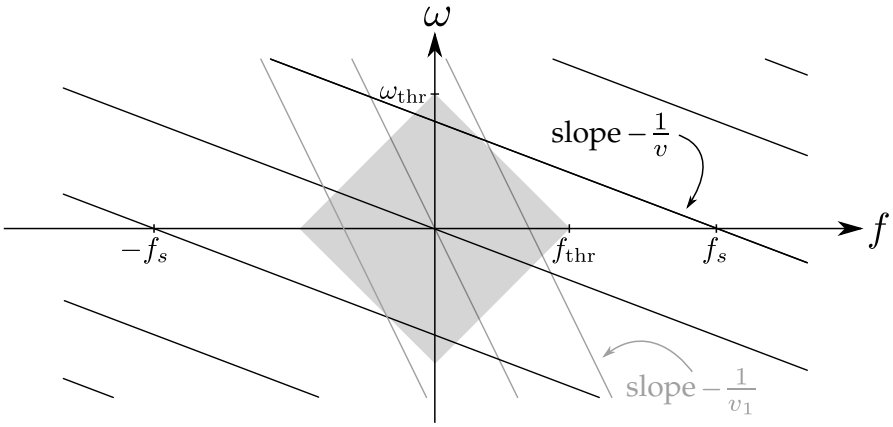
which uses the Fourier Shift theorem [117] to transform line (5.3) to line (5.4), and the Fourier pair $\mathcal{F}\{\exp(j\omega_0 t)\} = 2\pi\delta(f - \omega_0)$ [117] for the transformation of line (5.4) to line (5.5). Accordingly, Equation (5.6) describes a line with slope $-1/v$ in the f-ω plane, as shown by the black line in Figure 5.2b. Next, the point moving through space is temporally sampled with sampling frequency $f_s = 1/T_s$, as shown by the gray pluses in Figure 5.2a. Sampling in the temporal domain causes periodic replications of the original spectrum $P(\omega, f)$ at integer multiples of f_s, see the gray lines in Figure 5.2b.

Figures 5.1a and 5.2b are merged into Figure 5.3a for the following explanations. For purposes of this demonstration, the slope $-1/v$ is in Figure 5.3a smaller compared with the slope in Figure 5.2b. In Figure 5.3a, the slope $-1/v$ is chosen such that the replications of $P(\omega, f)$ intersect with the WoV. These intersections mean that the replications of the original spectrum pass through the pass-band of the low-pass filter, which is called aliasing. These aliasing artifacts are what humans perceive as jitter or jerkiness in a temporally sampled video. Therefore, if the frequency spectra of the replications lie outside the WoV, humans will not be able to distinguish the time-sampled representation of the moving point from the continuous representation.

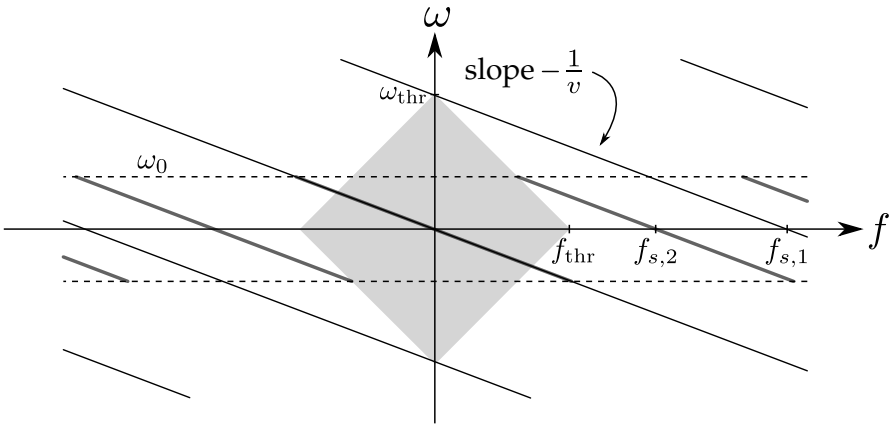
For the overlap of the replications with the WoV, two cases should be distinguished, see Fig. 5.3a: first, if the slope $1/v < \omega_{\text{thr}}/f_{\text{thr}}$, the replications overlap with the top and bottom of the WoV, at high spatial and low temporal frequencies. For such signals with high velocity (small $1/v$), the perceived effect is typically called "flicker" [42]. Second, for slopes $1/v_1 > \omega_{\text{thr}}/f_{\text{thr}}$, see the gray line in Fig. 5.3a, the replications overlap at the sides of the WoV, where low spatial frequency and high temporal frequency is present. This aliasing case is perceived as "multiple images" [42]. For the latter case, it is straightforward to determine the sampling frequency at which aliasing is avoided: if the replications are at positions greater than f_{thr}, no replications will overlap with the WoV. Thus, in the case of steep slopes, a sampling frequency greater than f_{thr} suffices to avoid aliasing. This is why in the following, we focus on the more interesting case in which the slope of the point spectrum $1/v$ is smaller than the slope $\omega_{\text{thr}}/f_{\text{thr}}$ of the WoV boundaries.

The authors in [42] used the WoV to derive an approximation of the minimum temporal sampling frequency, named critical sampling frequency f_c, which alleviates aliasing to a level below human perception thresholds. The relevant parts of the derivation [35], [42] are detailed in the following.

There will be imperceptible aliasing if the replicated spectra just touch the top and bottom corners of the WoV, as illustrated in Figure 5.3b. To satisfy this, the temporal sampling frequency $f_{s,1}$ has to be equal to or greater than the critical sampling frequency



(a) The moving point exhibits infinite spatial frequency, the spectrum replications overlap with the WoV for two different slopes.



(b) No periodic replications overlap with the WoV. Either because $f_{s,i}$ is chosen large enough (thin lines), or due to the limited spatial frequency ω_0 of the moving point (thick lines).

Figure 5.3: The Window of Visibility is depicted as gray diamond shape as in Figure 5.1a. The diagonal line (slope $-1/v$) passing through the origin represents the spectrum of the point moving at constant velocity v. Due to temporal sampling, the original spectrum is replicated at integer multiples of the sampling frequency f_s, as in Figure 5.2b.

$$f_{s,1} \geq f_c = v \cdot \omega_{\text{thr}}, \tag{5.7}$$

as can be derived from Figure 5.3b using the definition of a line slope. As a result of the point symmetry of the WoV and the spectrum of the moving point, this condition also satisfies the first periodic replication at $-f_s$. If the maximum spatial frequency ω_0 of the point (or a general object) is lower than the spatial perception threshold ω_{thr} (see Figure 5.3b), the condition for the sampling frequency $f_{s,2}$ accordingly changes to

$$f_{s,2} \geq f_c = f_{\text{thr}} - \frac{f_{\text{thr}}}{\omega_{\text{thr}}} \cdot \omega_0 + v \cdot \omega_0. \tag{5.8}$$

Merging conditions (5.7) and (5.8) yields the general condition

$$f_s \geq f_c = f_{\text{thr}} - \frac{f_{\text{thr}}}{\omega_{\text{thr}}} \cdot \min(\omega_{\text{thr}}, \omega_0) + v \cdot \omega_0, \quad \forall v \geq \frac{f_{\text{thr}}}{\omega_{\text{thr}}} \tag{5.9}$$

as the lower bound for the temporal sampling frequency. The consequential next step is to limit the temporal frequency to, for example, f_0. This is done in [35], but does not give further insight in our context. The analysis continues to show how the theoretically derived Equation (5.9) can be interpreted and applied to real video sequences.

Equation (5.9) leaves us with two key takeaways: first, objects with high speed v, yielding a flatter line in Figure 5.3, require higher temporal sampling frequencies f_s to attenuate aliasing artifacts, making jitter imperceptible. Second, at a given speed v and all spatial frequencies below the perception threshold ω_{thr}, objects with high maximum spatial frequencies (sharp edges) require higher temporal sampling frequencies than objects with lower maximum spatial frequencies (blurry/smooth edges).

Watson *et al.* [35] did not investigate the influence of the exposure time of a camera, and while Watson [42] provides an analysis of exposure, the analysis does not directly yield the critical sampling frequency as a function of camera exposure time. Camera exposure time influences the maximum spatial frequency, as described in the following. As shown in Section 2.3.1.2, during the exposure time of a (video) camera, photons keep falling onto the image sensor. If during the exposure time, the rate at which photons fall onto a pixel is constant, the output from this pixel will perfectly represent what can actually be seen from the pixel's perspective. If the rate of photons changes during exposure, for instance if the camera or object is moving, the pixel's output will represent the *average* light intensity recorded during the exposure period. This is why quickly moving objects can appear blurred in photos and videos, exhibiting a limited maximum spatial frequency. The following section formalizes how the integration during exposure affects the spatial frequency components of visual signals.

5.2.3 Analysis of Spatial Frequency Distribution in Real Video Sequences

An object moving through the field of view of a camera with finite exposure time will become blurred, its spatial patterns low-pass filtered. The following paragraphs show that blurring caused by exposure time integration will not yield an upper bound in terms of spatial frequency, if the original signal's spatial frequency was not limited. In addition, this section provides a function for the distribution of the spatial frequency components in dependence of object velocity and exposure time.

To illustrate exposure integration, imagine the one-dimensional spatial signal $g(x) = u(x - x_0)$ depicted in the top left graph in Figure 5.4. The following explanations extend naturally to more spatial dimensions. Function $u(\cdot)$ denotes the Heaviside step function or unit step function, which is equal to zero for negative arguments, and equal to one for non-negative arguments. For simplicity, the image signal is normalized to range $[0, 1]$ (where 0 and 1 correspond to black and white, respectively). Thus, the signal represents a perfectly sharp black-to-white edge. Its Fourier transform [117]

$$U(\omega) = \mathcal{F}\{u(x)\} = \pi\delta(\omega) + \frac{1}{j\omega} \tag{5.10}$$

contains the Dirac function $\delta(\cdot)$, which equals one if its argument equals zero, and equals

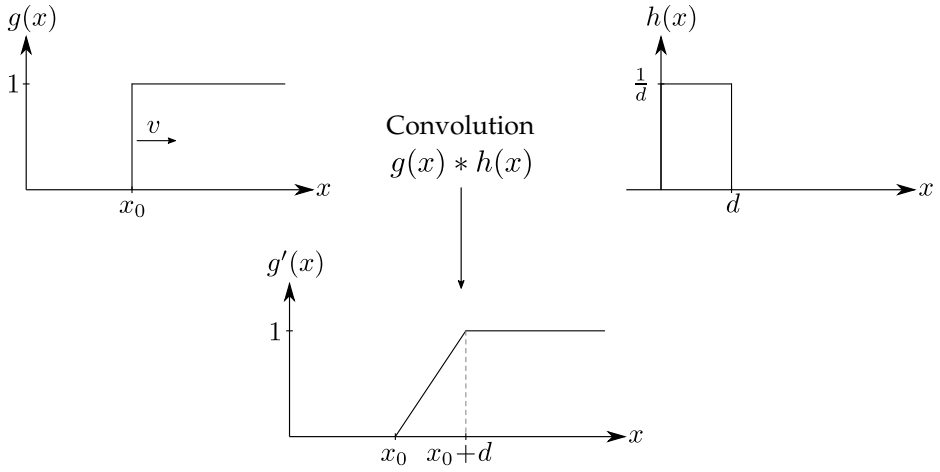


Figure 5.4: The input signal $g(x)$ moves right with velocity v. Convolution with camera filter $h(x)$ creates the blurred output signal $g'(x)$. Camera filter $h(x)$ actually is a temporal filter, but in this example represented by its spatial equivalent for simplicity. Filter width equals exposure time t_{exp} multiplied by speed v, see Equation (5.12). Adapted from [3], © 2018 IEEE.

zero otherwise. The Fourier transform of $u(\cdot)$ has a magnitude $|U(\omega)| \sim \frac{1}{\omega}$ when ignoring the $\delta(\cdot)$ function. Ignoring the Dirac function is reasonable because $\delta(\cdot)$ represents the frequency coefficient for $\omega = 0$, which is not relevant to the maximum frequency we are looking for.

The spatial signal $g(x)$ is a shifted version of $u(x)$, which corresponds to a multiplication of $U(\omega)$ with the complex exponential $e^{-j\omega x_0}$ [117]. This process does not change the magnitude of the original frequency spectrum of $u(x)$:

$$\begin{aligned} |G(\omega)| &= |\mathcal{F}\{g(x)\}| = |\mathcal{F}\{u(x)\}| \cdot |e^{-j\omega x_0}| \\ &= |\mathcal{F}\{u(x)\}| = |U(\omega)| \sim \frac{1}{\omega}, \end{aligned} \tag{5.11}$$

so it is still proportional to the reciprocal of ω. Signal $g(x)$ moves to the right at velocity v, as indicated in the top left graph in Figure 5.4. This process is recorded by a camera with exposure time t_{exp}. Consequently, during the exposure time, $g(x)$ moves

$$d = v \cdot t_{\text{exp}} \tag{5.12}$$

to the right. The resulting, blurred signal $g'(x)$ is depicted in the bottom graph of Figure 5.4. Note that at each location x, the signal is integrated over time. At locations $x < x_0$, input $g(x)$ will equal to zero at all times during exposure, and at locations $x > x_0 + d$, the input will equal one at all times during exposure. In these locations, $g(x) = g'(x)$ holds. At $x = x_0$, the signal on the (one-dimensional) camera sensor is for all times during exposure equal to zero, except for an infinitesimally small time at the very beginning. Thus, the resulting value in the blurred image in this location equals zero. At $x = x_0 + 0.1 \cdot d$, the signal is equal to one for ten percent of the exposure time, zero for the remaining 90 percent. Therefore, the resulting image value equals 0.1. The linear ascend spans the entire exposure time area, yielding the ramp that can be seen in the bottom graph of Figure 5.4. This linear dependency holds for every camera exposure process of an object moving at constant velocity.

The previous insight allows us to formalize how temporal integration during camera exposure affects the spatial frequency of an input signal $g(x)$ moving at constant velocity v. To simplify the following proof, temporal integration of $g(x)$ during exposure time is replaced by the convolution $g(x) * h(x)$. The convolution needs to yield the same $g'(x)$ as temporal integration. To this end, we design the dimensions of $h(x)$ relative to movement speed v and exposure time t_{exp} using Equation (5.12):

$$h(x) = \frac{1}{d}\left(u(x) - u(x-d)\right) \tag{5.13}$$

The convolution $g(x) * h(x)$ will always lead to the same $g'(x)$ as temporal integration with exposure time t_{exp} of a signal $g(x)$ moving at constant velocity v. Filter $h(x)$ is shown in the top right graph in Figure 5.4. The magnitude of its Fourier transform

$$|H(\omega)| = |\mathcal{F}\{h(x)\}| = \frac{1}{d}|\mathcal{F}\{u(x)\}| \cdot |1 - e^{-j\omega d}| \tag{5.14}$$

equals the magnitude of $U(\omega)$ multiplied with the inverse of d and the magnitude of $1 - e^{-j\omega d}$. Let us derive

$$\begin{aligned} c(\omega, d) &= |1 - e^{-j\omega d}| && (5.15)\\ &= |1 - \cos(\omega d) + j \cdot \sin(\omega d)| && (5.16)\\ &= \sqrt{(1 - \cos(\omega d))^2 + (\sin(\omega d))^2} && (5.17)\\ &= \sqrt{1 - 2\cos(\omega d) + (\cos(\omega d))^2 + (\sin(\omega d))^2} && (5.18)\\ &= \sqrt{2 - 2\cos(\omega d)} \in [0, 2]. && (5.19) \end{aligned}$$

It is important to notice that $c(\omega, d)$ is a periodic function in the domain $c(\omega, d) \in [0, 2]$. $c(\omega, d)$ will therefore not play a significant role for the following proportionality considerations. Consequently, the magnitude from Equation (5.14)

$$|H(\omega)| = \frac{c(\omega, d)}{d}|U(\omega)| \sim \frac{c(\omega, d)}{\omega d} \tag{5.20}$$

approximates $\frac{1}{d}|U(\omega)|$ and, more importantly, is approximately proportional to the reciprocal of the product ωd because $c(\omega, d) \in [0, 2]$. Thus, the filter width d, which is a function of t_{exp} and v, see Equation (5.12), defines how quickly the frequency coefficients of $|H(\omega)|$ will converge to zero. Finally, we retrieve the magnitude spectrum $G'(\omega)$ of the output signal $g'(x)$:

$$|G'(\omega)| = |\mathcal{F}\{g'(x)\}| = |\mathcal{F}\{g(x) * h(x)\}| \tag{5.21}$$
$$= |\mathcal{F}\{g(x)\}| \cdot |\mathcal{F}\{h(x)\}| \tag{5.22}$$
$$= |G(\omega)| \cdot |H(\omega)| \tag{5.23}$$
$$= |U(\omega)|^2 \frac{c(\omega, d)}{d} \quad \left(\sim \frac{c(\omega, d)}{\omega^2 d}\right) \tag{5.24}$$
$$= \left|\pi\delta(\omega) + \frac{1}{j\omega}\right|^2 \frac{\sqrt{2 - 2\cos(\omega d)}}{d} \tag{5.25}$$
$$= \frac{1}{\omega^2} \frac{\sqrt{2 - 2\cos(\omega t_{\text{exp}} v)}}{t_{\text{exp}} v} \quad \forall \omega \in \mathbb{R} \setminus \{0\} \tag{5.26}$$

The above calculation uses the convolution theorem [117] to transform line (5.21) to line (5.22), and Equation (5.11) and (5.20) to convert line (5.23) to line (5.24). In line (5.24), we observe that the magnitude spectrum $|G'(\omega)|$ of output signal $g'(x)$ decreases faster than the input signal at higher frequencies ω because of its approximate proportionality to the reciprocal of $\omega^2 d = \omega^2 t_{\text{exp}} v$, which gives it the blurred appearance. Still, the magnitude spectrum does not generally vanish at an upper limit ω_0, see Figure 5.5. Figure 5.5 shows the magnitude spectrum for two products d of t_{exp} and v and the upper limit for $d = 10$. The upper limit assumes $\cos(\omega t_{\text{exp}} v) = 1 \ \forall \omega$. The fact that all graphs approximate zero, but do not consistently equal zero after any frequency ω is why result (5.9) cannot be used directly on real signals and further investigations need to be done.

Equation (5.26) expresses the spectrum magnitude of output $g'(x)$ as a function of exposure time t_{exp} and speed v to give a better intuition of how these two variables influence the spatial frequency components. As we know from Equation (5.19), the numerator in Equation (5.26) can take values in [0, 2]. Therefore, the spectrum magnitude of $g'(x)$ is approximately proportional to, among others, $1/t_{\text{exp}}$. Thus, a longer exposure time t_{exp} will, for example, lead to faster decay of frequency components of the magnitude spectrum of $g'(x)$, allowing a smaller critical sampling frequency f_c.

The discussion in the previous paragraphs assumed infinitely high spatial resolution. In real systems, intensity signals are spatially sampled by the pixel matrix in the camera sensor. This leads to a staircase function instead of a ramp function for $g'(x)$ in Fig. 5.4. The staircase signal would be composed of unit step functions, yielding a spectrum magnitude that is vanishing slower than in Equation (5.24), hence causing a larger critical sampling frequency f_c. However, at normal viewing conditions, state-of-the-art displays have pixel densities of usually 35 to 150 pixels per degree (an overview is provided in Table 2.1). This is close to and often greater than the angular resolution of the human eye, approximately 0.02 degrees with optimal eyesight [32], corresponding to 50 pixels per degree. Hence, as done in [35], we disregard the negligible spatial discretization effect.

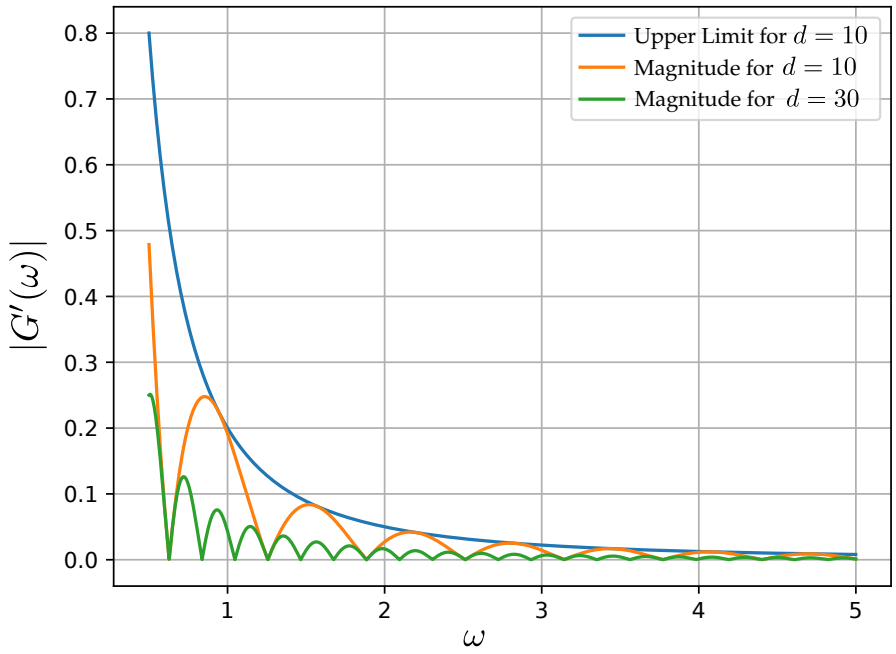


Figure 5.5: Magnitude spectrum $|G'(\omega)|$ of the original input signal $g'(x)$ for two parameter selections $d = t_{\text{exp}}v$. The upper limit assumes that the cosine term of Equation (5.26) constantly equals one. All spectra only approximate zero, but do not equal zero beyond a spatial frequency ω. Note that the results are plotted only for $\omega \geq 0.5$ to be able to highlight the spectrum characteristics.

5.2.4 Conclusions for the Perception of Temporal Sampling in Real Video Sequences

The derivations in Section 5.2.3 served three purposes: first, they show that there is no upper bound on the spatial spectrum of an output image created with a conventional camera. Consequently, the WoV is not directly applicable to real video sequences, it only provides an intuition. Second, the derivations introduce the reader to how the camera exposure process filters an input scene containing motion. Third, they provide the two central variables of interest for the experiments in this chapter: camera exposure time t_{exp} and object velocity v. They define the width d (Equation (5.12)) of the camera filter $h(x)$, the steepness of the ramp of the output image and how quickly the spatial frequency components of the output image will converge to zero. Furthermore, the steepness of the spatiotemporal spectrum is a function of v, see Figure 5.3. Higher velocity v decreases the steepness of the spectrum, leading to a higher f_c, see Equation (5.9). On the other hand, a higher speed v increases the filter width d (Equation (5.12)), which causes high-frequency components to decay more quickly (Equation (5.24)), accordingly reducing f_c. Section 5.4.1 will investigate which of these two opposing effects of v prevails.

Overall, these insights lead us to four hypotheses which Section 5.4 is going to verify through experiments:

1. Higher speed v changes f_c
2. Longer exposure time t_{exp} decreases f_c
3. Higher speed v increases the influence of t_{exp} on f_c
4. Longer exposure time changes the influence of v on f_c

5.3 Experimental Setup

The following sections will conduct a series of experiments to find the previously mentioned threshold for perceiving jitter in time-sampled motion sequences. This section describes the presented sequences, sequence creation, sequence playback, the participants and how participants performed the experiment.

Sequence creation and sequence playback were separated because real-time creation was not possible (see Section 5.3.3).

5.3.1 Video Sequences

As it is common practice in psychometrics, an object moving in one dimension is presented to the users [35], [37], [41]. The sequences are shown on a special computer monitor, which is described in more detail in Section 5.3.4. The shown sequences, for an example see Figure 5.6, contain a vertical white bar that spans the entire display height, and one fifth of the display width. The white bar moves from left to right over black background at constant speed. This setting gives us maximum contrast and clearly perceivable intensity edges. When general video content is presented to users, lower contrasts might be present, which would require lower frame rates. Therefore, the obtained results will serve as an upper bound for the critical sampling frequency f_c.

The bar starts at the left side of the monitor, and when it moves out of the display on the right side, it moves into the display again on the left side. The moving bar is computed with a simulation rate of 20 kHz. This allows us to simulate the relevant exposure times for the camera and the resulting motion blur. The simulation of exposure time is described in Section 5.3.3. The blurring caused by exposure time can be seen at the edges of the white bar in Figure 5.6.

5.3.2 Sequence Parameters

For each set of sequences, we set a constant speed v for the bar, and a constant exposure time t_{exp} for the camera. Each set contains sequences with frame rates f_{seq} ranging from 10 Hz to 90 Hz in steps of 2 Hz. For referencing, Table 5.1 labels the sets. In this table, the speed and exposure time increase from bottom to top, and from left to right, respectively. Hence, the camera filter width d (as defined in Equation (5.12)) decreases from top right to bottom left and consequently, the intensity of motion blur decreases from the top right to the bottom left. Only two subjects (first and second author of paper [3]) have conducted tests on all sets. The nine sets A to I have been used in experiments for all ten test subjects. This sub-sampling

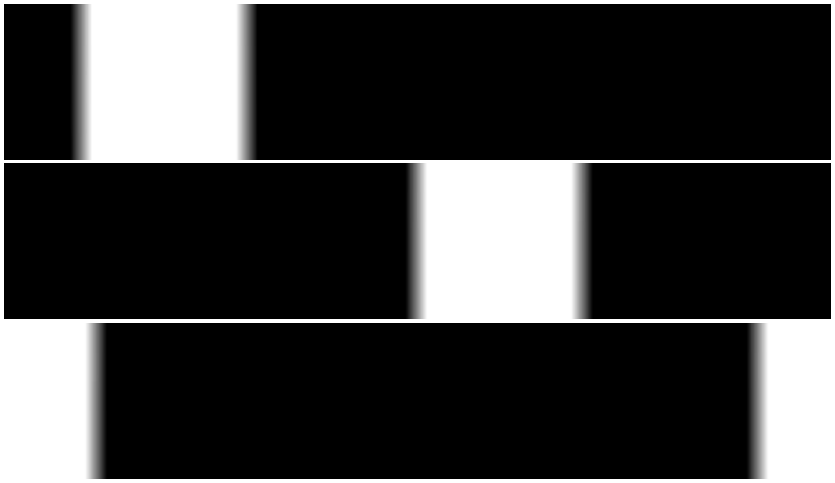


Figure 5.6: Sequence C (see Table 5.1) at three time instances during playback (from top to bottom): first, the bar is starting at the left side of the monitor, second after moving half way to the right, and third when the bar is crossing the right display border to reappear at the left border. In this figure, each image is shown at 1/3 vertical height (adapted from [3], © 2018 IEEE).

has been performed to reduce the duration of the experiment for the test subjects. For a detailed discussion of test length and difficulty, see Section 5.4.3. For all test subjects, the highest speed was chosen to be 48.3 deg/s (i.e., sequence sets A, B and C). 72.5 deg/s were not chosen, as at such a high speed, the bar is passing the display width so quickly that it is extremely exhausting for test subjects to register the presence or absence of jitter.

The parameter limits were chosen based on the following considerations: in conventional modern video applications, videos are recorded at frame (image) rates between 30 Hz and 60 Hz. This yields sampling periods between $33.\overline{3}$ ms and $16.\overline{6}$ ms, which serve as an upper bound for the exposure time because exposure of one image in a video may not take more time than the corresponding sampling period. We are in addition particularly interested in short exposure times of high frame rate videos with around 200 Hz of frame rate, as used in low delay video communication such as [4]. Therefore, the exposure range is set to [5.0, 25.0] ms to cover current and future applications.

The upper speed limit is decided by what participants were able to judge with reasonable effort. It was observed that higher speeds are difficult to follow given that the display covers only 46.4 degrees of the subject's field of view (FoV). The lower speed bound is dictated by the spatial resolution of the utilized computer display (details in Section 5.3.4), which provides 1920 pixels in horizontal direction. Early experiments found that at speeds below 6.0 deg/s, spatial discretization effects turned out to be perceivable, which distorted the results of the experiments. In summary, the subjects and discrete spatial resolution of the display set the upper and lower speed bounds, respectively, and this yields a speed range of [6.0, 72.5] deg/s.

The first two subjects performed the experiment for 7x5 pairs of speed and exposure times, and the remaining subjects for 3x3 pairs. It was observed that the sub-sampled pairs (speed, exposure) for all subjects are sufficient to capture all effects on the (speed, exposure) plane. It should be noted that the angular speed values in Table 5.1 are determined by the

v \ t_{exp}	5 ms	10 ms	15 ms	20 ms	25 ms
72.5 deg/s	J	K	L	M	N
48.3 deg/s	**A**	O	**B**	P	**C**
36.3 deg/s	Q	R	S	T	U
24.2 deg/s	**D**	V	**E**	W	**F**
18.1 deg/s	X	Y	Z	a	b
12.1 deg/s	c	d	e	f	g
6.0 deg/s	**G**	h	**H**	i	**I**

Table 5.1: Labels for the used sequence sets with various speeds v [degree/second] and exposure times t_{exp} [milliseconds]. The first and second author of paper [3] performed psychophysical tests on all sets, the remaining subjects only on sequence sets with bold letters and gray background(adapted from [3]).

chosen bar speed and the angle covered by the display in the FoV of the subjects (i.e., 46.4 degrees). During creation of the sequences, the bar speed was defined relative to the pixels on the display and values (for instance 0.025, 0.1 or 0.2 pixels per simulation step) were chosen to minimize spatial discretization effects at the expense of obtaining aesthetically less appealing angle speed values.

5.3.3 Sequence Creation: Simulation of Video Recording

A Python script creates the sequences using OpenCV [118], encodes the resulting video in H.264 [47] and writes it to a file for later presentation to the users. The script simulates the movement of the vertical bar at speed v in discrete-time steps at $f_{\text{sim}} = 20\,\text{kHz}$. This means that the script creates an image showing the bar at its current position 20,000 times per second in simulation time, and then applies the camera exposure time filter to create the output video sequence with a given frame rate, object speed, and exposure time. The simulation frequency was chosen considering that it provides a sufficiently small simulation complexity (all sequence sets could be created in approximately three days) and imperceptibly small time discretization effects. At $f_{\text{sim}} = 20\,\text{kHz}$, the temporal offset of capturing an event at an exact simulation time is at most $25\,\mu\text{s}$, which proved to be small enough for not being perceived by the test subjects. At lower simulation frequencies, time discretization effects become visible in some sequences.

The discretization effect can influence the resulting sequence in two perceivable ways: first, if temporal or spatial sampling frequencies are lower than human perceptual thresholds, test subjects will directly perceive temporal or spatial sampling. Second, imagine a bar that moves to the right 1.01 pixels per simulation step. After spatial discretization, its position will be incremented by one pixel, except for every 100th step, at which the position is incremented by two pixels. In this case, test subjects will be exposed to flicker with a

frequency which is 100 times smaller than the simulation frequency. The analogous insight holds for the discretization of time.

In the temporal domain, this discretization effect is attenuated below the human perceptual threshold by choosing a large enough simulation frequency of 20 kHz. However, in spatial domain, we are limited by the resolution of the used display panel. This proved to be insufficient, which is the reason why linear interpolation at the edges of the bar was added: imagine that the left edge of the bar should be drawn at pixel position 2.4 (between pixels 2 and 3). Without interpolation, pixel 1 would be black (value 0), and pixels 2 and 3 would be white (value 255). With linear interpolation, pixels 1 and 3 retain their value, but the value of pixel 2 equals to $(1 - 0.4) \cdot 255 = 153$, where the term $(1 - 0.4)$ generalizes to "One minus distance to the closest integer pixel position". The use of linear interpolation reduced spatial discretization below human perception thresholds.

Linear interpolation is not generally applicable. Fundamentally, the degree of interpolation of the pixel value at a pixel location has to equal the degree of the equation for object location. The location $x(t)$ of a moving object with respect to time t can in many cases be described by polynomials. For example, the one-dimensional location $x(t)$ of an object moving with constant velocity v can be expressed by the equation $x(t) = v \cdot t$. Assume that we know an object's locations $x(t_1)$ and $x(t_2)$ at times $t_2 > t_1$, and would like to compute the object's location at time $t_3 \in [t_1, t_2]$ without knowing the speed of the object. We only know that the object is moving at constant velocity. The resulting equation $x(t_3) = t_3 \cdot (x(t_2) - x(t_1))/(t_2 - t_1)$ implicitly computes object speed $v = (x(t_1) - x(t_2))/(t_1 - t_2)$ by interpolating the object locations in a linear manner. This result can be applied to pixel value interpolation: in the case of constant linear velocity, the object is moving through the pixel matrix at constant speed, therefore we may interpolate the pixels' values linearly. The corresponding insight holds for higher-order object location equations: linear acceleration for example requires quadratic interpolation of the pixel value. For object location functions that cannot be expressed through polynomials, (approximation through) splines would be a candidate for interpolation. The experiment could be done without interpolation in the highly improbable case in which at each simulation step (simulation frequency $f_{\text{sim}} = 20\,\text{kHz}$), both edges of the white bar are at integer pixel locations. In our case, the object (white bar) is moving at constant velocity, and thus linear interpolation is the suitable candidate.

Finally, the script needs to simulate motion blur caused by the exposure time of the camera and the movement of the bar. This is done in a rather simple way: the script always keeps the bar images from the past $k = f_{\text{sim}} \cdot t_{\text{exp}}$ simulation step images in a queue, see Figure 5.7. In order to create an output image of the camera, the script computes the average image of the past k images, as shown in Figure 5.7. Computing the average of the past k simulation step images equals convolving the input image with the camera filter $h(x)$ as in Figure 5.4. In each simulation step, the oldest simulation step image is popped from the queue, and a newly computed image is pushed onto the queue.

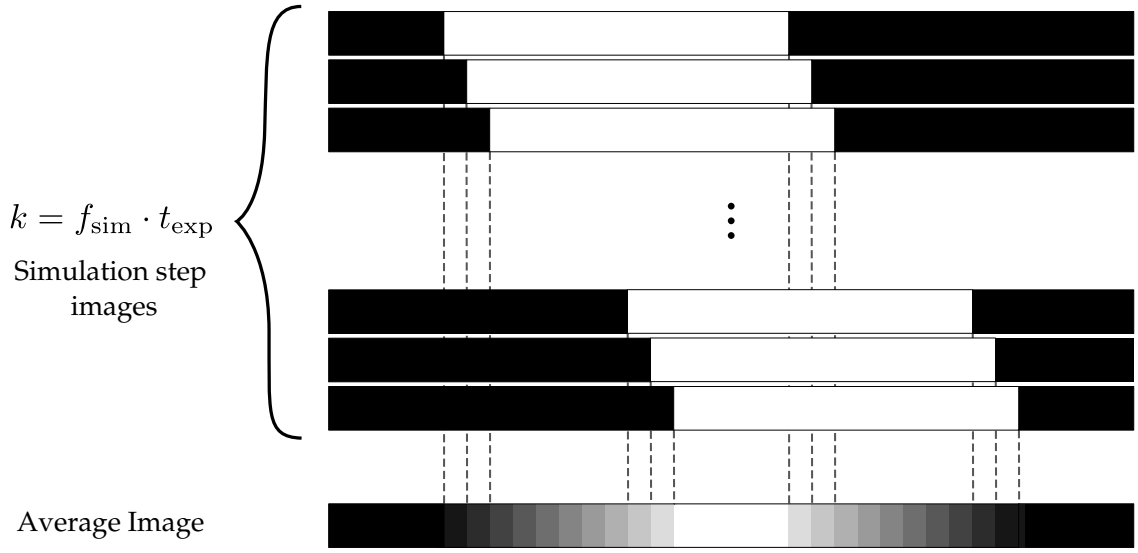


Figure 5.7: Simulation of the blurring caused by the exposure time. The simulation script keeps the past k images in which the white bar is at progressing positions. The output image is computed as the average of the k images. The color gradient representing the motion blur can clearly be seen in the output image. For simplicity, the plot does not show the pixel value interpolation of the simulation step images, they contain perfectly sharp black-white transitions. This example uses a small $k = 10$ to highlight the process of averaging. The actual implementation uses large $k > 100$, such that the color gradient appears smooth, see Figure 5.6.

5.3.4 Video Playback with Highly Precise Timing

This section discusses the sequence player, the display, and the physical experimental setup. The player should be able to play the high-resolution sequences from the Python script (Section 5.3.3) with precise frame timing. No available video player could meet this requirement, so a custom player was created. It uses libav[1] for decoding and SDL[2] for displaying. The decoder can decode any frame from any sequence within 1.5 ms, which is much smaller than the minimum frame period (i.e., 11.1 ms) we have in the used video sequences.

In the following, we assume that we want to display a video at a frame rate f_s with frame period $T_s = 1/f_s$. For frame timing, the *wait_until*() function of the *chrono* C++ library was used. Just before showing a frame/image on the screen, we record the current time in time stamp $t_0 = t$. Next, we update the screen content and perform all computationally intensive and time varying tasks such as file reading, decoding, color space conversion and copying to the GPU. Then we wait until $t_0 + T_s$, set the new time stamp to the current time $t_0 = t$ and update the screen content with the newly decoded image. This process is repeated until there is no frame left in the video file.

Because of processing overhead from setting the time stamp, the actual frame rate $\widetilde{f}_s$ of the player was lower than the target frame rate f_s. The difference $\Delta f_s = \widetilde{f}_s - f_s$ depends on the target frame rate f_s and is shown for $f_s \in [20, 120]$ Hz in Figure 5.8. The difference

[1] https://www.libav.org/, last visited 15.10.2018
[2] https://www.libsdl.org/, last visited 15.10.2018

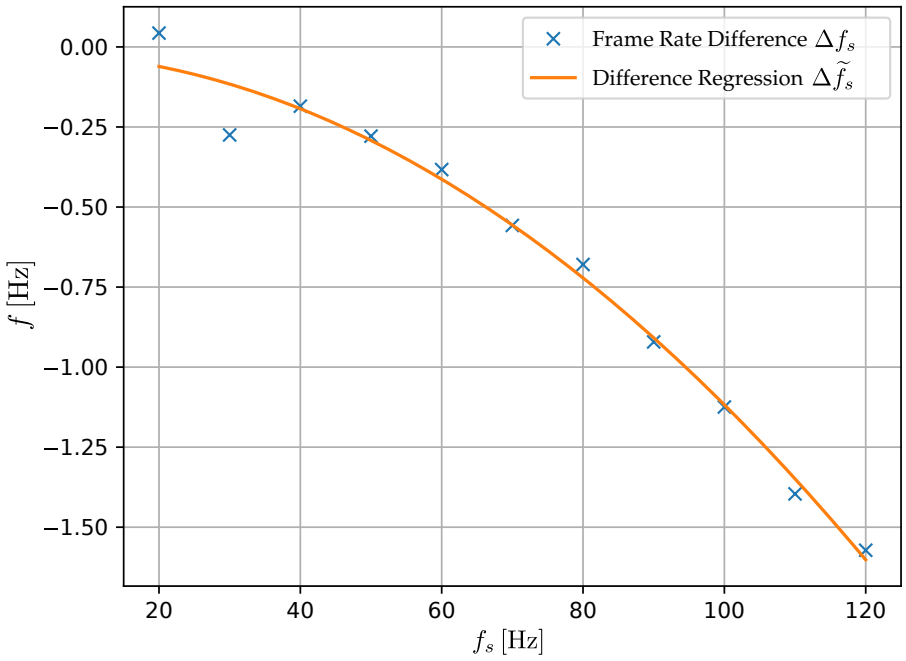


Figure 5.8: Frame rate difference Δf_s between the actual frame rate and target frame rate. The regression (5.27) of the differences closely approximates the frame rate difference, except for the outliers at low frame rates.

increases with increasing frame rates because at higher frame rates, frame periods become shorter, and consequently the constant contribution of the processing overhead becomes relatively larger. In Figure 5.8, we see that the difference Δf_s can be approximated using a quadratic regression. With the regression

$$\Delta \widetilde{f}_s(f_s) = -1.10 \cdot 10^{-4} \cdot f_s^2 - 0.0168, \tag{5.27}$$

we achieve a close approximation of the actual frame rate $\widetilde{f}_s$. Excluding the outliers at $f_s = 20\,\text{Hz}$ and $f_s = 30\,\text{Hz}$, the approximation exhibits a root mean squared error of 0.026 Hz. We use Equation (5.27) to set a new target frame rate $f_s + |\Delta \widetilde{f}_s(f_s)|$, such that the actual playback frame rate will more closely approximate the originally desired frame rate f_s. With that, the average playback frame rate did not deviate more than 0.05 Hz from the target frame rate. In addition, the playback frame periods of the resulting system was evaluated by measuring them in software. It was found that the actual playback frame periods $\widetilde{T}_s$ closely match the originally targeted T_s, with a maximum absolute deviation of $|T_s - \widetilde{T}_s| = 30\,\mu\text{s}$ per frame period among all measurements.

An Acer XB270H was used as display. It has a 27-inch TN-panel, with a resolution of 1920 horizontal times 1080 vertical pixels. The panel, which as a contrast ratio of 1000:1, was set to its maximum luminance, 300 candela per square meter. This way, the shown video sequences exhibit high contrast on a conventional monitor, again allowing the obtained results to serve as upper bounds for the critical frame frequency f_c. The display supports Nvidia

G-Sync, which refreshes the image whenever a new frame is available in the frame buffer (i.e., in contrast to conventional monitors) up to a maximum refresh rate of 144 Hz. With the used sequences, we do not exceed 144 Hz, so this display allows us to show videos with the exact frame rate defined by the custom video player. Conventional monitors have fixed refresh rates, leading to delayed presentation of some video frames.

Both the player's and the display's time precision were evaluated by encoding a video sequence that showed black and white frames interchangeably. While playing back this sequence, a PT was put on the display surface and its resistance sampled with 8 kHz to observe the brightness change of the monitor with a high time resolution. This experiment proved that the setup is working as expected, frame period variations were smaller than $1/8\,\text{kHz} = 125\,\mu\text{s}$.

The width of the display panel is $w = 60\,\text{cm}$, test subjects are seated with an eye-to-panel distance of approximately $d = 70\,\text{cm}$ in front of the monitor. Therefore, the display spans 46.4 degrees of the horizontal field of view of the test subjects. Lighting in the room was dominated by artificial light to eliminate the influence of daylight or weather conditions.

5.3.5 Participants

In addition to two authors of paper [3], seven male and one female subject were recruited to participate in the experiment. The age of all participants at experiment time ranged from 23 to 30 years, with a mean of 27.6 years. All participants have either normal or corrected-to-normal eyesight.

5.3.6 Experimental Procedure

Preliminary experiments used the method of constant stimuli [119] to retrieve a prior estimate of the absolute frame rate threshold, equal to the critical sampling frequency f_c from Section 5.2.2, at which humans can distinguish time-discrete presentation of a video sequence from the continuous presentation of the same scene 50 % of the time. In the method of constant stimuli, test subjects are randomly presented a stimulus (video sequence) from a range of stimulus parameter values, in our case from a frame rate range. Stimuli from the range have to be sampled with equal spacing, and each stimulus must have equal probability for presentation to avoid errors of habituation and expectation. Hence, this method frequently presents stimuli which are rather distant from threshold f_c and accordingly easy to classify. Consequently, these answers carry little novel information and are for this reason often redundant.

To circumvent this issue, a modified variant of the simple up-down or staircase method [120], [121] was used. In the up-down method, the next sequence is chosen based on the answer for the previous sequence: if jerkiness was perceived (answer **x** in Figure 5.9), the frame rate of the next sequence will be increased by for example 2 Hz, if no jerkiness was perceived (answer **o** in Figure 5.9), frame rate will be lowered by the same step size of 2 Hz, see the black line at test round number 13 and greater in Figure 5.9. This procedure ensures that frame rates close to threshold f_c will be frequently sampled, avoiding redundant stim-

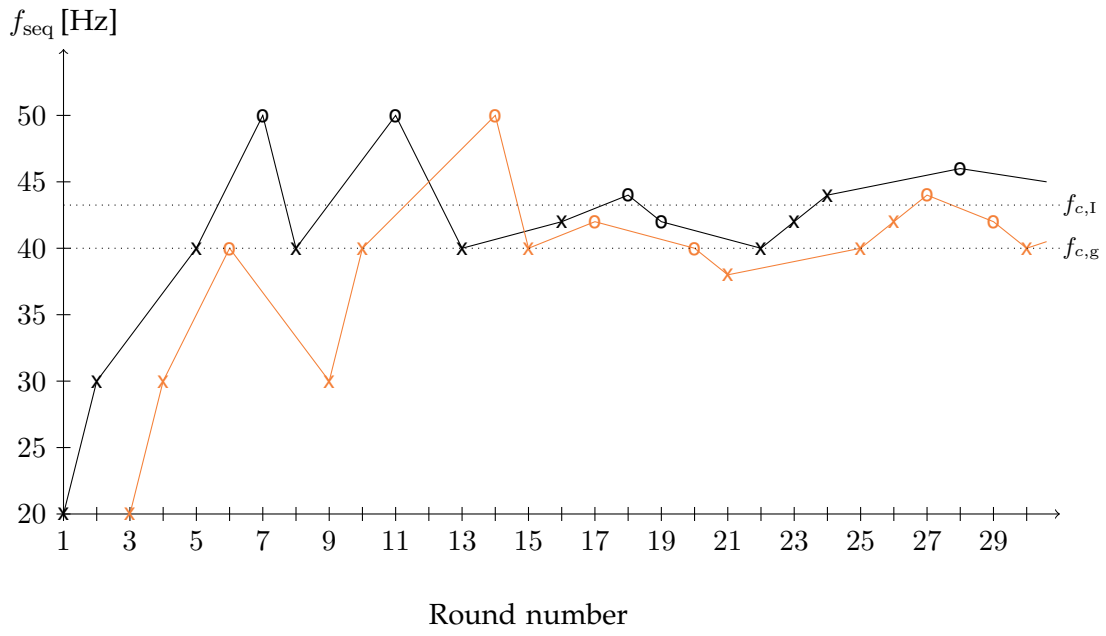


Figure 5.9: Example of an adaptive interleaved staircase test with two sequence sets. In each round, set I (black line, absolute frame rate threshold $f_{c,\text{I}}$) or set g (orange line, absolute frame rate threshold $f_{c,\text{g}}$) is chosen randomly. Within each set, the answer to a sequence (x: could perceive stutter, o: could not perceive stutter) increases (x) or decreases (o) frame rate f_{seq} of the next sequence from that set by the current step size. Step size is in the beginning 10 Hz, and reduced to 2 Hz after four reversals in that set (adapted from [3], © 2018 IEEE).

uli. However, test participants will be able to predict frame rate, which leads to errors of expectation.

This is why the interleaved staircase method [121] was used. In this method, the test subject is presented, for instance, sequence sets I and g (see Table 5.1 for parameters, and Figure 5.9 for an example). For each sequence set individually, the simple staircase method is used. In addition, the sequence sets are randomly interleaved, meaning that the next sequence to be presented is randomly chosen from the sequence sets I and g. After choosing the set of the next sequence, the sequence frame rate f_{seq} is determined according to the simple staircase based on the previous answer for the chosen set. This strongly impedes predicting frame rate, in particular if the number of sets is large enough and if the set parameters speed and exposure are sufficiently similar. In the experiments, at least three sequence sets were used for interleaving, making it impossible for test subjects to predict frame rates.

Further potential in increasing the efficiency of the experiment lies in its beginning: it starts at a low [121] frame rate $f_{\text{seq}} = 20$ Hz, which all test subjects classified as jerky for all sequence sets. The following considerations apply to one set: during the test, participants first have to converge to the absolute frame rate threshold f_c before being able to give the most relevant answers.

To speed up convergence, a step size larger than, for example, 2 Hz could be used, which would decrease the precision of the result. Therefore, an adaptive step size in two experiment phases [122], [123], as shown in Figure 5.9 is used: in the first phase, we prefer to

quickly converge to f_c, and thus choose a large step size of 10 Hz. This phase terminates when the test subject's answers have caused four reversals. A reversal takes place when the chosen frame rates f_{seq} pass through an extremum, meaning that frame rate was first decreased and then increased, or vice versa. After four reversals, the second phase, in which we use a smaller step size of 2 Hz, starts. The experiment finishes when twelve reversals have taken place during the second phase in each of the sequence sets.

An oral introduction was given to all test subjects. In addition, they received an introductory sheet, which described the experiment, the user interface and encouraged behavior for result consistency, such as keeping a constant distance to the monitor (not leaning towards or away from it) and taking breaks if needed. The sheet states that "sequences are randomly chosen without any correlation", concealing the underlying adaptive interleaved staircase method. Distinguishing monocular from binocular vision is not necessary for motion perception [34], so no precautions in this respect were taken.

The 35 sequence sets of the two author subjects from Table 5.1 were divided into eight interleaved experiment blocks with four sequence sets each, and one experiment block with three sequence sets. The nine sequences of the remaining eight test subjects, highlighted in Table 5.1, were divided into one experiment block with the four sequence sets A to D, and one experiment block with the five sequence sets E to I.

5.3.7 Data Analysis

After a participant has finished the experiment the frame rate threshold f_c for each sequence set is computed. For the adaptive step size staircase method, the threshold f_c equals the average of the last twelve reversals [123].

5.4 Experimental Results

Using the subjective tests of Section 5.3, perception thresholds for all ten participants are retrieved. These results, and their accordance to the four hypotheses from Section 5.2.4 are discussed in this section.

5.4.1 Preliminary Conclusions from Two Test Subjects

First, the experiment was conducted on all 35 sequence sets from Table 5.1 by two author test subjects. The resulting average f_c for all sets are depicted in Figure 5.10. In the horizontal layout of the graph in Figure 5.10, the array of sequence sets from Table 5.1 can be recognized.

No more precise plot or more values of f_c are given because this experiment is motivational and its exact values do not generalize, since it was conducted on only two users. Still, the experiment shows that test sequence set sub-sampling is feasible. Revisiting the four hypotheses from Section 5.2.4, we observe the following: first, we can see that higher speed v increases the absolute frame rate threshold f_c (hypothesis 1). Compare, for example, $f_{c,\text{J}} = 59.83\,\text{Hz}$, which shows the moving bar at an angular speed of 72.5 deg/s and

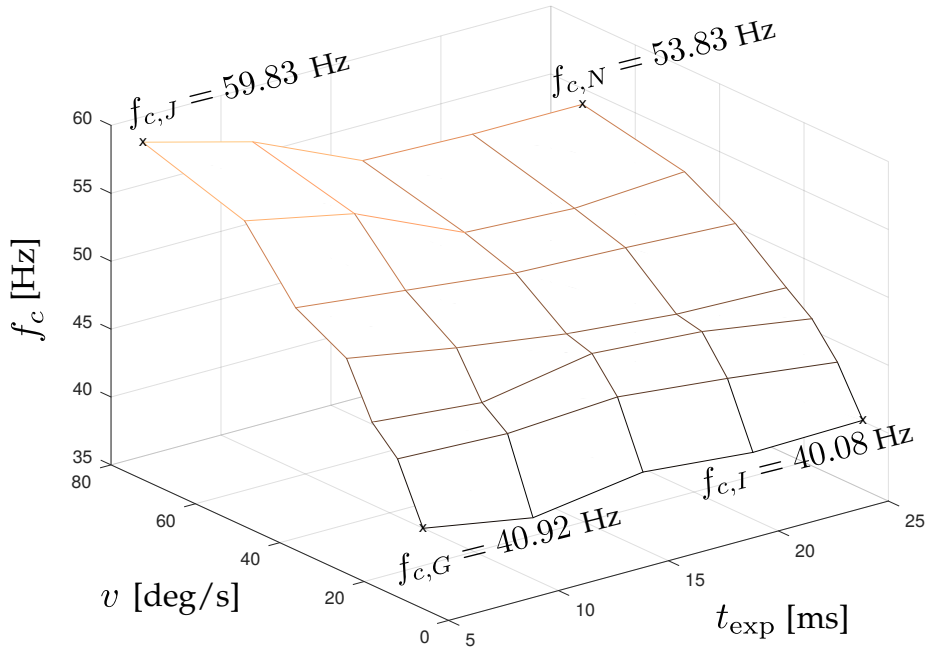


Figure 5.10: Surface plots of the average absolute frame rate thresholds (critical sampling frequencies) f_c for two test subjects on all 35 sequence sets. Corner points of the tetragons that constitute the bent planes correspond to the tested sequence sets and their absolute frame rate threshold f_c. The horizontal layout of the sample points in the velocity/exposure plane is based on Table 5.1 (adapted from [3], © 2018 IEEE).

$f_{\text{c,G}} = 40.92$ Hz, which has an angle speed of 6.0 deg/s (see Table 5.1). Hence, from the two opposing effects of speed v on critical sampling rate f_c presented in Section 5.2.4, the effect changing the steepness of the spatiotemporal spectrum (Figure 5.3 and Equation (5.9)) has more influence than the filter width d (Equation (5.12) and Equation (5.24)).

The second hypothesis, stating that higher exposure will decrease f_c, holds only for large enough speeds. There is a 6 Hz difference between $f_{\text{c,J}}$ and $f_{\text{c,N}}$, but only a 0.84 Hz difference between $f_{\text{c,G}}$ and $f_{\text{c,I}}$. The decrease in influence is understandable because the width of the camera filter $h(x)$ is equal to the product of speed and exposure time (Equation (5.12)). Also, this insight confirms the third hypothesis, stating that at reduced speed the influence of exposure time will diminish.

Finally, the difference between $f_{\text{c,J}}$ and $f_{\text{c,G}}$ is larger than the difference between $f_{\text{c,N}}$ and $f_{\text{c,I}}$, confirming the fourth hypothesis: longer exposure time (causing increased motion blur) decreases the influence of speed.

In addition to the four hypotheses, we note that Figure 5.10 shows points that cannot exist in real systems: for instance, $f_{c,N} = 53.83$ Hz, but in a real system, at an exposure time of 25 ms, video can be recorded at most at 40 Hz. It was possible to create these points for the experiment, as in the simulation, the frame rate is not constrained to the inverse of the exposure time. For real systems, one can conclude that in particular for long exposure times, there

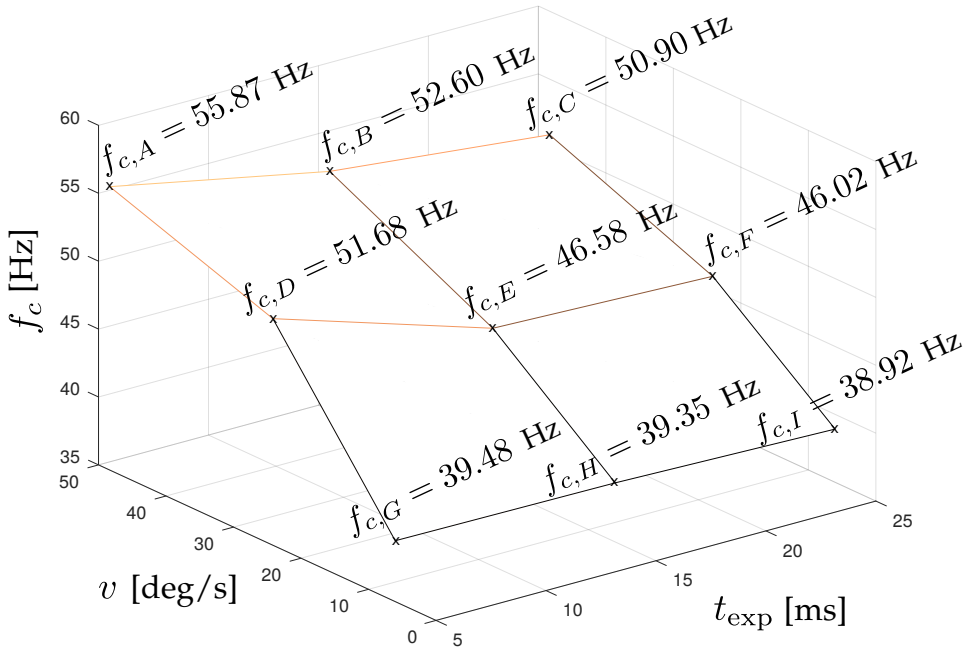


Figure 5.11: Surface plots of the average absolute frame rate thresholds (critical sampling frequencies) f_c for all test subjects on nine sequence sets (adapted from [3], © 2018 IEEE). Box plots corresponding to this figure are depicted in Figure 5.12.

exist object velocities at which humans are always going to perceive temporal sampling. This is most probably because at typical display brightness, the human visual system should integrate light for periods much shorter than 25 ms. This disparity between the camera's and the eyes' integration time is perceived by humans. Already for exposure times shorter than 15 ms all resulting critical sampling frequencies can be processed in real systems. In summary, for systems in which temporal sampling shall be made imperceptible, exposure time has to be chosen low enough to enable display at temporal sampling frequencies requested by Figure 5.10.

All previous insights also hold for all test subjects, as the following section shows.

5.4.2 Average Test Subject Behavior of All Test Subjects

The graph in Figure 5.11 illustrates the results for the average absolute frame rate threshold f_c. It can be seen that the conclusions from Section 5.4.1 generalize to all test subjects. The tendencies of frame rate thresholds of individual participants match the tendencies in Figure 5.11 well; corresponding box plots are given in Figure 5.12. We also see a grouping effect: sequence set D was presented in an interleaved experiment together with sets A, B, and C. This results in a comparably high $f_{\mathrm{c,D}} = 51.68\,\mathrm{Hz}$, in contrast to $f_{\mathrm{c,E}} = 46.58\,\mathrm{Hz}$ from sequence set E, which was interleaved with sets F to I.

Section 5.5 gives a further, detailed statistical analysis of the results, including statistical

confidence tests.

5.4.3 Test Length and Difficulty

The ten test subjects spent overall 6 hours and 40 minutes performing the experiments for sets A to I, giving 2734 answers. Experiment time per participant was between 25 and 61 minutes, with an average of 44 minutes. For each sequence, participants spent on average 8.8 seconds from starting to watch the sequence until submitting an answer (fastest participant: on average 5.8 seconds, slowest participant: on average 15.6 seconds).

Users took on average two seconds more per sequence on the first experiment block with sets A to D than on the second block with sequence sets E to I. This is understandable since they started with the first block and had to get used to the experiment. Otherwise, no statistically significant difference in the average time users spent on classifying one set, or the average number of answers given for a set, was found. A significant difference in these values between sets would indicate that subjects had more difficulties classifying sequences from one set than from the other. Given no statistically significant difference, we can say that for none of the chosen sequence set parameters, test subjects had difficulties to complete the experiment. In addition, the statistically insignificant time differences confirm that the experiment is short enough to avoid that tired users give careless answers.

5.4.4 Directional Independence

Existing literature does not discuss the relation of direction of motion and the corresponding absolute frame rate threshold f_c, at which humans perceive jitter 50% of the time. We expect that there will not be a significant difference of thresholds f_c for changing motion directions considering the experiments by Cox *et al.* [124] and Seiffert *et al.* [40]: the authors in [124] presented eight motion directions (up, right, down, left, up/right, down/right, up/left, and down/left) to participants and measured the corresponding temporal frequency threshold for a motion signal to be perceivable in contrast to no motion. They found only minor and inconsistent differences of perception thresholds for the different directions. This insight should yield a direction independence of f_c, because in Cox's and Derrington's experiment [124], the same spatiotemporal filters in the human visual cortex as in the experiment of this thesis are defining the thresholds. Furthermore, the authors in [40] use a rotating disk for presenting motion in their experiments and do not report any direction influence, which further indicates that motion direction does not influence frame rate threshold f_c. Finally, the arrangement of photoreceptor cells in the eye's retina does not exhibit a dominant direction [46], and the filter structure behind these cells does not apply particular weights to specific directions [124], [125].

To confirm this assumption, the screen was rotated 90 degrees, such that the bar was moving from top to bottom, and performed the interleaved experiment with sequence sets A to D on one participant. As expected, the differences to the thresholds f_c from the horizontal experiment were insignificant and inconsistent. Using this result together with the insights from the previous paragraph [40], [124], [125], we have no reason to believe that we will find

divergent thresholds for other directions. Thus, motion direction does not change the absolute frame rate perception threshold f_c. This result is used for simplifying the computation of required video frame rates in Section 6.2.2.

5.5 Statistical Analysis

In this thesis, a constant sampling frequency (CSF) system denotes a system which samples signals with a constant period of time between two successive samples. Therefore, sampling with constant frequency is, in this manuscript, also temporally uniform sampling. In contrast to that, adaptive sampling frequency (ASF) systems do not have a fixed sampling period. The period between two successive samples is adapted to the sampled data.

The terms refresh rate or update frequency for displays are not used in the following. Instead, as discussed in Section 3.4.5.1, we consider sampling rates or frequencies for cameras as well as for displays to make notation more consistent. In summary, this thesis refers to CSF and ASF cameras as well as CSF and ASF displays.

5.5.1 Statistical Significance of Tendencies

The statistical significance of the results from Section 5.4 are analyzed in the following. For a better understanding, box plots of the absolute frame rate thresholds f_c are depicted in Figure 5.12. In Figure 5.11 we see that increasing speed seems to increase the absolute frame rate threshold f_c for each of the exposure times 5 ms, 15 ms, and 25 ms. Applying Fisher's one-way analysis of variance (ANOVA) test [126] to the results from sets A, D, and G ($t_{\text{exp}} = 5\,\text{ms}$) reveals significant differences ($F(2, 27) = 37.24, p < 0.001$) of the thresholds. We find similar results ($F(2, 27) = 32.83, p < 0.001$) for sets B, E, and H ($t_{\text{exp}} = 15\,\text{ms}$) and for sets C, F and I ($t_{\text{exp}} = 25\,\text{ms}$), in which the ANOVA values ($F(2, 27) = 26.65, p < 0.001$) are once more statistically significant. The post-hoc Tukey test further revealed that the mean threshold of one set is statistically significantly different ($p < 0.01$) from the other two set means for each exposure time.

As can be seen in Figure 5.12, the influence of exposure time on threshold f_c is smaller than the effect of speed. Observing the box plots from sets A, B, and C, we also see that the variance of thresholds between sequence sets is small compared to the variance of thresholds within sets. Nevertheless, for the majority of test subjects, a strong and consistent correlation between the individual test subject thresholds and exposure time t_{exp} was seen. Therefore, the subjective differences between test subjects would conceal the existing dependency of threshold f_c on exposure time t_{exp} in a classical ANOVA test. This is why a repeated measures ANOVA (RANOVA) test [126] is applied to check statistical significance of the effect of exposure time t_{exp} on f_c. The RANOVA test computes the statistical significance of difference among data sets while ignoring variations between test subjects.

Applying RANOVA to sets A, B, and C yields significant differences ($F(2, 18) = 29.69$, $p < 0.001$) for thresholds f_c for varying exposure times at speed $v = 48\,\text{deg/s}$. In addition, the post-hoc Tukey test reports significant pairwise differences for all set mean thresholds. The difference is significant at the 0.01 level in set pairs (A,B) and (A,C) and significant at

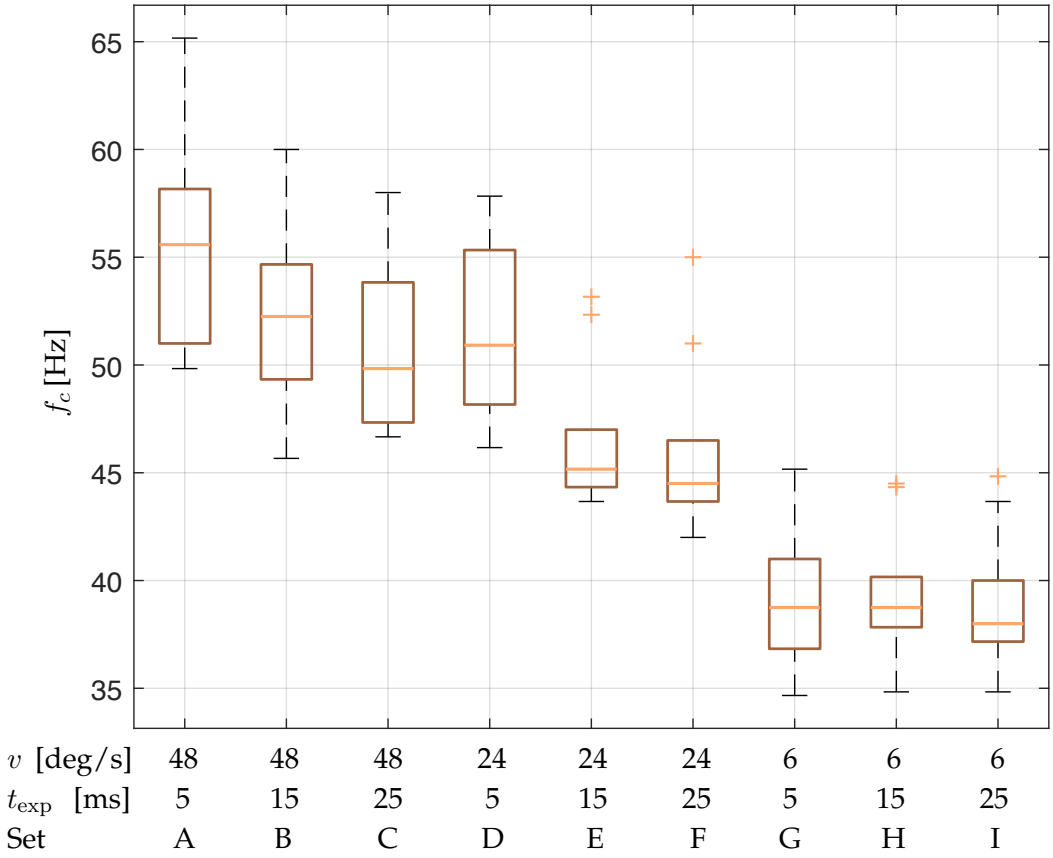


Figure 5.12: Box plots of all participant thresholds for each of the nine sequence sets A to I. The boxes contain 50% of the data, the horizontal lines in them represent the median. Whiskers are at most 1.5 times the box height (interquartile range) and end at the lowest or highest found threshold value. Pluses denote outliers (adapted from [3], © 2018 IEEE).

the 0.05 level in set pair (B,C). For sets D, E, and F, RANOVA ($F(2, 18) = 17.05, p < 0.01$) again yields significant differences, but the post-hoc Tukey test finds a significant difference only on the pairs (D,E) and (D,F) at the 0.01 level. This result has to be read with caution, since the threshold result $f_{c,D}$ is higher than what was expected because of the grouping effect discussed in Section 5.4.2. Finally, for $v = 6\,\text{deg/s}$ on sets G, H, and I, RANOVA ($F(2, 18) = 3.19, p = 0.065$) reveals that statistical significance at the 0.05 level is just missed. To back up this result, the post-hoc Tukey test finds no statistically significant differences in all three set pairs.

In summary, the tests especially confirm hypothesis 3: exposure time significantly affects f_c at high speeds v, while the influence vanishes at lower speeds. The remaining hypotheses are also confirmed, as the trends observable in Figure 5.11 are proven to be statistically significant.

5.5.2 Model of the Absolute Frame Rate Threshold

The previous sections found that both speed and exposure time have a significant influence on the absolute frame rate threshold, rendering Figure 5.11 a valid model for predicting f_c. To be able to apply these insights to general scenarios, a linear function is fitted to the data

from Figure 5.11:

$$f_c(t_{\text{exp}}, v) = 40.44 - 153.2 \cdot t_{\text{exp}} + 0.3230 \cdot v, \tag{5.28}$$

where t_{exp} is given in milliseconds and v in degree per second. A linear model was chosen as with higher degrees for the polynomial the model can achieve a closer fit of the model to the values present in Figure 5.11, but the higher degree polynomial does not extrapolate well to values of v and t_{exp} outside the region investigated in the previous experiment (overfitting). The linear model is based on the mean f_c, and hence critical sampling frequencies computed by it will make temporal sampling imperceptible to users 50% percent of the time. To satisfy a greater fraction of users, a frequency offset f_o can be added to Equation (5.28), as done in Algorithm 4.

Chapter 6

Methods for Delay Reduction

Based on the insights from Chapters 3 and 5, this chapter proposes various techniques to reduce G2G and G2A latency in video communication. All techniques employ high sampling rates, as required by the analysis in Section 3.7. The first method, detailed in Section 6.1, proposes a greedy frame skipping algorithm that minimizes both G2G latency, as well as the data rate of the transmitted video. The frame rate of the video resulting from the proposed greedy algorithm may, however, be extremely low, such that temporal sampling can become perceivable. This deteriorates the QoE of the video consumer, as analyzed in Section 6.1.4.5. In consequence, Section 6.2 proposes a content-adaptive minimum frame rate, at which humans do not perceive that temporal sampling is taking place. At the minimum frame rate, video consumers will not be able to distinguish the temporally sampled video from the original, continuous scene. Consequently, humans will also not be able to notice that the video is frame skipped, if the frame rate shown to them is always above the critical minimum frame rate required by Algorithm 4, proposed in Section 6.2. Both algorithms from Section 6.1 and Section 6.2 are merged in Section 6.3.

Frame skipping, as proposed in this thesis, generally distinguishes between key frames, which contain important novel visual information, and regular frames, which are transmitted to sustain a base frame rate required for good QoE. Since key frames carry visual information related to events, they determine the G2G delay of a video communication setup. To avoid that processing or transmission of a key frame has to be stalled because of regular frames, Section 6.4 proposes a preemption algorithm.

Finally, Section 6.5 compares the theoretical G2G delay model from Section 3.6.2 to delay measurements performed on the prototype from Section 6.1. This confirms the theoretical model, shows how to apply it, and gives a delay analysis of a low delay prototype that includes the methods proposed in this chapter.

Most of the contributions of this chapter have been published in [3] and [4].

6.1 Greedy Adaptive Frame Skipping

The approach presented in this section discards video frames that do not contain significant information. This is performed to reduce the frame rate, and hence data rate of a video, while

keeping the G2G delay at a constant level. In this implementation, the algorithm accepts a QoE loss (see Section 6.1.4.5) in favor of reduced data rate.

6.1.1 Observations

Increasing the sampling rates of cameras and displays is necessary to reduce the significant G2G delay contributors t_{CTS} and t_{DTS} representing the delays from the sampling processes. The maximum and mean delays of the uniform distributions of t_{CTS} and t_{DTS} (see Sections 3.4.1.1 and 3.4.5.1) are inversely proportional to the sampling rate. For example, when the small post-exposure delay t_{post} (see Section 3.4.1.1) is neglected, the delay contributed by a temporal sampling process with a frequency of 30 Hz is distributed following a $\mathcal{U}(0, 33.\overline{3})$ ms probability distribution. The sampling process in readily available 240 Hz cameras contributes a significantly lower delay of $\mathcal{U}(0, 4.1\overline{6})$ ms.

However, the higher frame frequency video will increase data and packet rates considerably and stress processing components such as the encoder and decoder. In addition, as Section 5.4.2 shows, humans do not require a high frame rate of for example 240 Hz such that temporal sampling is imperceptible. Similarly, most real-time machine vision procedures such as object tracking or mapping algorithms operate at frame rates smaller than 30 Hz [93], [94]. In summary, we need a high sampling frequency to ensure low latency in the video communication, but this increases stress on processing and transmission elements, and video consumers such as humans or computer vision algorithms are not able to process frames at such high rates. Consequently, insignificant frames of the high frame rate video should be discarded to reduce the frame rate, while keeping latency unchanged. The corresponding frame skipping algorithm is presented in the following.

6.1.2 Greedy Frame Skipping Algorithm

As shown in Figure 3.1, the frame skipping block is placed immediately after the camera. With this placement, a skip or forward decision for every frame is made before any further processing of that video frame is performed. The proposed frame skipping method avoids an increased output information rate from the encoder, which potentially overloads the transmission channel, and avoids overburdening any block in the video communication chain (Figure 3.1). The decision whether to further process or skip a frame is based on the following three criteria:

1. **Content:** The larger the amount of new information in a frame compared to the last non-skipped frame, the more likely a frame is to be chosen for further processing. New information can, for instance, be measured as the Mean Absolute Difference (MAD) or the Structural Similarity index (SSIM) [127]. Other frame skipping metrics for change detection can be used instead. Alternatively, if the frame skipping unit is placed after an intra-only encoder, the encoded frame could be analyzed. The following only considers frame skipping that selects raw frames. By transmitting the frames with new information, it is ensured that a new event which significantly changes the frame content achieves minimal delay. The process is illustrated in Figure 6.1. Frames with

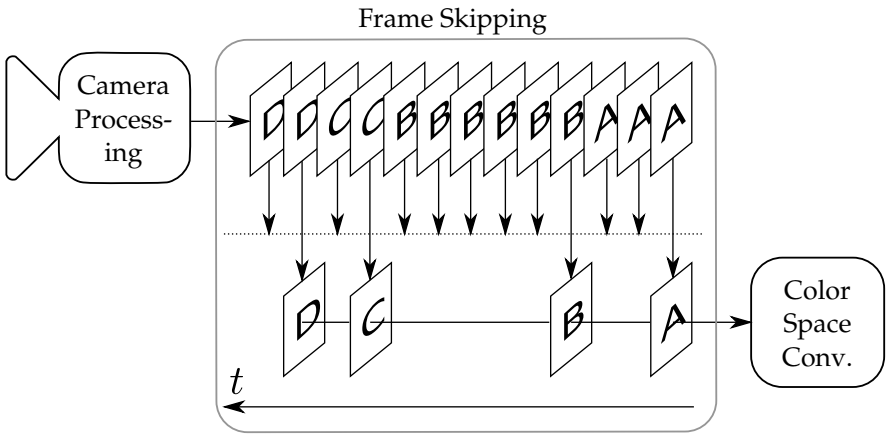


Figure 6.1: Frame skipping process: a subset of the frames from the camera is selected for encoding, the remaining frames are skipped (adapted from [4], © 2018 IEEE).

similar content have the same letter A, B, C, or D. From the three similar frames with the letter A, only the first frame needs to be transmitted, the subsequent two frames are not forwarded to the encoder. The next frame passed to the encoder is the first frame marked with B, which has a significant difference to the previous frames, for instance in terms of MAD. The choice of skipping or transmitting a frame can, for example, be based on a MAD threshold value.

Clearly, the rate of frames coming out of the frame selector is lower than the rate of incoming frames from the camera. The number of selected frames depends on the frame content and the skipping threshold. Note that in Figure 6.1, there are multiple frames in the frame selector for illustration purposes. A real implementation does not store the frames, but decides for every frame as fast as possible whether the frame should be skipped or transmitted. For MAD computation, the most recently transmitted frame is stored and compared with a new frame arriving at the frame skipping module. If the new frame is selected for transmission, the new frame replaces the old comparison frame as the most recent frame.

2. **Subjective Criteria:** A low frame rate and many frame drops decrease subjective video quality [128]–[130]. Therefore, we target to select regular frames with a minimum frame rate predefined by $1/t_{\max}$. $t_{\max}$ is chosen to keep the subjective quality as high as possible while satisfying other constraints, such as the delay and the channel data rate. Having a lower boundary for the frame rate is beneficial to both human observers and machine vision algorithms. The latter typically require regular updates to perform well.

3. **Bottleneck Component:** Every block of the chain has a constant or varying throughput rate in frames per second. The rate of selected frames may not exceed the rate of the slowest block. Otherwise, frames will have to be dropped or queued by these blocks which would lead to additional delay. Consequently, the frame skipping block selects frames for transmission at most at a frame rate equal to $1/t_{\min}$.

Algorithm 3: Greedy Frame Skipping

Data: Content difference ΔI, difference threshold I_{thr}, time Δt_{prev} since last frame, minimum time $t_{\min}$ between two frames, maximum time $t_{\max}$ between two frames

Result: Classification of frame as key frame (return 2), regular frame (return 1), or skip (return 0)

```
1  begin
2      if ΔI > I_thr // Significant content difference
3      then
4          if Δt_prev > t_min // Avoid bottleneck overburdening
5          then
6              return 2
7          else
8              return 0
9      else
10         if Δt_prev > t_max // Subjective criteria
11         then
12             return 1
13         else
14             return 0
```

Joining all three criteria into frame selector instructions yields the decision rules presented in Algorithm 3. ΔI is the content difference between the last transmitted frame and the current frame, which the frame selector shall classify as key or regular frame. The content difference can be quantified using MAD, SSIM, or other metrics. Δt_{prev} is the time since the last frame was transmitted. If ΔI exceeds a content difference threshold I_{thr} and Δt_{prev} is smaller than $t_{\min}$, then the frame is skipped in order to keep the frame rate below the rate supported by the bottleneck block.

If ΔI exceeds the threshold and $\Delta t_{\text{prev}} > t_{\min}$, then the frame is transmitted as key frame. If ΔI is smaller than I_{thr}, the frame is skipped, except when Δt_{prev} is greater than $t_{\max}$; in that case, the frame is transmitted as regular frame to maintain the minimal frame rate $1/t_{\max}$, which satisfies subjective criteria.

6.1.3 Prototype System Description

The two prototype components of relevance, the specific implementation of the frame skipping algorithm, and the prototype system are detailed in the following.

6.1.3.1 Algorithm implementation

Algorithm 3 was implemented based on a thresholded form of the MAD in the experimental prototype. For high performance, the following algorithm is implemented in C++, using OpenCV [118]. The thresholded form of the MAD first conducts a pixel-wise subtraction of

the current frame from the previously transmitted frame. For each pixel of frame i at position (m, n) in the difference frame the value equals

$$d_{i,m,n} = |l_{i,m,n} - l_{i-k,m,n}|. \tag{6.1}$$

This is the absolute difference of the pixel luminances l in the same spatial position in frames i and $i - k$, with the last frame being transmitted k frames ago. The absolute pixel values d of the difference frame are then thresholded to yield

$$\hat{d}_{i,m,n} = 255 \cdot \min\left(1, \max\left(0, d_{i,m,n} - 10\right)\right). \tag{6.2}$$

All pixel differences d greater than ten are set to the maximum value of $\hat{d} = 255$, while the remaining pixels in the difference image are set to the minimum value of $\hat{d} = 0$. This thresholding significantly reduces the influence of the camera's image sensor noise [95], which distorts each pixel value. Subsequently, the mean of the thresholded difference image ΔI is computed. The resulting thresholded MAD ΔI is compared with the fixed MAD content difference threshold $I_{\text{thr}} = 1.4$. The $I_{\text{thr}} = 1.4$ threshold value was determined empirically in order to be sensitive to events that are spatially small in a video with a resolution of 640x480 pixels. At the same time, the $I_{\text{thr}} = 1.4$ value is above the difference value of two images that differ only in noise.

In the prototype setup, the parameter $t_{\min} = 0$ ms is chosen, as all units are able to process images at more than 240 Hz, the utilized temporal sampling frequency in the prototype. Finally, $t_{\max}$ is set to 420 ms, yielding a lower target of 2.4 Hz for the frame rate, that is, 1 % of the camera frame rate.

The minimum frame rate $1/t_{\max}$ is adjusted depending on the video content in Section 6.2 using the critical sampling frequency f_c. Future research may explore further parameter adaptations. For instance, the content threshold I_{thr} could be adjusted based on video content, lighting conditions, and camera model. In addition, the bottleneck parameter $t_{\min}$ could be adapted according to the state of the communication network.

6.1.3.2 System Description

A prototype of the video communication system depicted in Figure 3.1 was implemented. A Ximea MQ022CG-CM USB3.0 camera is recording a video with a spatial resolution of 640x480 pixels in the RGB color space and a temporal resolution of 240 Hz. The camera is connected to a Ubuntu 16.04 Desktop PC running the frame skipping algorithm and an x264 encoder [61] on an Intel Core i7 quad core processor with 3.6 GHz per core. The encoder is tuned towards the lowest latency settings, with intra-only encoding using the tunings *zerolatency* and *fastdecode* as well as the preset *ultrafast*. The encoded video is then streamed to the decoder PC using the user datagram protocol (UDP) over a Gigabit Ethernet link with a data rate of $C = 1$ Gbit/s. At the decoder PC, the encoded video is decoded using the libav decoder[1] and displayed on an Acer XB270H monitor with a sampling frequency of 144 Hz.

[1] www.libav.org, last visited 15.10.2018

6.1.4 Experimental Results

G2G and G2A delay were measured using the system described in Chapter 4. In addition, the individual delays for frame skipping t_{FS}, encoding t_{Enc}, network t_{Netw}, decoding t_{Dec}, and color conversion t_{CC} were measured with the high resolution clock of the C++ chrono library. Also, the frame and data rate of the produced video are computed over windows of one second. We consider four scenarios with a static video background: first, transmission of all frames with high camera frame rate. Second, the same camera frame rate with frame skipping enabled. Third, transmission of all frames with a constant camera frame rate which equals the average frame rate after the frame skipper in the second scenario. Fourth, spatially more complex video content to demonstrate the influence of the content on delay. The video content in the first three scenes is the top of a table in front of a white wall with a few cables and part of a disabled monitor on it. In the fourth scenario, the front panel of an oscilloscope and a LEGO® construction are added for spatial complexity.

For every scenario and partial delay component, at least 500 samples were recorded. The results of these measurements are summarized in Table 6.1 and the empirical cumulative distribution functions for the measured G2G and G2A delays in the first scenario are plotted in Figure 6.2 with 95 % confidence envelope based on the Dvoretzky-Kiefer-Wolfowitz inequality [131], [132].

6.1.4.1 Full Frame Rate Transmission

When transmitting all frames at 240 Hz, the system achieves a mean G2G delay of 19.67 ms, as noted in Table 6.1. The maximum G2G delay is considerably higher at 35.65 ms. We observe from Figure 6.2 that the high maximum delay values are caused by a few outliers. These outliers are caused by interrupts and scheduling of the operating system of the computers involved in the video transmission. Without these interrupts, the maximum G2G delay would be approximately 25 ms.

For the G2A delay, the same phenomenon can be seen in Figure 6.2. Without the outliers, the maximum G2A delay would be approximately 13 ms. From Table 6.1, we observe that the difference between the average G2G delay and the average G2A delay is 8.62 ms. This difference represents the average delay contributed by the display processing chain, including the display refresh. The difference between the minimum G2G delay and the minimum G2A delay is $t_{\mathrm{DP}} = 5.69$ ms, which represents the minimum delay of the display processing, with a zero display refresh delay. The variance of the display processing delay t_{DP} is in this context negligible because display processing is implemented in hardware. Therefore, t_{DP} is approximated with a constant delay.

The minimum delays of frame skipping t_{FS}, encoder t_{Enc} and decoder t_{Dec}, network t_{Netw}, and two times color conversion t_{CC} sum up to 1.71 ms. The camera frame processing and transmission delay from the camera to the encoder computer was measured to be $t_{\mathrm{CP}} = 6.02$ ms with the G2X delay measurement system. Summing these up to 7.73 ms leaves 0.53 ms compared to the minimum measured G2A delay of 8.26 ms for the remaining delay components, such as network interfacing, memory access latencies and CPU thread start de-

Scenario	Component [unit]	Minimum	Q1	Mean	Median	Q3	Maximum	Std. Deviation
1) High fps, no skipping	t_{G2G} [ms]	13.95	17.79	19.67	19.46	21.19	35.65	2.86
	t_{G2A} [ms]	8.26	9.67	11.05	10.82	12.10	26.88	2.15
	t_{CP} [ms]	6.02	6.02	6.02	6.02	6.02	6.02	0.00
	t_{FS} [ms]	0.22	0.24	0.25	0.24	0.25	0.81	0.02
	t_{Enc} [ms]	0.78	0.85	0.88	0.88	0.92	1.67	0.05
	t_{Netw} [ms]	0.02	0.02	0.02	0.02	0.02	0.02	0.00
	t_{Dec} [ms]	0.17	0.24	0.27	0.26	0.29	0.89	0.06
	t_{CC} [ms]	0.26	0.30	0.32	0.31	0.33	0.68	0.05
	t_{DP} [ms]	5.69	5.69	5.69	5.69	5.69	5.69	0.00
	Frame rate [Hz]	240	240	240	240	240	240	0.00
	Data rate [kByte/s]	656	657	658	657	658	659	0.50
2) High fps, skipping	t_{G2G} [ms]	14.66	19.41	21.23	21.12	22.98	28.93	2.69
	t_{G2A} [ms]	8.51	10.31	11.48	11.27	12.55	16.77	1.56
	Frame rate [Hz]	2.40	6.00	7.40	7.00	8.00	13.00	1.57
	Data rate [kByte/s]	3.12	14.63	17.34	17.44	20.12	34.57	4.20
3) Low fps, no skipping	t_{G2G} [ms]	21.95	68.16	100.18	100.39	132.23	169.92	38.80
	t_{G2A} [ms]	12.16	57.35	89.93	90.21	122.04	162.18	38.97
	Frame rate [Hz]	7.40	7.40	7.40	7.40	7.40	7.40	0.00
	Data rate [kByte/s]	7.97	13.33	13.78	13.39	15.04	22.56	2.47
4) Complex scene, high fps, no skipping	t_{G2G} [ms]	15.49	19.33	21.20	20.87	22.63	37.18	3.08
	t_{G2A} [ms]	9.86	11.20	12.69	12.35	13.44	27.13	2.60
	t_{Enc} [ms]	0.90	1.00	1.08	1.05	1.10	3.58	0.15
	t_{Netw} [ms]	0.05	0.05	0.05	0.05	0.06	0.07	0.06
	t_{Dec} [ms]	0.36	0.45	0.52	0.48	0.53	3.28	0.14
	Data rate [kByte/s]	1393	1588	1634	1641	1675	1770	59.01

Table 6.1: Prototype measurement results: Minimum, first quartile (Q1), mean, median, third quartile (Q3), maximum and standard deviation of performance metrics (adapted from [4]).

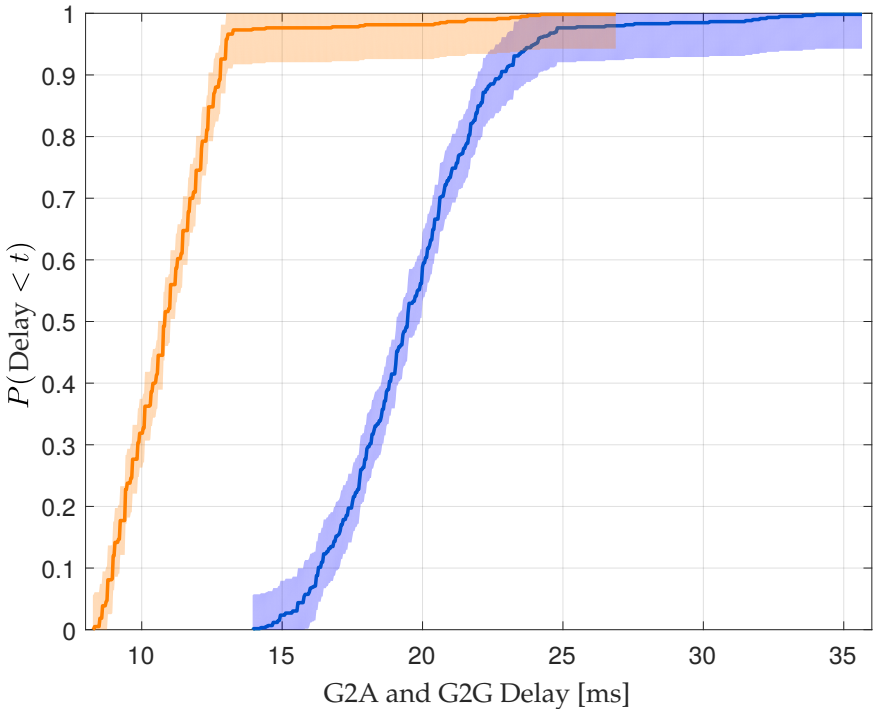


Figure 6.2: Empirical cumulative distribution functions for G2A (orange) and G2G (blue) delays for scenario 1 (high fps, no skipping), including 95 %-confidence envelopes around the graphs (adapted from [4], © 2018 IEEE).

lay, which are not individually measured. Furthermore, the G2A delay measurement has a precision of 0.5 ms per sample, so the measured mean G2A delay of 8.26 ms could in reality be smaller. The average data rate of the compressed video at full frame rate is 658 kByte/s.

6.1.4.2 Enabling Frame Skipping

With frame skipping enabled, we observe that the mean G2G delay is increased by 1.56 ms, while the mean G2A delay inflation is smaller, with an increase of 0.43 ms, compared to scenario 1 (see Table 6.1) without frame skipping. The difference in mean delay increase is caused by a more sensitive G2G delay detection in scenario 1. As described in Section 4.1.3.3, the LED lights up at a random time during one exposure period. The earlier that happens, the more light from the LED falls on the photo sensor, yielding an image of a seemingly brighter LED. On the other hand, if the LED is turned on close to the end of an exposure period, the LED will appear to be dimmer. In the full frame rate transmission scenario (scenario 1 in Table 6.1), frames with the dim LED are transmitted and trigger the rising edge detection Algorithm 2 after the PT. The PT's brightness increase detection algorithm is based on differences of the brightness in front of the PT and is tuned to be very sensitive. In comparison, the G2A detection (see Section 4.3.1) is based on pixel thresholding to keep computational complexity at a minimum. Pixel thresholding is less sensitive and may classify images in which the LED is dim, but lit up, as images in which the LED is still completely turned off. In consequence, in scenario 1, the G2G delay measurement detects some images as already containing the lit LED, while the G2A algorithm misses them, leading to a comparably higher

G2A delay for scenario 1.

With frame skipping enabled in scenario 2, the frame skipper decides which frames contain novel information and are passed on as key frames. The frame skipper uses thresholding and classifies images with an LED with low brightness as regular frames, skipping them. Therefore, these frames with low LED brightness cannot be detected early by the PT, yielding a higher mean G2G delay in scenario 2 with frame skipping than in scenario 1 without frame skipping. The delay increase with frame skipping is less pronounced for the G2A delay since the less sensitive pixel thresholding was not able to detect the LED with low brightness in scenario 1.

The benefit of frame skipping can clearly be seen in the reduced average frame rate and data rate; both drop approximately by a factor of 40. In particular, frame skipping reduces the mean data rate from an average of 658 kByte/s down to 17.34 kByte/s.

Frame skipping has the interesting side benefit of reducing the maximum G2G delay by 6.72 ms and the maximum G2A delay by 10.11 ms. These reductions of the maximum delays are due to the reduced processing loads of the sender and receiver PCs. Instead of processing 240 fps, they now process only 7.4 fps, reducing the processing load from 32.5 % to 22.5 % of the sender PC, such that now less than one of four cores is fully used. In the receiver PC, the processing load reduces from 10 % to an insignificant load compared to background tasks. This leaves more idle time to execute elementary tasks of the operating system, which makes these tasks less likely to interfere with video (de-)compression.

6.1.4.3 Low Frame Rate

In order to gain further insight into frame skipping (scenario 2), we compare with video transmission scenario 3. Video transmission scenario 3 has a constant low frame rate that is set to the average frame rate of the frame skipping scenario 2. That is, scenario 3 uses a camera running at a constant frame rate of 7.4 frames per second. We observe from Table 6.1 that this low frame rate has a significant impact on the mean G2G and G2A delays compared to the high frame rate with frame skipping scenario 2 in Table 6.1. The mean G2G delays rise from 21.23 ms to 100.18 ms, and the mean G2A delays increase from 11.48 ms to 89.93 ms. Moreover, the mean data rate of the low frame rate scenario is 13.78 kByte/s, which is slightly lower than the 17.34 kByte/s average data rate of the frame skipping scenario. Thus, we conclude that frame skipping employed with a high frame rate camera in the prototype setup requires only about the same (or slightly more) transmission data rate as a conventional low frame rate camera while drastically reducing the mean G2G and G2A delays in this prototype setup.

6.1.4.4 Varying Image Contents

The final scenario (scenario 4 in Table 6.1) demonstrates how a more complex video sequence affects the delays. Pointing the camera to a more complex scene increases the mean G2G and G2A delays by 1.63 ms and 1.64 ms, respectively. Encoding and decoding times are on average increased by 0.2 ms, while the data rate is doubled to tripled compared to the full trans-

mission scenario 1. The remainder of the increase in G2G and G2A delay can be explained by the higher amount of data that has to be processed by the networking hardware.

The increases from the G2G delays to the G2A delays are nearly the same for scenarios 1 and 4, which validates the measurements; since the display delay should not be affected by varying frame contents. It should be noted, however, that in a vision-based control system, more complex image contents can substantially increase the image processing delays.

6.1.4.5 Influence of Frame Skipping on Video Quality

With the parameter settings detailed in Section 6.1.3.1, frame skipping has a minor influence on the subjectively perceived video quality. In perfectly still scenes, the frame skipping block forwards images only when the $t_{\max}$ condition from Algorithm 3 forces an image transmission, even if the image content difference ΔI is small. The only difference between such images is the thermal noise caused by the camera sensor [95]. The low frame rate $1/t_{\max} = 2.4\,\text{Hz}$ is perceivable through image differences caused by the thermal noise.

The influence of frame skipping on the objective video quality was evaluated as follows. Objects were moved at various speeds in front of the camera. For each video sequence and prescribed content difference threshold value I_{thr}, the PSNR and SSIM values between the displayed frames and the corresponding skipped frames are computed. This yields approximately 5000 value pairs for each roughly 20-second sequence. The lowest PSNR and SSIM values across a video sequence indicate the largest deviation of the displayed frames from the skipped frames. The lowest values are also representative of the perceivable discontinuity when a displayed frame is replaced by a new frame. The lowest values provide conservative lower bounds compared to alternative approaches, such as considering the mean of the PSNR and SSIM values across the video stream [129], that have been developed for sub-sampling with a constant rate of displayed frames. The measurements indicate a lowest PSNR value of 38 dB and a lowest SSIM of 0.95 between the displayed and skipped frames for frame skipping with the default $I_{\text{thr}} = 1.4$ threshold.

The performance characteristics of frame skipping depend mainly on the content difference threshold I_{thr}. For a still scene, the thresholded MAD values between subsequent images in the prototype are approximately 1.1 due to camera sensor noise. As noted in Section 6.1.3.1, the $I_{\text{thr}} = 1.4$ default threshold is used to be sensitive to new events, but also robust to noise. For $I_{\text{thr}} < 1.1$, the frame skipping mechanism chooses all frames, while for increasing $I_{\text{thr}} > 1.1$, the algorithm starts skipping frames. For threshold values up to $I_{\text{thr}} = 3$, frame skipping gives nearly the same performance values as reported thus far; specifically, the lowest PSNR and SSIM values drop slightly to 36 dB and 0.94, respectively, for $I_{\text{thr}} = 3$. Increasing $I_{\text{thr}} > 3$ slowly deteriorates the video quality to lowest PSNR and SSIM values of 32 dB and 0.93 and an increased mean G2G delay of 23.73 ms for $I_{\text{thr}} = 10$. For even larger I_{thr}, the algorithm starts skipping frames that actually contain an event, severely decreasing the quality of experience and increasing delay. Overall, this thesis recommends a threshold I_{thr} just above the camera sensor noise for low delay, good video quality, and a low data rate.

6.1.5 Summary

The proposed greedy frame skipping algorithm successfully maintains the low latency of a video with high temporal sampling frequency, while the frame and data rates are significantly lower. Conversely, if the skipped variant is compared to a video with the same average frame rate, latency is significantly reduced. The algorithm, however, has a minor, negative impact on the QoE, as low frame rates are noticeable.

6.2 Perception-Optimized Adaptive Frame Skipping

To avoid that frame skipping deteriorates QoE, the results from Chapter 5 shall be utilized. They allow us to adjust the minimum frame rate such that the process of frame skipping becomes imperceptible to a human observer. Before the actual algorithm is introduced, three necessary observations are presented.

6.2.1 Observations

6.2.1.1 Effective Frame Rate of a Video Processing Chain

Nowadays, in the vast majority of used displays, the constant sampling frequency (CSF) is not synchronized to the image source, in contrast to the specific display used in the experiment from Section 5.3. In addition, practically all cameras in use (except dynamic vision sensors [43]) have constant sampling frequencies. Hence, this section shows how to map the frame rate threshold $f_c \in \mathrm{R}^+$ to its quantized version $\hat{f}_c$, dictated by the sampling frequencies of the involved CSF devices in Section 6.2.1.2. The effective sampling frequency f_s of images visible on the display is equal to the minimum of the involved sampling frequencies. Assume for example a 50 Hz camera and a 60 Hz display, or the other way around: the effective displaying rate will in both cases be 50 Hz, as the device with higher sampling frequency will sample at least once during the sampling period of the lower sampling frequency device. The example shows that it does not matter whether the camera or the display is sampling at a lower frequency. For the following thought experiments, we assume the display to have the lower, constant sampling frequency.

6.2.1.2 Frame Rate Quantization

A constant sampling frequency display can show videos with the original sampling rate f_s and integer fractions ($f_s/2$, $f_s/3$ etc.) of f_s by skipping one or more new images. Figure 6.3 shows the possible reduced display sampling frequencies for $f_s \in [30, 350]$ Hz. As target displaying frequencies for this thought experiment, we take the range of $f_c \in [40, 60]$ Hz from Figure 5.11 plus/minus a plausible safety margin of 20 Hz. Therefore, we target displaying frequencies in the range from 20 Hz to 80 Hz. We are in this example not interested in displaying higher frequencies than 80 Hz, as users will probably not perceive the increase in frame rate over 80 Hz, and we are also not interested displaying a video below 20 Hz, as humans will certainly perceive temporal sampling at that frequency. It can be seen in Figure 6.3 that

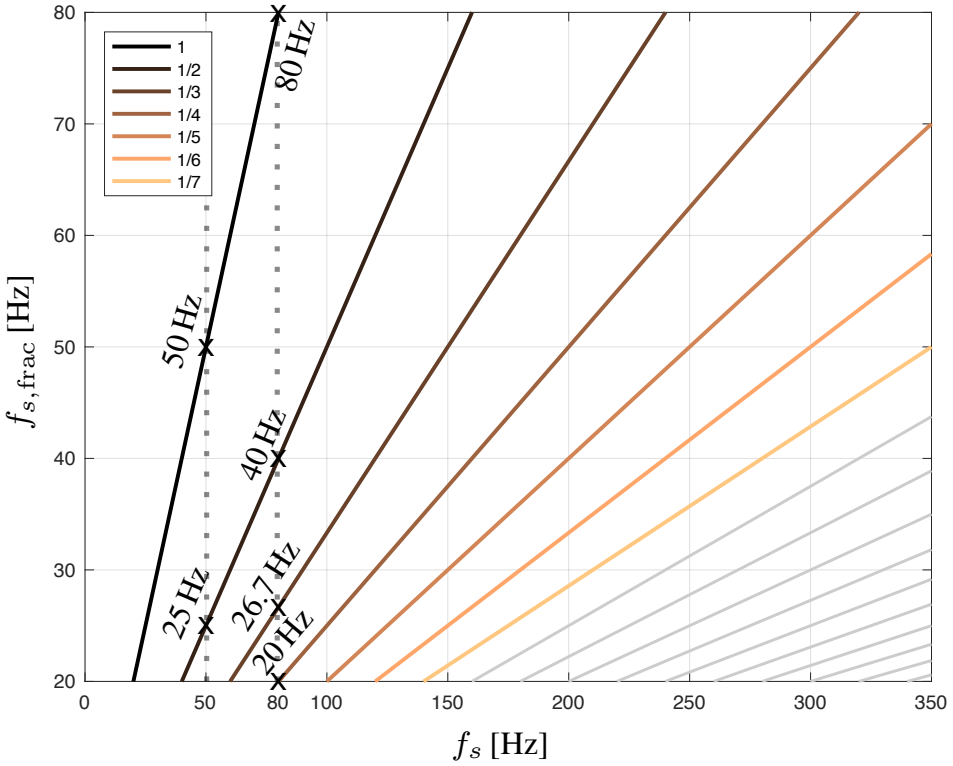


Figure 6.3: Possible display sampling frequencies $f_{s,\mathrm{frac}}$ for given original sampling frequency f_s. Available display sampling frequencies are highlighted for $f_s = 50\,\mathrm{Hz}$ and for $f_s = 80\,\mathrm{Hz}$. For clarity, the legend shows only the first seven integer fractions of each f_s. Greater fractions are depicted by gray lines (adapted from [3], © 2018 IEEE).

for low $f_s = 50\,\mathrm{Hz}$ we can only choose between two target frequencies (50 Hz and 25 Hz) in the relevant range between 20 Hz and 80 Hz, while for increased sampling frequencies there are more displaying frame rates available (for example for $f_s = 80\,\mathrm{Hz}$, these are 80 Hz, 40 Hz, 26.7 Hz, and 20 Hz). Section 6.2.1.3 analyzes how this frame rate quantization deteriorates the frame rate reduction that can be achieved by using the adaptive sampling rate proposed in this thesis.

Since the goal is that the user is unable to perceive the sampling process, we want to display the video with the absolute frame rate threshold f_c or a higher frequency. As an example, let us assume that we have a display with 120 Hz, and the video content characteristics demand that it is shown at $f_c = 50\,\mathrm{Hz}$. With the given display, we can display 120 Hz, 60 Hz, 40 Hz and so on. Thus, we choose to display it with the frame rate just above f_c, which is in this case $\hat{f}_c = 60\,\mathrm{Hz}$. The analog thought experiment can be done for a CSF camera. To formalize this quantization process, we first need to define the set

$$L = \left\{ \frac{f_s}{i} \right\}, \quad i \in \mathbb{N} \setminus \{0\} \text{ and } \frac{f_s}{i} \geq f_c \tag{6.3}$$

of available quantization levels for the given effective sampling rate f_s. Using set L of quantization levels, we apply quantization rule

$$\hat{f}_c = \lceil f_c \rceil_L \,, \tag{6.4}$$

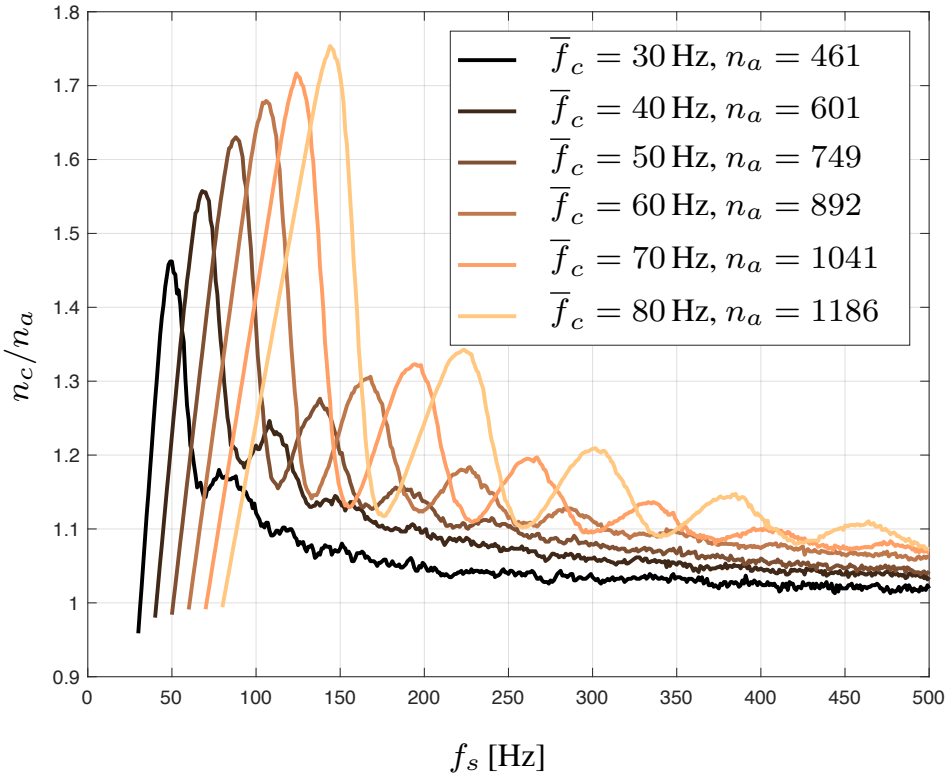


Figure 6.4: Ratio of required frames n_c for playback on CSF displays compared to the number of frames required for playback on ASF displays n_a. Note that for $f_s < \overline{f}_c + 15$, users will be able to perceive stutter because in this model, the maximum occurring f_c is in that case greater than f_s (adapted from [3], © 2018 IEEE).

where $\lceil \cdot \rceil_L$ quantizes the input to the smallest element from set L that is greater than or equal to the input. In Section 6.2.2, this temporal quantization is implemented. Figure 6.6 shows the trace of the desired f_c for a video sequence and the actually displayed quantized frame rate $\hat{f}_c$.

6.2.1.3 Loss of Frame Skipping Advantage as a Function of Effective Frame Rate

As can be seen in Figure 6.6, CSF systems require the video sequence to be displayed at a higher frequency $\hat{f}_c$ than requested by Equation (5.28). In this section, the number and ratio of additional frames needed to show the video when a constant sampling rate device is involved is analyzed.

The analysis is performed by simulating traces of f_c. The traces are created using an autoregressive model that reflects the trace characteristics seen in Figure 6.6 and Section 6.2.3. The mean absolute frame rate thresholds $\overline{f}_c$ range from 30 Hz to 80 Hz in steps of 10 Hz. The traces cover the 30 Hz interval $[\overline{f}_c - 15, \overline{f}_c + 15]$ Hz. For each of these traces, the number of displayed frames n_a required to show the corresponding video when only ASF devices are involved is computed.

In addition, the number of displayed frames n_c required to show the video if CSF processes (camera or display) with sampling frequencies $f_s \in [30, 500]$ Hz are involved is com-

puted. The ratio n_c/n_a depicted in Figure 6.4 expresses the amount of additionally shown frames of a CSF device compared to an ASF device. Most importantly, the overhead n_c/n_a decreases with increasing sampling rate f_s. This is expected, as with higher f_s, we have more levels for the quantization of f_c. The worst case investigated here is for the trace with $\overline{f}_c = 80\,\text{Hz}$ at $f_s = 144\,\text{Hz}$. In this case, the amount of additionally played frames is 75% (factor $n_c/n_a = 1.75$ in Figure 6.4). For commonplace 240 Hz devices, the ratio of additionally played frames never exceeds 20 %. Section 6.2.3 uses a CSF camera with $f_s = 339\,\text{Hz}$ and an ASF display. The mean display frame rates range from $\overline{f}_c = 46\,\text{Hz}$ to $\overline{f}_c = 76\,\text{Hz}$. Thus, the data rate saving results presented in Table 6.3 are approximately between 5% and 12% worse than what would be possible with $f_s \to \infty$.

6.2.2 Perception-Optimized Frame Skipping Algorithm

The previous insights lay the basis for the application of perception limits to video compression. Consequently, the procedure of extracting the adaptive playback frame rate from a coded video sequence is presented. Finally, experimental results of a non-real time implementation of the proposed technique are provided. For this application, the pixel densities of various display classes, given in Table 2.1, are utilized.

6.2.2.1 Display Pixel Density

Various display classes have different pixel densities ρ. The overview in Table 2.1 covers only horizontal size and angle, since the vertical size as well as angle directly follow for quadratic pixels. Hence, knowing the size in one dimension allows us to map motion given as pixel offset to motion in centimeters per second on the screen. Typical display dimensions, distance of the user to the display, user's FoV covered by the display, the number of pixels in a display row, and the pixel density ρ from user perspective are given in Table 2.1. In that table, we see that even within one display class pixel density can vary widely. The values in the last column of Table 2.1 can be used in Equation (6.6) to map the on-screen speed to an angular velocity v from a spectator's perspective.

6.2.2.2 Extracting f_c from Coded Video Sequences

From motion vectors of a coded video sequence (in this example the H.264 [47] video codec), we can obtain the absolute frame rate threshold f_c for each video frame. The procedure is summarized in Algorithm 4 and detailed in the following.

The H.264 codec utilizes inter prediction, described in Section 2.3.2.3. Inter prediction uses a reference frame to predict the contents of the current frame. To do so, the codec identifies a matching block in the reference frame for each block in the current frame. Ideally, the matching block corresponds to the same image content, which has been offset as a result of motion apparent in the video. This is why the vectors indicating the spatial relation between the two blocks are called motion vectors. Occasionally, block similarity is not caused by motion, but just by similarity of image content in different locations. In that case, motion vectors do not represent true motion. Hence, we need to filter the set of motion vectors.

Algorithm 4: Frame rate determination

Data: Frame i, effective sampling frequency f_s, frequency offset f_o
Result: Quantized critical frame rate $\hat{f}_c$

1 **begin**
2 Extract motion vectors $mv_{k,l,i}$
3 Scale motion vectors based on distance to reference frame
4 Apply 3D median filter (6.5) to the motion vectors
5 Compute maximum motion vector magnitude m_i
6 Calculate speed v_i using Equation (6.6)
7 Compute frame rate threshold $f_{c,i}$ using Equation (5.28)
8 Add desired frequency offset f_o to $f_{c,i}$
9 Quantize $f_{c,i}$ to retrieve $\hat{f}_{c,i}$ using Equation (6.4)

Given the motion vectors $mv_{k,l,i}$ at block locations k, l (vertical, horizontal) of frame i, we first scale the motion vector according to the temporal distance to its reference frame. If the reference frame is temporally adjacent, no scaling is done, if the reference is two frames away, the motion vector is divided by two, and so on. Afterwards, we apply the three-dimensional, spatiotemporal median filter (6.5) to suppress motion vectors that do not represent an actual motion. The filter

$$mvp_{k,l,i} = \underset{\substack{m,n\in[-2,2],\\ j\in[-6,0]}}{\text{median}} (mv_{k+m,l+n,i+j}) \tag{6.5}$$

takes the current motion vector and its spatially as well as temporally neighboring samples as input. The rationale for this filter is that the majority of motion vectors represent an actual motion, while there are few outlier vectors which are either too small or too large. An average of motion vectors would be distorted by these outliers. The median instead should give a good representative for the actual motion in the frame. This was the case in the conducted experiments, the filter dimensions from Equation (6.5) have yielded good results in empirical tests. The filter only uses past and current motion vector samples to ensure causality.

The spatiotemporal 3D median filter is applied to each spatial dimension of the two-dimensional motion vector separately. In the next step, we compute the motion vector magnitude using the Euclidean norm. We are not interested in the direction, as discussed in Section 5.4.4. Finally, for each frame i, we compute the maximum motion vector magnitude m_i from all motion vector magnitudes in the frame. This is done for the reason that the maximum motion apparent in the video defines the absolute frequency threshold for jitter perceptibility for the person viewing the video. The correctness of the steps of Algorithm 4 until step 5 were verified by watching the video and extracting the maximum object offset for a number of frames per hand. Comparing the manually extracted results (ground truth) to the values obtained with the algorithm presented in this section showed that steps 2-5 of Algorithm 4 are working precisely.

An alternative to using the results from video encoder motion analysis is applying a real-

time optical flow algorithm [133], [134] to extract motion vectors for each frame. This might be necessary if, for instance, video coding parameters restrict the motion vector length.

Next, we compute the angular speed v from the user's perspective. The maximum motion vector magnitude m_i within a frame divided by the temporal sampling frequency f_s of the video gives us the maximum occurring speed on the display. Together with display pixel density ρ (Table 2.1) we can map speed m_i relative to the display panel to an angular speed v_i from the user's perspective:

$$v_i(m_i, \rho, f_s) = \frac{m_i \cdot f_s}{\rho} \ [\mathrm{deg/s}] \tag{6.6}$$

The resulting speed from Equation (6.6) can be inserted into Equation (5.28) to compute the current critical sampling frequency $f_{c,i}$ for video frame i. Next, the frequency offset f_o is added to $f_{c,i}$ to modify the QoE. This is done because Equation (5.28) finds the critical sampling frequency at which humans can perceive temporal sampling 50 % of the time. For $f_o > 0$ Hz, a higher video quality in terms of temporal sampling frequency can be achieved. The experiments in Section 6.2.3 showed that the most sensitive test participants are not able to perceive temporal sampling for $f_o = 20$ Hz. Conversely, $f_o < 0$ Hz, can be used to reduce the data rate of the video. The advantage compared to CSF video playback at the same data rate is a more constant QoE. In the final step of Algorithm 4, the resulting $f_{c,i}$ including the offset is quantized to an integer fraction of f_s, as described in Section 6.2.1.2, according to Equation (6.4).

In a real implementation, $\hat{f}_{c,i}$ must be determined before encoding of the frame. This can be done by first computing only the motion vectors, then applying Algorithm 4, and afterwards making a frame skipping decision using $\hat{f}_{c,i}$. Finally, if the frame is to be further processed, it is encoded using the already available motion vectors.

6.2.3 Experimental Results

The methods proposed in Section 6.2.2.2 cannot be applied to widely used test sequences such as the "Foreman" sequence because most of them have been recorded at 30 Hz, which is below f_c for almost all content. Consequently, new test sequences were created: At an exposure time of 2.5 ms, a sampling frequency of $f_s = 339$ Hz, and a spatial resolution of 960×270 pixels videos were recorded using a XIMEA MQ022CG-CM camera. The raw image sequences were encoded using the x264 encoder [61], an implementation of the H.264 coding standard [47]. To ensure applicability in low delay video communication scenarios [4], the tunings *zerolatency* and *fastdecode* as well as preset *ultrafast* were used in the encoder.

Sequences in which single or multiple objects are moving in front of a static background and sequences in which the camera is panning at varying speed and in different directions over three dimensional scenes, yielding various on-screen speeds were recorded. Representative frames of these three videos are shown in Figure 6.5, a further description is given here:

- **Catwalk**: The camera is in a fixed position, and a person walks multiple times through the camera's field of view. The person swings his arms actively, so various objects are

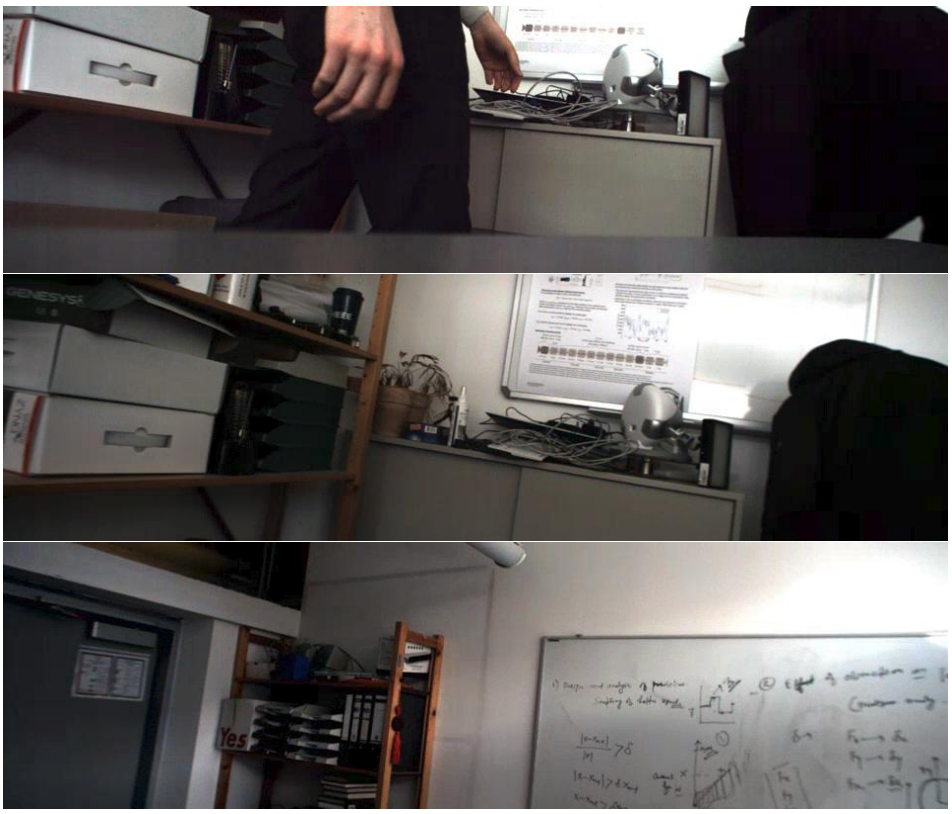


Figure 6.5: Video test sequences used the experiments in this section. Results for each of these sequences are summarized in Table 6.2. Sequence names from top to bottom: Catwalk, MessyRoom, Whiteboard.

moving at different speeds and directions at once. The average data rate of the compressed video at full frame rate equals $\overline{r} = 10.49\,\text{Mbit/s}$.

- **MessyRoom**: The camera pans over an untidy room with many details. Camera motion is random, and the camera almost never rests. The average data rate of the compressed video at full frame rate equals $\overline{r} = 23.25\,\text{Mbit/s}$.

- **Whiteboard**: The camera pans over a whiteboard with written text on it, and towards the end of the video, an arm is moving through the camera's field of view. The average data rate of the compressed video at full frame rate equals $\overline{r} = 21.01\,\text{Mbit/s}$.

In Figure 6.6, we can see the resulting f_c (orange) for a part of the Catwalk sequence. Despite using the ASF display from Section 5.3.4, we need to quantize f_c as a CSF camera is involved in the video processing chain. The quantization to integer fractions of $f_s = 339\,\text{Hz}$ is shown by the black line in Figure 6.6.

A note on how adding f_o nonlinearly affects the maximum playback frequency $f_{\max}$: in Figure 6.6 ($f_o = 0\,\text{Hz}$), the highest non-quantized f_c is approximately 90 Hz. The frequency quantization rule (Equation (6.4)) forces the playback frame rate to the next higher available frequency, therefore $f_{\max} = 339/3 = 113\,\text{Hz}$. In the example in Figure 6.6, if we add a frequency offset f_o of for example 20 Hz, the maximum playback frequency $f_{\max}$ will not

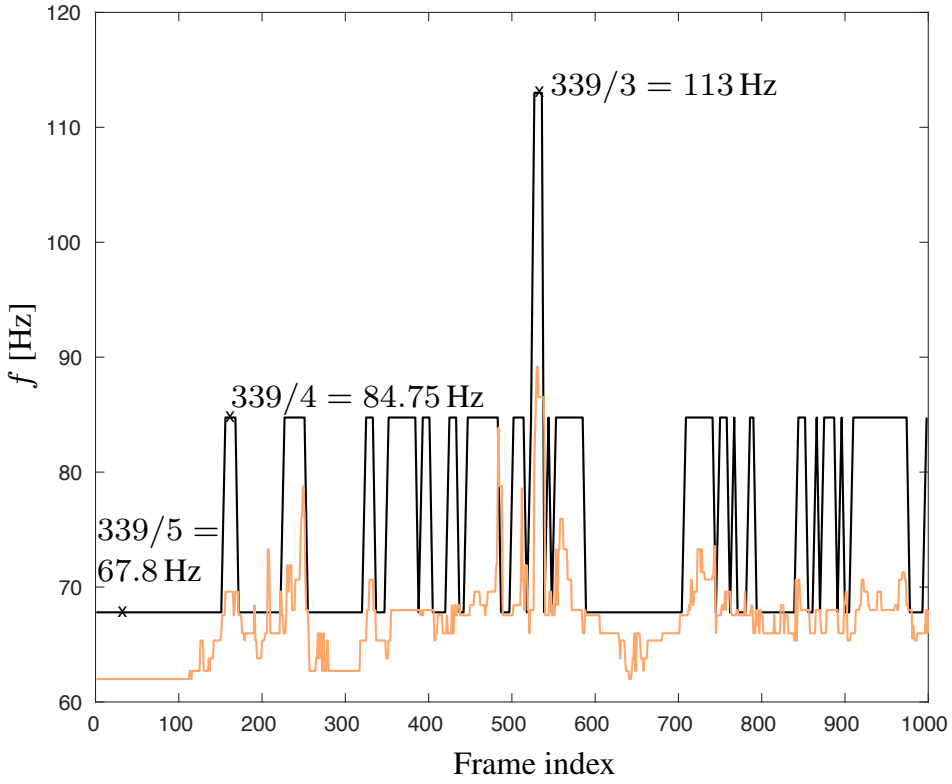


Figure 6.6: f_c (orange), and its quantization $\hat{f}_c$ (black) for part of the Catwalk video sequence with a constant sampling rate of $f_s = 339\,\text{Hz}$. We can see the three levels $339/3 = 113\,\text{Hz}$, $339/4 = 84.75\,\text{Hz}$, and $339/5 = 67.8\,\text{Hz}$ for $\hat{f}_c$ (adapted from [3], © 2018 IEEE).

change because $f_c + f_o = 90 + 20\,\text{Hz} < 113\,\text{Hz}$. If, for instance, we add a frequency offset of $f_o = 30\,\text{Hz}$ (now, $f_c + f_o = 90 + 30 = 120\,\text{Hz} > 113\,\text{Hz}$), the algorithm would quantize to the next higher available frequency, therefore $f_{\max} = 339/2 = 169.5\,\text{Hz}$. In conclusion: depending on its value, f_o might have no effect on $f_{\max}$, or it might push $f_{\max}$ to another available frequency quantization level from the set of quantization levels L, see Equation (6.3).

The model in Equation (5.28) and Algorithm 4 were verified by letting users view the sequences played back at $\hat{f}_c$. As predicted in Section 5.5.2, it was noted that different users require different frequency offsets f_o to be added to the result of model (5.28) in order not to notice any jitter. The most sensitive participant required $f_o = 20\,\text{Hz}$ to be added to Equationn (5.28). After this addition and video playback with the corresponding frequency, users did not notice any temporal sampling.

The data rate of a video played back with ASF can be compared to the required data rate of a video played back with CSF. The constant sampling and display frequency f_s has to correspond to the highest frequency $f_{\max}$ in the ASF sequence for humans never to be able to perceive temporal sampling.

Table 6.2 shows the frequency and average data rate results for the three sequences from Figure 6.5. All videos are encoded at the same default image quality settings to achieve the same (except negligible numerical variations) picture quality for all encoded videos. We investigate ASF Algorithm 4 for frequency offsets $f_o = 0\,\text{Hz}$ and $f_o = 20\,\text{Hz}$. In the corre-

Sampling	ASF, $f_o = 0\,\text{Hz}$			ASF, $f_o = 20\,\text{Hz}$			CSF: $\overline{r}(f_s)$		
Metric	$\overline{r}$	$\overline{f}_c$	$f_{\max}$	r	$\overline{f}_c$	$f_{\max}$	$\overline{r}(67.8)$	$\overline{r}(84.75)$	$\overline{r}(113)$
Catwalk	**2.64**	46.36	67.8	**3.73**	69.08	84.75	**3.12**	**4.47**	-
MessyRoom	**4.75**	50.45	84.75	**6.98**	75.88	113	-	**7.80**	**10.44**
Whiteboard	**4.54**	49.95	84.75	**6.58**	71.46	113	-	**7.86**	**10.24**

Table 6.2: Data rates of adaptive sampling frequency (ASF) and constant sampling frequency (CSF). Average data rate r is given in megabits per second, frequencies in Hz (adapted from [3]).

Sequence	$\Delta r(f_o = 0)$	$\Delta r(f_o = 20)$
Catwalk	15.4 %	16.6 %
MessyRoom	39.1 %	33.1 %
Whiteboard	42.2 %	35.7 %

Table 6.3: Data rate reduction of adaptive sampling frequency (ASF) and constant sampling frequency (CSF). The data rate reduction is the relative difference between the CSF data rate and the ASF data rate with the corresponding highest frequency $f_{\max}$ from Table 6.2 computed using Equation (6.7). Adapted from [3].

sponding columns, the entries show the average video data rate r in megabits per second (Mbps), mean playback frequency $\overline{f}_c$, and maximum playback frequency $f_{\max}$.

For CSF playback, Table 6.2 shows the average data rates r at sampling frequencies f_s corresponding to the maximum playback frequencies in the previous columns. The table compares the maximum frequency because in both the ASF and the CSF playback, humans may not be able to perceive temporal sampling. Hence, in the CSF case, the sequence has to be played back at $f_s = f_{\max}$. Thus, CSF data rates are presented at the maximum frequencies occurring in the ASF cases. Finally, Table 6.3 summarizes the average data rate reduction

$$\Delta r = \frac{\overline{r}_{\text{CSF}} - \overline{r}_{\text{ASF}}}{\overline{r}_{\text{CSF}}} \tag{6.7}$$

of ASF over CSF video processing.

We can see that the reduction of average data rate ranges from 15.4% to 42.2%. The exact amount of rate reduction depends mainly on the variance of motion speed v. If variance of speed v is large, we can expect a greater reduction: ASF would exhibit the greatest data rate reduction if there is high speed apparent for a short period of time (leading to a high $f_{\max}$, defining f_s for CSF), and most of the time there is little to no motion (leading to a low $\overline{f}_c$). On the other hand, with constant speed v, there would be no difference between ASF and CSF. From these examples it becomes clear that ASF does never increase data rate r compared to CSF if the video is played back at a frequency such that temporal sampling is imperceptible to humans.

6.2.4 Summary

The central contribution of this section, the ASF from Algorithm 4 achieves on average a 30.3 % data rate reduction compared to CSF video, while temporal sampling can not be perceived for both sampling strategies. The data rate comparison even gives an advantage to the conventional CSF strategy, because the sampling frequency of CSF is usually content-agnostic. Consequently, a fairer comparison would just assume a high CSF to avoid the perception of temporal sampling in a brute-force manner, in which case the data rate reduction would be even more significant.

The proposed Algorithm 4 can be used in general applications to reduce the video data rate. Even for video played back from a storage device, reducing the data rate in an imperceptible way is beneficial. The ASF algorithm is therefore not constrained to low delay video communication. Algorithm 4 is by itself not effective for reducing the delay of video communication, because it does not guarantee the transmission of visual events. This is why Section 6.3 merges the greedy Algorithm 3 with the perceptual Algorithm 4.

6.3 Merging the Proposed Algorithms

6.3.1 Observations

Algorithm 3 effectively reduces the sampling frequency, and hence the data rate of a video. It does so while keeping G2G latency almost as low as in the original video sequence with constant, high temporal sampling frequency. However, the video after frame skipping may exhibit a low temporal sampling rate that allows perception of the temporal sampling process, which leads to a loss of QoE.

On the other hand, Algorithm 4 reduces video sampling frequency such that the reduction of frame frequency is imperceptible, yielding an unchanged QoE. It does, however, provide no guarantee for low latency.

The two algorithms are thus complementary. The goal of this section is to merge the two algorithms to get the benefits of both: an imperceptible sampling frequency reduction, with no deteriorating effect on G2G latency.

6.3.2 Merged Frame Skipping Algorithm

The interfacing between algorithms 3 and 4 is done by using $t_{\max}$ from Algorithm 3. Variable $t_{\max}$ controls the minimum temporal sampling rate of the frame skipped image sequence produced by Algorithm 3. With unchanged video content, a lower $t_{\max}$ causes a higher rate of regular frames f_{reg}. As described in Section 6.1.2, $t_{\max}$ reflects subjective criteria such as the human perception of temporal sampling. Perception of temporal sampling is handled by Algorithm 4: for a given frame i, it finds a temporal sampling rate $\hat{f}_{c,i}$ at which temporal sampling is imperceptible to humans. Consequently, as shown in Algorithm 5,

$$t_{\max,i} = \frac{1}{\hat{f}_{c,i}} \tag{6.8}$$

Algorithm 5: Merged Frame Skipping

Data: Frame frequency offset f_o, set of video frames S
Result: Frame skipping decision (key/regular/skip frame) as in Algorithm 3
begin
  for *Frame* s_i *in* S **do**
    Algorithm 4: Compute quantized critical frame rate $\hat{f}_{c,i}$
    Equation (6.8): Compute $t_{\max,\mathrm{i}} = 1/\hat{f}_{c,i}$
    Algorithm 3: Compute frame skipping decision using $t_{\max,\mathrm{i}}$

is used to merge the algorithms. For each frame, we first apply Algorithm 4 to compute the critical sampling frequency $\hat{f}_{c,i}$. $t_{\max,i}$ equals the inverse of $\hat{f}_{c,i}$, as in Equation (6.8). Finally, $t_{\max,i}$ is utilized to give a lower bound for the frame rate computed by Algorithm 3. The entire procedure is depicted in Algorithm 5.

6.3.3 Experimental Results

The evaluation of merged Algorithm 5 focuses on how the two sub-algorithms 3 and 4 interact, and which data rate reduction the combined algorithm achieves. The effect of the merged algorithm on G2G latency is not evaluated because still, the greedy frame skipping (Algorithm 3) is used inside Algorithm 5. Additionally, the basis, in particular the key frame detection of the greedy algorithm, is unchanged, which is why for G2G latency we would see the same results as in Table 6.1.

The analogous point holds for the evaluation of perception-related effects. Section 6.2.3 proves that using Algorithm 4 reduces the frame rate of a video without any perceivable alteration of the video. In the merged algorithm, the perceptual sub-algorithm defines the lower frame rate limit. Consequently, the frame skipping in the video produced by Algorithm 5 will not be perceivable either.

The results of Algorithm 5 applied to the three video sequences from Section 6.2.3 are summarized in Table 6.4. Specifically, the average frequencies of key frames ($\overline{f}_{\mathrm{key}}$) and regular frames ($\overline{f}_{\mathrm{reg}}$), and how frame skipping according to the merged Algorithm 5 affects the data rate of the compressed video are analyzed. We compare the data rates to those of the videos recorded and encoded at full CSF $f = 339\,\mathrm{Hz}$, as presented in Section 6.2.3. This is useful because it was already proven that Algorithm 4 achieves a data rate reduction, even when compared to sub-sampled CSF videos that are played back at the lowest imperceptible sampling frequency. However, this frequency is usually not known and videos are processed at the frame frequency at which they were recorded. This is why we compute the relative data rate reduction Δr of the videos with full sampling frequency compared to the temporally sub-sampled videos (according to Algorithm 5) in Table 6.4.

First, we see that applying only the greedy Algorithm 3 yields few regular frames in all sequences, which is understandable given the setting $t_{\max} = 420\,\mathrm{ms}$ from Section 6.1.3.1. Applying only perceptual frame skipping Algorithm 4 yields no key frames, only regular frames. This is caused by the design of Algorithm 5, in which sub-algorithm 4 determines

Sequence	Algorithm	$\overline{f}_{\text{key}}$ [Hz]	$\overline{f}_{\text{reg}}$ [Hz]	$\overline{f}$ [Hz]	r [Mbit/s]	Δr [%]
Catwalk	Greedy	47.75	0.76	48.51	3.43	67.3
	Perception, $f_o = 0$	0.00	46.36	46.36	2.64	74.8
	Perception, $f_o = 20$	0.00	69.08	69.08	3.73	64.4
	Merged, $f_o = 0$	41.11	21.87	62.98	3.96	62.2
	Merged, $f_o = 20$	36.94	36.77	73.71	4.32	58.8
MessyRoom	Greedy	111.92	0.09	112.01	8.84	62.0
	Perception, $f_o = 0$	0.00	50.45	50.45	4.75	79.6
	Perception, $f_o = 20$	0.00	75.88	75.88	6.98	70.0
	Merged, $f_o = 0$	110.90	3.23	114.13	8.91	61.7
	Merged, $f_o = 20$	108.94	7.06	116.00	8.99	61.3
Whiteboard	Greedy	58.56	0.00	58.56	5.92	71.8
	Perception, $f_o = 0$	0.00	49.95	49.95	4.54	78.4
	Perception, $f_o = 20$	0.00	71.46	71.46	6.58	68.7
	Merged, $f_o = 0$	50.73	17.02	67.75	6.66	68.3
	Merged, $f_o = 20$	47.58	26.73	74.31	7.26	65.4

Table 6.4: Average frame rates $\overline{f}$, $\overline{f}_{\text{key}}$, and $\overline{f}_{\text{reg}}$, data rates $\overline{r}$, and data rate reductions Δr for greedy Algorithm 3, perceptual Algorithm 4, and merged Algorithm 5. We reuse the video sequences from Figure 6.5 and Section 6.2.3. $\overline{f}_{\text{key}}$ is the average frequency of key frames and $\overline{f}_{\text{reg}}$ represents average frequency of regular frames. Note that the average cumulative video frame rate equals the sum $\overline{f} = \overline{f}_{\text{key}} + \overline{f}_{\text{reg}}$.

only the frequency of regular frames.

Merging both algorithms in all cases lowers the frequency of key frames $\overline{f}_{\text{key}}$ as well as the frequency of regular frames $\overline{f}_{\text{reg}}$ in Table 6.4. This is due to the fact that sub-algorithms 3 and 4 compete for frames to classify: if the perceptual algorithm causes more regular frames, the stored frame for computing the content difference according to Equation (6.2) is updated more often. This reduces frame differences ΔI, and renders a key frame less likely to occur. Conversely, if a frame was just sent as key frame, we can wait for a longer time, precisely $1/\hat{f}_c$, before we need to send the next regular frame. Thus, a video with more key frames requires fewer regular frames. The sum $\overline{f}$ of the average key and regular frame frequencies equals the cumulative average temporal sampling rate of the video. Therefore, despite a reduction in one type of frames, the cumulative sampling rate $\overline{f}$ of a video from the merged algorithm is still increased compared to the result of an isolated algorithm.

The increase in temporal sampling rate $\overline{f}$ when transitioning from an isolated algorithm to the merged algorithm explains the loss in data rate reduction Δr. Data rate reduction Δr is computed as the relative difference between the data rate of the video with full sampling rate (see Section 6.2.3 and Algorithm 6.7) and the data rate of the frame skipped video. For

all sequences and frame skipping strategies, we see considerable data rate reductions Δr in Table 6.4. Depending on the video sequence, moving from an isolated frame skipping algorithm to the merged algorithm degrades data rate reduction by 3.3 % to 18.4 %. Still, compared to an original full frame rate video, the proposed Algorithm 5 achieves between 58.8 % and 68.3 % data rate reduction, while G2G latency and perceptual quality of the video remain unchanged. On average, the merged algorithm yields 64.1 % data rate reduction for $f_o = 0\,\text{Hz}$, and 61.8 % reduction for $f_o = 20\,\text{Hz}$.

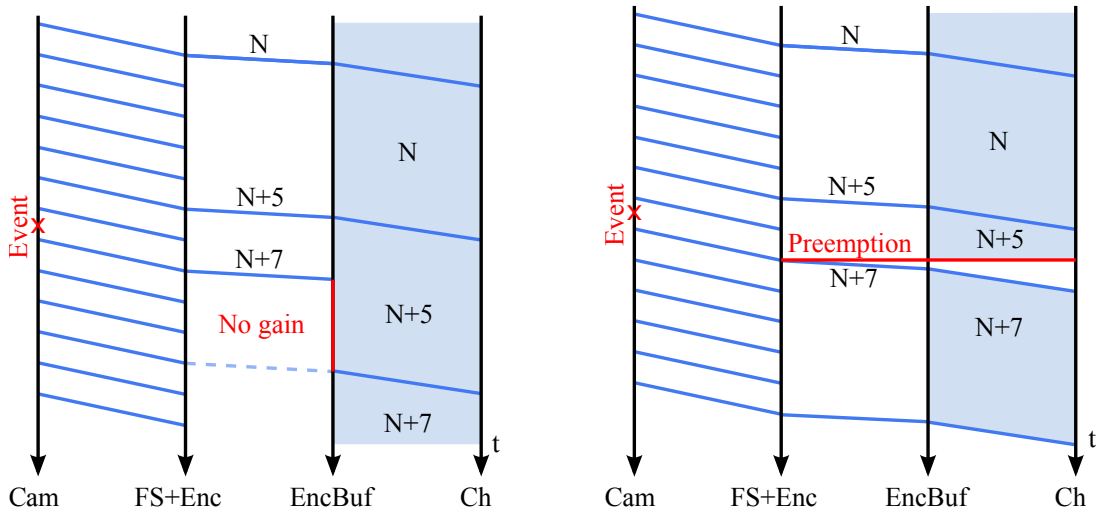
6.4 Preemption

In general computing, preemption means interrupting a process to first execute another process with higher priority. Applying preemption to the greedy or merged frame skipping algorithm first means to assign a higher priority to key frames than to regular frames. Second, the processing of regular frames in a block shall be aborted if a key frame requires processing because key frame carry visual event information, and therefore define the delay of a video communication chain. A further analysis of this issue, a way to implement preemption in the proposed setup, and an evaluation are given in the following.

6.4.1 Observations

This section focuses on the encoder buffer from Figure 3.1, as this is the most likely block at which queuing occurs because the subsequent network data rate is usually the most unreliable parameter in a video communication setup. Let us assume that a key frame is ready to be transmitted after being encoded, but an old regular frame is still being transmitted and therefore occupying the channel or the old regular frame is in the encoder buffer, waiting for transmission. In that case, the new key frame would have to wait in the encoder buffer until the old regular frame is completely transmitted. To avoid that this key frame has to wait, we flush the regular frames from the buffer and directly start transmitting the key frame. The flushing is beneficial because the regular frame does not contain significant information, in contrast to the arriving key frame. Such a scenario is depicted in the message sequence chart in Figure 6.7a, where frames are sent from the camera to the unit consisting of the frame selector and the encoder, from where they are forwarded to the encoder buffer. Frames are represented as blue lines. The light blue area between two frames is the frame data that is transmitted. In this example, only the frame transfer time on the transmission channel is relevant. The channel is fully used by regular frames, therefore the transmission time of a frame on the channel is exactly as long as one maximum frame period $t_{\max}$.

Suppose that $t_{\max}$ in Algorithm 3 is set such that every fifth frame from the camera is selected as a regular frame, as illustrated in Figure 6.7. The frame selector classifies frame $N + 7$ from the camera as a key frame. Frame $N + 7$ comes only two frames after regular frame $N + 5$. The key frame $N + 7$ will be transmitted after regular frame $N + 5$ is transmitted entirely (dashed line). It cannot be transmitted earlier because the channel is fully used. Accordingly, using a camera and frame selector on a fully loaded channel would not achieve any improvements over a conventional five times slower camera without frame selector.



(a) Preemption disabled (b) Preemption enabled

Figure 6.7: Key frame forwarding behavior in different preemption modes. In both scenarios, a visual event is recorded in key frame *N+7*. Without preemption, the key frame is buffered; with preemption, it is immediately transmitted to the channel (adapted from [4], © 2018 IEEE).

6.4.2 Preemption Algorithm

To achieve such an improvement, we need to take further measures when receiving a frame with an event: the previous, old regular frame that is still occupying the encoder, the encoder buffer, the channel, the decoder buffer, the decoder, or the graphics buffer feeding the display, has to be deleted (flushed) as shown in Figure 6.7b. The flushing is equally necessary for the general case, in which the channel is not fully used, but a new key frame arrives in the encoder buffer, while an older regular frame is still occupying the encoder buffer or is being transmitted.

The frame selector should still not choose events too often. If the frame selector chooses frames in quick succession, an event quickly following another will preempt the earlier event. Therefore, in a burst of events, only the last event would not be preempted, leading to a delayed transmission of the burst of events. The delayed burst transmission can be avoided by properly setting the time $t_{\min}$ in Algorithm 3 such that after an event occurred, a short timeout is enforced. During the timeout, no new key frame is transmitted and the frame containing the event can be safely transmitted. The preemption units should not preempt key frames in case of a too small $t_{\min}$, but drop the newly arriving ones. Not deleting leading key frames is reasonable because it is more important to transmit the initial event of a burst of events rather than a later one.

Encoders using inter-frame coding techniques may refer to frames that are later preempted. This causes artifacts in the decoded image because a frame that is being decoded references a previous frame that never arrived at the decoder. Thus, the decoder will abort decoding of the affected frame, leading to a temporal pause in video playback. This does not occur when using intra-only coding, since there are no dependencies between the frames, which is why intra-only coding in combination with preemption is used. However, intra-

only coding leads to worse compression efficiency because the temporal redundancy is not exploited to improve coding efficiency.

6.4.3 Experimental Results

The preemption mechanism was implemented with the Click Modular Router [135]. Specifically, the preemption functionality was added to the encoder buffer queueing block, see Figures 3.1 and 6.7, and fed the data stream from the encoder into the encoder buffer with preemption functionality. The first byte of each encoded frame identifies the frame either as a key frame or a regular frame, as determined by the thresholded MAD frame content assessment according to Line 2 of Algorithm 3 (the remainder of Algorithm 3 was not executed, that is, there was no frame skipping in order to evaluate the effects of preemption in isolation). To require buffering, the output rate of the encoder buffer was constrained to $C = 14\,\text{kByte/s}$ using the *BandwidthShaper* function of the Click Modular Router, while the actual channel data rate between the sender and the receiver was still $C = 1\,\text{Gbit/s}$.

The preemption evaluation focused on the flushing of full frames (that had not yet begun transmission) from the encoder buffer. The situation depicted in Figure 6.7, canceling the ongoing transmission of a regular frame to make way for a key frame, was not considered. This has not been done as interrupting the sending process of a packet is not possible to implement on the desktop prototype without kernel and driver changes. Including the cancellation of a sending process would cause a G2G delay reduction of up to one transmission period of a frame. For the current setup, this would be negligible since the actual transmission data rate of the connection between sender and receiver is 1 Gbit/s, giving frame transmission periods in the sub-millisecond range.

In Figure 6.8, we observe that preemption prevents large delays caused by filled buffers. While the maximum delay of the queued setup without preemption is 519.93 ms because of a filled buffer, the maximum delay is 43.97 ms with enabled preemption. Preemption also affects the average G2G delays, which are 111.13 ms and 17.10 ms for the setup without preemption and the setup with preemption, respectively. Thus, preemption is highly effective in the prototype, reducing the average G2G delays by roughly half an order of magnitude and the maximum G2G delays by a full order of magnitude.

Note that the average G2G delay for enabled preemption is smaller than the average delays in Table 6.1 because for this setup, the raw frame size was reduced to 320×240 pixels (with 240 frames per second). This frame size reduction was necessary so as to avoid IP packet fragmentation (for simplicity) in the prototype. With the 320×240 pixels frame size, the encoded frames are smaller than the maximum transmission unit (MTU). For the investigations in this section, encoder rate control was disabled to emphasize the advantage of preemption. Even enabled rate control can overshoot the target bits for one or more frames, in which case the preemption unit can flush them from the queue to make space for an incoming key frame.

We note that an alternative approach to preemption for avoiding overloading the encoder buffer and network channel could be to employ frame skipping with an increased $t_{\min}$ in Algorithm 3. However, increasing $t_{\min}$ may block a key frame (line 4 of Algorithm 3) that could

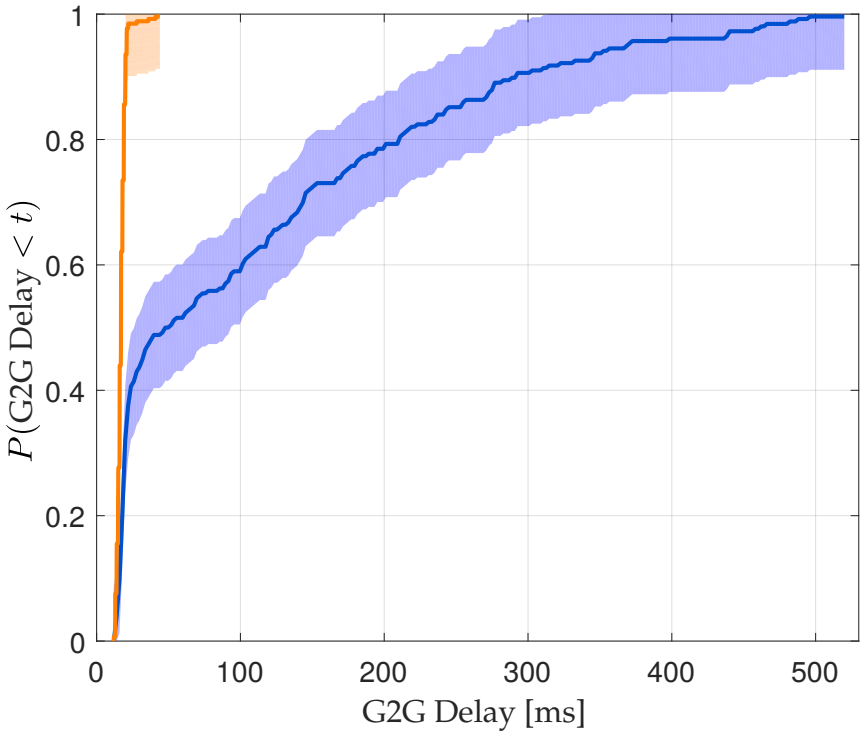


Figure 6.8: Cumulative G2G delay distribution function with preemption (orange) and without preemption (blue), including 95 %-confidence envelopes around the graphs (adapted from [4], © 2018 IEEE).

have passed through the encoder buffer using preemption. Consequently, increasing $t_{\min}$ would increase G2G delay. Therefore, preemption is the preferred method of dealing with frame rates that may temporarily exceed the processing capabilities of a block. On the other hand, employing frame skipping with a sufficiently large $t_{\min}$ is beneficial when a block is never able to process frames faster than $1/t_{\min}$. Frame skipping with a sufficiently large $t_{\min}$ is in this case superior to preemption because it is less complex than preemption and avoids unnecessary processing steps for frames that will later be preempted, hence saving computational resources and energy in the video transmission chain.

Overall, the detailed examination of the trade-offs between frame skipping and preemption as well as the performance characteristics of the combination of frame skipping and preemption is an interesting direction for future research. For instance, we expect that enabling frame skipping in addition to preemption will strongly reduce the data rate and slightly increase the latency, comparable to switching from scenario 1 to scenario 2 in Table 6.1.

6.5 Comparison of the Experimental Results to the Theoretical Delay Model

The setup from scenario 1 from Table 6.1 is used to confirm the theoretical G2G delay model (3.11). In Equation (3.11), the encoding and decoding delays t_{Enc} and t_{Dec} are modeled by triangular distributions, which approximate the underlying limited Gaussian distributions that were observed in the measurements in Table 6.1. The triangular distributions span from minimum to maximum delay, with the triangle tip at the mean. For the encoder

these three points are 0.78 ms, 0.88 ms, and 1.08 ms. The maximum $t_{\mathrm{Enc}} = 1.67\,\mathrm{ms}$ from Table 6.1 were not used because this value was influenced by operating system interrupts, which are not included in the proposed model. Analogously, the triangle corner positions for the decoder are at 0.17 ms, 0.27 ms, and 0.54 ms, taken from Table 6.1 with the exception of the maximum value, which was again an outlier. The triangles are normalized to cover an area of one.

The remaining components comprising the camera processing, frame skipping, two color conversions, network, and display processing contribute an average delay of

$$\begin{aligned} t_{\mathrm{rem}} &= t_{\mathrm{CP}} + t_{\mathrm{FS}} + 2 \cdot t_{\mathrm{CSC}} + t_{\mathrm{Netw}} + t_{\mathrm{DP}} \\ &= (6.02 + 0.25 + 2 \cdot 0.32 + 0.02 + 5.69)\ \mathrm{ms} \\ &= 12.62\ \mathrm{ms}. \end{aligned} \tag{6.9}$$

For these blocks the mean measurement results from Table 6.1 are used, where t_{DP} is the sum of display processing and display pixel response. Buffering delays are not included because they are negligible for the $C = 1\,\mathrm{Gbit/s}$ channel. The remaining delay t_{rem} is assumed to be constant because there is little variance in its summands, see the rightmost column of Table 6.1. Hence the distribution $p_{\mathrm{rem}}(t) = \delta(t - t_{\mathrm{rem}})$ is a unit impulse at t_{rem} and zero otherwise.

Replacing the blocks that are already represented in Equation (6.9) for t_{rem} allows us to simplify model (3.11) to

$$t_{\mathrm{G2G}} \sim p(t) = (p_{\mathrm{CTS}} * p_{\mathrm{Enc}} * p_{\mathrm{Dec}} * p_{\mathrm{DTS}} * p_{\mathrm{Rem}})(t). \tag{6.10}$$

In Equation (6.10), the models (3.2) and (3.5) are utilized for the distributions of the temporal sampling processes p_{CTS} and p_{DTS}, respectively. The cumulative probability distribution of the G2G delay resulting from Equation (6.10) is depicted by the blue graph in Figure 6.9 for the given parameters. The orange graph in Figure 6.9 shows the cumulative G2G delay distribution of the first scenario in Table 6.1 without the outliers caused by operating system interrupts. Both the limits and the shape of the distributions match very well, which confirms the validity of the theoretical model. The theoretical model is consistently on the left side of the orange graph, but still, the graphs are close to each other. The consistent underestimation of G2G delay is caused by the deliberately neglected buffer delays, which introduce only a small error in the model, but simplify the model significantly. The operating system interrupts were not included because for many video coding systems, this is not an issue, since they either do not fully use the processor or have another implementation, such as a real-time operating system or a hardware implementation, which do not suffer from delay outliers due to interrupts and scheduling.

Note that the preceding model neglects buffering delays and approximates the network delay with a fixed value (that corresponds essentially to the transmission delay). This is reasonable for video communication systems with negligible buffer (queueing) delays as well as systems with buffers and enabled preemption. To model buffered video communication sys-

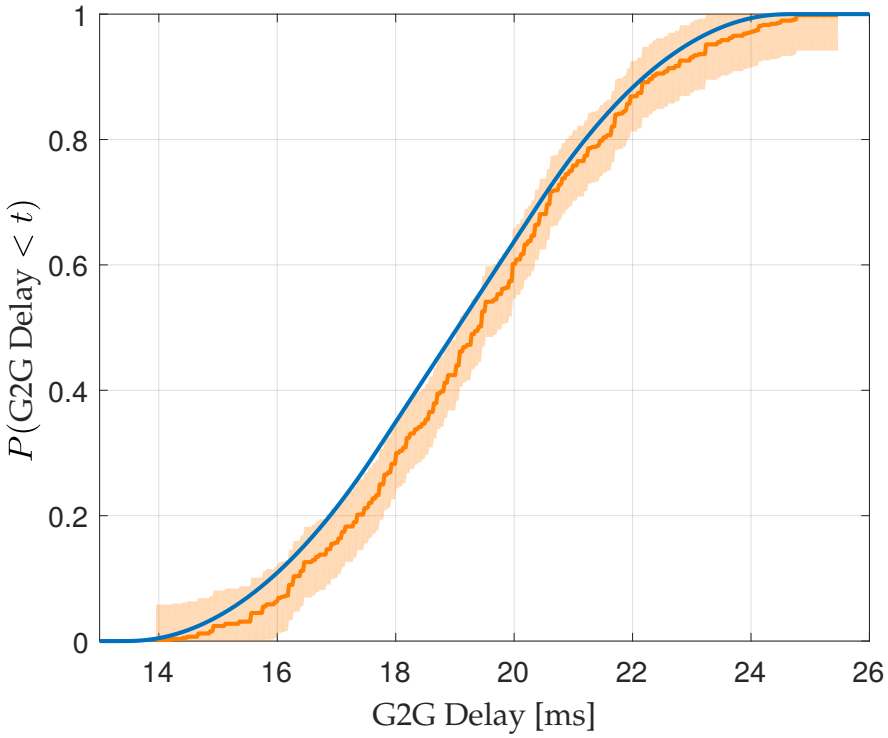


Figure 6.9: Cumulative distribution function obtained from the probabilistic analysis in Section 6.5 (blue) and empirical cumulative distribution function of the measurement samples from the prototype for full transmission (high fps, no skipping, Section 6.1.4.1) (orange). For the empirical distribution, the 95 %-confidence envelope is also given (adapted from [4], © 2018 IEEE).

tems without preemption, queueing models for the typically highly variable waiting times in buffers would have to be included.

6.6 Chapter Summary

In this chapter, we have seen that the greedy frame skipping algorithm shows a strong data rate reduction compared with high frame rate uniform sampling while keeping G2G latency low. To overcome the QoE degradation of the greedy algorithm, a perception-based frame skipping algorithm which performs frame skipping (creating an ASF video) such that it is imperceivable is proposed. Compared to a CSF video with lowest imperceivable sampling frequency, it achieves on average a 26 % data rate reduction. Combining both algorithms yields a video streaming system which exhibits low latency, imperceivable frame skipping with a short enough exposure time, and a considerable data rate reduction compared to the originally recorded video, on average approximately 63 %. Data rate reduction depends on video content and recording parameters: for a lower temporal sampling frequency of the camera, smaller data rate reductions can be expected. However, even compared to an ideally low sampling rate as in Section 6.2.3, the proposed algorithm achieves on average more than 20 % data rate reduction.

Additionally, the proposed preemption mechanism effectively avoids stalling or buffering of key frames, and the theoretical model from Section 3.6.2 has been confirmed.

Chapter 7

Conclusion and Future Work

Ultra low delay video communication enables many novel applications. For a human consumer, there is a great number of potential applications in the field of teleoperation, where low delay video communication is particularly advantageous when the human operator is wearing an HMD. Low delay video communication is equally important for machine vision. Applications such as interconnected autonomous driving, observing conveyor belts, or extracting state information from robot arms require a reliable low latency. In particular autonomous driving shows great potential for improving comfort and safety in street traffic, but low latency sensor signals are required for safe and fast control in real-world scenarios.

Despite all the possible applications and benefits, and despite recent advances towards the tactile internet, latency in video communication is a little researched topic. The reason for this is that in the past few years, we have seen significant advances in video recording and displaying technologies, and increasing computational resources in processing units. Only with the latest advances, widespread consumer application of remote control and visually controlled machines is possible. The great potential of these applications motivates low delay video communication, and consequently this thesis.

7.1 Conclusion

In this thesis, delay in video communication was defined and analyzed. A detailed block model divided the G2G delay of video communication into separate blocks such as camera, encoder, and network. The blocks and their delay contribution were further analyzed, which allowed us to formalize the delay of each block. The delay models of all blocks of a video communication chain in turn enabled the creation of a comprehensive theoretical model for G2G delay. The G2G delay model was further used with parameters from real world implementations to identify the blocks with the greatest G2G delay reduction potential. It was found that in a modern high-end video communication setup, the temporal sampling rates of camera and display should be increased to curtail G2G delay.

For the development and evaluation of low G2G delay video communication solutions, it is necessary to measure G2G delay. A survey found that existing solutions were insufficient. They lacked at least one of the following characteristics: performing automated measurements, non-intrusiveness, decorrelated measurements, affordable price, or precision. Hence,

a novel measurement system which utilizes an LED to create a visual event in the camera's FoV, and a PT to detect when the visual event is shown on the display was developed. Based on an Arduino platform, the system incorporates all beneficial characteristics mentioned previously. In addition, the flexibility of the system facilitates modifications to measure not only G2G delay, but also G2A delay, and any delay along the video communication chain. The measurement system was used to create a survey of G2G delays in state-of-the-art video communication applications and devices such as video conferencing, teleoperation, and smartphones. For many implementations, a G2G delay between 100 ms and 200 ms was measured. Such high G2G delays are prohibited by the latency constraints of future low delay video applications. Thus, the survey showed that the majority of modern video communication solutions is not ready for usage in applications such as autonomous driving. This fact further motivates the G2G delay reduction methods proposed in this thesis.

To reduce G2G delay, increasing sampling rates is a straightforward approach. However, this increases data rates and computational requirements of all involved blocks. The strategy proposed in this thesis is to skip irrelevant video frames, and only process video images containing significant information. Ideally, this frame skipping process is imperceptible to humans. To keep any frame skipping processes outside the domain of human perception, one part of this thesis investigated how humans perceive temporally sampled visual signals. In particular, there are two scenarios to be considered: if a video is recorded with a low temporal sampling frequency (frame rate), humans can easily distinguish the video from the original scene. If the video is recorded and displayed at an extremely high temporal sampling rate, humans are unable to distinguish video and original, in terms of temporal sampling. For frame skipping, it was necessary to define at which sampling frequency humans transition from perceiving temporal sampling to not perceiving it. This is called the critical sampling rate. The theory behind human perception leads us to expect that the critical sampling rate depends mainly on the exposure time of the recording camera as well as on the motion speed apparent in the video. Through a carefully designed psychophysical study, these presumptions were confirmed and a model for the critical sampling rate depending on exposure time and object speed was created.

The results obtained from delay analysis and perception analysis were utilized to propose various techniques to reduce delay in video communication. All following implementations rest on employing a high sampling frequency camera and performing subsequent temporal sub-sampling of the frames. In a first approach, perception-related results were disregarded and a greedy frame skipping algorithm was designed. The procedure decides to forward a frame as key frame if it contains significant novel content, or forward it as regular frame to keep frame rate above a predefined lower limit. Compared to video communication at the original frame rate, this algorithm achieves a reduction of a factor 40 in terms of compressed video data rate, while keeping latency extremely low. The prototype which employs greedy frame skipping achieved G2G and G2A delays of 21.2 ms and 11.5 ms, respectively. Still, the data rate reduction comes at a price: greedy frame skipping is perceivable in most scenarios, and degrades QoE.

To resolve this, the insights related to the perception of temporally sampled visual signals

were used. An algorithm around the model for critical sampling frequency was proposed, rendering the model applicable for real video sequences. The method computes the apparent motion speed, and maps the on-screen speed to a speed relative to the video consumer's perspective. The resulting critical sampling rate is finally quantized to one of the available sampling rates, which are integer fractions of the original sampling frequency. This procedure achieves an average data rate reduction of 26 % compared to a video which is sampled with constant frequency, chosen as low as possible with humans being unable to perceive temporal sampling. This method, however, does not relate to low latency video communication and thus does not guarantee to meet any latency constraints.

This is why the greedy and the perceptual methods were merged. The fundamental frame skipping decision still comes from the greedy algorithm, but the perceptual algorithm sets the lower frame rate limit. Using this setup, the low delay from the greedy algorithm is guaranteed, and frame skipping is imperceptible, thanks to the perceptual algorithm. The merged algorithms achieve an average data rate reduction of 63 % compared to the video with full frame rate.

It was observed that key frames are at times stalled, when processing of a preceding regular frame is not yet finished in a block. Stalling increases latency, therefore a mechanism called preemption was proposed to avoid stalling of key frames in the encoder buffer. If a key frame arrives in the buffer, the queue of waiting regular frames is flushed, enabling direct forwarding of the key frame. Experimental results showed a G2G delay reduction of half an order of magnitude if the transmission channel is causing considerable buffering and video rate control is disabled.

Finally, the prototypes and the delay measurement system were utilized to confirm the validity of the theoretical G2G delay model. It was shown that the model closely matches measurements with various parameters, which is why the model can be assumed to be correct.

In short, this thesis has analyzed delay in video communication, proposed a system to measure G2G delay in video communication, and found a model for choosing the frame rate at which humans can just perceive or not perceive temporal sampling. These insights have been used to propose a frame skipping algorithm that combines a greedy decision with perceptual considerations. The proposed techniques achieved a significant G2G latency reduction, while keeping the increase in data rate closely constrained.

7.2 Future Work

This manuscript has not answered all questions related to low latency video communication. Many open research questions remain, and a few of them shall be pointed out in this section.

1. The greedy frame skipping algorithm can be refined, for example to include adaptive thresholds. The thresholds for image difference and minimum and maximum time between two frames could adapt to the video content, available channel rate, or the computational load of involved blocks such as coders or buffers.

2. Cut-through operation could be implemented in many blocks of a video processing chain, and the corresponding reduction in G2G latency could be evaluated.

3. The perception of temporal sampling can be researched in more detail. In this thesis, a model for a representative luminance, contrast and for the worst case video content was found. Worst case in this context means the video content with the highest spatial frequency, thus the quickest color intensity transition. The model can be extended to take other contrast situations, and the influence of the spatial frequencies visible in the video into account. A model extended in this way would allow a closer fit of the modeled to the true critical frame rate, and therefore greater frame and data rate reductions.

4. The insights gained from the perception of temporal sampling can be applied to other fields than low latency video communication. For example in computer graphics such as gaming or augmented reality, QoE can be improved by blurring content or adapting the temporal sampling frequency to the current content.

5. The model for the perception of temporal sampling gives the critical sampling rate at which temporal sampling is perceivable 50 % of the time. In the experiments of this thesis, a positive frequency offset was added to make temporal sampling imperceptible. Conversely, a study could investigate the effects of a negative frequency offset on QoE, or in general of a range of offset values.

6. Further research can examine how different visual event classes such as rapidly changing illumination and continuous motion affect frame skipping. This can lead to frame skipping methods that adapt to visual event classes.

7. The effect of frame skipping on machine vision algorithms can be investigated. If necessary, application-specific frame skipping algorithms can be proposed.

8. Finally, a study that summarizes scientific work about human visual latency perception limits would be highly useful. Depending on the application, such as inking on a touchscreen, wearing an HMD, or remotely controlling a slow robot, humans show different latency perception thresholds. A survey of latency perception thresholds could help to define G2G latency targets for future low delay video research.

Bibliography

Publications by the author

Journal publications

[1] B. Cizmeci, X. Xu, R. Chaudhari, C. Bachhuber, N. Alt, and E. Steinbach, "A multiplexing scheme for multimodal teleoperation," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 13, no. 2, p. 21, 2017. DOI: 10.1145/3063594.

[2] T. Aykut, M. Karimi, C. Burgmair, A. Finkenzeller, C. Bachhuber, and E. Steinbach, "Delay compensation for a telepresence system with 3D 360° vision based on deep head motion prediction and dynamic FoV adaptation," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4343–4350, Oct. 2018. DOI: 10.1109/LRA.2018.2864359.

[3] C. Bachhuber, A. Bhardwaj, R. Pries, and E. Steinbach, "On the minimum perceptual temporal video sampling rate and its application to adaptive frame skipping," *IEEE Transactions on Circuits and Systems for Video Technology (early access)*, 2018. DOI: 10.1109/TCSVT.2018.2870256.

[4] C. Bachhuber, E. Steinbach, M. Freundl, and M. Reisslein, "On the minimization of glass-to-glass and glass-to-algorithm delay in video communication," *IEEE Transactions on Multimedia*, vol. 20, no. 1, pp. 238–252, 2018. DOI: 10.1109/TMM.2017.2726189.

Conference publications

[5] C. Bachhuber, S. Conrady, M. Schütz, and E. Steinbach, "A testbed for vision-based networked control systems," in *11th International Conference on Computer Vision Systems*, Springer, 2017, pp. 26–36. DOI: 10.1007/978-g3-g319-g68345-g4_3.

[6] C. Bachhuber and E. Steinbach, "Are today's video communication solutions ready for the tactile internet?" In *Wireless Communications and Networking Conference Workshops (WCNCW)*, IEEE, 2017, pp. 1–6. DOI: 10.1109/WCNCW.2017.7919060.

[7] C. Bachhuber and E. 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. DOI: 10.1109/ICIP.2016.7532735.

General publications

[8] F. Chaumette and S. Hutchinson, "Visual servo control, part II: Advanced approaches," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[9] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Conference on Computer Vision and Pattern Recogn. (CVPR)*, IEEE, 2012, pp. 3354–3361.

[10] K. Okumura, H. Oku, and M. Ishikawa, "High-speed gaze controller for millisecond-order pan/tilt camera," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2011, pp. 6186–6191.

[11] N. Alt, C. Claus, and W. Stechele, "Hardware/software architecture of an algorithm for vision-based real-time vehicle detection in dark environments," in *Conference on Design, Automation and Test in Europe*, ACM, 2008, pp. 176–181.

[12] N. Alt and E. Steinbach, "Visuo-haptic sensor for force measurement and contact shape estimation," in *International Symposium on Haptic Audio Visual Environments and Games (HAVE)*, IEEE, 2013, pp. 24–28.

[13] G. C. Walsh, H. Ye, and L. G. Bushnell, "Stability analysis of networked control systems," *IEEE Transactions Control Systems Technology*, vol. 10, no. 3, pp. 438–446, May 2002.

[14] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control systems," *IEEE Control Systems*, vol. 21, no. 1, pp. 84–99, 2001.

[15] Y. Shi and B. Yu, "Output feedback stabilization of networked control systems with random delays modeled by Markov chains," *IEEE Transactions on Automatic Control*, vol. 54, no. 7, pp. 1668–1674, 2009.

[16] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceedings of the IEEE*, vol. 95, no. 1, p. 138, Jan. 2007.

[17] D. Yue, Q.-L. Han, and C. Peng, "State feedback controller design of networked control systems," in *IEEE International Conference on Control Applications*, vol. 1, 2004, pp. 242–247.

[18] Y. Tipsuwan and M.-Y. Chow, "Control methodologies in networked control systems," *Control Engineering Practice*, vol. 11, no. 10, pp. 1099–1111, 2003.

[19] L. Zhang, H. Gao, and O. Kaynak, "Network-induced constraints in networked control systems—a survey," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 403–416, 2013.

[20] R. Yang, G.-P. Liu, P. Shi, C. Thomas, and M. V. Basin, "Predictive output feedback control for networked control systems," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 1, pp. 512–520, 2014.

[21] M. B. Cloosterman, N. Van de Wouw, W. Heemels, and H. Nijmeijer, "Stability of networked control systems with uncertain time-varying delays," *IEEE Transactions on Automatic Control*, vol. 54, no. 7, pp. 1575–1580, Jul. 2009.

[22] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5g," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74–80, 2014.

[23] G. P. Fettweis, "The tactile internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, 2014.

[24] M. Annett, A. Ng, P. Dietz, W. F. Bischof, and A. Gupta, "How low should we go?: Understanding the perception of latency while inking," in *ACM Proceedings of the Graphics Interface*, 2014, pp. 167–174.

[25] R. Jota, A. Ng, P. Dietz, and D. Wigdor, "How fast is fast enough?: A study of the effects of latency in direct-touch pointing tasks," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, 2013, pp. 2291–2300.

[26] K. Mania, B. D. Adelstein, S. R. Ellis, and M. I. Hill, "Perceptual sensitivity to head tracking latency in virtual environments with varying degrees of scene complexity," in *ACM Symposium on Applied Perception in Graphics and Visualization*, 2004, pp. 39–47.

[27] S. Kawamura and R. Kijima, "Effect of head mounted display latency on human stability during quiescent standing on one foot," in *IEEE Proceedings in Virtual Reality (VR)*, 2016, pp. 199–200.

[28] A. Ng, J. Lepinski, D. Wigdor, S. Sanders, and P. Dietz, "Designing for low-latency direct-touch input," in *Proceedings of the 25th annual ACM symposium on User interface software and technology*, ACM, 2012, pp. 453–464.

[29] R. Rutschmann, "Perception of temporal order and relative visual latency," *Science*, vol. 152, no. 3725, pp. 1099–1101, 1966.

[30] I. J. Hirsh and C. E. Sherrick Jr, "Perceived order in different sense modalities.," *Journal of experimental psychology*, vol. 62, no. 5, p. 423, 1961.

[31] D. N. Robinson, "Visual discrimination of temporal order," *Science*, vol. 156, no. 3779, pp. 1263–1264, 1967.

[32] M. Yanoff, B. S. Fine, and J. D. M. Gass, *Ocular pathology*. Mosby-Wolfe London, 1996.

[33] Y. Bababekova, M. Rosenfield, J. E. Hue, and R. R. Huang, "Font size and viewing distance of handheld smart phones," *Optometry and Vision Science*, vol. 88, no. 7, pp. 795–797, 2011.

[34] G. Johansson, "Visual motion perception," *Scientific American*, vol. 232, no. 6, pp. 76–89, 1975.

[35] A. B. Watson, A. J. Ahumada, and J. E. Farrell, "Window of visibility: A psychophysical theory of fidelity in time-sampled visual motion displays," *Journal of the Optical Society of America A*, vol. 3, no. 3, pp. 300–307, 1986.

[36] S. Hecht and E. L. Smith, "Intermittent stimulation by light: Vi. area and the relation between critical frequency and intensity," *The Journal of general physiology*, vol. 19, no. 6, pp. 979–989, 1936.

[37] A. M. Derrington, H. A. Allen, and L. S. Delicato, "Visual mechanisms of motion analysis and motion perception," *Annual Review of Psychology*, vol. 55, pp. 181–205, 2004.

[38] T. Ledgeway and A. T. Smith, "Evidence for separate motion-detecting mechanisms for first-and second-order motion in human vision," *Vision research*, vol. 34, no. 20, pp. 2727–2740, 1994.

[39] C. Chubb and G. Sperling, "Drift-balanced random stimuli: A general basis for studying non-fourier motion perception," *Journal of the Optical Society of America A*, vol. 5, no. 11, pp. 1986–2007, 1988.

[40] A. E. Seiffert and P. Cavanagh, "Position displacement, not velocity, is the cue to motion detection of second-order stimuli," *Vision research*, vol. 38, no. 22, pp. 3569–3582, 1998.

[41] E. H. Adelson and J. R. Bergen, "Spatiotemporal energy models for the perception of motion," *Journal of the Optical Society of America A*, vol. 2, no. 2, pp. 284–299, 1985.

[42] A. B. Watson, "High frame rates and human vision: A view through the window of visibility," *SMPTE Motion Imaging Journal*, vol. 122, no. 2, pp. 18–32, 2013.

[43] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120 dB 15us latency asynchronous temporal contrast vision sensor," *Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[44] D. Honegger, H. Oleynikova, and M. Pollefeys, "Real-time and low latency embedded computer vision hardware based on a combination of FPGA and mobile CPU," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2014, pp. 4930–4935.

[45] J. Fung and S. Mann, "Computer vision signal processing on graphics processing units," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, IEEE, vol. 5, 2004, pp. V–93.

[46] P. D. Gowdy and C. M. Cicerone, "The spatial arrangement of the L and M cones in the central fovea of the living human eye," *Vision research*, vol. 38, no. 17, pp. 2575–2589, 1998.

[47] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[48] G. J. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, *et al.*, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[49] I.-K. Kim, J. Min, T. Lee, W.-J. Han, and J. Park, "Block partitioning structure in the HEVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1697–1706, 2012.

[50] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1685–1696, 2012.

[51] X. Ji, D. Zhao, W. Gao, Q. Huang, S. Ma, and Y. Lu, "New bi-prediction techniques for B pictures coding," in *International Conference on Multimedia and Expo*, IEEE, vol. 1, 2004, pp. 101–104.

[52] Á. Huszák and S. Imre, "Analysing GOP structure and packet loss effects on error propagation in mpeg-4 video streams," in *International Symposium on Communications, Control and Signal Processing (ISCCSP)*, IEEE, 2010, pp. 1–5.

[53] C.-Y. Chang, C.-F. Chou, D.-Y. Chan, T. Lin, and M.-H. Chen, "Accurate bitrate model and greedy-based rate controller for low delay video transmission," *IEEE Systems Journal*, vol. 6, no. 3, pp. 414–425, Sep. 2012.

[54] M. Gao, B. Cizmeci, M. Eiler, E. Steinbach, D. Zhao, and W. Gao, "Macroblock level rate control for low delay H.264/AVC based video communication," in *IEEE 21st International Packet Video Workshop (PV), 2015*, Jun. 2015.

[55] H. Lin, X. He, Q.-Z. Teng, W. Fu, and S. Xiong, "Adaptive bit allocation scheme for extremely low-delay intraframe rate control in high efficiency video coding," *SPIE Journal of Electronic Imaging*, vol. 25, no. 4, pp. 043 008–1–043008–13, Jul. 2016.

[56] P. Navakitkanok and S. Aramvith, "Improved rate control for advanced video coding (AVC) standard under low delay constraint," in *IEEE International Conference on Information Technology: Coding and Computing*, vol. 2, 2004, pp. 664–668.

[57] J. Ribas-Corbera and S. Lei, "Rate control in DCT video coding for low-delay communications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 1, pp. 172–185, Feb. 1999.

[58] S. Sanz-Rodriguez, T. Mayer, M. Alvarez-Mesa, and T. Schierl, "A low-complexity parallel-friendly rate control algorithm for ultra-low delay high definition video coding," in *IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2013, pp. 1–4.

[59] F. Zhang and E. Steinbach, "Improved ρ-domain rate control with accurate header size estimation," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2011, pp. 813–816.

[60] F. Bossen *et al.*, "Common test conditions and software reference configurations," *Joint Collaborative Team on Video Coding*, vol. 12, 2013.

[61] L. Merritt and R. Vanam, "X264: A high performance H.264/AVC encoder," *https://www.videolan.org/developers/x264.html*, Accessed: 26.09.2018.

[62] *X265 - a free open-source H.265 encoder*, http://www.videolan.org/developers/x265.html, Accessed: 26.09.2018.

[63] R. M. Schreier, A. M.T. I. Rahman, G. Krishnamurthy, and A. Rothermel, "Architecture analysis for low-delay video coding," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2006, pp. 2053–2056.

[64] R. M. Schreier and A. Rothermel, "A latency analysis on H.264 video transmission systems," in *IEEE International Conference on Consumer Electronics*, 2008, pp. 1–2.

[65] A. Vinel, E. Belyaev, K. Egiazarian, and Y. Koucheryavy, "An overtaking assistance system based on joint beaconing and real-time video transmission," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 5, pp. 2319–2329, Jun. 2012.

[66] R. Song, Y.-L. Wang, Y. Han, and Y.-S. Li, "Statistically uniform intra-block refresh algorithm for very low delay video communication," *Journal of Zhejiang University SCIENCE C*, vol. 14, no. 5, pp. 374–382, May 2013.

[67] M. Baldi and Y. Ofek, "End-to-end delay analysis of videoconferencing over packet-switched networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, pp. 479–492, Aug. 2000.

[68] P. Holub, J. Matela, M. Pulec, and M. Šrom, "Ultragrid: Low-latency high-quality video transmissions on commodity hardware," in *Proceedings of the ACM International Conference on Multimedia*, 2012, pp. 1457–1460.

[69] T. Inatsuki, M. Matsuura, K. Morinaga, H. Tsutsui, and Y. Miyanaga, "An FPGA implementation of low-latency video transmission system using lossless and near-lossless line-based compression," in *IEEE International Conference on Digital Signal Processing (DSP)*, 2015, pp. 1062–1066.

[70] M. U. K. Khan, J. M. Borrmann, L. Bauer, M. Shafique, and J. Henkel, "An H.264 Quad-Full HD low-latency intra video encoder," in *Conference on Design, Automation and Test in Europe (DATE)*, 2013, pp. 115–120.

[71] R. Hill, C. Madden, A. v. d. Hengel, H. Detmold, and A. Dick, "Measuring latency for video surveillance systems," in *Digital Image Computing: Techniques and Applications, 2009. DICTA'09.*, IEEE, 2009, pp. 89–95.

[72] J. MacCormick, "Video chat with multiple cameras," in *Proceedings of the 2013 conference on Computer supported cooperative work companion*, ACM, 2013, pp. 195–198.

[73] M. C. Jacobs, M. A. Livingston, and A. State, "Managing latency in complex augmented reality systems," in *Proceedings of the 1997 symposium on Interactive 3D graphics*, ACM, 1997, 49–ff.

[74] T. Sielhorst, W. Sa, A. Khamene, F. Sauer, and N. Navab, "Measurement of absolute latency for video see through augmented reality," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, IEEE Computer Society, 2007, pp. 1–4.

[75] O. Boyaci, A. Forte, S. A. Baset, and H. Schulzrinne, "vDelay: A tool to measure capture-to-display latency and frame rate," in *Multimedia, 2009. ISM'09. 11th IEEE International Symposium on*, IEEE, 2009, pp. 194–200.

[76] J. Jansen and D. C. Bulterman, "User-centric video delay measurements," in *Proceedings of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ACM, 2013, pp. 37–42.

[77] M. van der Schaar and Y. Andreopoulos, "Rate-distortion-complexity modeling for network and receiver aware adaptation," *IEEE Transactions on Multimedia*, vol. 7, no. 3, pp. 471–479, 2005.

[78] K. Choi and E. S. Jang, "Early TU decision method for fast video encoding in high efficiency video coding," *Electronics letters*, vol. 48, no. 12, pp. 689–691, 2012.

[79] K. Choi and E. S. Jang, "Fast coding unit decision method based on coding tree pruning for high efficiency video coding," *Optical Engineering*, vol. 51, no. 3, p. 030 502, 2012.

[80] G. Chen, Z. Pei, L. Sun, Z. Liu, and T. Ikenaga, "Fast intra prediction for HEVC based on pixel gradient statistics and mode refinement," in *International Conference on Signal and Information Processing (ChinaSIP), IEEE China Summit*, IEEE, 2013, pp. 514–517.

[81] W. Jiang, H. Ma, and Y. Chen, "Gradient based fast mode decision algorithm for intra prediction in HEVC," in *International Conference on Consumer Electronics, Communications and Networks (CECNet)*, IEEE, 2012, pp. 1836–1840.

[82] M. U. K. Khan, M. Shafique, and J. Henkel, "An adaptive complexity reduction scheme with fast prediction unit decision for HEVC intra encoding," in *International Conference on Image Processing (ICIP)*, IEEE, 2013, pp. 1578–1582.

[83] J. Støttrup-Andersen, S. Forchhammer, and S. M. Aghito, "Rate-distortion-complexity optimization of fast motion estimation in H.264/MPEG-4 AVC," in *International Conference on Image Processing (ICIP)*, IEEE, vol. 1, 2004, pp. 111–114.

[84] Y. Hu, Q. Li, S. Ma, and C.-C. J. Kuo, "Joint rate-distortion-complexity optimization for H.264 motion search," in *IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2006, pp. 1949–1952.

[85] J. Vanne, M. Viitanen, T. D. Hamalainen, and A. Hallapuro, "Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1885–1898, 2012.

[86] J. Vanne, M. Viitanen, and T. D. Hämäläinen, "Efficient mode decision schemes for HEVC inter prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 9, pp. 1579–1593, 2014.

[87] G. Cote, B. Erol, M. Gallant, and F. Kossentini, "H.263+: Video coding at low bit rates," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 849–866, 1998.

[88] B. E. Bayer, *Color imaging array*, US Patent 3,971,065, Jul. 1976.

[89] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Computer Networks (1976)*, vol. 3, no. 4, pp. 267–286, 1979.

[90] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation method: An approach to store-and-forward communication network design," *Networks*, vol. 3, no. 2, pp. 97–133, 1973.

[91] P. E. Jorgensen, "The road to reality: A complete guide to the laws of the universe," *The Mathematical Intelligencer*, vol. 28, no. 3, pp. 59–61, 2006.

[92] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[93] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418.

[94] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[95] M. A. Farooque and J. S. Rohankar, "Survey on various noises and techniques for denoising the color image," *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, vol. 2, no. 11, pp. 217–221, 2013.

[96] H. Grassmann, "Zur Theorie der Farbenmischung," *Annalen der Physik*, vol. 165, no. 5, pp. 69–84, 1853.

[97] K. L. Kaiser, *Transmission lines, matching, and crosstalk*. CRC Press, 2005.

[98] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[99] A. M. Sheikh, A. Fiandrotti, and E. Magli, "Distributed scheduling for low-delay and loss-resilient media streaming with network coding," *IEEE Transactions on Multimedia*, vol. 16, no. 8, pp. 2294–2306, Dec. 2014.

[100] T. H. Szymanski, "An ultra-low-latency guaranteed-rate internet for cloud services," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 123–136, 2016.

[101] J. Wu, C. Yuen, N.-M. Cheung, and J. Chen, "Delay-constrained high definition video transmission in heterogeneous wireless networks with multi-homed terminals," *IEEE Transactions on Mobile Computing*, vol. 15, no. 3, pp. 641–655, 2016.

[102] S. P. Weber, X. Yang, J. G. Andrews, and G. De Veciana, "Transmission capacity of wireless ad hoc networks with outage constraints," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4091–4102, 2005.

[103] M. Schwartz, *Telecommunication networks: protocols, modeling and analysis*. Addison-Wesley Reading, MA, 1987, vol. 7.

[104] S. Lee, S. Lee, H. Kim, J. Kim, S. Hong, Y. Jeong, C. Park, Y. Choi, J. Lee, J. Koh, *et al.*, "29.2: 18.1" Ultra-FFS TFT-LCD with super image quality and fast response time," in *SID Symposium Digest of Technical Papers*, Wiley Online Library, vol. 32, 2001, pp. 484–487.

[105] A. K. Jain, *Fundamentals of digital image processing*, 1st ed. Prentice Hall, Englewood Cliffs, NJ, 1989.

[106] W. Feller, *An Introduction to Probability Theory and Its Applications*, 3rd ed. John Wiley & Sons, Inc., 1968, vol. 1.

[107] Y. Watanabe, G. Narita, S. Tatsuno, T. Yuasa, K. Sumino, and M. Ishikawa, "High-speed 8-bit image projector at 1,000 fps with 3 ms delay," in *Proceedings of the International Display Workshops*, 2015, pp. 1064–1065.

[108] I. Ishii, T. Tatebe, Q. Gu, Y. Moriue, T. Takaki, and K. Tajima, "2000 fps real-time vision system with high-frame-rate video recording," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2010, pp. 1536–1541.

[109] Y. Watanabe, T. Komuro, and M. Ishikawa, "955-fps real-time shape measurement of a moving/deforming object using high-speed vision for numerous-point analysis," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2007, pp. 3192–3197.

[110] T. Lee, "Effect of junction capacitance on the rise time of LED's and on the turn-on delay of injection lasers," *Bell Systems Technical Journal*, vol. 54, no. 1, pp. 53–68, 1975.

[111] V. N. Tran, R. Stuart, and H. Bhavsar, "Phototransistor switching time analysis," *California Eastern Laboratories*, 2009.

[112] J.-K. Sheu, S.-J. Chang, C. Kuo, Y.-K. Su, L. Wu, Y. Lin, W. Lai, J. Tsai, G.-C. Chi, and R. Wu, "White-light emission from near UV InGaN-GaN LED chip precoated with blue/green/red phosphors," *IEEE Photonics Technology Letters*, vol. 15, no. 1, pp. 18–20, 2003.

[113] Y. Lu, Y. Zhao, F. Kuipers, and P. Van Mieghem, "Measurement study of multi-party video conferencing," in *International Conference on Research in Networking*, Springer, 2010, pp. 96–108.

[114] Y. Xu, C. Yu, J. Li, and Y. Liu, "Video telephony for end-consumers: Measurement study of Google+, iChat, and Skype," in *Proceedings of the 2012 ACM conference on Internet measurement conference*, ACM, 2012, pp. 371–384.

[115] M. R. Mine, "Characterization of end-to-end delays in head-mounted display systems," *The University of North Carolina at Chapel Hill, TR93-001*, 1993.

[116] J. G. Robson, "Spatial and temporal contrast-sensitivity functions of the visual system," *Journal of the Optical Society of America A*, vol. 56, no. 8, pp. 1141–1142, 1966.

[117] R. N. Bracewell, *The Fourier Transform and its Applications*, 3rd ed. McGraw-Hill New York, 1986.

[118] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*, 1st ed. O'Reilly Media, Inc., 2008.

[119] D. Laming and J. Laming, "F. Hegelmaier: On memory for the length of a line," *Psychological research*, vol. 54, no. 4, pp. 233–239, 1992.

[120] H. Levitt, "Transformed up-down methods in psychoacoustics," *The Journal of the Acoustical society of America*, vol. 49, no. 2B, pp. 467–477, 1971.

[121] T. N. Cornsweet, "The staircase-method in psychophysics," *The American Journal of Psychology*, vol. 75, no. 3, pp. 485–491, 1962.

[122] M. R. Leek, "Adaptive procedures in psychophysical research," *Attention, Perception, & Psychophysics*, vol. 63, no. 8, pp. 1279–1292, 2001.

[123] F. Barbagli, K. Salisbury, C. Ho, C. Spence, and H. Z. Tan, "Haptic discrimination of force direction and the influence of visual information," *ACM Transactions on Applied Perception (TAP)*, vol. 3, no. 2, pp. 125–135, 2006.

[124] M. J. Cox and A. M. Derrington, "The analysis of motion of two-dimensional patterns: Do fourier components provide the first stage?" *Vision research*, vol. 34, no. 1, pp. 59–72, 1994.

[125] H. R. Wilson, V. P. Ferrera, and C. Yo, "A psychophysically motivated model for two-dimensional motion perception," *Visual neuroscience*, vol. 9, no. 1, pp. 79–97, 1992.

[126] R. Gueorguieva and J. H. Krystal, "Move over ANOVA: Progress in analyzing repeated-measures data and its reflection in papers published in the archives of general psychiatry," *Archives of general psychiatry*, vol. 61, no. 3, pp. 310–317, 2004.

[127] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[128] Y.-F. Ou, T. Liu, Z. Zhao, Z. Ma, and Y. Wang, "Modeling the impact of frame rate on perceptual quality of video," in *IEEE International Conference on Image Processing*, Oct. 2008, pp. 689–692.

[129] Y.-F. Ou, Z. Ma, T. Liu, and Y. Wang, "Perceptual quality assessment of video considering both frame rate and quantization artifacts," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 3, pp. 286–298, Mar. 2011.

[130] Z. Lu, W. Lin, B. C. Seng, S. Kato, S. Yao, E. Ong, and X. K. Yang, "Measuring the negative impact of frame dropping on perceptual visual quality," in *Human Vision and Electronic Imaging X*, 2005, pp. 554–562.

[131] P. Massart, "The tight constant in the Dvoretzky-Kiefer-Wolfowitz inequality," *The Annals of Probability*, vol. 18, no. 3, pp. 1269–1283, Jul. 1990.

[132] A. Dvoretzky, J. Kiefer, and J. Wolfowitz, "Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator," *The Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 642–669, Sep. 1956.

[133] T. Camus, *Real-time optical flow*. Society of Manufacturing Engineers, 1994.

[134] J. Díaz, E. Ros, F. Pelayo, E. M. Ortigosa, and S. Mota, "FPGA-based real-time optical-flow system," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 2, pp. 274–279, 2006.

[135] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *ACM Transactions on Computer Systems (TOCS)*, vol. 18, no. 3, pp. 263–297, Aug. 2000.

List of Figures

List of Tables