Received December 30, 2019, accepted March 28, 2020, date of publication April 8, 2020, date of current version April 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2986580

Power Modeling for Video Streaming Applications on Mobile Devices

CHRISTIAN HERGLOTZ[1], (Member, IEEE), STÉPHANE COULOMBE[1], (Senior Member, IEEE), CARLOS VAZQUEZ[1], (Senior Member, IEEE), AHMAD VAKILI[2], ANDRÉ KAUP[3], (Fellow, IEEE), AND JEAN-CLAUDE GRENIER[2]

[1]Department of Software and IT Engineering, École de Technologie Supérieure (ÉTS), Montréal, QC H3C 1K3, Canada
[2]Summit Tech Multimedia Communications Inc., Montréal, QC H2N 1N2, Canada
[3]Multimedia Communications and Signal Processing, Friedrich-Alexander University Erlangen-Nürnberg (FAU), 91058 Erlangen, Germany

Corresponding author: Christian Herglotz (christian.herglotz@fau.de)

This work was supported by the Mitacs and Summit Tech Multimedia through the Mitacs Accelerate Program.

ABSTRACT In this paper, we derive an accurate power model for video streaming which we condense to the essential components contributing the most to the overall power consumption. As a use case, we choose mobile devices on the receiver side performing video streaming in broadcasting or end-to-end scenarios. In modeling, we consider the complete video streaming toolchain, which mainly consists of data acquisition, video processing, display, and audio handling. We compose an overall power model with the help of models from the literature and propose a dedicated feature selection approach to reveal the most important factors related to power consumption. The resulting models achieve mean estimation errors below 7.61%. Results from feature selection indicate that the display brightness, the bitrate, and the frame rate have the highest impact on the power consumption.

INDEX TERMS Video coding, modeling, power measurement, mobile communication, streaming, H.264/AVC, HEVC.

I. INTRODUCTION

In recent years, video streaming in end-to-end or broadcasting scenarios has become the main contributor to global Internet traffic. Forecasts indicate that the share of video data will even increase to more than three fourths in 2022 [1]. Worldwide, all this data is received by millions of portable devices, such as smartphones or tablet personal computers (PCs), which must process the data to be able to display video and play audio for high-quality user experiences. Furthermore, a recent study revealed that 1% of the global greenhouse gas emissions is caused by video communications [2], which shows that modern video systems contribute significantly to climate change. As a consequence, research on energy efficient video communication solutions is of vital importance.

In this respect, a large amount of research is performed to optimize the energy efficiency or the complexity of distinct components, such as the encoder on the sender side [3]–[5], the delivering network [6]–[9], or the video decoder [10]–[13] and the display on the receiver side [14].

The associate editor coordinating the review of this manuscript and approving it for publication was Honggang Wang.

In this paper, we focus on a mobile device on the receiver side whose battery is highly burdened by the streaming process, which is characterized by a computationally complex software application and power-intensive hardware modules.

A high-level analysis of the power consumption of a modern smartphone revealed that during local video playback with hardware decoding, more than 1 W of power is constantly needed [15]. Consequently, a fully charged battery is drained after a maximum of 15 hours (at a capacity of 3000 mAh and a supply voltage of 5 V). For virtual reality applications, we have shown that the reduction of the resolution helps in saving power [16]. Furthermore, it was shown that reducing the resolution of a streamed video instead of increasing the quantization is beneficial both in terms of subjective quality and power consumption [17].

In contrast to the aforementioned publications, which only considered pure video streaming, this paper deals with the complete pipeline including transmission and audio handling. To this end, we split up the streaming process into multiple functional tasks running in parallel. From this perspective, we can break down the complete streaming process into the components depicted in Fig. 1. As shown on the left side,

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

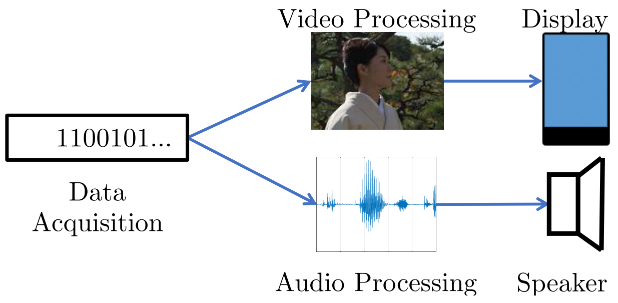


FIGURE 1. Flowchart of the video streaming process with the components data acquisition, video processing, display, audio processing, and speaker.

the streaming process starts with the acquisition of audio and video data, which can be read from the local memory or received through a network. After demuxing, the video data is processed (top branch), which means that it is decoded, rendered, and displayed. On the bottom branch, the audio data is decoded and the reconstructed raw audio signal is played using a speaker.

In the literature, one can find a wide variety of works that consider the power consumption of single parts of this pipeline. For the source network, Zou et al. compared the power efficiency of a Wi-Fi connection to a long-term evolution (LTE) connection when using H.264/AVC coded videos [18]. A very detailed analysis, which focused solely on Wi-Fi power consumption including interference handling, was performed by Sun *et al.* [19]. For audio decoding using MP3, Simunic et al. constructed a model for software processing and suggested power saving algorithms based on code optimizations [20]. Furthermore, Dolezal and Becvar [21] considered audio decoding and output via a speaker.

For video, many researchers focus on the video decoding process. For software decoders, high-level models [22] using few bit stream properties as well as advanced models using up to 90 bit stream features for accurate energy estimations were proposed in [23]. Furthermore, the impact of coding standards, such as MPEG-2, H.263, and H.264/AVC, on power consumption was analyzed in detail in [24]. For hardware decoders, models were developed for power estimation in [10] and for energy estimation in [25].

All these papers reflect the state of the art allowing for accurate and reliable power and energy estimates of distinct processes or hardware modules in the streaming pipeline. However, in applied video streaming, these modules are running in parallel and their functionality and processing flow is interconnected, such that the power and the energy consumption might differ significantly. To the best of our knowledge, this paper is the first to analyze the power of the complete pipeline in detail.

To analyze the end-to-end process, we construct a linear model reflecting the functionality of the components shown in Fig. 1 at a high abstraction level. Measuring in various test conditions, we find the dependency with respect to streaming parameters such as the bitrate, the frame rate, or the resolution of the video sequence. Therefore, we perform a high number of measurements on two applications and three devices, train the proposed model in a least-squares sense, and ensure validity by performing cross-validation. Then, a feature selection algorithm is presented which is used to identify the most important parameters related to the power consumption. In summary, this work provides the following contributions, which from our review of the state of the art, has not been studied before:

- Consideration of the complete streaming pipeline.
- Construction of an overall power model.
- Derivation of main parameters related to the power consumption.

In Section II, we review various energy and power models from the literature and insert them into a power model for accurate power estimation of the complete streaming process. Then, in Section III, we discuss the power measurement setup as well as the tested hardware and software solutions. Furthermore, the set of measurements is described including the set of input sequences and the set of input audio files. Afterwards, Section IV presents the training and validation method, which is used to assess the proposed energy models. Section V introduces our proposed method for feature selection and finally, Section VI interprets the results and presents a power analysis of the tested streaming applications. Section VII concludes this paper.

II. POWER MODELING

For ease of application and to allow the use of linear algebra, we propose to use a linear power model in the form

$$\hat{P} = \boldsymbol{v} \cdot \boldsymbol{x}, \tag{1}$$

where $\hat{P}$ is the estimated power of the device during streaming, $\boldsymbol{v}$ a row vector of variables describing the streaming process, $\boldsymbol{x}$ a column vector of model parameters of the same size, and the symbol '$\cdot$' denotes the dot product. In the evaluation, we will show that this linear approach is sufficient when mean estimation errors below 10% are acceptable.

In the remainder of this paper, the variables in $\boldsymbol{v}$ are denoted by Roman letters. These variables describe the streaming process, for instance, using properties of the video stream, such as the frame rate f_{v} or the resolution S, or using flags indicating whether or not a component, such as the audio branch, is active. These flags are denoted by the letter F, and can obtain a value of '1' for active or '0' for non-active. The parameters in $\boldsymbol{x}$ are denoted by uppercase and lowercase Greek letters. The parameters with uppercase letters relate to the flags F, and can be interpreted as constant offsets. The lowercase Greek parameters for their part are attributed to the variables describing the streaming process.

In a first step, we assume that each component shown in Fig. 1 contributes to the complete power in an additive fashion. As we assume streaming to be a static process, we can neglect a time dependency and model the complete streaming power P_{stream} by

$$P_{\mathrm{stream}} = \Pi_0 + P_{\mathrm{data}} + P_{\mathrm{video}} + P_{\mathrm{display}} + P_{\mathrm{audio}} + P_{\mathrm{speaker}}. \tag{2}$$

In addition to the components shown in Fig. 1, we consider an offset parameter Π_0. The corresponding variable in $\boldsymbol{v}$ is constantly set to one. P_data can be interpreted as the power needed for data acquisition through a network, P_video is the power needed for video decoding and rendering (e.g., transformation from YCbCr to RGB), and P_display is the power needed to display the video or to send it to a high-definition multimedia interface (HDMI) port. P_audio is the power needed for audio processing and P_speaker the power needed for playing the sound via a speaker or sending it to an audio output port (HDMI or headset). In the following subsections, we discuss the modeling of these components in detail.

A. DATA ACQUISITION

First, we consider the power for data acquisition P_data. In the literature, the power needed to read audio and video data from the local memory is often considered to be part of the audio and video processing powers P_audio and P_video [10], [23]. Consequently, we only consider network streaming in the data acquisition power. A simple model is proposed for 4G LTE networks by Huang *et al.* [26]; the same model was indeed used for Wi-Fi networks by Xu *et al.* [27]. Because it is straightforward, and has been proven to be valid for these two network technologies (4G and Wi-Fi), we assume that it can also be valid for other network technologies, such as 3G or hardwired Ethernet.[1] It is given by

$$\hat{P}_\text{net} = \Gamma + \beta \cdot b, \tag{3}$$

where b is the incoming bitrate of the stream, which is the sum of the audio bitrate b_a and the video bitrate b_v; Γ is a constant offset, and β is a parameter describing the linear relation between the bitrate b and the receiving power. In practice, during measurements, we maintain only the target network connection, which is hence used for downloading the stream (e.g., during Wi-Fi streaming, GSM networks, such as 3G or LTE, are disabled). The resulting model reads

$$\hat{P}_{n,\text{data}} = \beta_n \cdot b_n + \Gamma_n \cdot F_n, \tag{4}$$

where n denotes the network type, which in this work can be Wi-Fi, 3G, or Ethernet (RJ45). The overall data acquisition power can then be modeled as the sum over all considered networks

$$\hat{P}_\text{data} = \hat{P}_\text{Wi-Fi,data} + \hat{P}_\text{3G,data} + \hat{P}_\text{ethernet,data}, \tag{5}$$

where in practice, none or only one of the summands is nonzero because the other networks are switched off.

B. VIDEO PROCESSING POWER

In the literature, many models were proposed for estimating the decoding energy and the decoding power [10], [11], [22], [23], [28], [29]. As most modern smartphones provide hardware video decoder modules that are more power-efficient than software video decoders [15], we only consider a hardware decoder model proposed in [25]. The original model was designed to estimate the decoding energy and reads

$$\hat{E}_\text{vdec} = \Xi_0 + \Psi_0 \cdot T + \epsilon_\text{frame} \cdot N_\text{frames} + \epsilon_\text{bit} \cdot B_\text{v} + \pi_S \cdot S \cdot T, \tag{6}$$

with Ξ_0 being a constant offset energy for the initialization of the process, Ψ_0 a constant offset power during processing, T the duration of the sequence, ϵ_frame the energy per frame, N_frames the number of coded frames, ϵ_bit the bit-dependent energy, B_v the size of the video bit stream in bits, π_S the power needed to reconstruct a single pixel, and S the number of pixels per frame, which is the product of the luma pixel width with the luma pixel height. The power per pixel was used instead of the energy per pixel to implicitly take the frame rate f_v into account by

$$f_\text{v} = \frac{N_\text{frames}}{T}. \tag{7}$$

To apply this model in the power domain, we divide by the duration T and let T approach infinity assuming an endless sequence as

$$\begin{aligned}\hat{P}_\text{vdec} &= \lim_{T\to\infty} \frac{\hat{E}}{T} \\ &= \lim_{T\to\infty} \left[\frac{\Xi_0}{T} + \Psi_0 + \epsilon_\text{frame} \cdot \frac{N_\text{frames}}{T} + \pi_S \cdot S \right. \\ &\quad \left. + \epsilon_\text{bit} \cdot \frac{B_\text{v}}{T} \right] \\ &= \Psi_0 + \epsilon_\text{frame} \cdot f_\text{v} + \pi_S \cdot S + \epsilon_\text{bit} \cdot b_\text{v},\end{aligned} \tag{8}$$

where we exploit (7) and the relation between the bitrate and the file size

$$b_\text{v} = \frac{B_\text{v}}{T}. \tag{9}$$

In this model, the bit stream properties frame rate f_v, the pixels per frame S, and the video bitrate b_v are used as variables.

In addition, the rendering of the output pixels needs to be taken into account. We did not find any suitable model for this power in the literature. Therefore, we propose using the number of pixels to be rendered per second as

$$\hat{P}_\text{render} = \rho \cdot S \cdot f_\text{v}, \tag{10}$$

where ρ is the model parameter.

It is worth mentioning that we consider the application to be a black box in which rendering and decoding cannot be separated in measurements. Furthermore, we take into account two different video codecs (HEVC and H.264/AVC) and find empirically that the codec only has a marginal influence on the overall power consumption. Hence, we combine (8) and (10), neglect the codec, and obtain

$$\hat{P}_\text{video} = F_\text{v} \cdot \Psi_0 + \epsilon_\text{frame} \cdot f_\text{v} + \pi_S \cdot S + \epsilon_\text{bit} \cdot b_\text{v} + \rho \cdot G \tag{11}$$

with the new variable $G = S \cdot f_\text{v}$ that describes the pixels per second. F_v indicates whether a video is transmitted.

[1]This assumption will be validated in Section VI.

Please note that when $F_\mathrm{v} = 0$, the other variables are zero, too, such that the overall estimated power $\hat{P}_\mathrm{video}$ also yields zero.

C. DISPLAY OPERATION POWER

In this work, we use liquid crystal displays (LCD) in measurements. Kim et al. found that the power of such displays is linearly related to the backlight brightness H [30]. However, our measurements as well as observations made by Carroll and Heiser [31] suggest that the power can also have a quadratic relation with the backlight brightness. Hence, to cover both cases, we propose using a quadratic model as

$$\hat{P}_\mathrm{display} = F_\mathrm{display} \cdot \Omega + \kappa \cdot H + \lambda \cdot L, \qquad (12)$$

where F_display indicates whether the display is switched on, $L = H^2$ is the square of the backlight brightness, and Ω, κ, as well as λ are the modeling parameters. The tested Android-based smartphones provide a range of brightness levels of $\{0, 1, \dots, 255\}$.

D. AUDIO PROCESSING POWER

As an audio processing power model, Simunic et al. proposed to use processor events, such as the number of instructions and memory reads [20]. As our goal is to model the power exploiting high-level parameters, we avoid performing a complex analysis on processor events. Therefore, similar to the video processing case, we consider the input audio bitrate b_a, the sampling rate f_a, and the number of audio output channels C. The proposed power model reads

$$\hat{P}_\mathrm{audio} = F_\mathrm{a} \cdot \Phi + \eta \cdot b_\mathrm{a} + \theta \cdot f_\mathrm{a} + \zeta \cdot C + \xi \cdot R, \qquad (13)$$

where $R = f_\mathrm{a} \cdot C$ is the number of audio samples per second. Similar to video decoding, we tested two different audio codecs (MP3 and AAC) but found no significant difference in the overall power consumption. The model parameters are Φ, η, θ, ζ, and ξ. F_a states whether an audio stream is available.

E. SPEAKER OPERATION POWER

For the speaker, we adopt two variables. First, we adopt the output volume O, which, in Android, can be set in the range of $\{0, 1, \dots, 16\}$. Additionally, we assume that the power of the discrete audio signal has an influence on the power consumption. The signal power is given by

$$p_\mathrm{signal} = \frac{1}{N} \sum_{l=1}^{N} (x[l])^2, \qquad (14)$$

where $x[l] \in [-1, 1]$ is the normalized amplitude of the audio signal (16 bit per sample) at the time l. N is the complete number of audio samples [32]. The model reads

$$\hat{P}_\mathrm{speaker} = \omega \cdot O + \mu \cdot p_\mathrm{signal} \qquad (15)$$

with the parameters ω and μ, which, respectively, describe the impact of the volume and the signal power on the power consumption.

TABLE 1. Input variables and parameters for modeling the power consumption of video streaming.

Component	Eq.	Input variables	Parameters
General	(2)	1	Π_0
P_data	(4)	F_n, b_n	Γ_n, β_n
P_video	(11)	$F_\mathrm{v}, f_\mathrm{v}, S, b_\mathrm{v}, G$	$\Psi_0, \epsilon_\mathrm{frame}, \pi_S, \epsilon_\mathrm{bit}, \rho$
P_display	(12)	F_display, H, L	Ω, κ, λ
P_audio	(13)	$F_\mathrm{a}, b_\mathrm{a}, f_\mathrm{a}, C, R$	$\Phi, \eta, \theta, \zeta, \xi$
P_speaker	(15)	O, p_signal	ω, μ

Table 1 summarizes all considered power components and lists the input variables and parameters. The vector $\boldsymbol{x}$ consists of all the parameters, and the vector $\boldsymbol{v}$ is derived using the variables in the third column.

III. POWER MEASUREMENT SETUP

The literature proposes a significant number of power measurement methods. Simple methods rely on power monitors integrated into a chip. For modern Intel processors, the running average power limit (RAPL) tool [33] was used in [23] and [34]. Using complexity profiling tools, such as Valgrind [35] or PAPI [36], metrics such as the number of central processing unit (CPU) instructions and the memory access can be obtained. These tools are additionally used for energy or power estimation [11], [37], [38]. Finally, power meters can be exploited. One measurement solution is to derive the power of distinct hardware modules using integrated shunts [15], which is time consuming and costly. The method used in this work relies on the power consumption of the complete device (i.e., a smartphone or an evaluation board). To this end, we measure the power consumption through the main supply jack or the battery connectors. Similar approaches were used in [10], [23].

For the purposes of this research, this method was chosen to ensure that all relevant hardware modules in the streaming process are taken into account. These modules include the network interface, the memory (random-access-memory, cache, buffers), the CPU, the graphics processing unit (GPU), and the display. Hence, the measured powers correspond to the complete power needed in real applications.

Going forward, Subsection III-A presents the general measurement setup, while Subsection III-B introduces the measured portable devices, and Subsection III-C shows the tested software. Afterwards, Subsection III-D presents the results of a typical single power measurement and explains how we can ensure that measured power values are reliable and statistically sound. Finally, Subsection III-E lists all video and audio files used in separate measurement instances.

A. GENERAL MEASUREMENT SETUP

The setup for measuring the complete power consumption of a device under test (DUT) is depicted in Fig. 2. The DUT is located top-center, and can be a smartphone, an evaluation board, or any other device meant for live streaming. The DUT is energized by the power meter on the left using the

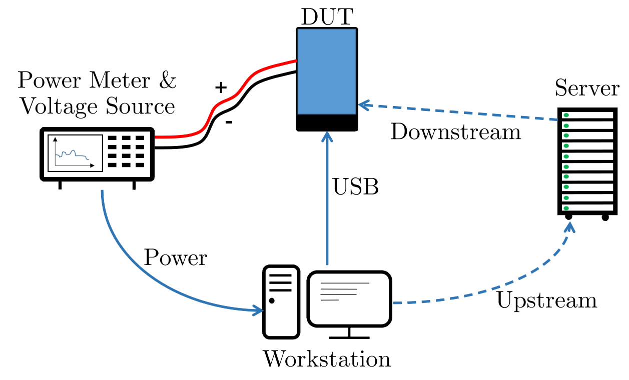


FIGURE 2. Measurement setup with DUT, power meter, streaming server, and workstation for measurement automation.

battery connectors or the voltage input jack. A remote broadcasting server is used to simulate real-time video streaming (right). Depending on the use case, hardwired Ethernet or wireless connections such as Wi-Fi or LTE are tested. Finally, the workstation performs automatic power measurements using universal serial bus (USB) control for the power meter and for the DUT. During power measurements, the USB connection to the DUT is disabled. A hardwired Ethernet network connection is used for streaming the test sequences from the workstation to the broadcasting server.

The power meter provides an integrated power supply. The supply voltage can be set manually, depending on the demands of the DUT. The power values are sampled at 5 kHz and are transmitted via USB to the workstation.

The server is an open real-time messaging protocol (RTMP) broadcasting server. Using FFmpeg, Flash video (FLV) containers [39] can be uploaded to this server, which broadcasts this data to the Internet. The stream can then be requested and received by any online device worldwide using wired or wireless connections.

B. DUTS

The first device (Device A) used as a DUT for power measurements is the Eragon820 software development kit (SDK). As the main processing unit, it includes an Eragon820 system-on-module (SOM) which implements a Qualcomm Snapdragon 820E processor. The main properties of this chip are listed at the top of Table 2.

The board is configured to enable stable and reliable power measurements. Therefore, all unnecessary services, such as positioning, Bluetooth, and Wi-Fi (if it is unneeded for measurements) are stopped. Unneeded applications are uninstalled or disabled. The application cache is emptied to avoid disturbances from applications running in the background.

The board is set to developer mode and it is rooted to allow control of the power management policy. With a special software application, the two high-performance processor cores are disabled and the two low-power processor cores are set to high-performance processing because initial tests showed that at least two processors need to be enabled for fluid playback. This setting is chosen to reduce the effect of dynamic voltage and frequency scaling (DVFS, [40]). Furthermore, the performance governor of the GPU is also set to 'performance'. Consequently, the processing units always run at the highest voltage and frequency.

TABLE 2. Main specifications of the three tested DUTs. The displays of device B and C use the in-plane switching (IPS) technology for thin-film-transistor (TFT) LCDs.

	Module	Properties
Device A	CPU	Quad-core (64 bit)
		2 low-power cores (max 1.593 GHz)
		2 high-performance cores (max 2.15 GHz)
	GPU	Up to 624 MHz, 319.4 GFLOPS
	Memory	4 GB of LPDDR 4 (up to 1866 MHz)
	Multimedia	4K video decoding at 60 fps (H.264 or HEVC)
Device B	CPU	Four 32 bit cores (max 2.5 GHz)
	GPU	Up to 578 MHz, 147.9 GFLOPS
	Memory	2 GB of LPDDR 3 (up to 1866 MHz)
	Multimedia	4K video decoding at 60 fps (H.264 or HEVC)
	Display	5 inches, 1080 × 1920 pixels, TFT/IPS
Device C	CPU	6-core
		4 low-power cores (max 1.4 GHz)
		2 high-performance cores (max 1.8 GHz)
	GPU	Up to 600 MHz, 153.6 GFLOPS
	Memory	3 GB of LPDDR 3 (up to 1866 MHz)
	Multimedia	4K video decoding at 60 fps (H.264 or HEVC)
	Display	5.5 inches, 1440 × 2560 pixels, TFT/IPS

The second device (Device B) is a consumer smartphone released in December 2015 (Fairphone 2). For our measurements, no application is stopped or cleared from the cache. A network connection to providers is always established to ensure a realistic environment. In terms of background services, Bluetooth and global positioning system (GPS) are always disabled because they are not needed for streaming. Two networks are tested: 3G and Wi-Fi, where only one network is enabled during the streaming process. Power management is set to the default settings.

Device C is another consumer smartphone (LG H812). Except for it being rooted and not connected to a network provider, its configuration is similar to Device B.

C. SOFTWARE

Two Android applications are tested which implement the video streaming process. App I is the VLC media player [41]. It is an open source video streamer and video player implementing the complete toolchain depicted in Fig. 1. It supports a vast number of codecs (H.264/AVC, HEVC, VP9, etc.), container formats (AVI, MP4, FLV, etc.), and streaming protocols (RTP, RTMP, HTTP, etc.). Therefore, it is highly suited for detailed streaming power analysis.

The second application (App II) is mainly designed for rich communication services (RCS). The application's pipeline works as follows. First, the communication stack receives RTMP packets that are decoded into an H.264-coded video and an audio stream. The video stream is decoded through the Android platform's hardware decoder into raw YCbCr frames, which are then sent to a pre-allocated OpenGL

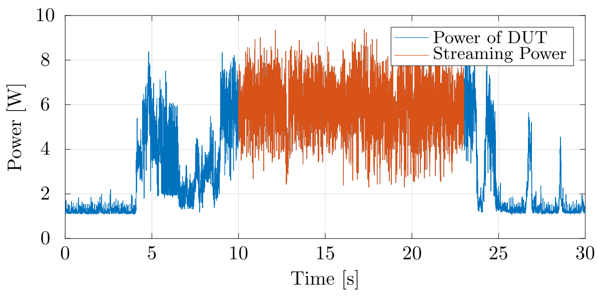


FIGURE 3. Power consumption of showcase streaming process. The blue parts correspond to the initialization of the application (left) and the exit (right), while the red part corresponds to the static streaming.

surface. A game engine reads from this surface to render the video and provides a shader to transform the raw YCbCr frame into a regular RGB frame ready for viewing.

D. POWER MEASUREMENTS

Fig. 3 shows the streaming power for a high-definition (HD) sequence (device A, App I, streaming via Ethernet). The curve shows that at the beginning of the measurement, the device is in idle mode ($\sim$ 1.8 W). At approximately 4 s, the streaming application is launched. Loading and preparing the stream takes up to approximately 10 s, when the video is starting to be displayed on the screen (red curve). At 24 s, the process is stopped and the device returns to idle mode.

In this work, we concentrate on the pure streaming power (red curve), and as such, the blue parts of the curve are ignored. Considering a live streaming solution in which the duration is unknown, this is a valid assumption. The advantage is that depending on the streaming duration, the energy can be easily calculated by multiplying the mean power with the duration.

Initial tests showed that when the same measurement is performed multiple times, the resulting mean power can vary significantly, for instance, because of background processes. Consequently, there is no assurance that a single measurement will return a meaningful average power value. Hence, each streaming configuration is measured up to 15 times. To speedup the measurement procedure, a confidence interval test [42] is applied for each streaming configuration if a minimum number of 5 measurements was done. To this end, the average of the mean power values $\bar{\bar{P}}$ and the confidence interval with a probability of 99% are calculated. If the confidence interval is located inside the interval $[0.99 \cdot \bar{\bar{P}}; 1.01 \cdot \bar{\bar{P}}]$, the measurement is assumed to be accurate, and is terminated early. Otherwise, the maximum of 15 measurements is performed and an outlier check is performed manually.

The resulting power values $\bar{\bar{P}}$ include the static and the dynamic power, i.e., the power consumption when the device is idle plus the power needed for processing. For the smartphones (devices B and C), the idle power, which yields less than 1 mW, is neglected. However, the evaluation board (device A) shows a high static power when idle, which can be attributed to the peripheral modules (e.g., Ethernet connector, HDMI output). This power of approximately 1.19 W is subtracted from the overall power for further analysis.

TABLE 3. Test sequences used for power measurements. All sequences have a duration of 10 s. The sixteen sequences at the top are taken from the HEVC common test conditions [43] and the two sequences at the bottom show a static white screen and a static black screen, respectively (all luminance values yield 255 and 0).

Class	Name	Resolution S	frame rate f_v
B	BQTerrace	1920 × 1080	60 Hz
	BasketballDrive	1920 × 1080	50 Hz
	Cactus	1920 × 1080	50 Hz
	Kimono	1920 × 1080	24 Hz
C	BQMall	832 × 480	60 Hz
	BasketballDrill	832 × 480	50 Hz
	PartyScene	832 × 480	50 Hz
	Flowervase	832 × 480	30 Hz
D	BQSquare	416 × 240	60 Hz
	BlowingBubbles	416 × 240	50 Hz
	BasketballPass	416 × 240	50 Hz
	RaceHorses	416 × 240	30 Hz
E	FourPeople	1280 × 720	25 Hz
	Johnny	1280 × 720	60 Hz
F	SlideEditing	1280 × 720	30 Hz
	SlideShow	1280 × 720	20 Hz
-	WhiteScreen	1920 × 1080	50 Hz
	BlackScreen	1920 × 1080	50 Hz

E. SET OF MEASUREMENTS

The single instances of the set of measurements are chosen based on the following guidelines:

- The measured variables shall avoid high correlations.
- For each variable, at least four independent measurement instances must be performed to avoid overfitting.
- Each measurement instance should be applicable to common streaming applications.

To have a representative set of video sequences, we take 16 sequences from the HEVC common test conditions [43]. Each sequence is coded with four quality levels (x264 and x265 encoding, medium preset for both H.264/AVC and HEVC video codecs, constant rate factors 18, 23, 28, 33). Hence, we measure a large number of different video frame rates, resolutions, and bitrates. The input sequences and their main properties are listed in Table 3.

For the display power, we assume that the chosen video files are representative in terms of the mean luminance (range [0, 1]), which varies between 0.227 and 0.636. The BlackScreen and WhiteScreen sequences additionally ensure that the lower and upper limits of the achievable range (0 and 1) are represented. Assuming different brightness levels, we restrict our analysis to the three states minimum, medium, and maximum brightness $\{0, 0.5, 1\}$, which were measured for four different input sequences plus BlackScreen and WhiteScreen.

TABLE 4. Source audio files for power measurements and main properties. C is the number of channels and the power is the digital signal power of the 16 bit samples (range $[-1, 1]$).

Name	f_audio	C	Signal Power	Source
Silence	48 kHz	2	0	-
Airport	48 kHz	8	0.030	[44]
Cafeteria	48 kHz	8	0.010	[44]
Conference	48 kHz	8	$2.7 \cdot 10^{-4}$	[44]
Crossroadnoise	48 kHz	8	0.062	[44]
FullSizeCar	48 kHz	8	0.287	[44]
InsideBus	48 kHz	8	0.265	[44]
Orchestra	48 kHz	8	0.023	[44]
Pub	48 kHz	8	0.040	[44]
RockMusic	48 kHz	8	0.019	[44]
SalesCounter	48 kHz	8	0.006	[44]
TrainStation	48 kHz	8	0.074	[44]
ChildSpeech1	48 kHz	1	0.001	[45]
FemaleSpeech1	48 kHz	1	0.003	[45]
FemaleSpeech2	48 kHz	1	0.001	[45]
MaleSpeech1	48 kHz	1	0.002	[45]
MaleSpeech2	48 kHz	1	0.001	[45]

For audio measurements, we employ the signals listed in Table 4. The properties comprise different content, sampling rates, number of channels, and audio signal powers. Most of the files are sampled at 48 kHz. We downsampled four of them to 16 kHz and 44.1 kHz. Furthermore, seven of the sequences originally providing 8 channels are encoded with two channels (channel 1 and channel 2) to obtain stereo audio. All sequences are encoded at a 128 kbps bitrate. Additionally, four of them are encoded at 64 kbps, 192 kbps, and 256 kbps. In our measurements, we consider both MP3- and AAC-encoded audio files.

Considering the speaker power, we assume that the powers of the chosen audio files are representative. A power of zero is obtained using the 'Silence' audio file. The volume is considered by measuring some audio files plus the silent file in the mute, medium volume, and maximum volume cases.

Most measurements are conducted performing pure audio or pure video streaming. However, we take into account that the power in simultaneous audio and video streaming could differ from the power in pure audio or video streaming such that additional combined streams are tested. Therefore, we choose a subset of 8 video sequences and a subset of 8 audio files for combined measurements. Table 5 lists these files and the combinations.

Due to restrictions on the software or the DUT, and to reduce the number of required measurements, several components were not measured for all test cases. These are the source (which can be the local memory or a network), the display as well as the speaker (which are external for device A), and the video as well as the audio codec. The exact tested settings for each hardware-software combination (HSC) are listed in Table 6.

TABLE 5. Video sequences and audio files for audio-video measurements. The combinations are shown in the two rightmost columns.

Video		Audio		Combination	
	b_v		b_a		
Name	[Mbps]	Name	[kbps]	i	ii
1. BQSquare	1.7	a. FemaleSpeech2	64	1a	1d
2. BQSquare	0.16	b. FemaleSpeech2	128	2b	2e
3. BQTerrace	37	c. FemaleSpeech2	256	3c	3f
4. BQTerrace	1.6	d. Orchestra	128	4d	4g
5. Kimono	14.2	e. Callcenter1	128	5e	5h
6. Kimono	1.50	f. RockMusic	128	6f	6a
7. RaceHorses	1.56	g. Conference1	128	7g	7b
8. RaceHorses	0.22	h. Silence	128	8h	8c

TABLE 6. Tested settings for the hardware-software combinations (HSCs) indicated on the first two lines.

Hardware	A		B	C
Software	I	II	I	I
Local	YES	NO	YES	YES
RJ45	YES	YES	NO	NO
Wi-Fi	YES	YES	YES	YES
3G	NO	NO	YES	NO
H.264	YES	YES	YES	YES
HEVC	YES	NO	NO	NO
AAC	YES	YES	YES	YES
MP3	YES	NO	NO	NO
Display	NO	NO	YES	YES
Speaker	NO	NO	YES	YES

IV. MODEL TRAINING AND VALIDATION

We perform model training and validation separately for each HSC. This is done because the power characteristics of different hardware and software implementations can differ significantly. We consider application I on all three devices and application II only on device A.

For model training on each HSC, we perform M measurements, where for each measurement, the corresponding vector of input variables $\boldsymbol{v}$ of length K is determined. Then, we accumulate all row vectors $\boldsymbol{v}$ into the $M \times K$-matrix $\boldsymbol{V}$, such that we can rewrite (1) to

$$\hat{\boldsymbol{P}} = \boldsymbol{V} \cdot \boldsymbol{x}, \tag{16}$$

where $\hat{\boldsymbol{P}}$ is the vector of M estimated powers.

To be able to interpret the optimal parameter values in $\boldsymbol{x}$, we normalize all variables in $\boldsymbol{V}$ to the range $[0, 1]$ and obtain $\boldsymbol{V}_\text{norm}$. For example, the brightness levels H, which were originally given in the range $H \in [0, 255]$, are divided by the respective maximum value $H_\text{max} = 255$.

Then, to obtain optimal parameter values for the vector $\boldsymbol{x}$, we minimize the sum of squared errors between the measured powers P_m and the estimated powers $\hat{P}_m$ as

$$\min_{\boldsymbol{x} \in \mathbb{R}^K} \left(\varepsilon = \sum_{m=1}^{M} \left(\hat{P}_m - P_m \right)^2 \right), \tag{17}$$

where P_m and $\hat{P}_m$ denote the m-th entry of the vectors $\boldsymbol{P}$ and $\hat{\boldsymbol{P}}$, respectively. This is done by using a trust-region reflective algorithm [46]. Furthermore, as we expect all variables to have a positive relation to the power consumption, we define the lower bound of all parameters $\boldsymbol{x}$ to be zero.

To show the validity of the model, we perform cross-validation to strictly separate the training from the validation data. Therefore, we split the set of measurements into five subsets. The first subset contains measurements with the 'BlackScreen' and the 'WhiteScreen' video sequences and the 'Silent' audio file. For the other four subsets, the remaining measurements are ordered by audio and video sequences and split in such a way that no source file (audio or video) occurs in two subsets. Then, in five iterations, four of the five sets are used for model training, and the remaining set is used for validation.

The models are assessed using the mean of the absolute relative estimation error given by

$$\bar{\varepsilon} = \frac{1}{M}\sum_{m=1}^{M}\left|\frac{\hat{P}_m - P_m}{P_m}\right|. \tag{18}$$

V. MODEL OPTIMIZATIONS

The main target of this paper is to determine the most important factors describing the overall power consumption during video streaming. Therefore, we propose to use a feature selection approach to prune the original set of input variables to the most important ones.

The proposed selection algorithm is based on the well-known wrapper approach [47] with a special selection criterion. In this approach, the optimal set of features is determined iteratively, where in each iteration k, one feature is dropped after one training instance (backward selection). During each training instance (loop over c), all HSCs are trained separately. The whole process is described in Algorithm 1.

```
1 for $k \leftarrow \{K, K-1, K-2, \ldots, 2, 1\}$ do
2     for $c \leftarrow \{A-I, A-II, B-I, C-I\}$ do
3         Train $\boldsymbol{x_c}$ with $k$ remaining variables (17);
4     end
5     $\bar{\boldsymbol{x}} = \mathrm{mean}(\boldsymbol{x}_{\mathrm{A-I}}, \boldsymbol{x}_{\mathrm{A-II}}, \boldsymbol{x}_{\mathrm{B-I}}, \boldsymbol{x}_{\mathrm{C-I}})$;
6     $i \leftarrow \arg\min \bar{\boldsymbol{x}}$;
7     Remove the feature with the index $i$ from $\boldsymbol{V}$;
8 end
```

Algorithm 1 Feature Selection Algorithm [47]

In the first iteration, we begin using all features, i.e., all variables from Table 1 are included in the model ($k = K$). In ll. 2-4, we train the model using (17) for all HSCs and save the corresponding estimation errors (18). Then, we consider the values of the trained parameters $\boldsymbol{x_c}$ in order to determine the feature i that will be dropped at the end of the iteration. This is done by calculating the mean of the parameter values over all HSCs $\bar{\boldsymbol{x}}$ (l. 5), where unused parameters are neglected by the mean-operator.[2] The mean parameter yielding the smallest value is then assigned the index i (l. 6). As all input variables are normalized to the range [0, 1] (cf. Section IV), the small value of the mean parameter indicates that the corresponding feature i has the lowest mean impact on the overall value of the estimated power consumption $\hat{\boldsymbol{P}}$ for all HSCs. Hence, in l. 7 we remove the corresponding variable from $\boldsymbol{V}$ because we expect it to have the smallest impact on the mean estimation error. In the next iteration, we repeat this process with $k = K - 1$ variables and again remove the variable showing the smallest impact. This process is repeated until no variable is left ($k = 0$).

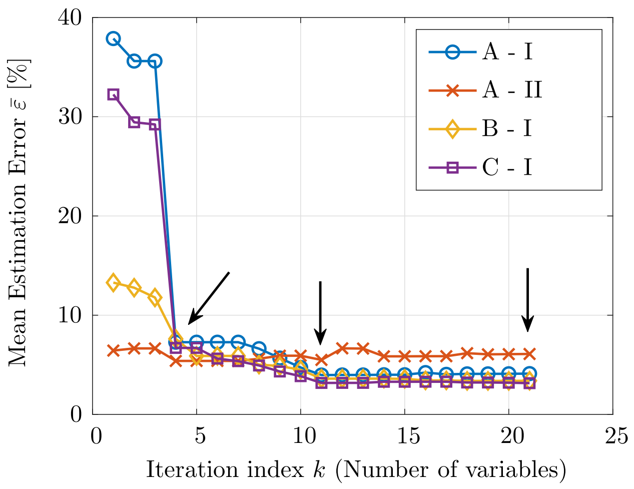


FIGURE 4. Mean estimation errors for all HSCs (legend) per iteration index k (cross-validated). Points of interest, which are further discussed in Section VI, are marked by black arrows.

In the algorithm, the mean value over the four HSCs is chosen because of the following reason. The goal of the proposed power model is to find variables that allow accurate power modeling independent from the used hardware and software. Therefore, in each iteration, we train separately (because the trained parameter values depend on the used hardware and software), but we discard variables that are of low importance for all HSCs, which is reflected by an overall small mean value.

The result of this process is depicted in Fig. 4. We can see the mean estimation errors $\bar{\varepsilon}$ (vertical axis) for all HSCs over the iteration index k (horizontal axis). Due to backwards counting in the loop, the first iteration corresponds to $k = 21$. Additionally, k represents the number of variables used for modeling per iteration.

VI. DISCUSSION

As points of interest, we choose $k = 21$, $k = 11$, and $k = 4$ (see black arrows). The former point is chosen because it represents the full model from Table 1. The point $k = 11$ is

[2]Unused parameters occur when a certain part of the streaming pipeline is not part of the measured HSC. For example, the display is not part of the A-I setup such that the corresponding parameters F_{display}, H, and L are not used to calculate $\bar{\boldsymbol{x}}$.

TABLE 7. Mean estimation errors $\bar{\varepsilon}$ for the models with $k = 4$, $k = 11$, and $k = 21$ features for all HSCs.

Model	A - I	A - II	B - I	B - II
$k = 4$	7.28%	5.39%	7.61%	6.70%
$k = 11$	3.97%	5.49%	3.61%	3.17%
$k = 21$	4.11%	6.09%	3.39%	3.15%

chosen because it reaches similar estimation errors compared to $k = 21$ with a significantly reduced number of variables. Finally, $k = 4$ is chosen because it reaches estimation errors below 10% with the smallest number of variables.

The estimation errors for the three considered models are summarized in Table 7. We can see that in terms of modeling accuracy, it is generally not helpful to consider all variables listed in Table 1, because in some cases, the estimation error is even higher for $k = 21$ than for $k = 11$, which can be explained by overfitting. Still, it is possible to obtain decent estimation errors (below 8%) using four variables ($k = 4$), which include a constant offset power Π_0, the bitrate in the case of Wi-Fi streaming b_{wifi} (audio plus video bitrate when the Wi-Fi connection is used for streaming), the linear brightness of the display H, and the frame rate of the video f_{v}.

To evaluate the impact of the features in more detail, Table 8 lists the optimally trained parameter values for the model case of $k = 11$ and HSC B-I. The order of the variables is chosen in such a way that the index k corresponds to the iteration in which the corresponding variable as well as the corresponding parameter was removed. Please note that trained values differ for the other HSCs and for other iterations.

Due to training on normalized variables, the values can be interpreted as the maximum power contribution of the corresponding variable. For example, when the frame rate attains its maximum value $f_{\mathrm{v}} = 60$ (cf. Table 3), the power attributed to the frame rate would yield the product of the parameter value with the normalized frequency given as $1 \cdot 0.35\,\mathrm{W} = 0.35\,\mathrm{W}$. When the frame rate is halved, the power contribution yields $\frac{30\,\mathrm{Hz}}{60\,\mathrm{Hz}} \cdot 0.35\,\mathrm{W} = 0.175\,\mathrm{W}$.

We can see that the parameter $\Pi_0 = 0.90\,\mathrm{W}$ shows the highest value, which means that most of the power is constant, i.e., it cannot be controlled by changing input parameters. However, power can be saved by changing the display brightness (maximum contribution 0.79 W) and the video's frame rate (maximum contribution 0.35 W at the maximum frame rate).

Furthermore, the bitrate of the video stream has a high influence (0.38 W), which is even higher when using the Wi-Fi connection (an additional power of 0.21 W). Please note that the Wi-Fi streaming bitrate b_{wifi} is only nonzero when the Wi-Fi connection is used for streaming. As the bitrates for Ethernet and 3G streaming are already dropped for $k = 11$, this is an indication that Wi-Fi streaming needs significantly more power than the other networks. Besides, it is striking that for $k = 4$, only the Wi-Fi streaming bitrate

TABLE 8. Features per iteration k for $k = 1$ to $k = 11$ variables. For each iteration k, all variables with a smaller k are included. Trained parameter values on the right are given for HSC B-1, $k = 11$. All entries are included in the model with $k = 11$, bold entries are also included in the model with $k = 4$.

k	Variable	Description	Param. value ($k = 11$)	
1	1	**Constant offset**	$\Pi_0 =$	0.90
2	b_{wifi}	**Bitrate (Wi-Fi)**	$\beta_{\mathrm{wifi}} =$	0.21
3	H	**Lin. display brightness**	$\kappa =$	0.79
4	f_{v}	**Video frame rate**	$\epsilon_{\mathrm{frame}} =$	0.35
5	$F_{3\mathrm{G}}$	3G connection offset	$\Gamma_{3\mathrm{G}} =$	0.49
6	L	Quadr. disp. brightness	$\lambda =$	0.00
7	F_{v}	Video decoding offset	$\Psi_0 =$	0.00
8	b_{v}	Video bitrate	$\epsilon_{\mathrm{bit}} =$	0.38
9	F_{wifi}	Wi-Fi connection offset	$\Gamma_{\mathrm{wifi}} =$	0.17
10	F_{a}	Audio decoding offset	$\Phi =$	0.11
11	G	Pixels per second	$\rho =$	0.20

remains in the model, which shows that the bitrate in Wi-Fi streaming has a higher influence on the streaming power than on the video processing power.

For the video processing branch, the most important variables are the linear display brightness H and the frame rate f_{v}, because they are the only variables left for $k = 4$. For a higher estimation accuracy, the quadratic display brightness, the decoding offset F_{v}, the video bitrate b_{v}, and the pixels per second G can additionally be considered as they are selected for $k = 11$. The fact that both linear and quadratic display brightness are selected for $k = 11$ indicates that a purely linear approach, which was proposed in [30], is not sufficient to accurately model the power consumption of a display.

For the audio branch, results indicate that audio processing and the speaker have a rather small influence, which can sufficiently well be modeled by a constant ($\Phi = 0.11\,\mathrm{W}$). Considering a very simple model with $k = 4$, audio can even be neglected because no corresponding variable is left.

VII. CONCLUSION

This paper analyzed the power consumption of video streaming applications on portable devices. It is the first work that considers the power behavior of the complete device. To this end, a model summarizing various high-level models of the most important functional components was constructed. An analysis performed with the help of feature selection revealed that there are three major components that can be exploited to manipulate and optimize the power consumption: display brightness, the frame rate of the video, and the streaming bitrate.

Future work can exploit this information to determine the video parameters leading to the best compromise between power consumption and perceptual video quality. Furthermore, modern video content such as 4K video, 3D, or the 360° projection formats can be taken into account. Finally, it would be interesting to perform such an evaluation on the sender side including video capture, encoding, and transmission.

REFERENCES

[1] Cisco. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022*. Accessed: Feb. 2019. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html

[2] The Shift Project. (2019). *Climate Crisis: The Unsustainable Use of Online Video': Our New Report on the Environmental Impact of ICT*. [Online]. Available:'https://theshiftproject.org/en/article/unsustainable-use-online-video/

[3] M. Jamali and S. Coulombe, "Fast HEVC intra mode decision based on RDO cost prediction," *IEEE Trans. Broadcast.*, vol. 65, no. 1, pp. 109–122, Mar. 2019.

[4] K.-H. Tai, M.-Y. Hsieh, M.-J. Chen, C.-Y. Chen, and C.-H. Yeh, "A fast HEVC encoding method using depth information of collocated CUs and RD cost characteristics of PU modes," *IEEE Trans. Broadcast.*, vol. 63, no. 4, pp. 680–692, Dec. 2017.

[5] L. Zhu, Y. Zhang, Z. Pan, R. Wang, S. Kwong, and Z. Peng, "Binary and multi-class learning based low complexity optimization for HEVC encoding," *IEEE Trans. Broadcast.*, vol. 63, no. 3, pp. 547–561, Sep. 2017.

[6] E. Yaacoub, Z. Dawy, S. Sharafeddine, and A. Abu-Dayya, "Joint energy-distortion aware algorithms for cooperative video streaming over LTE networks," *Signal Process., Image Commun.*, vol. 28, no. 9, pp. 1114–1131, Oct. 2013.

[7] Y. I. Choi and C. G. Kang, "Scalable video coding-based MIMO broadcasting system with optimal power control," *IEEE Trans. Broadcast.*, vol. 63, no. 2, pp. 350–360, Jun. 2017.

[8] L. Zou, R. Trestian, and G.-M. Muntean, "E3DOAS: Balancing QoE and energy-saving for multi-device adaptation in future mobile wireless video delivery," *IEEE Trans. Broadcast.*, vol. 64, no. 1, pp. 26–40, Mar. 2018.

[9] R. Martinez Alonso, D. Plets, M. Deruyck, L. Martens, G. Guillen Nieto, and W. Joseph, "TV white space and LTE network optimization toward energy efficiency in suburban and rural scenarios," *IEEE Trans. Broadcast.*, vol. 64, no. 1, pp. 164–171, Mar. 2018.

[10] X. Li, Z. Ma, and F. C. A. Fernandes, "Modeling power consumption for video decoding on mobile platform and its application to power-rate constrained streaming," in *Proc. Vis. Commun. Image Process. (VCIP)*, San Diego, CA, USA, Nov. 2012, pp. 1–6.

[11] T. Mallikarachchi, D. S. Talagala, H. K. Arachchi, and A. Fernando, "A feature based complexity model for decoder complexity optimized HEVC video encoding," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Las Vegas, NV, USA, Jan. 2017, pp. 366–369.

[12] C. Herglotz, A. Heindel, and A. Kaup, "Decoding-energy-rate-distortion optimization for video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 1, pp. 171–182, Jan. 2019.

[13] R. Yang, M. Xu, Z. Wang, Y. Duan, and X. Tao, "Saliency-guided complexity control for HEVC decoding," *IEEE Trans. Broadcast.*, vol. 64, no. 4, pp. 865–882, Dec. 2018.

[14] Q. Liu, Z. Yan, and C. W. Chen, "Cloud-based video streaming with systematic mobile display energy saving: Rate-distortion-display energy profiling," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Phoenix, AZ, USA, Sep. 2016, pp. 1504–1508.

[15] A. Carroll and G. Heiser, "The systems hacker's guide to the galaxy energy usage in a modern smartphone," in *Proc. 4th Asia–Pacific Workshop Syst. (APSys)*, Singapore, 2013, pp. 1–7.

[16] C. Herglotz, S. Coulombe, A. Vakili, and A. Kaup, "Power modeling for virtual reality video playback applications," in *Proc. IEEE 23rd Int. Symp. Consum. Technol. (ISCT)*, Ancona, Italy, Jun. 2019.

[17] C. Herglotz, A. Kaup, S. Coulombe, and A. Vakili, "Power-efficient video streaming on mobile devices using optimal spatial scaling," in *Proc. IEEE 9th Int. Conf. Consum. Electron. (ICCE-Berlin)*, Berlin, Germany, Sep. 2019, pp. 1–6.

[18] L. Zou, A. Javed, and G.-M. Muntean, "Smart mobile device power consumption measurement for video streaming in wireless environments: WiFi vs. LTE," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Cagliari, Italy, Jun. 2017, pp. 1–6.

[19] L. Sun, R. K. Sheshadri, W. Zheng, and D. Koutsonikolas, "Modeling WiFi active power/energy consumption in smartphones," in *Proc. IEEE 34th Int. Conf. Distrib. Comput. Syst.*, Madrid, Spain, Jun. 2014, pp. 41–51.

[20] T. Simunic, L. Benini, and G. De Micheli, "Energy-efficient design of battery-powered embedded systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 1, pp. 15–28, Feb. 2001.

[21] J. Dolezal and Z. Becvar, "Methodology and tool for energy consumption modeling of mobile devices," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Apr. 2014, pp. 34–39.

[22] P. Raoufi and J. Peters, "Energy-efficient wireless video streaming with H.264 coding," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, San Jose, SA, USA, Jul. 2013, pp. 1–6.

[23] C. Herglotz, D. Springer, M. Reichenbach, B. Stabernack, and A. Kaup, "Modeling the energy consumption of the HEVC decoding process," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 1, pp. 217–229, Jan. 2018.

[24] D. Sostaric, D. Vinko, and S. Rimac-Drlje, "Power consumption of video decoding on mobile devices," in *Proc. ELMAR*, Sep. 2010, pp. 81–84.

[25] C. Herglotz and A. Kaup, "Decoding energy estimation of an HEVC hardware decoder," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Firenze, Italy, May 2018, pp. 1–5.

[26] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, New York, NY, USA, 2012, pp. 225–238.

[27] F. Xu, Y. Liu, Q. Li, and Y. Zhang, "V-edge: Fast self-constructive power modeling of smartphones based on battery voltage dynamics," in *Proc. 10th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, vol. 13, 2013, pp. 43–56.

[28] R. Ren, M. Raulet, C. Sanz, E. Juarez, and F. Pescador, "Energy estimation models for video decoders: Reconfigurable video coding-CAL case-study," *IET Comput. Digit. Techn.*, vol. 9, no. 1, pp. 3–15, Jan. 2015.

[29] Z. He, W. Cheng, and X. Chen, "Energy minimization of portable video communication devices based on power-rate-distortion optimization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 5, pp. 596–608, May 2008.

[30] J.-W. Kim, K.-H. Lee, J.-G. Bae, H.-G. Kim, and J.-O. Kim, "Optimal liquid crystal display backlight dimming based on clustered contrast loss," *Opt. Eng.*, vol. 54, no. 10, Oct. 2015, Art. no. 103112.

[31] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proc. USENIX Conf.*, Boston, MA, USA, Jun. 2010, p. 21.

[32] B. Girod, R. Rabenstein, and A. Stenger, *Signals and Systems*. New York, NY, USA: Wiley, 2001.

[33] D. Hackenberg, R. Schone, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer, "An energy efficiency feature survey of the intel haswell processor," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshop*, May 2015, pp. 896–904.

[34] D. Correa, G. Correa, D. Palomino, and B. Zatt, "OTED: Encoding optimization technique targeting energy-efficient HEVC decoding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[35] N. Nethercote and J. Seward, "Valgrind: A framework for heavyweight dynamic binary instrumentation," *ACM SIGPLAN Notices*, vol. 42, no. 6, pp. 89–100, Jun. 2007.

[36] (Mar. 2020). *Performance Application Programming Interface (PAPI)*. [Online]. Available: http://icl.cs.utk.edu/papi/

[37] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz, "Performance and computational complexity assessment of high-efficiency video encoders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1899–1909, Dec. 2012.

[38] E. Monteiro, M. Grellert, S. Bampi, and B. Zatt, "Energy-aware cache assessment of HEVC decoding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Montreal, QC, Canada, May 2016, pp. 574–577.

[39] *Adobe Flash Video File Format Specification Version 10.1*, Adobe Syst., San Jose, CA, USA, Aug. 2010. [Online]. Available: https://www.adobe.com/content/dam/acom/en/devnet/flv/video_file_format_spec_v10_1.pdf

[40] S.-Y. Tseng, K.-H. Lin, W.-S. Wang, C.-T. King, and S.-H. Chang, "Performance and power consumption analysis of DVFS-enabled H.264 decoder on heterogeneous multi-core platform," in *Proc. 10th IEEE Int. Conf. Comput. Inf. Technol.*, Bradford, U.K., Jun. 2010, pp. 1758–1763.

[41] VideoLAN. (2020). *VLC Media Player*. Accessed: Mar. 2020. [Online]. Available: http://www.videolan.org/vlc/

[42] J. Bendat and A. Piersol, *Random Data: Analysis and Measurement Procedures*. Hoboken, NJ, USA: Wiley, 1971.

[43] F. Bossen, *Common Test Conditions and Software Reference Configurations*, Standard JCTVC-L1100, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Geneva, Switzerland, Switzerland, Tech. Rep, Jan. 2013.

[44] *Speech and Multimedia Transmission Quality (STQ); Speech Quality Performance in the Presence of Background Noise; Part 1: Background Noise Simulation Technique and Background Noise Database*, Standard ETSI ES 202 396-1 V1.7.1 (2017-10), 2017.

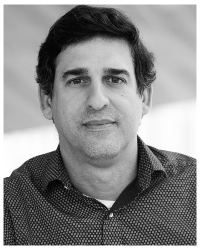
[45] P. Kabal. (2002). TSP Speech Database. McGill University, Database Version. [Online]. Available: http://www-mmsp.ece.mcgill.ca/Documents/Data/

[46] T. F. Coleman and Y. Li, "An interior trust region approach for nonlinear minimization subject to bounds," *SIAM J. Optim.*, vol. 6, no. 2, pp. 418–445, May 1996.

[47] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "Feature selection for high-dimensional data," *Prog. Artif. Intell.*, vol. 5, no. 2, pp. 65–75, 2016.

CHRISTIAN HERGLOTZ (Member, IEEE) received the Dipl.-Ing. degree in electrical engineering and information technology and the Dipl.-Wirt. Ing. degree in business administration and economics from Rheinisch-Westfälisch Technische Hochschule (RWTH) Aachen University, Germany, in 2011 and 2012, respectively, and the Dr.-Ing. degree from Friedrich-Alexander University Erlangen-Nürnberg (FAU), Germany, in 2017. Since 2012, he has been a Research Scientist with the Chair of Multimedia Communications and Signal Processing, Friedrich-Alexander University Erlangen-Nürnberg (FAU), Germany. In 2018 and 2019, he worked as a Postdoctoral Fellow at the École de technologie supérieure (ÉTS) in collaboration with Summit Tech Multimedia Communication Inc., Montréal, QC, Canada, on energy efficient VR technologies. Since 2019, he has also been a Senior Scientist with Friedrich-Alexander University Erlangen-Nürnberg (FAU). His current research interests include energy efficient video communications and video coding.

STÉPHANE COULOMBE (Senior Member, IEEE) received the B.Eng. degree in electrical engineering from the École Polytechnique de Montréal, Canada, in 1991, and the Ph.D. degree in telecommunications (image processing) from INRS-Telecommunications, Montreal, in 1996. From 1997 to 1999, he was with the Nortel Wireless Network Group in Montreal, and from 1999 to 2004, he worked with the Nokia Research Center, Dallas, TX, USA, as a Senior Research Engineer and as a Program Manager in the Audiovisual Systems Laboratory. He joined ÉTS, in 2004, where he currently carries out research and development on video processing and systems, compression, and transcoding. From 2009 to 2018, he has held the Vantrix Industrial Research Chair in Video Optimization. He is also a Professor with the Department of Software and IT Engineering, École de Technologie Supérieure (ÉTS is a constituent of the Université du Québec network).



CARLOS VAZQUEZ (Senior Member, IEEE) received the B.Eng. and M.Sc. degrees from the Technical University of Havana (CUJAE), in 1992 and 1997 respectively, and the Ph.D. degree from the INRS-Energy, Materials and Telecommunications (EMT) in Montréal, QC, Canada, in 2003. He has been an Assistant Professor with the Department of Software and IT Engineering, École de Technologie Supérieure (ÉTS-Montréal), since 2013. His research interests are in the general fields of image and video processing and computer vision, more specifically he is interested in 3D reconstruction from images, quality evaluation of immersive visual content, augmented reality and 3D-360 video processing, and analysis and coding.

AHMAD VAKILI received the B.Sc. and M.Sc. degrees in Electrical Engineering from Tehran Polytechnic, and the Ph.D. degree in telecommunications from INRS, Montreal. Prior to his Ph.D., he had more than six years of professional work experience, as a Lecturer and an Electrical Engineer. He is currently the Chief Research and Development Officer with Summit Tech Multimedia Communications Inc. His recent research mainly focuses on 360-8K-VR video codecs and transmission, AI projects, such as voice authentication and recognition, RCS in the IoT, and so on. His research interests include multimedia communication services, VR, video codec, RCS, AI, QoS, and QoE management.

ANDRÉ KAUP (Fellow, IEEE) received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering from RWTH Aachen University, Aachen, Germany, in 1989 and 1995, respectively. He was with the Institute for Communication Engineering, RWTH Aachen University, from 1989 to 1995 and joined Siemens Corporate Technology, Munich, Germany, in 1995, where he became the Head of the Mobile Applications and Services Group, in 1999. Since 2001 he has been a Full Professor and the Head of the Chair of Multimedia Communications and Signal Processing, Friedrich-Alexander University Erlangen-Nürnberg (FAU), Germany. From 2015 to 2017, he has served as the Head of the Department of Electrical Engineering and the Vice Dean of the Faculty of Engineering, FAU. He has authored around 350 journal and conference papers and has over 120 patents granted or pending. His research interests include image and video signal processing and coding. He is a member of the IEEE Multimedia Signal Processing Technical Committee and a member of the scientific advisory board of the German VDE/ITG. He was named Siemens Inventor of the Year 1998 and received several best paper awards. His group won the Grand Video Compression Challenge at the Picture Coding Symposium 2013. The Faculty of Engineering at FAU honored him with the Teaching Award, in 2015. He is a member of the Bavarian Academy of Sciences.

JEAN-CLAUDE GRENIER enlisted at Ahuntsic College for a degree in software development until 2004. Afterwards, in 2005, he moved on to the Université du Québec à Montréal, where he received the B.Sc. degree in software Engineering, in 2008. After a few internships during his studies, he accepted a position in a video game company due to his interest for the vast complexity the domain offered. He worked in this industry for almost ten years before moving to the field of telecoms. He started working at Summit Tech Multimedia in 2015 on a small team exploring the possibilities virtual reality had to offer to the telecommunications field. He is currently the Lead Developer of an expanding team of passionate people pushing the boundaries of communications.