On the Minimization of Glass-to-Glass and Glass-to-Algorithm Delay in Video Communication

Christoph Bachhuber, Eckehard Steinbach, *Fellow, IEEE*, Martin Freundl, and Martin Reisslein, *Fellow, IEEE*

***Abstract*—Video cameras are increasingly used to provide real-time feedback in automatic control systems, such as autonomous driving and robotics systems. For such highly dynamic applications, the glass-to-glass (G2G) and glass-to-algorithm (G2A) latencies are critical. In this paper, we analyze the latencies in a point-to-point video transmission system and propose novel frame skipping and preemption approaches to reduce the G2G and G2A delays. We implement the proposed approaches in a prototype that shows significantly reduced G2G and G2A latencies as well as reduced transmission bitrate requirements compared with traditional video transmission schemes. In our low-delay video communication prototype, a VGA resolution video is transmitted with average G2G and G2A delays of 21.2 and 11.5 ms, respectively, with off-the-shelf hardware.**

***Index Terms*—Delay analysis, delay measurement, frame skipping, preemption, prototype system.**

I. Introduction

A. Motivation

CAMERAS have the potential to replace many sensors for process control [2], [3]. 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 [4] and mechanical sensors. Another advantage of video camera sensing is that a single camera can replace multiple sensors, such as a camera observing a robot arm, substituting multiple angle and force sensors [5]. 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.

There are still many challenges 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 [6] deteriorates the stability [7], and when the latency exceeds an allowable limit, leads to instability [8]. Researchers thoroughly investigated the effect of latency on NCS stability [9], and proposed many algorithms to compensate delay [10]–[13]. Creating a low-delay sensor-to-controller transmission is hence advantageous [14] for stabilizing a system and enables control of more dynamic applications.

For visually controlled systems, the challenge corresponding to 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, see Fig. 1. 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 50–80 ms. In contrast, the end-to-end (E2E) delays of applications envisioned for the “Tactile Internet” should be very low (e.g., < 10 ms), in extreme cases <1 ms [15]. 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. Related research on network transmission systems is progressing towards reducing other E2E delay components, see for instance [16]–[26].

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, see Fig. 1. More specifically, the G2G delay is the time period from the time instant when the event’s photons first pass through the camera lens glass to the time instant when the corresponding photons pass through the display glass.

Manuscript received January 30, 2017; revised May 12, 2017 and June 14, 2017; accepted July 9, 2017. Date of publication July 12, 2017; date of current version December 14, 2017. This work was supported in part by a Friedrich Wilhelm Bessel Research Award from the Alexander von Humboldt Foundation. This paper was presented in part at the IEEE International Conference on Image Processing, Phoenix, AZ, USA, September 2016. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Xiaoqing Zhu. *(Corresponding author: Eckehard Steinbach.)*

C. Bachhuber, E. Steinbach, and M. Freundl are with the Chair of Media Technology, Technical University of Munich, Munich 80333, Germany (e-mail: christoph.bachhuber@tum.de; eckehard.steinbach@tum.de; martin.freundl@tum.de).

M. Reisslein is with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287-5706 USA (e-mail: reisslein@asu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2017.2726189

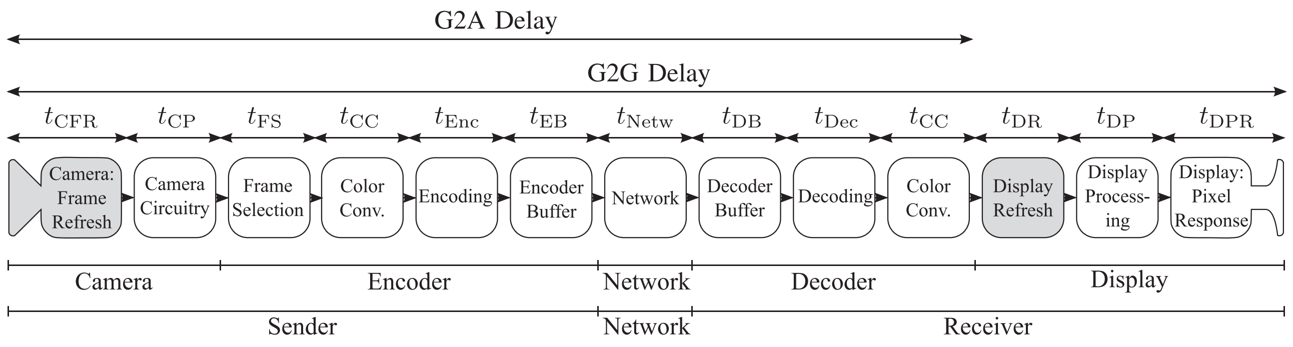

Fig. 1. Our model of a video communication chain. Blocks with gray background represent a non-processing delay caused by the physical setup of the chain. White blocks represent processing delays.

Depending on the application, humans can visually perceive latencies as low as 6 ms for inking on a touchscreen [27], or around 50 ms for interactive applications, such as gaming or graphics tools [28]. In general, the just noticeable difference (JND) for visual delay differentiation by trained observers lies between 8 ms and 17 ms [29]. Kawamura *et al.* [30] 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, Kawamura *et al.* found that a 1 ms delay setup (realized via pose prediction) gave superior task performance compared to the 23 ms delay case. Thus, we conclude that humans can not only perceive latencies below 66 ms, but latency reductions down to the JND also benefit task performance.

In this study, we investigate approaches for achieving very low latency for real-time video transmission, narrowing the gap to the JND. Our system achieves a mean G2G delay of 21.2 ms. Thus, our system achieves significantly shorter latencies than state-of-the-art systems, which feature G2G delays between 120 ms and 386 ms for conversational applications, and approximately 50 ms for teleoperation and augmented reality products [31]. We proceed to review the related literature and then outline how our contributions in this paper advance low-latency video transmission.

B. Related Work

A few fields relate to this paper: delay analysis of video transmission systems, low latency video transmission systems, as well as keyframe selection and buffer analysis of video transmission applications.

Baldi and Ofek [32] analyzed the E2E delay of videoconferencing applications over packet switched networks, but they did not include the delays due to frame refreshes in the camera and display. Refresh processes in the camera (resp. display) sample (resp. update) the scene at constant time intervals, which may cause delays for events not synchronized to those intervals. Furthermore, Baldi and Ofek did not analyze any delays after the decoding unit, but lumped all these delays together as a processing delay. Vinel *et al.* [33] presented a coarse delay analysis as part of their overtaking assistance system. Vinel *et al.* considered five delay contributors: encoder, transmitter buffer, channel, receiver buffer, and decoder. The coarse delay analysis did not consider the camera refresh delay, the camera processing delay, nor any delays related to the display.

Song *et al.* [34] have conducted a detailed analysis of a video encoder that uses statistically uniform intra-block refresh. They concluded that the worst case (maximum) E2E delay is caused by the maximum size frame. Song *et al.* consequently proposed an intra refresh coding scheme, in which alternating parts of the image are intra coded and the remainder is inter-coded. Their analysis does not include any delay from the recording camera or processing delay of the display. Furthermore, Song *et al.* did not conduct a statistical analysis of the delays. Schreier *et al.* [35], [36] analyzed the latency from the input of the encoder to the output of the decoder of different H.264 coding modes. In their analysis, the largest delay contributors are the coders as well as the transmitter and receiver buffers. The analysis does not include the camera and display delays, i.e., does not address the G2G delay.

Video communication setups with low latency have previously been researched by Holub *et al.* [37] who implemented low-latency video transmission over 10 Gigabit Ethernet (10 GE) networks. Holub *et al.* leveraged the processing power of graphics processing units, achieving an “end-to-end latency” of 2.5 to 5 frames, corresponding to 75 to 166 ms for a video with 30 fps. However, the end-to-end latency is not defined in detail and the delay measurements are not described in detail in [37]. It appears that the study in [37] performed a simple calculation of the delays caused by processing and buffering, but did not measure G2G delay. Other studies [38], [39] have optimized the encoder to achieve low latency, but ignored the delays of several other components, e.g., the display and camera.

Keyframe selection or frame skipping describes the strategy of not transmitting, storing, or showing every video frame produced by a camera. Keyframe selection is used to reduce the bitrate of the video. Liu *et al.* [40] drop insignificant frames of often static lecture videos, whilst not considering tight delay constraints. Doulamis *et al.* [41] utilize a neural network to predict key frames in real time to cope with low and variable rate connections. Other approaches that are not applicable to our low-delay use case find positions for independently coded frames by optimizing rate and distortion over an entire video [42], [43].

There are many studies, in particular video rate control studies, that have analyzed transmission buffers in detail. Rate control is a technique used in the video encoder that tries to match the data rate of the produced compressed video stream to the available data rate of the transmission channel. With a more precise rate control, smaller transmission buffers can be used. Navakitkanok *et al.* [44] have analyzed the effect of a delay constraint on the encoder buffer. The buffer after the encoder drops frames when it becomes too full and therefore would cause a delay exceeding the constraint. Consequently, Navakitkanok *et al.* have proposed a rate control method to allocate the number of bits per encoded frame. Rate control generally improves the buffer behavior and reduces the number of dropped frames [45]–[47]. Other studies, such as Ribas-Corbera *et al.* [48] as well as Zhang and Steinbach [49], have extensively analyzed the transmission buffer and proposed rate control via frame-level bit allocation. Low decoding latencies specifically for GPU buffers have been studied in [50].

Low delay video transmission has also received considerable attention in popular video standards, such as H.264/AVC [51] and H.265/HEVC [52]–[54]. For example, H.264/AVC supports an arbitrary ordering of image slices during sending and reception [51]. HEVC offers parallel processing of image segments [54] which reduces encoding latency compared to sequential single-thread processing because parallel processing fully uses modern multi-core architectures. Furthermore, the most popular implementations of both standards, $\times$264 [55] and $\times$265 [56], provide zero latency tuning options, offering parameter sets to decrease encoding and decoding delay, albeit for reduced rate-distortion performance.

C. Contributions

Our first main contribution with respect to the existing literature is a detailed analysis of all delay components in a video communication system in Section II, including the camera, coding, and display delays. Furthermore, we propose an intelligent frame skipping method to transmit high-priority event frames and a preemption mechanism to guarantee the fastest possible transmission of these high-priority frames in Section III. Finally, we are—to the best of our knowledge—the first to systematically examine the impact of differing frame rates for the camera, encoder, network, and display on the G2G and G2A delays in a video transmission prototype. We introduce the prototype, which includes a novel G2G and G2A delay measurement system, in Section IV. Section V presents evaluation results for the frame skipping and preemption algorithms and confirms the correctness of the theoretical model for predicting the delay of video communication.

II. Modeling and Analyzing Delay in Video Transmission

We analyze the fundamental point-to-point video transmission chain for machine vision (G2A) and for a human observer (G2G), as depicted in Fig. 1. The video transmission chain in Fig. 1 comprises the entire G2G delay for a human observer. The model for a machine vision setup is different in that the display part is omitted when computing the G2A delay. The display part is only required when the video is presented to humans. Machine vision setups typically employ an image processing algorithm, which then causes a physical action, e.g., a motor movement. In the end-to-end analysis of vision-based control systems, the image processing, control algorithm, and actuator delays have to be added to the G2A delay. This paper will not further detail the image processing, control algorithm, and actuator delays, as these delays are not directly related to video communication. In the remainder of this section we analyze the G2G delay T for a human observer.

A. Cut-Through Operation

Some of the blocks in Fig. 1 can operate in a cut-through (CT) mode. A CT operation means that a block a starts sending a data segment (to the next block b), before the segment has been completely received and/or processed by block a. Without loss of generality, we assume that a data segment contains one video frame, which we refer to as "frame" for brevity. In addition to starting the sending while processing the segment, processing of a segment can be started before it is entirely received, such that receiving, processing, and sending can take place at the same time. CT operations are common practice in camera and display electronics. The received first Bytes of a frame are processed once they are available and forwarded to the next block in the chain. Such a tight CT operation is possible because of the reliable and fast connections between the chain blocks.

The alternative to the CT operation is the store-and-forward operation: the data segment needs to be completely received and stored in memory before processing starts. After the processing of the segment is finished, the forwarding to the next block in the chain is initiated. Examples for the store-and-forward operation mode can be found in software encoders and decoders as well as in packet network switches.

B. Block Model

The blocks of the video communication chain as well as their parameters, metrics, and, where available, cut-through abilities, are explained in the following. A summary of the definitions, parameters, and metrics is given in Table I.

1) Camera Frame Refresh: In general, the camera refresh instants are not synchronized with the time instants when real-world events occur. Therefore, an event can occur at any time between two successive camera exposures. We assume in the following that the camera sensor is exposed to light during an entire frame period t_{Cam}, such that the exposure time equals the frame period t_{Cam}. There are implementations in which the sensor is exposed to light for a time period shorter than a frame period. However, shorter exposure is not desirable for cameras with high frame rate that want to utilize the entire frame period for exposure so as to maximize the signal to noise ratio on the sensor. As will be shown later, very low G2G or G2A delay requires the use of a camera with a high refresh rate, and hence the assumption that the exposure time is equal to the frame period does not limit the following analysis. The frame period t_{Cam} in the camera is the inverse of the camera refresh rate f_{Cam}. In our model, we assume that different blocks can have different and varying frame rates, as shown in Table I.

TABLE I
PARAMETERS AND METRICS OF THE VIDEO TRANSMISSION DELAY MODEL, WITH PARAMETER SETTINGS FOR THE PROTOTYPE EXPERIMENTS

Type	Variable	Relations	Comment
Definitions	Θ	$\Theta = \{$ CFR, CP, FS, CC, Enc, EB, Netw, DB, Dec, DR, DP, DPR $\}$	Set of video delay components
	$t_i, i \in \Theta$ [s]		Delay of component i
	$p_i(t), i \in \Theta$	$t_i \sim p_i(t)$	Probability density function of the delay of component i
Parameters	$t_{\min}$ [s]	$t_{\min} = \max_{i\in\Theta\backslash\{\mathrm{Netw}\}}(t_i)$	Frame selection: Minimum time between two frames.
	$t_{\max}$ [s]	$f_{\mathrm{Base}} = \frac{1}{t_{\max}}$	Frame selection target: Maximum time $t_{\max}$ between two frames to approximate a minimum frame rate f_{Base}
	f_{Cam} [Hz]	$t_{\mathrm{CFR}} \sim \mathcal{U}\left(0, \frac{1}{f_{\mathrm{Cam}}}\right), t_{\mathrm{Cam}} = \frac{1}{f_{\mathrm{Cam}}}$	Camera frame rate, in our prototype $f_{\mathrm{Cam}} = 240$ Hz.
	f_{Enc} [Hz]	$t_{\mathrm{Enc}} = \frac{1}{f_{\mathrm{Enc}}}, f_{\mathrm{Base}} \le f_{\mathrm{Enc}} \le f_{\mathrm{Cam}}$	Encoder frame rate
	f_{Dis} [Hz]	$t_{\mathrm{DR}} \sim \mathcal{U}\left(0, \frac{1}{f_{\mathrm{Dis}}}\right)$	Display frame rate, also represents the machine vision frame rate, in our prototype $f_{\mathrm{Dis}} = 144$ Hz.
	C [Mbps]		Channel transm. bit rate, in our prototype $C = 1$ Gbps.

The readout time of the image sensor and the limited bandwidth of the interface to the next processing block limit the achievable frame rate of a camera. For example, a camera with a 60 Hz refresh rate has a frame period of approximately 16.7 ms. If an event takes place just after the frame period started, the exposure has to be finished before the frame containing the event can be further processed, adding almost one entire frame period to the delay. On the other extreme, if an event occurs just before the end of a frame period, then there is almost no additional delay due to the camera refresh. Overall, the delay t_{CFR} introduced by the camera refresh process can be modeled as a uniform distribution between zero and the camera frame period t_{Cam} because the event can take place at any time during exposure, independently of the end/beginning of exposure. The higher the camera frame rate, the smaller is the worst case delay introduced by the camera frame refresh process. A 500 Hz camera for example has a worst case delay t_{CFR} of 2 ms.

2) Camera Circuitry: The camera electronics comprise two main parts. First the camera sensor, from which the pixel data is read out. Second, the camera processing block applies elementary processing operations, such as gain, offset, white balance, and analog to digital conversion. As these functions are implemented in hardware, the resulting processing delay t_{CP} is typically in the sub-millisecond range. For example, in the Allied Vision Guppy Pro[1] cameras, these processing steps take $t_{\mathrm{CP}} = 710\ \mu\mathrm{s} \pm 62.5\ \mu\mathrm{s}$.[2]

The camera processing is considered to be a CT operation. Using the Ximea Mako as an example, we saw that industrial cameras connected over USB 3.0 adjust the transmission rate such that the transmission time of a frame from the camera to the frame selector equals the frame period t_{CFR}. The transmission delay of a frame from the camera to the frame selector is the size of the raw frame divided by the camera interface bitrate. We include this transmission delay in the camera processing delay t_{CP}.

3) Frame Selection (Frame Skipping): In standard video transmission pipelines, all camera frames are forwarded. In contrast, a key component of our low-delay approach is that we skip selected frames and forward only the non-skipped frames in order to reduce the bit rate of the encoded video or the G2G delay, as described in Section III-A. In our CPU-based prototype, the time needed for frame selection t_{FS} varies around a mean of 246 μs up to a maximum delay of 810 μs for frames with VGA (640 $\times$ 480 pixels) resolution. Currently the frame selection operates in store-and-forward mode in our implementation. Optimized algorithms and hardware implementations are expected to significantly reduce the frame selection delay t_{FS}.

When implemented in hardware, a CT mode is feasible. In software setups, cameras return a pointer to the frame data when they finish writing the frame data in the memory. There is currently no way of reliably reading from memory into which the camera is still writing data. This is different for hardware implementations, for which data bits from the camera can immediately be processed after passing through small buffers. In the CT mode of hardware implementations, only a part of the frame is analyzed before the decision whether the frame shall be skipped or not is made. The investigation of how large the frame part needs to be for reliable frame skipping decisions is an important direction for future research.

4) Encoding (Incl. Color Conversion): The encoding of a video frame can take widely differing times t_{Enc}, depending on the implementation, hardware or software encoder, and the employed standard, e.g., AVC/H.264 or HEVC/H.265. In a low-delay video application, it is prohibitive to use a group of pictures structure in which inter-coded frames are bidirectionally predicted and depend on frames from the future. Such non-causal coding requires frame reordering, which introduces several frame periods of delay. Instead, unidirectional prediction-based inter coding or intra-only encoding is used in typical low-delay video applications. Intra-only encoding encodes the frames independently of each other. Intra-only encoding has a significantly smaller compression ratio than inter-coding, which exploits temporal dependencies between successive frames. Therefore, the encoder is the block in which we mainly trade off latency against compression efficiency. Intra-only hardware encoders have an encoding delay of as little as 250 μs.[3] This

[1] [Online]. Available: http://www.alliedvision.com, accessed on: 2017-01-30.

[2] See page 136 of the Allied Vision Technologies Guppy Pro Technical Manual V4.1.0.

[3] SOC Technologies MPEG-4 H.264/AVC Video Encoder.

delay describes the propagation latency of a pixel through the encoder, not the time t_{Enc} it takes to encode an entire frame. The frame encoding time t_{Enc} is equal to the inverse of the maximum frame rate f_{Enc} at which an encoder can process frames.

If necessary, color space conversion with delay t_{CC} is applied before encoding because usually, cameras stream frames in RGB format and encoders use frames in YUV format. Also, creating the packets and packet headers is part of encoding, but takes a negligible amount of time.

From the cut-through perspective, we distinguish software and hardware encoders. State-of-the-art software encoders are unable to perform cut-through processing. They write the completely encoded frame to memory when they have finished encoding. Thus, for software encoders, the frame encoding time t_{Enc} (and not the pixel propagation latency through the encoder) is relevant for the G2G delay. In the remainder of this paper, we focus on software encoders.

5) Encoder Buffer, Decoder Buffer: Buffer behavior has been extensively examined, see e.g., [44], [48], [49], especially in video rate control studies. A highly filled buffer introduces a large encoder buffer delay t_{EB} or decoder buffer delay t_{DB}. In the evaluations in Section IV, the channel rate $C = 1$ Gbps is high compared to the video bit rate, which is in all cases below 2 MByte/s, see Table II. Therefore, the buffering delay is negligible and not further investigated. In general, using rate control, the bitrate of the video is matched as closely as possible to the available bitrate of the channel. Assuming a well-working rate control, which is able to quickly and accurately adapt to the channel bitrate, the buffer load is kept as small as possible to minimize the resulting delay.

6) Network: The network delay t_{Netw} accounts for all delays for the network transport of an encoded frame from the encoder to the decoder. In the simple case of a direct link (channel) from encoder to decoder, t_{Netw} accounts for the delays for the transmission of the frame onto the direct link and the signal propagation over the link. For a more complex packet-switched network, t_{Netw} accounts also for the processing, queueing, transmission, and propagation delays due to the intermediate switches. Complex networks covering large distances can contain the packets of multiple video frames at a given time instant. Hence, such networks exhibit t_{Netw} delays much larger than one frame period. The network delay t_{Netw} is therefore not considered for the calculation of $t_{\min}$ in Table I.

We briefly elaborate on the simple case of a direct channel, which is considered in our prototype. The frame transmission delay equals the frame size s [bit] divided by the transmission bit rate C [bps] of the channel. For example, a 1500 Byte frame on a $C = 1$ Gbit/s channel (as considered in our experiments in Section V-A) incurs a transmission time of 12 μs. The propagation speed of electrical signals on a channel is commonly expressed relative to the speed of light $c \approx 3 \cdot 10^8$ m/s in vacuum. Electrical signals in copper travel at a speed between $0.66c$ (RG-58/U coaxial cable, ca. $1.98 \cdot 10^8$ m/s) and $0.95c$ (open wire, ca. $2.85 \cdot 10^8$ m/s) [57]. Considering a 100 m long copper cable, the resulting delays range from 0.35 μs to 0.51 μs and are negligible for the low-delay video application. In our experiments, the connection distance is shorter than 100 meters, yielding a propagation delay shorter than 1 μs. For longer connections, the propagation delay becomes significant. For example, for 300 km of connection distance, the propagation delay, lower bounded by the speed of light, is at least 1 ms. Overall, for our prototype setup with a short direct channel between encoder and decoder, the network delay corresponds essentially to the transmission delay, i.e., $t_{\mathrm{Netw}} = s/C$.

7) Decoding (Incl. Color Conversion): Decoding reverses the encoding step and produces a raw frame that can be sent to the display. Highly optimized decoders, such as the libavcodec h.264 decoder used in our prototype, have an average decoding delay of $t_{\mathrm{Dec}} = 272$ μs. If necessary, a color space conversion with delay t_{CC} is applied on the raw frame, which adds 321 μs on average for our setup. After that, the frame resides in the graphics buffer, waiting for the next display refresh.

8) Display Refresh: Analogous to the camera frame refresh, the display panel is refreshed at a fixed rate f_{Dis} in classical panels. The display refresh process comprises the read out of the frame data from the graphics buffer and the transfer to the display electronics in periodic time intervals of constant duration $1/f_{\mathrm{Dis}}$. The limited rate is caused by the limited bandwidth of the data transmission interface of the display and the time the panel requires to draw one frame. In traditional displays, refresh time is not related to when the decoder finishes decoding a frame. In the best case, the decoder finishes a frame just before the next display refresh, causing almost no additional delay t_{DR}. In the worst case, the decoder puts out a frame immediately after a display refresh, in which case the most recent frame is drawn on the display by the next refresh, which takes place after almost one display frame period. The frame display period is the inverse of the display refresh rate f_{Dis}.

New panel types with dynamic refresh using synchronization techniques such as NVidia G-Sync$^{\mathrm{TM}}$ or AMD FreeSync$^{\mathrm{TM}}$ have recently reached the market. These panels can synchronize their display refresh to the refresh of the graphics buffer. The synchronization reduces t_{DR} drastically at low video frame rates compared to using a fixed refresh display. But these displays also have a minimal period of time they need to draw one frame, imposing a maximum refresh rate. If the video frame rate equals the maximum refresh rate, there is no more gain from display refresh synchronization techniques because the entire receiving unit is running at its maximum frame rate, and will therefore refresh independently of when a frame from the sender arrives. If every block in the video transmission chain is operating at the same frame rate, synchronization can eliminate the display refresh delay and only the camera refresh delay varies relative to the event time.

9) Display Processing: The time t_{DP} describes the delay between when a new pixel value is sent to the display and when the display electronics are ready to change the corresponding pixel with the next panel refresh. The processing delay varies widely for different displays: we measured values from less than 1 ms on a Samsung 2233BW monitor up to 23 ms in a DELL U2412M. The transmission time from the decoder to the display is the size of the decoded frame in bits divided by the transmission bitrate of the connection interface, for example DisplayPort 1.3 provides 8.1 Gbps per lane.[4] We include the transmission delay to the display in the display processing delay t_{DP}.

10) Display Pixel Response: Pixels in LCD displays do not respond instantaneously to a changed voltage. In modern displays, such as the Acer XB270H, the grey to grey response time t_{DPR} goes down to 1 ms. In our prototype, we observed a display refresh plus processing plus pixel response delay $t_{\mathrm{Dis}} + t_{\mathrm{DP}} + t_{\mathrm{DPR}} = 8.6$ ms as the difference between the mean G2G and G2A delay.

C. Remarks

1) Delay Variations: All these delays are not perfectly constant due to real-world imperfections. Nevertheless, significant delay variations are generally only expected from the frame refresh in the camera, the encoder, the decoder, and the display refresh. When transmitting the video over a network with high transmission delay and delay jitter, then t_{Netw} also becomes significant. In a video transmission setup with low latency components, the camera and display contribute significantly to the G2G delay, as found in [1]. The delays from these two components can account for over half of the average G2G delay. Therefore, the delays from the camera and the display offer significant potential for G2G delay reduction.

2) Potential Gains of CT Operation: In the sender (Fig. 1), at most one camera frame transmission period (which is equal to a camera frame period in our setup) can be saved through CT operation. This is because in store-and-forward, the frame has to be received only once, and then the processing units, for example the frame selector, color conversion and encoder, can subsequently apply their operations on the frame in the shared memory. In the receiver, at most the frame transmission delay onto the link leading to the receiver can be saved (which is shorter than or equal to the network delay t_{Netw}). The delay reduction is limited to the frame transmission delay on the link to the receiver because in the store-and-forward mode, the receiver has to wait until the frame has fully arrived and can then quickly operate on it. Note that delays due to multi-hop store-and-forward network transport are not part of these considerations and are not affected by CT operation in the receiver. CT operation implies very small tolerances for delay jitter from the blocks delivering data to sender and receiver. While the jitter is not an issue for the sender, which is fed with frames from the camera, the jitter has to be carefully considered for the receiver when designing a video transmission setup with CT operation.

3) Influence of Spatial Image Resolution: A raw three-channel color image with a resolution of 640 × 480 pixels with 1 Byte pixel depth contains $640 \cdot 480 \cdot 3 = 0.92$ MByte of data. In comparison, a 1920 × 1080 color image has 6.2 MByte, almost seven times as much data. The sizes of the corresponding encoded images usually relate with a similar factor (or a slightly smaller factor because encoding of higher resolution images can better exploit spatial redundancy). Many processing step delays are proportional to the amount of processed data: frame selection, color conversion, encoding, network, and decoding operate sequentially or in a parallel manner with a limited number of parallel threads and delays are therefore directly proportional to the amount of data that these steps process. For instance, reducing the image size to 50% vertically and horizontally, reduces the delays due to these steps to 25% of the original values. The camera frame refresh is a special case: the maximum achievable frame rate of a camera depends on the vertical image resolution because the vertical readout process in the camera sensor dictates the shortest achievable sensor readout time. Therefore, with half the vertical resolution, twice the frame rate f_{Cam} can be achieved, which results in halving the camera refresh delay t_{CFR}.

We performed a 50% vertical and horizontal image size reduction in our prototype, which yielded a 320 × 240 pixel video at 480 Hz (the camera frame rate f_{Cam} doubled due to the halved vertical resolution). The resulting mean G2G delay was measured as 17.27 ms, i.e., 2.4 ms less than the 19.67 ms mean G2G delay for the first scenario in Table II, which considered a 640 × 480 pixel video at 240 Hz. The delay difference of 2.4 ms can approximately be obtained by multiplying the mean delay values of scenario 1 from Table II for the frame selector, color conversion, encoder, network, and decoder with 0.75 (we reduce the amount of processed data by 75%), and adding them up. The sum is then added to the difference in mean t_{CFR} delays, which is half the difference in camera frame periods (of the 240 fps and 480 fps cameras), i.e., 1.04 ms. This calculation gives 2.35 ms, a reasonable approximation of the measured 2.4 ms, considering the measurement precision of 0.5 ms, see Section IV-B.

III. Low Delay Video Transmission Mechanisms

In this section, we present two novel approaches for reducing the G2G and G2A delays in a video transmission chain. The first approach is a frame skipping method that discards video frames without significant information. The second approach is a buffer preemption mechanism that shortens the waiting time of frames containing significant novel information, i.e. frames that differ largely from their preceding frame, so called key frames. Frames without significant novel information are called regular frames.

A. Frame Skipping

As noted in Section II-C, the delay introduced by the camera refresh process offers significant potential for reducing the G2G delay. To leverage this potential we reduce the delay t_{CFR} by using a camera with high frame rate. Our core contribution is that we enable the use of a high frame rate camera without requiring changes of other system parameters, such as the processing speed of the encoder or the required transmission rate of the communication network. The proposed frame skipping method avoids an increased output information rate which potentially overloads the transmission channel. To control the camera output information rate so that no block is overburdened, we propose to select the frames which are to be transmitted immediately after the capturing process at the camera. The de-

[4][Online]. Available: https://www.vesa.org/featured-articles/vesa-publishes-displayport-standard-version-1-4/, accessed on: 2017-01-30.

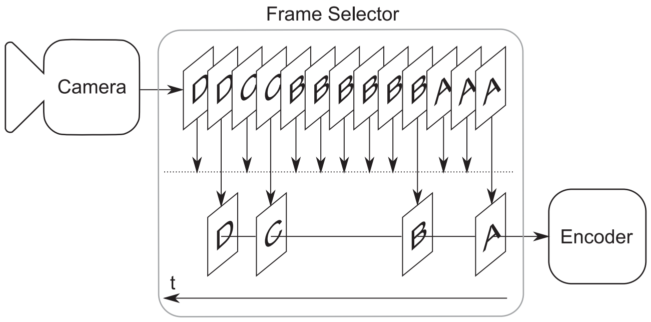

Fig. 2. Frame selection process: a subset of the frames from the camera is selected for encoding, the remaining frames are skipped.

cision 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 processing. New information can, for instance, be measured as the **m**ean **a**bsolute **d**ifference (MAD) or the **s**tructural **sim**ilarity index (SSIM) [58]. Other frame selection metrics for change detection can also be used. Alternatively, if the frame skipping unit is placed after an intra-only encoder, the encoded frame could be analyzed. In the following we only consider frame skipping that selects raw frames. By transmitting the frames with new information, we ensure that a new event which significantly changes the frame content achieves minimal delay. The process is illustrated in Fig. 2. Frames with 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 example 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 selection threshold. Note that in Fig. 2, 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 [59]–[61]. 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 bitrate. 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.

Algorithm 1: Frame Selection

```
1:  if $\Delta I > I_{\mathrm{thr}}$ then                         ▷ Content
2:     if $\Delta t_{\mathrm{prev}} < t_{\min}$ then               ▷ Bottleneck
3:        skip frame
4:     else
5:        transmit key frame
6:     end if
7:  else
8:     if $\Delta t_{\mathrm{prev}} > t_{\max}$ then       ▷ Subjective criteria
9:        transmit regular frame
10:    else
11:       skip frame
12:    end if
13: end if
```

3) Bottleneck Component: Every block of the chain has a constant or varying throughput rate in frames per second, as described in Section II-B. 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. Therefore, the frame skipper selects frames for transmission at most at a frame rate equal to $1/t_{\min}$.

4) Frame Selection Algorithm: Joining all three criteria into frame selector instructions yields the decision rules presented in Algorithm 1. ΔI is the content difference between the last transmitted frame and the current frame, which the 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 $\Delta t_{\mathrm{prev}} > t_{\min}$, 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 to approximate the minimal frame rate $1/t_{\max}$.

5) Prototype Implementation of Frame Selection: We implemented Algorithm 1 based on a thresholded form of the MAD in our experimental prototype. The thresholded form of the MAD first conducts a pixel-wise subtraction of the current frame from the previously transmitted frame. The absolute pixel values of the difference frame are then thresholded. All pixel differences greater than ten are set to the maximum value of 255, while the remaining pixels are set to the minimum value of zero. This thresholding significantly reduces the influence of small additive values in all pixels in every new frame, known as thermal camera sensor noise. Subsequently, the mean of the thresholded difference image is computed. The resulting thresholded MAD is compared with the fixed MAD content difference threshold $I_{\mathrm{thr}} = 1.4$. The $I_{\mathrm{thr}} = 1.4$ threshold value was determined empirically in order to be sensitive to events that are spatially small. At the same time, the $I_{\mathrm{thr}} = 1.4$ value is above the difference value of two images that differ only in noise.

We set $t_{\min} = 0$ ms in our prototype setup because all units are able to process images at more than 240 Hz. Finally, we set

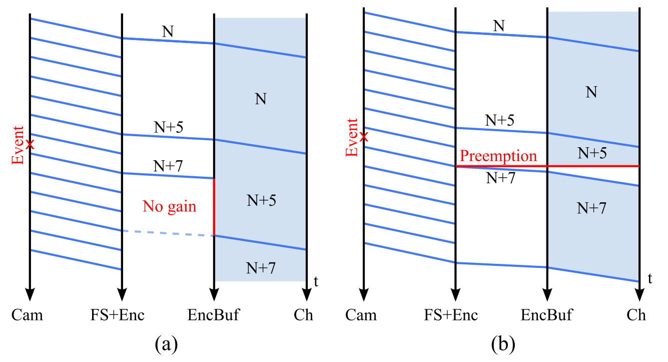

Fig. 3. Frame transmission (a) without preemption and (b) with preemption.

$t_{\max} = 420$ ms, yielding a lower target of 2.4 Hz for the frame rate, i.e., 1 % of the camera frame rate.

Future research may explore parameter adaptations. For instance, the content threshold I_{thr} could be adjusted based on video content, lighting conditions, and camera model. The bottleneck parameter $t_{\min}$ could be adapted according to the state of the communication network. Additionally, $t_{\max}$ could be adjusted depending on the requirements of the machine vision algorithm processing the received video sequence.

B. Preemption

Preemption reduces the G2G or G2A delay when 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 this key frame waiting, 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 Fig. 3(a), 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 transferred. 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 $t_{\max}$ is set such that every fifth frame from the camera is selected as a regular frame, as illustrated in Fig. 3(a). 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. Therefore, 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.

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 encoding buffer, the channel, the decoder buffer, the decoder, or the graphics buffer feeding the display, has to be deleted (flushed) as shown in Fig. 3(b). 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 $t_{\min}$ 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. Therefore, 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 we use intra-only coding in combination with preemption. However, intra-only coding leads to worse compression efficiency because the temporal redundancy is not exploited to improve coding efficiency.

IV. Prototype Setup

We implemented a prototype of the video communication system depicted in Fig. 1. In this section, we first describe the overall system and then detail our system for accurately measuring G2G and G2A delays.

A. System Description

A Ximea MQ022CG-CM USB3.0 camera recording a 640 × 480 pixels RGB video is connected to a Ubuntu 16.04 Desktop PC running the frame skipping algorithm and an ×264 encoder 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. The encoded video is then streamed to the decoder PC using UDP, where it is decoded using libavcodec and displayed on an Acer XB270H monitor. The system parameters are noted in Table I.

B. Measuring System for G2G and G2A Delay

In [1], we proposed a system to measure G2G delay with a precision of 0.5 ms. In this subsection, we first give a short

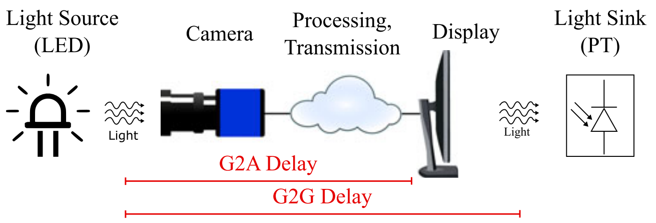

Fig. 4. Illustration of the G2G and G2A delay measurement system: the delay of the propagation of the light information of an LED until the corresponding image is decoded is the G2A delay, while the delay of the propagation of the light information from the LED to a PT gives the G2G delay.

overview of the G2G delay measurement system and then extend the system to measure G2A delays.

1) G2G Measurement System: The G2G delay measurement system measures time delay introduced (added) by a video transmission system under test to the propagation of light from a light-emitting diode (LED) to a phototransistor (PT), as illustrated in Fig. 4. To perform the measurement, the measurement controller turns on the LED which is positioned in the field of view of the camera and notes the associated time instant. A PT is attached to the display at the position where the LED is shown. The resistance of the PT decreases when the LED lights up in the displayed image. The controller notes the time instant when the resistance decreases. The controller then computes the G2G delay as the time span between the time instant of turning on the LED and the time instant of the decrease of the PT resistance.

The precision of the measurements depends on the sampling rate of the PT. In our system, the sampling rate is 2 kHz, yielding a precision of 0.5 ms. Previous delay measurement systems that relied on filming the output of the display with a video camera, e.g., [62]–[65], had limited precision of, e.g., 16.7 ms for a 60 Hz video camera. Further advantages of the proposed system over previous work [66], [67] are that our system conducts measurements automatically without requiring human assistance, is non-intrusive and therefore works in any video transmission system, and has low cost of less than 40 Euros per system. To make the system widely accessible,[5] we developed an Android application that guides the users through the measurement process.

2) Adding G2A Measurement Capability: For G2A delay, we modify the G2G measurement system as follows. At the receiver, we note the time instant when the first image of the lit up LED is decoded and color converted so that the image would be ready for an image processing algorithm, see Fig. 1. Specifically, using pixel value thresholding, the bright LED is recognized by the decoding PC, which sends out a signal over the serial port. The serial port is connected to the measurement system, which notes the time instant of the serial signal. The time span from the lighting up of the LED to the serial signal from the decoding PC is the G2A delay.

We also used the serial port approach to measure the delay until the image is ready for encoding in the PC that is directly connected to the camera, yielding the camera circuitry delay $t_{\mathrm{CP}} = 6.02$ ms. We assume this delay to be constant because camera processing is implemented in hardware, resulting in negligible camera circuitry delay variance in our evaluation context.

[5]The building and measurement instructions, the Android application, and the system source code are available under http://tinyurl.com/G2GDelay.

V. Evaluation Results

In Section V-A, the delay reduction effect of frame skipping as described in Section III-A is investigated, without the preemption from Section III-B, since preemption was not necessary for the considered $C = 1$ Gbit/s channel. In Section V-B, we then evaluate the prototype with preemption with a constrained encoder buffer output rate of 14 kByte/s, which emulates a channel rate of $C = 14$ kByte/s. In Section V-C, we compare a probabilistic model for predicting a G2G delay distribution with an empirical cumulative distribution function of measurements from our prototype.

A. *Effectiveness of Frame Skipping*

We measured G2G and G2A delay using the system described in Section IV-B. In addition, we measured the individual delays for frame selection t_{FS}, encoding t_{Enc}, transmission t_{Netw}, decoding t_{Dec}, and color conversion t_{CC} with the high resolution clock of the C++ time library. Also, the frame and bit rate of the produced video are computed over windows of one second. We consider four scenarios with static 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. And, 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, we add the front panel of an oscilloscope and a LEGO construction for spatial complexity.

For every scenario and partial delay component, we measured at least 500 samples. The results of these measurements are summarized in Table II and the empirical cumulative distribution functions for the measured G2G and G2A delays in the first scenario are plotted in Fig. 5 with 95% confidence envelope based on the Dvoretzky-Kiefer-Wolfowitz inequality [68], [69].

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 II. The maximum G2G delay is considerably higher at 35.65 ms. We observe from Fig. 5 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, we observe the same phenomenon in Fig. 5. Without the outliers, the maximum G2A delay would be approximately 13 ms. We observe from Table II 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 dis-

TABLE II
PROTOTYPE MEASUREMENT RESULTS: MINIMUM, FIRST QUARTILE (Q1), MEAN, MEDIAN, THIRD QUARTILE (Q3), MAXIMUM, AND STANDARD DEVIATION OF PERFORMANCE METRICS

Scenario	Component [unit]	Minimum	Q1	Mean	Median	Q3	Maximum	Std. Dev.
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
	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
	Frame rate [Hz]	240	240	240	240	240	240	0
	Bit 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]	2.4	6.0	7.4	7.0	8.0	13.0	1.57
	Bit 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]	7.4	7.4	7.4	7.4	7.4	7.4	0
	Bit 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
	Bit rate [kByte/s]	1393	1588	1634	1641	1675	1770	59.01

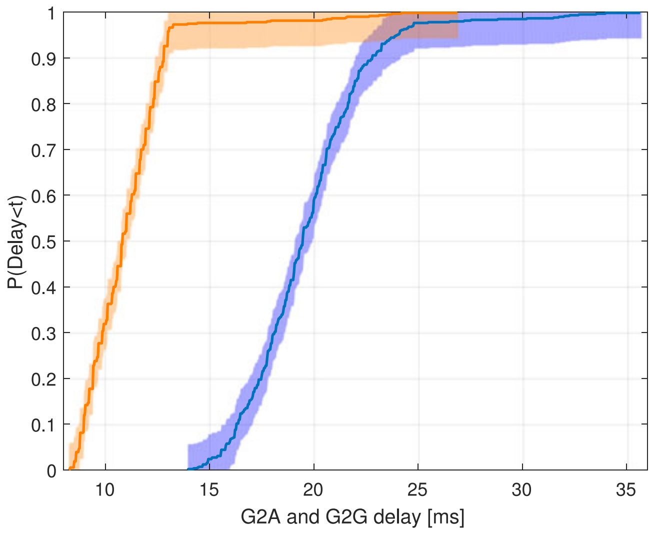

Fig. 5. Empirical cumulative distribution functions for G2A (orange) and G2G (blue) delays for scenario 1 (high fps, no skipping), including 95%-confidence envelopes around the graphs.

play 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 we approximate t_{DP} with constant delay.

The minimum delays of frame selection 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. We measured the camera frame processing and transmission delay from the camera to the encoder computer to be $t_{\mathrm{CP}} = 6.02$ ms. 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 delay, 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 bitrate is 658 kByte/s.

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 increase is smaller, with an increase of 0.43 ms, compared to scenario 1 (see Table II) 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 II-B.1, 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 II), frames with the dim LED are transmitted and trigger the detection algorithm after the phototransistor (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 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. Therefore, 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 bitrate; both drop approximately by a factor of 40. In particular, frame skipping reduces the mean bitrate 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.

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 II 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 II. 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 bit rate of the low frame rate scenario is 13.78 kByte/s, which is slightly lower than the 17.34 kByte/s average bitrate of the frame skipping scenario. Thus, we conclude that frame skipping employed with a high frame rate camera in our prototype setup requires only about the same (or slightly more) transmission bitrate as a conventional low frame rate camera while drastically reducing the mean G2G and G2A delays in this prototype setup.

4) Varying Image Contents: The final scenario (scenario 4 in Table II) 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 bit rate is doubled to tripled compared to the full transmission 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.

5) Influence of Frame Skipping on Video Quality: With the parameter settings detailed in Section III-A.5, 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 Section III-A.2 forces an image transmission, even if the image content difference ΔI is small. The only difference between such images is the Gaussian distributed thermal noise caused by the camera sensor. The low frame rate $1/t_{\max} = 2.4$ Hz is perceivable through image differences caused by the thermal noise. But low frame rates for still scenes were found to have little influence on the quality of experience, as the image differences are insignificant. Also, when moving an object very slowly through the scene, the low frame rate is perceivable, however, this does not significantly degrade the quality of experience.

We have evaluated the influence of frame skipping on the objective video quality as follows. We moved objects at various speeds in front of the camera. For each video sequence and prescribed content difference threshold value I_{thr}, we compute the PSNR and SSIM values between the displayed frames and the corresponding skipped frames. This yields approximately 5000 value pairs for each approximately 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. Considering the lowest values provides conservative lower bounds compared to alternative approaches, such as considering the mean of the PSNR and SSIM values across the video stream [60], that have been developed for subsampling with a constant rate of displayed frames. Our 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 $I_{\mathrm{thr}} = 1.4$ 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 III-A.5, we utilized the $I_{\mathrm{thr}} = 1.4$ default threshold to be sensitive to new events, but also robust to noise. For $I_{\mathrm{thr}} < 1.1$, the frame skipping mechanism chooses all frames, while for increasing $I_{\mathrm{thr}} > 1.1$, the algorithm starts skipping frames. For threshold values up to $I_{\mathrm{thr}} = 3$, 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 $I_{\mathrm{thr}} = 3$. Increasing $I_{\mathrm{thr}} > 3$ 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 $I_{\mathrm{thr}} = 10$. 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, we recommend a threshold I_{thr} just above the camera sensor noise for low delay, good video quality, and a low data rate.

B. *Effectiveness of Preemption*

We implemented the preemption mechanism described in Section III-B with the Click Modular Router [70]. Specifically, we added the preemption functionality to the encoder buffer queueing block, see Figs. 1 and 3, 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 1 of Algorithm 1 (the remainder of Algorithm 1 was not executed, i.e., there was no frame skipping in order to evaluate the effects of preemption in isolation). To require buffering, we constrained the output rate of the encoder buffer to $C = 14$ 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 $C = 1$ Gbit/s.

Our preemption evaluation focused on the flushing of full frames (that had not yet begun transmission) from the encoder buffer. The situation depicted in Fig. 3, i.e., the cancellation of the ongoing transmission of a regular frame to make way for a key frame, was not considered. This is because 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 bitrate of the connection between sender and receiver is 1 Gbit/s, giving frame transmission periods in the sub-millisecond range.

In Fig. 6, 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 our 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 II because for this setup we reduced the raw frame size 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

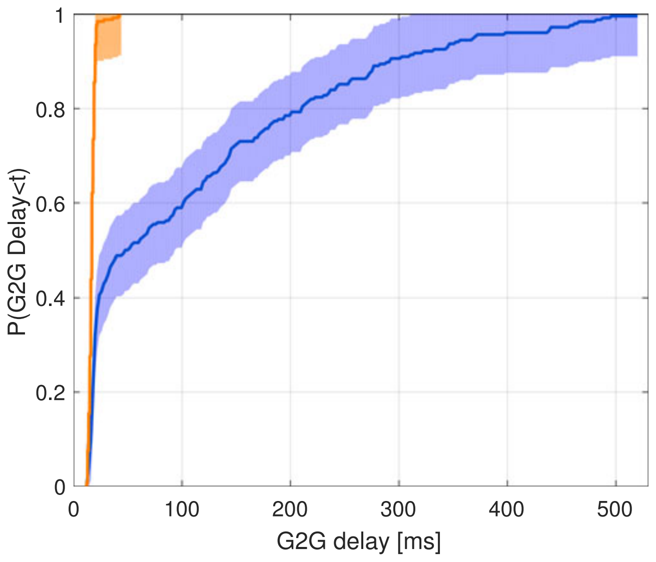

Fig. 6. Cumulative G2G delay distribution function with preemption (orange) and without preemption (blue), including 95%-confidence envelopes around the graphs.

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 $t_{\min}$. However, increasing $t_{\min}$ may block a key frame that could 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 II.

C. *Comparison to Theoretical Delay Model*

We conduct a probabilistic analysis to derive an approximation of the G2G delay distribution. We start by modelling the delays of the blocks from Section II-B. Both the camera refresh delay t_{CFR} and the display refresh delay t_{DR} are modeled as uniform random variables, as defined in Table I.

We model the encoding and decoding delays t_{Enc} and t_{Dec} by triangular distributions, which approximate the underlying limited Gaussian distributions that we observed in our measurements. 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. We did not use the maximum $t_{\mathrm{Enc}} = 1.67$ ms from Table II because this value was influenced by operating system interrupts, which we do not include in our model. Analogously, the triangle corner positions for the decoder are at 0.17 ms, 0.27 ms, and 0.54 ms, taken from Table II 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 selection, color conversion, network, and display processing contribute an average delay of

$$\begin{aligned} t_{\mathrm{rem}} &= t_{\mathrm{CP}} + t_{\mathrm{FS}} + t_{\mathrm{CC}} + t_{\mathrm{Netw}} + t_{\mathrm{DP}} \\ &= (6.02 + 0.25 + 0.32 + 0.02 + 5.69)\ \mathrm{ms} \\ &= 12.30\ \mathrm{ms}. \end{aligned} \tag{1}$$

For these blocks we use the mean measurement results from Table II. We do not include buffering delays because they are negligible for the 1 Gbit/s channel. The remaining delay t_{rem} is assumed to be constant, therefore the distribution p_{rem} is a unit impulse at t_{rem} and zero otherwise.

The sum of these block delays results in the G2G delay T. We obtain the probability density function (PDF) of the G2G delay T by convolving the PDFs of the delays

$$T \sim p(t) = (p_{\mathrm{CFR}} * p_{\mathrm{Enc}} * p_{\mathrm{Dec}} * p_{\mathrm{DR}} * p_{\mathrm{Rem}})(t). \tag{2}$$

For (2) to be valid, the delays have to be mutually independent. Mutual independence is not given for p_{Enc} and p_{Dec}, but we presume that violating this assumption will still give a reasonable approximation. The cumulative probability distribution of the G2G delay resulting from (2) is depicted by the blue graph in Fig. 7 for the given parameters. The orange graph in Fig. 7 shows the cumulative G2G delay distribution of the first scenario in Table II 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. We did not include the operating system interrupts 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 systems without preemption, queueing models for the typically highly variable waiting times in buffers would have to be included.

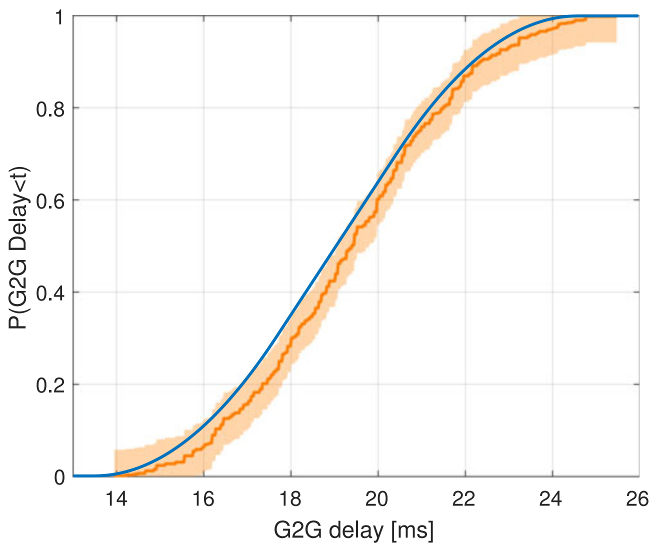

Fig. 7. Cumulative distribution function obtained from the probabilistic analysis in Section V-C (blue) and empirical cumulative distribution function of the measurement samples from the prototype for full transmission (high fps, no skipping, Section V-A.1) (orange). For the empirical distribution, we also give the 95%-confidence envelope.

VI. Conclusion

We conducted a detailed analysis of the delay sources in a point-to-point video transmission system and found that the sampling delay of the camera contributes a large proportion to the Glass-to-Glass (G2G) delay and the Glass-to-Algorithm (G2A) delay. We proposed two methods, namely frame skipping and preemption, to enable the use of a high frame rate camera and consequently reduce the G2G and G2A video delays.

We examined the effectiveness of frame skipping in a video communication system prototype built with typical off-the-shelf video communication components. Adding frame skipping to a high frame rate camera system reduced the required transmission bitrate by a factor of almost forty, while maintaining nearly the same low G2G and G2A delays. On the other hand, frame skipping (in conjunction with a high frame rate camera) reduced the G2A delay by almost a full order of magnitude compared to a low frame rate camera system while maintaining the same (or slightly increasing the) required transmission bitrate as the low frame rate camera system. The preemption mechanism can effectively avoid waiting times of key frames in the encoder buffer. In our evaluation prototype, preemption reduced the average G2G delays by half an order of magnitude. Moreover, we proposed a theoretical model for predicting the G2G delay distribution of a video transmission system. The model can be used when setting up the parameters of a video transmission system.

There are several interesting directions for future research on low latency video transmission. One direction is to implement and evaluate cut-through operation, for which a hardware prototype has to be set up, e.g., based on field-programmable gate arrays. Another direction is to optimize the frame skipping decision and to study the effects of frame skipping on the subjective and objective video quality in more detail. Furthermore, application-specific frame skipping can be investigated. It is possible that humans require frame skipping decision rules that are different from decision rules that are optimal for machine vision algorithms.

ACKNOWLEDGMENT

The authors would like to thank A. B. Méndez for setting up parts of the video transmission prototype.

REFERENCES

[1] C. Bachhuber and E. Steinbach, "A system for high precision glass-to-glass delay measurements in video communication," in *Proc. IEEE Int. Conf. Image Process.*, 2016, pp. 2132–2136.

[2] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 3354–3361.

[3] K. Okumura, H. Oku, and M. Ishikawa, "High-speed gaze controller for millisecond-order pan/tilt camera," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 6186–6191.

[4] N. Alt, C. Claus, and W. Stechele, "Hardware/software architecture of an algorithm for vision-based real-time vehicle detection in dark environments," in *Proc. ACM Conf. Des., Autom. Test Eur.*, 2008, pp. 176–181.

[5] N. Alt and E. Steinbach, "Visuo-haptic sensor for force measurement and contact shape estimation," in *Proc. IEEE Int. Symp. Haptic Audio Vis. Environ. Games*, 2013, pp. 24–28.

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

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

[8] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proc. IEEE*, vol. 95, no. 1, pp. 138–162, Jan. 2007.

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

[10] D. Yue, Q.-L. Han, and C. Peng, "State feedback controller design of networked control systems," in *Proc. IEEE Int. Conf. Control Appl.*, 2004, vol. 1, pp. 242–247.

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

[12] L. Zhang, H. Gao, and O. Kaynak, "Network-induced constraints in networked control systems—A survey," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 403–416, Feb. 2013.

[13] R. Yang, G.-P. Liu, P. Shi, C. Thomas, and M. V. Basin, "Predictive output feedback control for networked control systems," *IEEE Trans. Ind. Electron.*, vol. 61, no. 1, pp. 512–520, Jan. 2014.

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

[15] G. P. Fettweis, "The tactile internet: Applications and challenges," *IEEE Veh. Technol. Mag.*, vol. 9, no. 1, pp. 64–70, Mar. 2014.

[16] G. Baltas and G. Xylomenos, "Evaluating the impact of network I/O on ultra-low delay packet switching," in *Proc. IEEE Symp. Comput. Commun.*, 2015, pp. 397–402.

[17] O. Bondarenko, K. De Schepper, I.-J. Tsang, B. Briscoe, A. Petlund, and C. Griwodz, "Ultra-low delay for all: Live experience, live analysis," in *Proc. ACM Int. Conf. Multimedia Syst.*, 2016, pp. 33:1–33:4.

[18] B. Briscoe *et al.*, "Reducing internet latency: A survey of techniques and their merits," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2149–2196, Third Quarter 2016.

[19] C. Greco, M. Cagnazzo, and B. Pesquet-Popescu, "Low-latency video streaming with congestion control in mobile ad-hoc networks," *IEEE Trans. Multimedia*, vol. 14, no. 4, pp. 1337–1350, Aug. 2012.

[20] U. Jennehag, S. Forsstrom, and F. V. Fiordigigli, "Low delay video streaming on the internet of things using Raspberry Pi," *Electronics*, vol. 5, no. 3, pp. 1–11, 2016.

[21] Y. Liu, D. Niu, and B. Li, "Delay-optimized video traffic routing in software-defined interdatacenter networks," *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 865–878, May 2016.

[22] J. Pilz, M. Mehlhose, T. Wirth, D. Wieruch, B. Holfeld, and T. Haustein, "A tactile internet demonstration: 1 ms ultra low delay for wireless communications towards 5G," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2016, pp. 862–863.

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

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

[25] 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 Trans. Mobile Comput.*, vol. 15, no. 3, pp. 641–655, Mar. 2016.

[26] J. Wu, B. Cheng, C. Yuen, N.-M. Cheung, and J. Chen, "Trading delay for distortion in one-way video communication over the internet," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 4, pp. 711–723, Apr. 2016.

[27] 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 *Proc. Graph. Interface Conf.*, 2014, pp. 167–174.

[28] 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 *Proc. ACM SIGCHI Conf. Human Factors Comput. Syst.*, 2013, pp. 2291–2300.

[29] 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 *Proc. ACM Symp. Appl. Perception Graph. Vis.*, 2004, pp. 39–47.

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

[31] C. Bachhuber and E. Steinbach, "Are todays video communication solutions ready for the tactile internet?" in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops*, 2017, pp. 1–6.

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

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

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

[35] R. M. Schreier, A. M. T. I. Rahman, G. Krishnamurthy, and A. Rothermel, "Architecture analysis for low-delay video coding," in *Proc. IEEE Int. Conf. Multimedia Expo.*, 2006, pp. 2053–2056.

[36] R. M. Schreier and A. Rothermel, "A latency analysis on H.264 video transmission systems," in *Proc. IEEE Int. Conf. Consum. Electron.*, 2008, pp. 1–2.

[37] P. Holub, J. Matela, M. Pulec, and M. Šrom, "Ultragrid: Low-latency high-quality video transmissions on commodity hardware," in *Proc. ACM Int. Conf. Multimedia*, 2012, pp. 1457–1460.

[38] 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 *Proc. IEEE Int. Conf. Digit. Signal Process.*, 2015, pp. 1062–1066.

[39] 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 *Proc. EDA Conf. Des., Autom. Test Europe*, 2013, pp. 115–120.

[40] T. Liu and C. Choudary, "Real-time content analysis and adaptive transmission of lecture videos for mobile applications," in *Proc. ACM Int. Conf. Multimedia*, 2004, pp. 400–403.

[41] A. Doulamis and G. Tziritas, "Content-based video adaptation in low/variable bandwidth communication networks using adaptable neural network structures," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2006, pp. 4037–4044.

[42] P. Usach-Molina, J. Sastre, V. Naranjo, L. Vergara, and J. M. L. Muñoz, "Content-based dynamic threshold method for real-time keyframe selecting," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 7, pp. 982–993, Jul. 2010.

[43] H. Lee and S. Kim, "Rate-constrained key frame selection using iteration," in *Proc. IEEE Int. Conf. Image Process.*, 2002, vol. 1, pp. I-928–I-931.

[44] P. Navakitkanok and S. Aramvith, "Improved rate control for advanced video coding (AVC) standard under low delay constraint," in *Proc. IEEE Int. Conf. Inf. Technol., Coding Comput.*, 2004, vol. 2, pp. 664–668.

[45] 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 Syst. J.*, vol. 6, no. 3, pp. 414–425, Sep. 2012.

[46] 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 J. Electron. Imag.*, vol. 25, no. 4, pp. 043008–1–043008–13, Jul. 2016.

[47] 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 *Proc. IEEE Int. Conf. Multimedia Expo. Workshops*, 2013, pp. 1–4.

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

[49] F. Zhang and E. Steinbach, "Improved ρ-domain rate control with accurate header size estimation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2011, pp. 813–816.

[50] S. Y. Chien, K. H. Lok, and Y. C. Lu, "Low-decoding-latency buffer compression for graphics processing units," *IEEE Trans. Multimedia*, vol. 14, no. 2, pp. 250–263, Apr. 2012.

[51] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[52] D. Grois, D. Marpe, T. Nguyen, and O. Hadar, "Comparative assessment of H. 265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders for low-delay video applications," in *Proc. SPIE*, 2014, pp. 92170Q-1–92170Q-10.

[53] T. Mallikarachchi, D. S. Talagala, H. K. Arachchi, and A. Fernando, "Content-adaptive feature-based CU size prediction for fast low-delay video encoding in HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, to be published.

[54] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[55] "x264—A free open-source H.264 encoder," Jun. 2007. [Online]. Available: http://www.videolan.org/developers/x264.html. Accessed on: Jul. 20, 2017.

[56] "x265—A free open-source H.265 encoder," Apr. 2014. [Online]. Available: http://www.videolan.org/developers/x265.html. Accessed on: Jul. 20, 2017.

[57] K. L. Kaiser, *Transmission Lines, Matching, and Crosstalk*. Boca Raton, FL, USA: CRC Press, 2005.

[58] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[59] Y.-F. Ou, T. Liu, Z. Zhao, Z. Ma, and Y. Wang, "Modeling the impact of frame rate on perceptual quality of video," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2008, pp. 689–692.

[60] Y.-F. Ou, Z. Ma, T. Liu, and Y. Wang, "Perceptual quality assessment of video considering both frame rate and quantization artifacts," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 3, pp. 286–298, Mar. 2011.

[61] Z. Lu *et al.*, "Measuring the negative impact of frame dropping on perceptual visual quality," *Proc. SPIE*, vol. 5666, pp. 554–562, 2005.

[62] R. Hill, C. Madden, A. van den Hengel, H. Detmold, and A. Dick, "Measuring latency for video surveillance systems," in *Proc. IEEE Digit. Image Comput., Technol. Appl.*, 2009, pp. 89–95.

[63] J. Jansen and D. C. Bulterman, "User-centric video delay measurements," in *Proc. ACM Workshop Netw. Oper. Syst. Support Digit. Audio Video*, 2013, pp. 37–42.

[64] J. MacCormick, "Video chat with multiple cameras," in *Proc. ACM Conf. Comput. Supported Cooperative Work Companion*, 2013, pp. 195–198.

[65] T. Sielhorst, W. Sa, A. Khamene, F. Sauer, and N. Navab, "Measurement of absolute latency for video see through augmented reality," in *Proc. IEEE/ACM Int. Symp. Mixed Augmented Reality*, 2007, pp. 1–4.

[66] O. Boyaci, A. Forte, S. A. Baset, and H. Schulzrinne, "vDelay: A tool to measure capture-to-display latency and frame rate," in *Proc. IEEE Int. Symp. Multimedia*, 2009, pp. 194–200.

[67] M. C. Jacobs *et al.*, "Managing latency in complex augmented reality systems," in *Proc. ACM Symp. Interact. 3D Graph.*, 1997, pp. 49–54.

[68] P. Massart, "The tight constant in the Dvoretzky–Kiefer–Wolfowitz inequality," *Ann. Probability*, vol. 18, no. 3, pp. 1269–1283, Jul. 1990.

[69] A. Dvoretzky, J. Kiefer, and J. Wolfowitz, "Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator," *Ann. Math. Statist.*, vol. 27, no. 3, pp. 642–669, Sep. 1956.

[70] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, Aug. 2000.

Christoph Bachhuber received the B.Sc. degree in electrical engineering and information technology in 2012 and the M.Sc. degree in 2014, both from the Technical University of Munich (TUM), Munich, Germany, where is currently working toward the Ph.D. degree.

He joined the Chair of Media Technology, TUM, in 2014, where he is currently a member of the research and teaching staff. His research interests include delay in video communication, vision-based control, and video coding.

Eckehard Steinbach (S'96–A'99–M'04–SM'08–F'15) studied electrical engineering at the University of Karlsruhe, Karlsruhe, Germany, the University of Essex, Colchester, U.K., and ESIEE, Paris, France, and received the Engineering Doctorate degree from the University of Erlangen-Nuremberg, Erlangen, Germany, in 1999.

From 1994 to 2000, he was a member of the research staff of the Image Communication Group, University of Erlangen-Nuremberg. From February 2000 to December 2001, he was a Postdoctoral Fellow with the Information Systems Laboratory, Stanford University, Stanford, CA, USA. In February 2002, he joined the Department of Electrical Engineering and Information Technology, Technical University of Munich, Munich, Germany, where he is currently a Full Professor of media technology. His current research interests include the area of audio-visual-haptic information processing and communication as well as networked and interactive multimedia systems.

Martin Freundl received the B.Sc. degree in electrical engineering and information technology and the M.Sc. degree from the Technical University of Munich (TUM), Munich, Germany, in 2014 and 2016, respectively.

He worked on the implementation and analysis of key frame selection in a latency-constrained video transmission at the Chair of Media Technology, TUM, during his M.Sc. thesis. He is currently a Software Engineer in the field of automation technology with SAS-Softec GmbH, Weiden, Germany.

Martin Reisslein (S'96–A'97–M'98–SM'03–F'14) received the Ph.D. degree in systems engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 1998.

He is currently a Professor with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ, USA.

Prof. Reisslein is currently an Associate Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TXRANSACTIONS ON EDUCATION, IEEE ACCESS, and *Computer Networks* and *Optical Switching and Networking*. He is an Associate Editor-in-Chief of the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS.