Power Overwhelming: Quantifying the Energy Cost of Visualisation

Christoph Müller* Visualisierungsinstitut der Universität Stuttgart
Moritz Heinemann† Visualisierungsinstitut der Universität Stuttgart
Daniel Weiskopf‡ Visualisierungsinstitut der Universität Stuttgart
Thomas Ertl§ Visualisierungsinstitut der Universität Stuttgart

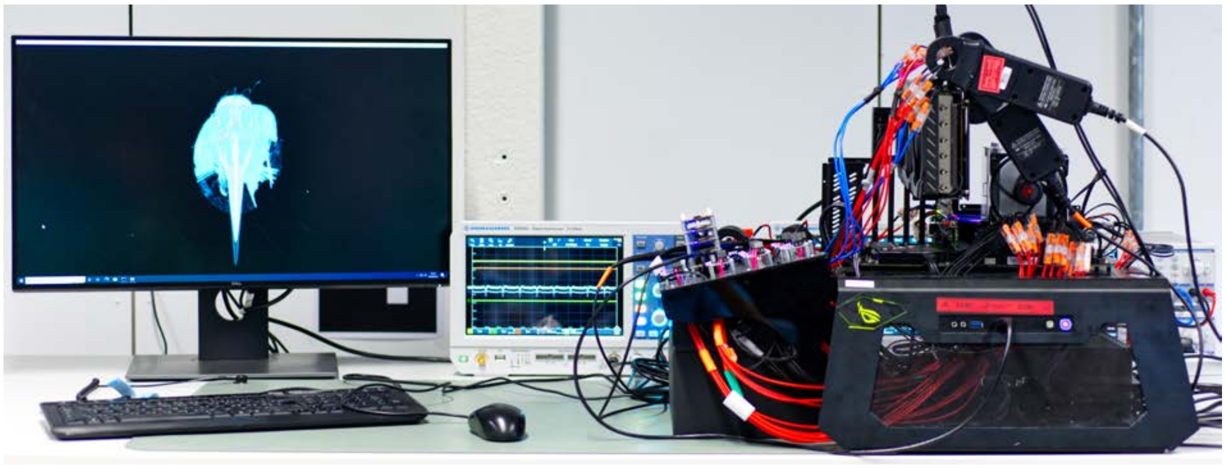


Figure 1: The test bench used in our experiments: All significant power rails of the system are redirected through Texas Instruments *INA226* power monitors on Tinkerforge bricklets. Additionally, a Rohde & Schwarz *HMC8015* power analyser is used to log the total power draw of the power supply. Oscilloscopes can be used to probe the power rails of the GPU at the probing points of the riser card installed between the PEG slot and the GPU.

Abstract

GPUs are the power-hungry tool of many visualisation researchers. However, their energy consumption has mostly been investigated outside the visualisation community, albeit our algorithms can generate more complex workloads than compute kernels. Additionally, a raising number of web-based visualisations potentially makes consumers other than the GPU more relevant. We present measurement setups for quantifying the energy cost of visualisation, ranging from software sensors over external power meters and micro controller-based setups to using oscilloscopes. These setups cover energy consumption of GPUs, CPUs and other components of a computing system. Using raycasting of spherical glyphs, volume rendering and D3 visualisations as examples, we show that there are viable options for evaluating most kinds of visualisations. We conclude by stating the challenges to a broader application of these techniques and by making recommendations on how to overcome these.

Index Terms: Hardware—Power and energy—Impact on the environment; Hardware—Hardware test—Board- and system-level test; Human-centered computing—Visualization—Visualization systems and tools

1 Introduction

Graphics processing units (GPUs) and their enormous parallelism and computational power are a key tool enabling many of the advances in visualisation in the past decades. However, this computational power comes at the cost of GPUs being the single most energy-hungry component in a computer nowadays, wherefore one would expect that energy consumption and the influence of visualisation algorithms on it being under close scrutiny [23]. This has not been the case so far, and we argue for a systematic investigation of energy consumption of visualisations. Several methods of different complexity exist for doing that, which we will compare in this paper using a few exemplary algorithms as a blueprint for what will hopefully expand into a broader initiative, eventually providing an overall understanding of the energy cost of visualisation.

The measurement procedures are largely borrowed from research on energy consumption of graphics accelerators and its optimisation in software in the context of high-performance computing (HPC), where GPUs facilitate massively increasing the compute density in clusters [7,30]. On the flip side, their energy hunger becomes the limiting factor when reaching the next petaflop or exaflop target, albeit a superior computation-per-Watt efficiency compared to traditional processors caused a leading manufacturer of discrete graphics cards to tout that "The future of HPC is green". Furthermore, GPU-based systems are nowadays prevalent when it comes to training machine learning (ML) models and performing inference. These are widely used in industry where the electricity bill becomes a relevant factor, which sparked some research in optimising power consumption of ML kernels [19,21]. And in the mobile space, running on a battery necessitates at least some economic handling of power to maximise battery life [15].

Visualisation research, in contrast, seems to widely ignore this aspect while focusing on positive effects that might inspire the public to save electricity [17]. In turn, we know surprisingly little about the influence of software and algorithms on energy consumption in particular. On the lowest level, one might ask whether one of two distinctly different algorithms for achieving the same result is more energy efficient than the other, or whether and when a higher frame rate correlates or anticorrelates with a higher power draw. Does increasing the sampling rate – and thus the image quality – of a volume renderer also increase energy consumption?

*e-mail: christoph.mueller@visus.uni-stuttgart.de
†e-mail: moritz.heinemann@visus.uni-stuttgart.de
‡e-mail: daniel.weiskopf@visus.uni-stuttgart.de
§e-mail: thomas.ertl@visus.uni-stuttgart.de

Are dedicated ray-tracing units on the GPU more power-efficient than using compute or pixel shaders? Do browser-based visualisation techniques primarily draw power via the CPU or via the GPU? How does interacting with a visualisation influence the energy cost? Are the findings from the HPC area transferable to the graphics pipeline? The answer to these questions is to a large extent: We don't know.

Information and communications technology (ICT) as a whole was estimated to account for more than 2 % of the yearly global emissions of carbon dioxide in 2018 [24] – roughly the same as all air traffic. Predictions see it account for 20 % of the total electricity demand within a decade or two, albeit computing hardware becoming more and more power efficient at the same time [4, 20] – virtually a prototype of the so-called *rebound effect*. The largest fraction of this increase is assumed to be for the "communication" part of ICT [24], so one could argue that it is unlikely that the relatively small number of users pushing their discrete GPUs to the limit for visualisation would have a big impact. However, we argue that it is irresponsible to rely on "believing" and "guessing" for facts that can be empirically quantified by simply measuring them.

As a first step in this direction, we evaluated the energy consumption of two scientific visualisation techniques and several web-based information visualisations on *Observable HQ*. We performed measurements using different techniques, describe how to set these up (cf. Fig. 1) and which of them are most suitable for evaluating visualisation algorithms. As we obtained all numbers at the same time, we can compare the results against each other, thus finding a reasonable trade-off between reliable numbers and the financial and work effort required for obtaining them. Our numbers show that there are interesting things to discover in the area while obtaining reliable measurements is almost trivial under certain conditions. We also discuss challenges and problems for the non-trivial cases and contemplate how we as a research community can foster the collection of power-related data in publications.

2 Background on measuring power consumption

Before going into details, we briefly want to recapitulate some basics: By measuring the electric potential difference, or voltage $U[\mathrm{V}]$, and the electric current $I[\mathrm{A}]$, we can derive the instantaneous apparent power $S[\mathrm{V\,A}] = U \cdot I$. As the circuits on the low-voltage side of a PC power supply run on direct current (DC), the real power is the same: $P[\mathrm{W}] = S = U \cdot I$. Voltage can be measured using voltmeters or oscilloscopes. Measuring current is, in our scenario, typically performed by either introducing shunt resistors in the circuit or non-destructively by using Hall-effect sensors [36]. When using shunts, we exploit Ohm's law, stating for a resistor of known resistance $R[\Omega]$ the voltage is proportional to the current: $U = R \cdot I$. When measuring the voltage drop across the shunt, we can simply solve this for I. Hall-effect sensors exploit the fact that charges running through a conductor in a magnetic field that is perpendicular to the conductor experience a force which in turn induces the so-called Hall voltage. This voltage is again proportional to the current.

2.1 GPU management libraries

The easiest way to obtain power measurements from a GPU without adding sensors at all is by querying the driver for it. Both major manufacturers of discrete GPUs expose such functionality via their management libraries, the *NVIDIA Management Library (NVML)* [33] and the *AMD Display Library (ADL)* [3]. Obviously, the reliability of that information depends on where the software gets its input data from. In case of NVIDIA's product, we know from the activities of the overclocking community (who engage in a technique called *shunt modding* to trick the GPU into allowing higher currents by changing the overall resistance of the shunts) that the manufacturer controls the overall power consumption of the whole board quite tightly [2]. In order to do so, NVIDIA has placed shunt resistors on effectively all of the power inputs of the

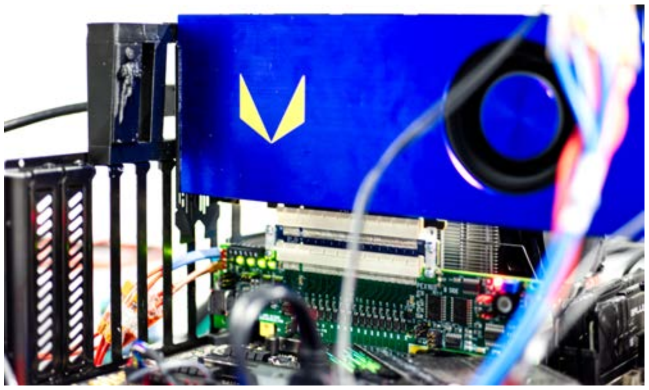


Figure 2: A GPU attached to a riser card used for intrusive measurement methods. The riser has probing points for the power provided through the PEG slot. On its back side, two cables for the 12 V and the 3.3 V rails have been soldered on the card. These are used to funnel the power through the power sensors on two Tinkerforge bricklets.

board, power cables and the PCI Express Graphics (PEG) slot [40]. We attribute this design to the fact to the assertions NVIDIA makes about the power draw of their data centre products, wherefore they require accurate measurements to throttle the hardware accordingly.

Less is known about AMD's implementation, except for that their numbers only include the core components, mostly the chip and the memory, but not all power-drawing components on the board [40]. Besides the insecurity about the source of the measurements, using management libraries has the obvious disadvantage that they only cover the GPU itself, not the other components in a computer.

2.2 External power meters

External power meters solve the latter problem as they measure the total power consumption of the computer. They are added between the wall socket and the power plug and are therefore easy to use. The devices come in different varieties ranging from simple ones for home use just displaying the instantaneous power draw to professional power analysers with continuous logging capabilities. If the goal is measuring the overall power cost of visualisation, this might be sufficient. However, such measurements do not provide any insight into the behaviour of individual consumers in the system, nor do they provide clear guidance for optimising power consumption.

2.3 Intrusive measuring methods

A third option is adding power meters between the power source and the consumer within the system. For GPUs, an appliance-like device exists in form of the *Power Capture Analysis Tool (PCAT)*, which is part of the *NVIDIA Reviewer Toolkit for Graphics Performance* [37]. This appliance comprises a riser card for measuring the power drawn from the PEG slot as well as pass-through sensors for the PCI Express (PCIe) power cables, allowing for externally measuring the overall power drawn by a PCIe card. Unfortunately, NVIDIA distributes these kits only to selected tech journalists and we could not get our hand on any of them. However, it is possible to build a similar setup using micro controllers. The idea is cutting the PCIe and ATX (Advanced Technology Extended) power cables in two and adding a probe with voltage meters and shunt resistors in between. This allows for measuring not only the GPU, but also the CPU and other components in the system. Our setup uses Tinkerforge *Bricklets* in a way similar to Wallossek [40] in a combination with a PCIe riser card (see Fig. 2) that exposes probing points for the power lines of the bus. The setup has a decent temporal resolution to observe running software over an extended period of time and can be controlled via a variety of programming and scripting languages.

2.4 Oscilloscopes

Oscilloscopes offer the possibility to measure voltage changes – and using special current probes also electric current – at a high temporal resolution. At a sampling rate of several hundreds of megahertz, it becomes possible to observe the power draw of the GPU within a single frame. To capture the complete behaviour of the GPU, a voltage and a current probe are required for each of the 12 V and the 3.3 V lines of the PEG slot as well as for each of the 12 V PCIe power cables [40]. As the voltage probes can be stuck into the power plugs and the current probes are contactless Hall sensors, the procedure is only minimally invasive. Physical modifications are only required to the riser card, which is needed to access the PEG slot: a wire loop needs to be soldered to the board to clamp the current probe around it. For measuring the voltage, some riser cards already provide dedicated measuring points where the probe can be attached. As most oscilloscopes have only two or four connectors, two devices need to be stacked to capture the whole picture, making this a particularly expensive solution. Furthermore, it generates a significant amount of data, which might not be necessary for learning about, or optimising, the power consumption of visualisation software.

3 Related work

As early as 2012, Johnsson et al. [23] performed a study of the power consumption of OpenGL- and OpenGL ES-based rendering algorithms, followed by a more detailed investigation of how to measure per-frame consumption [22]. They argue that power is becoming the limiting factor for further improvement in rendering performance due to the increase of current leakage when scaling down the size of transistors [6]. In their experiment, they used a custom-made sensing board using a combination of Hall-effect sensors and shunt resistors to measure all power inputs to discrete GPUs from AMD and NVIDIA as well as the integrated graphics of an Intel CPU and an *iPhone* SOC. While they observed that deferred rendering was more power efficient and faster than forward rendering in most cases, their results show that this is not trivially predictable and most notably that better frame rates do not consistently require more or less power than worse ones. Therefore, they concluded "that [power consumption] will become an integral part of most graphics research papers in the near future. We speculate that it will become as common to report joules per pixel as it is to report milliseconds per frame today". At least for visualisation research, this has not happened as of today – authors report energy only if they specifically focus on this topic. For instance, Heinemann et al. [15] analysed the influence of rendering parameters and rendering APIs on the power consumption of a volume raycaster on a mobile device. We can only surmise the reasons for that lack of empirical results: perhaps, there is a lack of awareness of the significant power draw of our tools; perhaps, our community just does not care; perhaps, building a custom sensing board is too high of an entry barrier.

Consequently, most of the research on power consumption focuses on the general purpose GPU (GPGPU) or HPC cases where scaling up the GPU density quickly raises the need for energy. Mittal et al.'s survey [30] not only investigates power efficiency of GPUs in this context, but also includes techniques for improving energy efficiency. Bridges et al. [7] later extended the problem space by not only surveying measuring methods, but also including model building for [18, 27, 28, 32] and simulation of [26, 35, 39] the power consumption of GPU clusters. Such models derive the power draw from low-level hardware counters, e. g. from tallying the number of shader invocations or texture accesses, instead of measuring it directly. Bridges et al. see a strong correlation between these counters and power use, although the complexity and parallelism of modern GPUs presents new hurdles for power modelling.

Starting with its *Kepler* architecture, NVIDIA provides relatively accurate numbers for instantaneous power draw from on-board sensors. Burtscher et al. [9] provide a methodology to derive an accurate value for energy consumption from these readings. However, they did not obtain ground truth using external ways of sensing like oscilloscopes and external power meters [11, 12], but based their method on a theoretical profile of the GPU's power draw.

Instead of using counters, models can also be derived from NVML power readings during micro benchmarks and based e. g. on instructions in NVIDIA's PTX assembly of the benchmarks. Arafa et al. [5] performed a series of micro benchmarks of PTX for measuring the power usage of low-level instructions like addition, division etc. They compared the results of sampling NVML in the thread submitting the GPU workload as well as in parallel with the more abstract and vendor-independent Performance Application Programming Interface (PAPI) [42] and ground truth obtained from hardware-based measurements. These were gained from a custom-made riser with shunt resistors for the PEG slot and an oscilloscope with a voltage probe and a Hall-effect power clamp for the PCIe cables. The authors conclude that sampling the internal sensors via NVML in a parallel thread to the actual workload yields the most accurate results. While the methodology of how to use NVML is transferable to graphics and visualisation workloads, micro benchmarking is a technique tailored towards the GPGPU application case. In contrast, graphics workloads are pipelined, comprise non-programmable stages like the rasteriser and output merging and are – depending on the API – subject to under-the-hood optimisations of the driver, such that it is more difficult to predict what the GPU is actually doing at a single point in time.

Dynamic voltage and frequency scaling (DVFS) is a way to actively influence GPU energy consumption. It adjusts the core and RAM frequency and voltage of GPUs to optimise the power draw while almost not affecting performance [1]. In some cases, significant energy savings up to 20 % can be achieved using this technique, but the success is highly dependent on the actual application [29].

Aside from the many GPGPU application cases, the power consumption of computer vision [34] and lately artificial intelligence applications [19, 21] have been researched. In case of the latter, the main goal is to balance power and performance based on the observation that GPU clusters for machine learning are most efficient if the GPUs are fully saturated. Finally, there exists work on modelling and reducing the power consumption of displays via optimised colour mappings for visualisations [10] and on optimising the energy budget for rendering applications [41].

Studying the energy consumption of visualisation algorithms requires benchmarks for such algorithms. Our work is relying on the one of Bruder et al. [8], who performed a systematic study of the runtime performance of rendering spherical glyphs and of volume rendering. While their sole focus was runtime performance and the relevant factors that need to be covered when evaluating it, their framework provides an automated way to test combinations of these factors, which we extended to include power measurements.

4 Experimental setup

We set up an experiment to test and compare the techniques described in Sect. 2, namely (i) using on-chip sensors accessible by management libraries provided by the GPU vendor, (ii) measuring the overall system power draw by means of an external power analyser, (iii) redirecting PCIe cables and the PEG slot through Tinkerforge sensors and (iv) attaching an oscilloscope.

4.1 Hardware

Our test hardware is a typical PC configuration: AMD *Ryzen 5900X*, ASUS *ROG Strix X570-E Gaming*, be quiet *Dark Power Pro* 1200 W power supply unit (PSU), 64 GB RAM, solid state drive (SSD), mounted on a test bench allowing for easy access to all wires, as well as for changing the GPU quickly (see Fig. 1). As GPUs, we employ an NVIDIA *RTX 3090* (*Ampère* microarchitecture) and an AMD *Radeon Pro W6800* (*RDNA 2* microarchitecture).

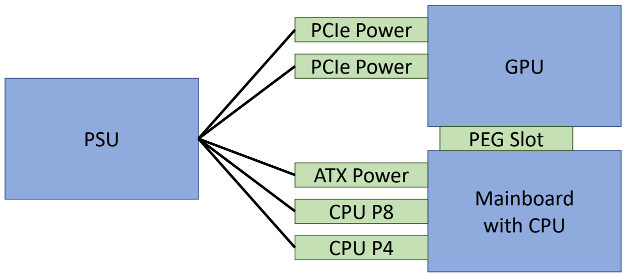


Figure 3: Schematic overview of all relevant power connections (green) in a typical PC system.

External power meter Obtaining the overall power consumption is fairly easy: We put a Rohde & Schwarz *HMC8015* power analyser between the wall socket and the PSU. The advantage of such a professional device over consumer hardware displaying only the instantaneous power draw is that it has logging capabilities to record data over an extended period of time and an API which allows for automating the measurements. Using the API, we synchronise the internal clock of the device to the current time of the PC whenever we start a measurement series to improve the temporal correlation between data from the PC and the external numbers.

Intrusive measurements Using Tinkerforge bricklets, we measure the power usage of most components in the PC independently. To do so, we cut the power connections on the low-voltage side of the PSU (see Fig. 3) in two and place *Voltage/Current 2.0 Bricklets* with Texas Instruments *INA226* power monitors in between. All of these connections comprise multiple cables, either because they provide multiple voltages like the main ATX power connection or because the larger overall wire cross section of a bundle of cables is required for the current on the connection. As the bricklets only have a single input and output connector for wires up to 2.5 mm^2, we need to bundle each connection into a single wire on the "last mile". Fortunately, the last mile is only a few centimetres in length, and we could not measure higher temperatures on this section of the wire than on the original ones shipped with the power supply, indicating that the reduced cross section is not an issue.

As already mentioned, GPUs not only get power via cables, but also from the PEG slot itself. Therefore, in order to know exactly how much power the GPU is using, we need to measure at the slot. This can be achieved by using a riser board with power probing points (see Fig. 2). While typically used for developing PCIe devices, these boards can also be used to solely gain access to the power connection. The Adex Electronics *PEX16IX* isolation extender we are using has pluggable fuses in the power rails, which we replaced by wires redirecting the connection through our power sensors. This use has been envisaged by the manufacturer, and as we are not testing any pre-production components, all hardware involved should operate within their specifications as if directly connected to the mainboard.

All in all, we are using ten sensors for the following measurement points (Fig. 4): ATX 3.3 V, ATX 5 V, ATX 12 V, CPU P4, CPU P8, up to three PCIe power cables and PEG 3.3 V and PEG 12 V. The SSD is the only component we are not measuring at the moment. Each of the power rails is sensed by a separate bricklet. We asynchronously sample these every 5 ms, which seems to be the fastest update rate before running into bandwidth issues in our setup. The *INA226* power monitor on the Tinkerforge bricklets measures current and voltage sequentially one after each other in short time windows, called *conversion time* in electronics, which we set to 588 µs. Having the sensor average current and voltage over four sequential samples of 588 µs each leads to a new measurement on the chip every 4.7 ms, which is the closest possible to the 5 ms readout rate. Averaging and multiplication are done in background and on chip without interrupting the measurement, which further optimises bandwidth utilisation as we do not transfer current and voltage samples we are not interested in for our current experiment.

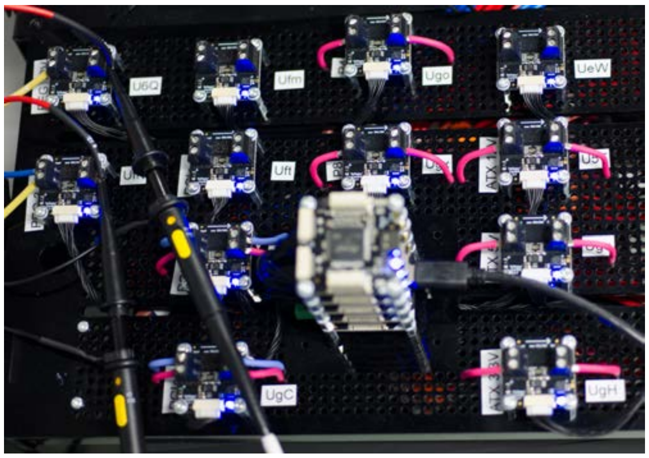


Figure 4: The Tinkerforge bricks (stacked in the middle) and sensor bricklets (arranged around the bricks) we used for measuring individual components in our system. We use the connection to the bricklets also as measurement points for the oscilloscopes.

Oscilloscopes Finally, we have two Rohde & Schwarz *RTB2004* digital oscilloscopes attached to the probing points of the riser and the PCIe power cables of the GPU. As we need a voltage probe and an *RT-ZC03* current clamp to compute the apparent power of each connection, two devices with a total of eight ports are required. As can be seen in Fig. 5 for the PCIe power cables, such a setup allows for inspecting how the power consumption develops within a frame. It should be noted, however, that we used it only to manually check the plausibility while rendering the exact same frame for a prolonged time and such high-resolution measurements are not included in our automated benchmarking results.

4.2 Software

The software infrastructure for obtaining all power measurements is implemented in a single C++ library[1] that is designed to run in-process in the benchmark programme and linked to the two applications we used for our experiments.

Power measurement library The collection of the data in process contributes to the power consumption of the system. We expect this contribution to be negligible, but quantifying it remains for future work. The rationale behind the in-process solution is twofold: first, measurements from the ADL and NVML management libraries can only be obtained on the machine the test is running on. Second, to achieve the best possible correlation between test cases and sensor readings, we can rely solely on consistent timestamps from the benchmarking machine wherever possible. One notable exception is the external power analyser: in principle, it is possible to poll it for momentary readings via USB, but enabling logging mode on the device itself and having it write its readings at its own sampling rate of ten samples per second to a USB thumbdrive produces more consistent results over long periods of time. These results are later correlated with the other sensor readings and benchmark activities via timestamps, for which we programmatically synchronise the clock of the device to the system clock of the benchmarking machine.

[1]`https://github.com/UniStuttgart-VISUS/power-overwhelming`

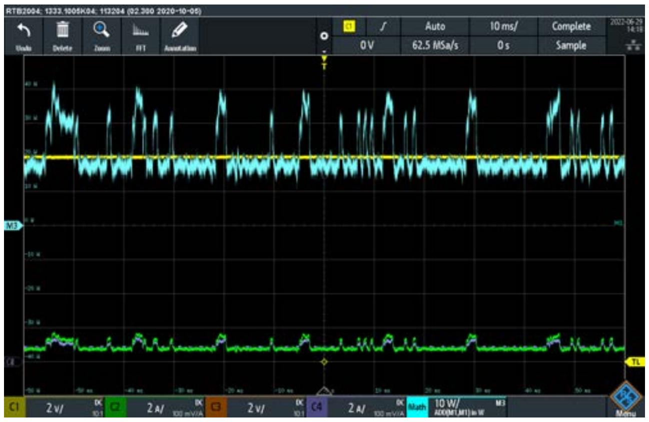


Figure 5: Measurements of the voltage and current probes of an oscilloscope attached to the PCIe cables of an NVIDIA *RTX 3090* while performing a volume rendering task. The odd-numbered probes C1 and C3 are the voltage probes, whereas the even-numbered ones C2 and C4 are the respective current clamps. The cyan-coloured math chart is the combined apparent power of both power lanes.

Following the insights by Arafa et al. [5], the data we collect in process are obtained in threads running in parallel to the benchmark itself. Correlation between the sensor readings and the benchmark activities is achieved by the benchmarking thread reporting its progress to the collector thread, which also writes the data to disk. The way how multi-threading works differs from sensor to sensor: NVIDIA's NVML is fully synchronous, i. e. the API needs to be polled regularly for new readings. In case it is called more frequently than it produces data, the same value is returned repeatedly – in our experience, new values are provided approximately every 100 ms. AMD's ADL implementation is in principle asynchronous in that the *PMLog* functions return a buffer where they regularly write new data once the sensor has been started with a user-defined sampling interval. Unfortunately, the API is not very well documented and there seems to be no synchronisation mechanism whatsoever, which would ensure consistency between the driver writing new data and a client reading it. Although samples are produced asynchronously by ADL, we copy the samples from the aforementioned buffer to our log file in a thread separate from the benchmark. Tinkerforge offers a synchronous and an asynchronous API. As mentined before, we use the latter to avoid polling unchanged values multiple time, thus wasting USB bandwidth. For the same reason, we register only for the power callback, omitting voltage and current values, and have the sensor compute the apparent power from consistent voltage and current samples on chip. The samples we receive from the bricklets are buffered and written to disk by the same thread that samples the ADL or the NVML sensor, respectively.

Benchmarking software The scientific visualisation benchmarks are based on the framework by Bruder et al. [8], which we extended by our aforementioned library to include power measurements in the benchmark code. We largely reused their implementation for raycasting spherical glyphs, but replaced the OpenCL implementation for volume rendering by a new one using Direct3D compute shaders in order to be able to compare pure compute shaders and the combination of compute shaders with the graphics pipeline.

The second set of benchmarks are web-based visualisations. We built a separate application using Microsoft's *WebView2* component, which is basically *Edge* as a widget that we can control programmatically, in order correlate the benchmarks with the measurements from out library. By registering the appropriate event handlers, we obtain notifications about what the browser is currently doing, most notably, when a page is being loaded and when loading completed.

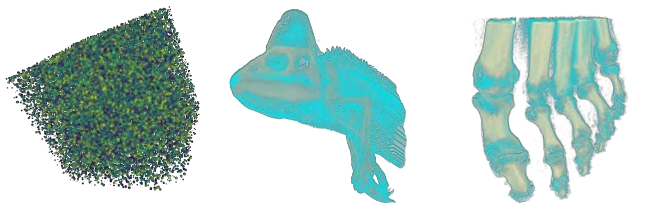


Figure 6: Renderings of the data sets used for the scientific visualisation benchmarks. From left to right: Randomly generated positions of spherical glyphs, the chameleon volume and the foot volume.

5 Experiment

We measure the energy consumption of three visualisations in a highly automated way, aggregating the sensor samples in a post-processing step as necessary.

5.1 Test cases

The two test cases for scientific visualisation, rendering spherical glyphs and volume rendering, are based on Bruder et al.'s [8] framework. This framework processes each test configuration in several phases: the first one is a prewarming phase, ruling out any unexpected initialisation cost in the subsequent measurements and providing an estimate for how long a single frame needs to complete. This estimate is later used to run the benchmark for a user-defined minimum time to measure the frame rate. We exploit this feature in that we request the benchmark to compute the frame rate by counting frames over at least 5 s. During this time window, we sample the sensors every 5 ms, which slightly undersamples the Tinkerforge sensors, but significantly oversamples the software ones. Given that we are averaging over 5 s in the end, we think this kind of sampling issues will not introduce much measurement bias.

Spherical glyphs Sphere rendering uses Direct3D 11 as rendering API and covers the two main techniques available in the benchmarking framework: raycasting on sprites generated in different ways – we restrict ourselves to the generation of ray-aligned and screen-aligned quads in the geometry shader and to instancing – and tessellating the geometry of full spheres and hemispheres [8, 13, 14]. We use two data sets of randomly (with fixed seed value) positioned spheres, one comprising 500,000 elements, the other 5,000,000 (see Fig. 6). The spheres have individual radii and a scalar intensity value that is mapped to a colour via a transfer function. We sample five positions on a path through the data set on the z-axis and an orbit around the y-axis as well as five random positions. All tests are performed on a 1024×1024 and a 2048×2048 viewport.

Volume rendering We have added two Direct3D 11 implementations for volume rendering, the first being a single-pass renderer fully implemented in a compute shader. The second one uses the graphics pipeline to compute all rays at once in an additional pass [25] while the iteration through the scalar field is done in a compute shader. Although not a reasonable approach as of today, we include it to test the influence of using the graphics pipeline and different shader stages. In both cases, we use the 256^3 *foot* data set and the $1024 \times 1024 \times 1080$ *chameleon* (see Fig. 6) and perform a pass without early ray termination and a second one with a threshold of 0.99. Step sizes vary between 0.5, 1 and 2 voxel distances, and the aforementioned camera positions and viewports are used.

Browser-based visualisation Measuring the energy consumption of browser-based visualisations is substantially more challenging, because we have far fewer options to isolate low-level operations. Moreover, Joules per frame as the quantity of interest is somewhat meaningless if the application is not continuously redrawing and it is impossible to determine at which point in time how many frames

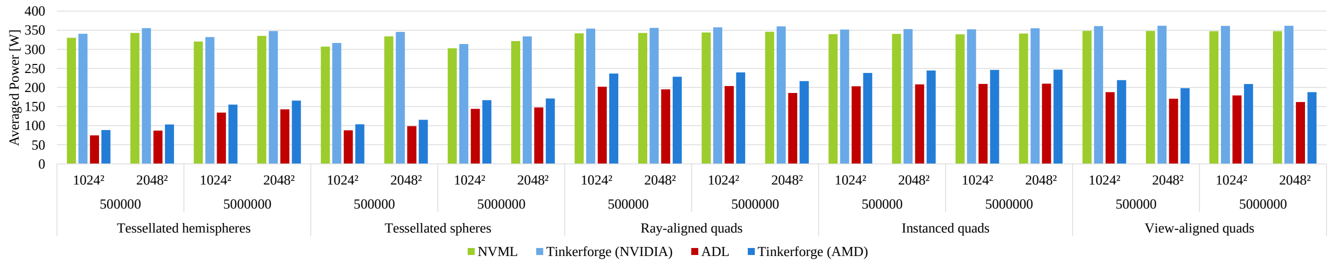


Figure 7: The averaged instantaneous power consumption for different sphere rendering techniques as measured by the NVML and ADL libraries as well as using Tinkerforge bricklets. The horizontal axis displays the controlled parameters, which are from top to bottom: the size of the viewport, the number of spheres and the rendering technique.

have been completed. Our approach is therefore different here: we determine how long the page is loading and sample the power sensors during this phase and continue for 8 s while the loaded page is being displayed. Afterwards, we show `about:blank` for 2 s and repeat the process four times for each page. As test pages, we use D3 demos for basic charts (bar charts, line charts etc.), more complex visualisations like a chord diagrams and parallel coordinates and WebGL graphics, most notably "Dirty Planet", which is a globe with a data overlay. A full list including all URLs can be found in the data set for this publication [31]. As we have noticed *WebView2* seemingly perform some GPU-heavy tasks in the background right after startup, we wait for 5 s before performing the first test in order to exclude this effect, which is obviously unrelated to the benchmark, from our measurements. All browser-based measurements are performed in a 3840×2160 full-screen window. One remaining problem of this approach is that it does not account for the power required to serve the pages nor for the network infrastructure.

5.2 Results

The amount of sensor readings we produced in this experiment are enormous. Therefore, most of the numbers here are aggregated or are chosen to highlight specific details, and we refer the gentle reader to the whole data set [31] for the full set of measurement results.

Spherical glyphs Fig. 7 illustrates the averaged instantaneous power readings for five methods of rendering spherical glyphs. Comparing the values from NVML and the bricklets, we see a deviation between 3 % and 11 % for these tests. For ADL, this gap is between 14 % and 20 %. Looking at the actual values, it becomes obvious why previous work [23] suggested using relative numbers: For the *RTX 3090*, neither the technique nor the data set seem to have a big impact on power consumption: the card is mostly running close to its 350 W limit. In case of the *W6800*, the methods causing more vertex load seem to require less energy than the raycasting-based ones, which get close to the limit of 250 W for this card.

The whole behaviour is due to the fact that the benchmarks have been designed to measure the prevalent metric for evaluating GPU-based scientific visualisation: frames per second. In turn, power efficiency boils down to the question of how many frames the algorithm can produce within a given power limit, where Joules per frame is directly dependent on the number of frames as long as the absolute power draw remains the same. However, from an environmental point of view, absolute numbers do matter, and it is not reasonable to render thousands of frames that are unnecessary, even if each of them is very cheap. If we limit the frame rate to the refresh rate of the monitor (60 Hz), the absolute power draw (Tinkerforge sensors) drops drastically in effectively all cases: for instance, tessellating 500,000 hemispheres requires between 35 W (NVIDIA) and 41 W (AMD), roughly the same as raycasting on instanced quads on a 1024^2 viewport. While increasing the number of pixels to 2048^2 raises the power draw of tessellation to 39 W and 44 W, the power draw for raycasting more than doubles on NVIDIA (76 W) and reaches 55 W on the AMD card. For the larger data set, the effect is even more pronounced (300 W and 146 W for the 2048^2 viewport, as the renderer does not support conservative depth). At the same time, Fig. 8 shows that the per-frame efficiency drops when reducing the frame rate, making the frame rate an unreliable indicator for ecoconscious visualisation. Overall, the effect of a limited frame rate on the per-frame energy varies between a factor of 0.8 and 3.3. While there is a general trend that the efficiency of tessellation is hit harder, which we attribute to the fact that the additional shader stages prevent the GPUs from efficiently using their resources, it is actually the raycasting on the *W6800* for which the energy consumption raises from 0.24 J to 0.78 J per frame, wherefore we consider it reasonable to actually measure until we have developed reliable power models for visualisation algorithms.

Volume rendering Limiting the frame rate consistently decreases per-frame power efficiency in this case, likely due to the fact that compute shaders have a very uniform workload. For a sampling distance of 0.5 voxels and the larger *chameleon* data set, we see very consistent drop in efficiency by a factor of around 1.3 in all cases. Again, these factors grow as the computational complexity decreases: for the small *foot* data set and a sampling distance of 2 voxels, the *RTX 3090* uses 2.5 times the energy per frame for the single-pass renderer and 1.9 times for the two-pass renderer – which is somewhat unexpected as we anticipated the additional shader stages to have a negative impact. Again, a call for empirical measurements.

Browser-based visualisation Fig. 9 illustrates the instantaneous power draw over time by reference to the "Dirty Planet" visualisation rendering a spinning globe using WebGL. Maxing out at approximately 120 W on the *RTX 3090*, the technique is not even close to the power limit of the board, but the GPU is obvisously working. The behaviour of the four test runs is, however, inconsistent, which might be partially caused by varying loading times and illustrates the imponderables of evaluating the energy consumption of this kind of visualisations. Interestingly, the *RDNA 2*-based AMD card has a high and oscillating power draw on the CPU, which can be consistently observed throughout all of the visualisations, be it the ones using WebGL or the basic charts drawing Scalabe Vector Graphics (SVG). We have no explanation for that behaviour at this moment, but as the power draw of the GPU is increasing as well, the 3D work is not solely performed on the CPU. We also ran the same tests on a *Radeon Vega Frontier Edition* based on the previous *GCN 5* architecture, which had a CPU power draw more in line with the NVIDIA card – however with a much larger discrepancy between the ADL readings and our Tinkerforge measurements than in case of the *W6800*. In general, we observe that the energy consumption of SVG charts is mostly dominated by the CPU during page loading, which peaks between 90 W and 100 W (for the dedicated power cables) plus 40 W supplied via the ATX power connector

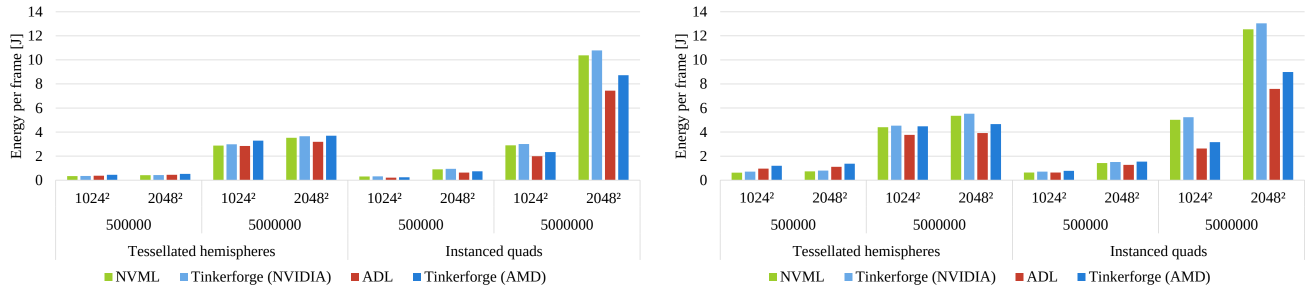


Figure 8: Energy consumption per frame for the geometry-heavy tessellation method and the rasterisation/pixel shader-bound raycasting on instanced quads. The left diagram shows the energy per frame when the GPU can render as many frames as possible, on the right side, the buffer swap is synchronised to the monitor and therefore rate-limited.

minus the power drawn via the PEG slot (a large fraction of that should go to the CPU as well), while the GPU is close to idle power most of the time. The D3 "Chord Dependency Diagram" sample is an interesting exception, built on SVG paths, but having a notable GPU power draw of up to 140 W on the *RTX 3090* and 60 W on the *W6800.* One conclusion we therefore can draw is that for faithfully representing the energy consumption of browser-based visualisations, a more holistic measurement setup than merely relying on on-board sensors on the GPU is necessary – although that still does not consider the server and transport aspects. Not only is the CPU responsible for most of the power draw, but it is also not obvious when the browser will use the GPU and to which extent. Finally, if the power draw is low or unsteady, it becomes increasingly difficult to correctly attribute the measurements to the visualisation as other work performed by the operating system might interfere.

6 Recommendations and Challenges

Based on our experience and the state of the art in visualisation literature and beyond, we see the following aspects that sould be addressed to foster improved use of energy measurements and reporting in visualisation research.

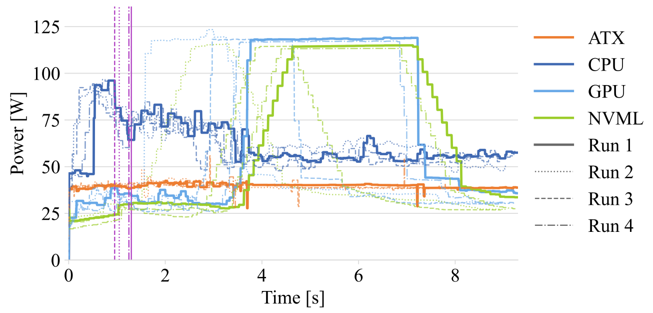
There is a need for **practical, accessible and reliable power measurement setups**, which are most likely not a one-fits-all solution, but dependent on the research question. If the goal is investigating different shaders within a frame, expensive oscilloscopes might be the only solution. If one can afford averaging over time, cheaper sensors with a lower sampling rate might be advisable. In many cases, NVIDIA's software sensors provide remarkably accurate data, effectively for free. Of course, the latter is only practical when using NVIDIA cards and restricts the approach to a very specific subset of applications heavily relying on GPUs – mostly in scientific visualisation. While there are surely information visualisation techniques hammering the GPU [16], gaining a complete picture of the power consumption of the growing number of browser-based applications is not possible this way. However, an interesting development in this context is the latest version of *Firefox* including a power profiler.

Tinkerforge bricklets or alike offer a solution for that by allowing for measuring most power rails in a system individually. Compared to oscilloscopes, they are also a relatively affordable solution, costing approximately one tenth in our case. However, there are disadvantages, e. g. the need to cut dozens of wires and then reassembling them correctly. We are also not fully comfortable with the bricklets having a current limit of 20 A, which is lower than what our PSU can theoretically provide on some of the power rails – but practically does not in the current setup. Furthermore, we add quite some extra cables and clamps to the circuit, changing the resistance of the connections. In combination with off-the-shelf riser cards, the overall solution having a far lower entry barrier than custom printed circuit boards as used by Johnsson et al. [23] is still a big plus.

Large series of benchmarks covering a variety of algorithms, data sets and hardware must be automated. Fortunately, all methods we used provide some kind of API allowing for automation, albeit different ones in most cases (both Rohde & Schwarz instruments have an API based on the Virtual Instrument Software Architecture (VISA), making them at least similar). Having a **unified software ecosystem** for power measurements supporting a variety of instruments is therefore desirable, again to lower the entry barrier and to avoid mistakes. In this work, we hit obscure software quirks on multiple occasions, which should be abstracted from visualisation researchers who only want to evaluate the energy efficiency of their algorithms. Ideally, the software ecosystem supports intrusive measurements that run in the process of the benchmark as well as non-intrusive observations for closed-source processes. We contribute our aforementioned library used for this research to this effort, including a hands-on description of the hardware setup, on GitHub (cf. Sect. 4.2).

Reproducible, replicable and comparable power measurements for visualisation algorithms, which should eventually merge into a consistent understanding of the characteristics of the algorithms, do not only rely on measurement setups and software frameworks, but also on **benchmarks,** most importantly data sets, which have been found to be a major influencing factor on runtime performance [8]. We admit that we chose our benchmarks based on what has been used for evaluating runtime performance of volume and sphere rendering, and for the web-based test cases, we wanted to cover basic charts, node-link diagrams and most importantly WebGL. However, there are arguably even more interesting cases like techniques relying on the graphics pipeline in multiple rendering passes or visual analytics applications, which often combine the difficult information visualisation case with calls to machine learning kernels every so often. Furthermore, in order to form a comprehensive picture of the energy consumption of the *algorithm*, we should identify relevant properties of data sets that affect power draw and eventually resort to generative data models that evenly cover the problem space [38].

We also need to agree on **how to analyse and report** power measurements. Given the high temporal resolution of some sensors, which accommodate the high frequency at which GPUs regulate power, it is possible to collect enormous amounts of data, which need to be massively condensed for publication. Johnsson et al. [23] suggest reporting Joules per pixel as the most general way of expressing the resolution-independent energy efficiency. But this might not necessarily be the best metric for visualisation where, in contrast to most rendering applications, oftentimes not the whole screen is filled or there is heavy overdraw – e. g. in case of volume rendering. Joules per fragment might solve this issue at the cost of making the evaluation more complex as additional statistics queries need to be issued, wherefore Joules per frame in the style of the prevalent frames per second or milliseconds per frame might be the most pragmatic starting point. In cases where the algorithm is not clearly GPU-bound,



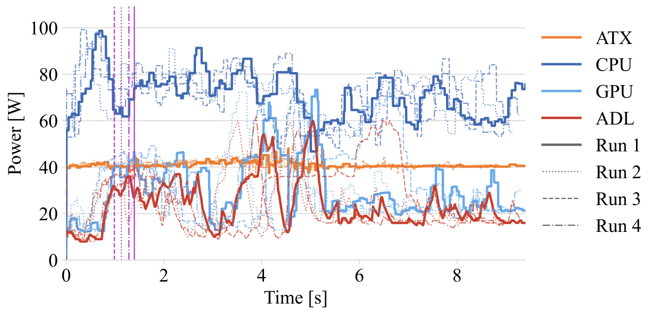


Figure 9: Instantaneous power draw of the "Dirty Planet" visualisation from *Observable HQ*. The "ATX" lines indicate the power measured on all ATX power rails minus the power measured leaving via the PEG slot. The "CPU" lines indicate the power measured on the dedicated EPS (Entry-Level Power Supply Specification) P4 and P8 power cables for the CPU. Therefore, the total CPU power consumption is the sum of "CPU" and a fraction of "ATX", which we unfortunately cannot measure independently. The "GPU" lines indicate the aggregated power measured on the PEG slot and the PCIe power cables. The "NVML" and "ADL" lines indicate the power measured via the respective software sensors by NVIDIA and AMD. The point in time when the browser indicated that the page was loaded is shown by vertical magenta lines. Three additional, lightly coloured measurement series are provided to illustrate that the power draw might differ drastically from run to run. We assume the noticeable steps in the NVML line being the result of internal averaging, making it a less than optimal choice for real-time observations.

reporting GPU power figures might be insufficient and we might need to find ways to present complex, multi-level, multi-faceted data – possibly by means of novel visualisations. For techniques like browser-based visualisations, it might also necessary to include whole time series of sensor data instead of blindly aggregated values.

For many of our measurements at hand, we see a fair amount of **uncertainty**, mostly regarding timestamps and thus correlation between activities and measurements. Only the ADL provides timestamps along with its measurements – a benefit negated by the lack of protection against race conditions, making it impossible be sure that the timestamp and the value stem from the same measurement. For all other types of sensors, we record the timestamp as we receive the sample. Additionally, we do not know the sampling rate of all sensors. For instance, NVML seems to have new values approximately every 100 ms, but timings vary. Similarly, we do not receive samples from the Tinkerforge bricklets at a rate as consistent as we wished for, hinting that we are close to some bandwidth bottleneck, be it USB or the internal bus between bricks and bricklets. Finally, running measurement code in process requires – even if not much – energy, which we currently do not account for. All of these issues should ideally be resolved over time by improving measurement methods – or at least considered when reporting results.

Once having the problems of empirically evaluating the energy consumption of visualisation algorithms sorted out, we should move to building **theoretical models** as the long-term goal. In the GPGPU and HPC context, it has been established that GPU performance counters and assembly instructions are viable options for building predictive models for estimating power consumption instead of measuring it. While the hardware counters representing the utilisation of the GPU are surely a promising option for our field as well, we might also include graphics pipeline statistics in such models and replace CUDA PTX by DXIL and SPIR-V shader assembly languages.

Having theoretical models will also help in finding **guidelines on the right measurement technique for every visualisation technique.** We see evidence in our experiment that high-resolution time series of multiple sensors are required for web-based methods, whereas uniform GPU-heavy workloads can be described by averaging measurements. Sometimes, it might be sufficient to measure the system as a whole and combine these numbers with hardware counters and models to obtain an acceptable prediction of the energy consumption – or even to rely solely on the models.

Raising the bar for what to expect from evaluation of visualisation might not be welcome for everyone. We therefore suggest establishing an **incentive structure** alike to what we do for research data in the form of paper badges. We also hope for the discretion of reviewers pushing the topic if and only if it is appropriate as we clearly see that measuring power might not be appropriate for all research. For instance, given our current approach, we are dubious about using it on highly interactive applications with quickly varying, non-repetitive computational demands. We are convinced that for the sake of a responsible use of resources, we should work towards an understanding of the energy consumption of our work, but we do not believe overly stringent regulations strangling creativity and exploration being advantageous for any field of research.

7 Conclusion and Future Work

We made the case that the visualisation community should look into the energy consumption of their algorithms and showed that nowadays, it is possible for everyone with basic knowledge about electrical systems and computers to build a measurement setup that allows for looking into individual power rails. In some cases, software-based sensors are a built-in option for measuring energy consumption with an extremely low barrier of entry and surprisingly accurate results, but with moderate temporal resolution. In our opinion, with this sensor available, there is no reason not to include such measurements along with the commonly reported frame times if a frame can be repeatedly rendered over some time. We reported results of a small measurement series for GPU-based and web-based visualisations and laid out the challenges related to such an endeavour based on our experiences in the experiment.

Obviously, our exemplary measurement series is not more than a first step on the way to understanding the power consumption of visualisation and we see a variety of future research directions: We should strive to build a reliable, easy-to-use software layer for evaluation, identify the data that are representative for characterising energy consumption of algorithms and simply perform and share a lot of measurements. Based on these, we see the opportunity to build models for power prediction as it has been done in the HPC area before. Furthermore, an increased awareness for power consumption of visualisation algorithms might also lead into developing power-efficient algorithms as a new area of research, again following examples from the HPC and graphics communities.

Acknowledgments

This work was partially funded by Deutsche Forschungsgemeinschaft (DFG) as part of SFB/Transregio 161 (project ID 251654672). The authors want to thank Tobias Rau for his assistance while 3D-printing parts for this work.

References

[1] Y. Abe, H. Sasaki, M. Peres, K. Inoue, K. Murakami, and S. Kato. Power and performance analysis of GPU-accelerated systems. In *Proc. Workshop Power-Aware Comput. Syst.*, 2012.

[2] X. Amberger. Unleashed unicorn on a flight of fancy – shunt-mod for NVIDIA GeForce RTX 3090 FE and the Alphacool Eisblock GPX-N, 2021. Online: `https://www.igorslab.de/en/unleashed-unicorn-spreads-his-wings-from-shunt-modding-nvidia-rtx-3090-founders-edition/`, last accessed 2022/06/30.

[3] AMD. AMD Display Library (ADL) SDK. Online: `https://gpuopen.com/adl/`, last accessed 2022/06/30.

[4] A. S. G. Andrae and T. Edler. On global electricity usage of communication technology: trends to 2030. *Challenges*, 6(1):117–157, 2015. doi: 10.3390/challe6010117

[5] Y. Arafa, A. ElWazir, A. ElKanishy, Y. Aly, A. Elsayed, A.-H. Badawy, G. Chennupati, S. Eidenbenz, and N. Santhi. Verified instruction-level energy consumption measurement for NVIDIA GPUs. In *Proc. Int'l Conf. Comput. Front.*, pp. 60–70, 2020. doi: 10.1145/3387902.3392613

[6] S. Borkar and A. A. Chien. The future of microprocessors. *Commun. ACM*, 54(5):67–77, 2011. doi: 10.1145/1941487.1941507

[7] R. A. Bridges, N. Imam, and T. M. Mintz. Understanding GPU power: a survey of profiling, modeling, and simulation methods. *ACM Comput. Surv.*, 49(3), 2016. doi: 10.1145/2962131

[8] V. Bruder, C. Müller, S. Frey, and T. Ertl. On evaluating runtime performance of interactive visualizations. *IEEE Trans. Vis. Comput. Graph.*, 26(9):2848–2862, 2020. doi: 10.1109/TVCG.2019.2898435

[9] M. Burtscher, I. Zecena, and Z. Zong. Measuring GPU power with the K20 built-in sensor. In *Proc. Workshop Gen. Purp. Proc. GPUs*, pp. 28–36, 2014. doi: 10.1145/2588768.2576783

[10] J. Chuang, D. Weiskopf, and T. Möller. Energy aware color sets. *Comput. Graph. Forum*, 28(2):203–211, 2009. doi: 10.1111/j.1467-8659.2009.01359.x

[11] C. Collange, D. Defour, and A. Tisserand. Power consumption of GPUs from a software perspective. In *Proc. Int'l Conf. Comput. Sci.*, pp. 914–923, 2009.

[12] M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky. A comparative study of methods for measurement of energy of computing. *Energies*, 12(11), 2019. doi: 10.3390/en12112204

[13] S. Grottel, M. Krone, C. Müller, G. Reina, and T. Ertl. Megamol—a prototyping framework for particle-based visualization. *IEEE Trans. Vis. Comput. Graph.*, 21(2):201–214, 2015. doi: 10.1109/TVCG.2014.2350479

[14] S. Gumhold. Splatting illuminated ellipsoids with depth correction. In *Proc. Symp. Vision, Model., Vis.*, pp. 245–252, 2003.

[15] M. Heinemann, V. Bruder, S. Frey, and T. Ertl. Power efficiency of volume raycasting on mobile devices. In *EuroVis 2017 – Posters*, 2017. doi: 10.2312/eurp.20171166

[16] J. Heinrich and D. Weiskopf. Continuous parallel coordinates. *IEEE Trans. Vis. Comput. Graph.*, 15(6):1531–1538, 2009. doi: 10.1109/TVCG.2009.131

[17] T. G. Holmes. Eco-visualization: combining art and technology to reduce energy consumption. In *Proc. Conf. Creat. & Cogn.*, pp. 153–162, 2007. doi: 10.1145/1254960.1254982

[18] S. Hong and H. Kim. An integrated GPU power and performance model. In *Proc. Int'l Symp. Comput. Arch.*, pp. 280–289, 2010. doi: 10.1145/1815961.1815998

[19] Q. Hu, P. Sun, S. Yan, Y. Wen, and T. Zhang. Characterization and prediction of deep learning workloads in large-scale GPU datacenters. In *Proc. Int'l Conf. High Perf. Comput., Netw., Stor., Anal.*, 2021. doi: 10.1145/3458817.3476223

[20] IEA. Data centres and data transmission networks, 2021. `https://www.iea.org/reports/data-centres-and-data-transmission-networks`.

[21] A. Jahanshahi, H. Z. Sabzi, C. Lau, and D. Wong. GPU-NEST: characterizing energy efficiency of multi-GPU inference servers. *IEEE Comput. Archit. L.*, 19(2):139–142, 2020. doi: 10.1109/LCA.2020.3023723

[22] B. Johnsson and T. Akenine-Möller. Measuring per-frame energy consumption of real-time graphics applications. *J. Comput. Graph. Tech.*, 3(1), 2014.

[23] B. Johnsson, P. Ganestam, M. Doggett, and T. Akenine-Möller. Power Efficiency for Software Algorithms Running on Graphics Processors. In *Proc. Symp. High Perf. Graph.*, 2012. doi: 10.2312/EGGH/HPG12/067-075

[24] N. Jones. The information factories. *Nature*, (561):163–166, 2018. doi: 10.1038/d41586-018-06610-y

[25] J. Krüger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *Proc. Vis.*, pp. 287–292, 2003. doi: 10.1109/VISUAL.2003.1250384

[26] J. Lim, N. B. Lakshminarayana, H. Kim, W. Song, S. Yalamanchili, and W. Sung. Power modeling for GPU architectures using McPAT. *ACM Trans. Des. Autom. Electron. Syst.*, 19(3), 2014. doi: 10.1145/2611758

[27] C. Luo and R. Suda. A performance and energy consumption analytical model for GPU. In *Proc. Conf. Depend., Auton., Secure Comput.*, pp. 658–665, 2011. doi: 10.1109/DASC.2011.117

[28] X. Ma, M. Dong, L. Zhong, and Z. Deng. Statistical power consumption analysis and modeling for GPU-based computing. In *Proc. Workshop Power-Aware Comput. Syst.*, 2009.

[29] X. Mei, L. S. Yung, K. Zhao, and X. Chu. A measurement study of GPU DVFS on energy conservation. In *Proc. Workshop Power-Aware Comput. Syst.*, 2013. doi: 10.1145/2525526.2525852

[30] S. Mittal and J. S. Vetter. A survey of methods for analyzing and improving GPU energy efficiency. *ACM Comput. Surv.*, 47(2), 2014. doi: 10.1145/2636342

[31] C. Müller, M. Heinemann, D. Weiskopf, and T. Ertl. Energy consumption of scientific visualisation and data visualisation algorithms, 2022. Data set. doi: 10.18419/darus-3044

[32] H. Nagasaka, N. Maruyama, A. Nukada, T. Endo, and S. Matsuoka. Statistical power modeling of GPU kernels using performance counters. In *Proc. Int'l Conf. Green Comput.*, pp. 115–122, 2010. doi: 10.1109/GREENCOMP.2010.5598315

[33] NVIDIA. NVIDIA Management Library (NVML). Online: `https://developer.nvidia.com/nvidia-management-library-nvml`, last accessed 2022/06/30.

[34] M. Qasaimeh, K. Denolf, J. Lo, K. Vissers, J. Zambreno, and P. H. Jones. Comparing energy efficiency of CPU, GPU and FPGA implementations for vision kernels. In *Proc. Int'l Conf. Embed. Softw. Syst.*, pp. 1–8, 2019. doi: 10.1109/ICESS.2019.8782524

[35] K. Ramani, A. Ibrahim, and D. Shimizu. Powerred: a flexible modeling framework for power efficiency exploration in GPUs. In *Proc. Workshop Gen. Purp. Proc. GPUs*, vol. 7, 2007.

[36] E. Ramsden. *Hall-Effect Sensors*. Newnes, Oxford, 2006.

[37] S. Schneider. NVIDIA reviewer toolkit for graphics performance, 2020. Online: `https://www.nvidia.com/en-us/geforce/news/nvidia-reviewer-toolkit/`, last accessed: 2022/06/30.

[38] C. Schulz, A. Nocaj, M. El-Assady, S. Frey, M. Hlawatsch, M. Hund, G. Karch, R. Netzel, C. Schätzle, M. Butt, D. A. Keim, T. Ertl, U. Brandes, and D. Weiskopf. Generative data models for validation and evaluation of visualization techniques. In *Proc. Workshop Beyond Time and Errors on Novel Eval. Meth. for Vis.*, pp. 112–124, 2016. doi: 10.1145/2993901.2993907

[39] J. W. Sheaffer, D. Luebke, and K. Skadron. A flexible simulation framework for graphics architectures. In *Proc. Conf. Graph. Hardw.*, pp. 85–94, 2004. doi: 10.1145/1058129.1058142

[40] I. Wallossek. Read out graphics cards power consumption via software instead of costly measurement? easy with a NVIDIA GeForce and almost impossible with an AMD Radeon, 2022. Online: `https://www.igorslab.de/en/graphics-cards-and-their-consumption-read-out-rather-than-measured-why-this-is-easy-with-nvidia-and-nearly-impossible-with-amd/`, last accessed 2022/06/30.

[41] R. Wang, B. Yu, J. Marco, T. Hu, D. Gutierrez, and H. Bao. Real-time rendering on a power budget. *ACM Trans. Graph.*, 35(4):111:1–111:11, 2016. doi: 10.1145/2897824.2925889

[42] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore. Measuring energy and power with PAPI. In *Proc. Int'l Conf. Par. Proc. Workshops*, pp. 262–268, 2012. doi: 10.1109/ICPPW.2012.39