Calibratio: A small, low-cost, fully automated Motion-to-Photon Measurement Device

Sebastian Pape* Marcel Krüger* Jan Müller* Torsten W. Kuhlen*

Figure 1: Left: Our implementation of the device concept. Visible are the tracking body on top, the metal contacts attached to it, the servo motor it is mounted to, the green led and the microcontroller in its bottom. Right: The device while measuring in our CAVE.

Abstract

Since the beginning of the design and implementation of virtual environments, these systems have been built to give the users the best possible experience. One detrimental factor for the user experience was shown to be a high end-to-end latency, here measured as motion-to-photon latency, of the system. Thus, a lot of research in the past was focused on the measurement and minimization of this latency in virtual environments.

Most existing measurement-techniques require either expensive measurement hardware like an oscilloscope, mechanical components like a pendulum or depend on manual evaluation of samples. This paper proposes a concept of an easy to build, low-cost device consisting of a microcontroller, servo motor and a photo diode to measure the motion-to-photon latency in virtual reality environments fully automatically. It is placed or attached to the system, calibrates itself and is controlled/monitored via a web interface. While the general concept is applicable to a variety of VR technologies, this paper focuses on the context of CAVE-like systems.

Index Terms: Computer systems organization—Architectures—Other architectures—Special purpose systems Computing methodologies—Computer graphics—Graphics systems and interfaces—Virtual reality Hardware—Hardware validation—Post-manufacture validation and debug—Design for debug General and reference—Cross-computing tools and techniques—Performance—

1 Introduction

While developing software for virtual environments, the user experience is one of the central points to keep in mind. In graphically intensive applications, the performance of the application and the resulting latency always needs to be considered. In our research group the software stack for the development of all virtual reality software was the ViSTA Virtual Reality Toolkit [1], which supported CAVEs, HMDs and other systems. Since the Unreal Engine Version 4.20 received a plugin for multi display rendering, the idea of switching to a game engine for faster development cycles came up. The *nDisplay* plugin initially did not support the rendering in a Linux cluster which was added by one member of our group for test purposes. Shortly after the engine was integrated, some Unreal demos were programmed to run on the aixCAVE, but were reported to have a higher response time than the previous ViSTA demos by some team members. To test this hypothesis we decided to get a rough estimate of the response time by taking high frame rate videos with a smartphone. This underlined the initial perceived higher response times, so the Unreal Engine was slower than the previous software stack. Since the initial smartphone measurements needed multiple people and took much time to evaluate, we needed a fast and easy process to measure the motion-to-photon latency. Despite many known techniques from literature (c.f. Section 2), we decided to build our own device for measuring combating some weaknesses in the design of the other techniques. This device should be fast and straightforward to use, fully automated, controllable from a distance, easy to build and cheap. Besides the obvious points, the operability from a distance allows the user to work on other parts of the system without physically controlling the device. This especially allows the closing of the aixCAVE door while measuring.

2 Related Work

The idea proposed in this paper is not entirely new, but iterates on previously published designs. A lot of measurement devices were proposed and built by many teams around the world. DiLuca summarized some of these methods in [3]. The methods described in the literature can be assigned to three categories.

– The easiest category of measurement concepts employs a camera to capture the movement and response. From the recorded footage, the motion-to-photon latency can be calculated by counting the frames

*e-mail: {pape|krueger|j.mueller|kuhlen}@vr.rwth-aachen.de

between the motion and visual change [4].
This always limits the temporal resolution to the frame rate of the camera. Since high frame rate cameras are often too expensive for a normal laboratory setup, multiple teams replaced the camera assembly by simpler sensors, like single- or multi-point light sensors.
Miller et al. employed two light sensor arrays and calculated a centroid of brightness [5]. To infer movement from the user and the output screen at the same time, one sensor was pointed towards the user and the other one was pointed at the screen. Afterwards the speed of the user and the virtual counterpart were calculated and plotted. From these curves the phase offset, which corresponds to the motion-to-photon latency, was calculated.
In the approach of DiLuca only a single light sensor was used [3]. In his setup the sensor is moved simultaneously over a virtual and a real black to white gradient. From the recorded brightness data, the phase offset and thus the motion-to-photon latency can be calculated as in [5].
Notably, these methods allow the measurement of real user movement instead to synthetic movement done by actuators.

– The second category contains approaches based on a pendulum to which a tracker is attached. As the pendulum is swung, it performs a sine movement and a virtual counterpart follows the physical one. By measuring the offset between the real and the virutal pendulum the motion-to-photon latency can be inferred. An implementation of this method introduced by Mine et al. relied on a setup built from an oscilloscope, a pendulum and a light sensor [6]. The system measures the delay from the tracker being at the lowest point on the pendulum swing to a change of brightness on a monitor.
Zhao et al. presented a similar method, but measured not only at the lowest point, but the complete swings of the pendulum with a rotary encoder [10].
Steed extended this idea by using a wireless tracker, a pendulum and a standard video camera [9]. The camera is set up to film the pendulum and a virtual counterpart on the screen. Afterwards two sine curves are fitted into the recorded video. The phase offset between the fitted sine curves gives the motion-to-photon delay.
Unfortunately these methods could produce wrong results, if the rendering software extrapolates user movement to reduce delays. This would result in lower delays in predictable movement situations, but higher delays in abrupt, non-predictable motion scenarios. Some of these methods also require a complicated calibration setup with multiple devices to set up.

– In the third category, the tracker is moved and the time between the movement and the response on the screen is measured. As a trigger for the response, a light sensing device is used.
Seo et al. employed a rotary base to which a disassembled HMD is mounted [8]. The base can turn and tilt to simulate all possible rotations. The team attached multiple light sensors to the display of the HMD. Using multiple sensors, motion patterns on the display are measured and a timer is stopped if movement is detected, which was started on the movement of the rotary platform. The measured time is the motion-to-photon latency. Papadakis et al. built a system, which moves a tracker with a motion-platform in a continuous motion [7]. To measure the response, a light sensor is attached to the output screen of the virtual environment, which changes the color based on a threshold angle of the tracker. The time from passing this angle on the motion platform to the light sensor responding reflects the motion-to-photon latency.
This method is very similar to the method presented in this paper. However, the method presented here differs in two ways. On the one hand, our motion platform executes abrupt movements, which cannot be extrapolated from previous data by the application. This also changes the trigger mechanism of the color change. On the other hand the device and hardware architecture in this paper propose a device that does not need an external Oscilloscope to measure the delay, which reduces the cost of the whole setup immensely.

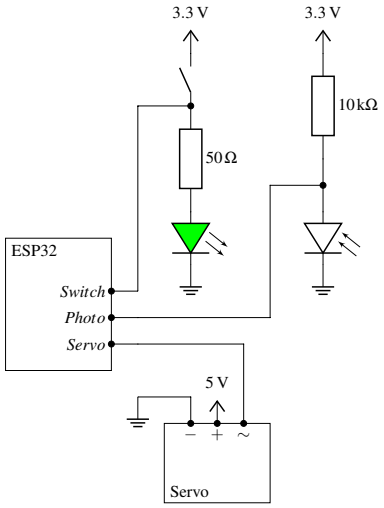

Figure 2: The circuit design of Calibratio. The 3.3V and 5V can be taken from the used development board directly.

Some methods (e.g. the method proposed by Choi et al. [2]) measure a modified version of the real hardware, by exchanging the original output screen for a normal computer monitor. This could result in a different motion-to-photon latency as the latency of the two screens could differ. Other devices like a device used by Oculus also use a light sensor based latency measurement [8]. This measurement was meant to work with the Oculus Development Kit 1, which had no translational tracking. Thus, the device connects to the computer via a USB connection to send a signal to change the screen's brightness. This results in a signal-to-photon latency rather than a motion-to-photon latency, which does not reflect the performance of the complete system.

3 Hardware

The concept of Calibratio, the system proposed in this work, is a portable, tether-free, fully automated and cheap device.
To achieve this, a microcontroller development board based on the ESP32 chipset was used. These boards are available for ≤ 5 USD. While the chipset is available without a development board, it provides the voltage levels, which are needed for the peripheral electronics, and adds a USB port for programming and powering the chipset.
The ESP32 was chosen, since it features Wi-Fi capabilities, allowing for a tether-free control mechanism. Additionally, it has a dual core CPU, which can be used for simultaneous measurements and control. For measuring the fast analog to digital converter (ADC) of the chipset is used, which can sample up to 200k samples per second.
As light sensor, a ≤ 0.50 USD photoresistor similar to the LDR07 was used. To enable reading the photodiode, a simple voltage-divider was built and attached to an ADC pin of the micro controller (c.f. Figure 2).
For mechanical movement of the tracker, a regular RC servo motor (≤ 3 USD) is used. The tracked device is attached to the servos head. This could be a complete HMD, a standalone tracker for HMDs or, like in our case, an opto-electrical tracking body. Unfortunately, the mechanical actuation introduces a timing offset due to inertia of the hardware and the signal processing of the servo. Regular RC servos communicate via a single data line and receive the input angle via a Pulse Width Modulation (PWM) signal. The data in this signal is transmitted at about 50Hz, which in the worst case, introduces a 20ms delay between the input and the start of physical actuation of the tracking device. Since the latter delay can't be predicted easily, an external measurement of the start of the physical actuation is

needed. This is done by introducing an electrical switch, built from two touching contacts, which is opened when the tracking device begins to move. The electrical contact allows the precise start of the timer, while the photoresistor is polled to stop the timer.
As a last part, an LED is connected in series with the electrical switch, such that it lights up, if the switch is closed. Thus, it functions as a pull down resistor and indicates that the switch is closed, which is used in Section 5.
The whole circuit diagram for this is shown in Figure 2. Here the $3.3\,V$ and $5\,V$ are used from the microcontroller development board. As a power supply for the whole system a USB battery pack is used and connected to the development board via a micro USB cable.

3.1 Specific Hardware Setups

The concept described in Section 3 can be used to measure the delay in all sorts of systems, that use a screen and some kind of tracking device, which makes it to a certain degree future-proof for upcoming generations of VR environments. For different virtual environments the attachment method of the tracking device to the servo might be more complex. In our case, the passive tracking body was directly screwed to the servo. If a heavier device, like an HMD is used, a stronger servo and a housing with some kind of rotary table, similar to [8], might be used.
To house all components, a small transparent lunchbox was used, since it does not interfere with the photoresistor reading, if the resistor is mounted in its bottom. In some systems, the photoresistor should be connected to a cable outside the housing to allow the mounting on other screen types. In our setup the whole box is simply placed on the CAVE floor and measures the area directly underneath.
The implementation used, discussed in Section 4, effectively allows reading of the photoresistor with $\sim 88\,\text{kHz}$ (every $11.36\,\mu\text{s}$). If this sample rate should not be accurate enough, the reading via ADC could be avoided by replacing the $10\,\text{k}\Omega$ resistor with a potentiometer and using simple digital sampling. The potentiometer can be used to set a voltage threshold, between the readings of the white image and the black image.
If the sampling rate used should still be too low, a Sound Card [3] or a logic analyzer in combination with a faster processing device like the *Raspberry Pi Zero W* could be utilized.

4 Software

The software on the measurement device fulfills the purpose of performing the measurements, providing a user interface to control the measurements and storing the results. For the response of the application a simple measurement protocol is implemented. Every application side programming is specifically kept easy, to allow the integration in every software environment. To ease software development on the microcontroller side, the Arduino framework[1] is used. The device code is available on Git [2].

4.1 Measurement Protocol

Since the microcontroller is equipped with a Wi-Fi module, it could connect itself to the VR application to perform calibration. We specifically chose not to connect the two systems to avoid dependencies that might alter the measurement result. Thus, the devices follow the unidirectional measurement protocol shown in Figure 3. The application displays a dynamically colored splat, centered below the tracker, which signals a response to the movement in the form of black and white color. The protocol uses a third color (red), which is only used for visualization of the internal calibration and is not used in the measurement process. In the first step, the application tries to

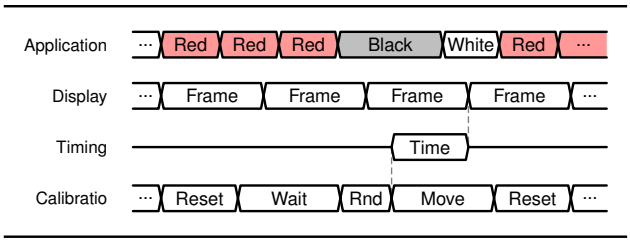

Figure 3: The unidirectional measurement protocol that is followed by the application and the measurement device.

calibrate the rotation of the tracking body. In this phase the splat is colored red. This is done by taking the rotation of the tracker over a few frames and computing the minimum and maximum values for the measured axis. This dead zone is extended by adding a constant angle to both extremes, which corresponds to the trigger-threshold for the application. If the tracker was moving while this calibration occurred, noticeable by a large dead zone, the calibration is restarted. Once the calibration completed successfully, the splat is colored black and the application waits for a motion input of the tracker. Once this is received, the splat is colored white and the cycle is repeated.
On the side of the measurement device, the first step resets the tracking device into a known state. For this the servo is moved until the switch is closed. Then a fixed time period is waited and an additional random time period (Rnd) is added.
The random time period, which is between 1-10ms, prevents a systematical measurement error that could occur due to an unfavorable alignment of the whole measurement cycle to a multiple of the frame timing. This would add a constant offset to the average response time.
After waiting, a rotation is issued to the servo motor and the timer is started as soon, as the switch is opened. Here a debouncing of the electrical contact is strictly necessary, as physical switches tend to issue multiple signals while opening or closing. To avoid this, on each (re-) closing of the switch the timer is restarted. The device now measures the time until the photoresistor passes a predefined threshold. For this, the application needs to color the splat white and show it on screen.

4.2 User Interface

To allow the user to easily perform measurements, the device's Wi-Fi capabilities are used to open an ad hoc network. The user can connect to this network with every Wi-Fi device and open the user interface under the IP address of the device via a web browser. This displays a web page, which allows the user to configure parameters and perform measurements or download the results.
The first parameter offered is the threshold that is considered white for the photoresistor. To determine it, a measurement of the current value can be made to get the value of black and white to choose the threshold accordingly. As the second parameter, the amount of servo deflection in the "move" period can be altered.
To avoid altering the measurements by user actions, the multi-core functionality of the microcontroller is used to handle user input on another core than the measuring process.

5 Test Measurements

To evaluate the device, it was placed in our CAVE system and a demo application was started. An Unreal application was chosen, which consisted of an empty room with four dancing virtual humans around the center. These moving virtual humans with high polygon count should represent an average load, which could occur from a more complex scene.

To verify the precision of the device, it was tested against a 240fps (≈ 4.16ms temporal resolution) smartphone camera in 5

[1] https://www.arduino.cc/

[2] http://devhub.vr.rwth-aachen.de/VR-Group/unreal-development/calibratio

runs. For this test, the smartphone recorded the device and the screen at the same time. The frames from start of the movement, which shuts off the led, and the floor changing its color are counted. All measurements resulted in a difference smaller than the timing resolution of the camera.

To measure the latency of the application, it was run for 10min prior to the measurements to avoid any loading phases. Then the measurement was started via the web interface. A total of 3000 measurements was performed over the course of 2h and the results are shown in form of a histogram in Figure 4.
In this Histogram measurements < 50ms and > 400ms were excluded as measurement errors, which discarded 25 data-points (0.83%). The y-axis states the probability of each measurement to result in the corresponding 1ms wide bin on the x-axis. The average motion-to-photon latency is ≈ 124.62ms. To describe the data, three curves were fit, which correspond to a Gaussian Mixture Model (purple) with two components (red and green). The shown components are parameterized with $\mu_1 \approx 119$ms, $\sigma_1 \approx 6.58$ms, $\mu_2 \approx 131.9$ms, $\sigma_2 \approx 6.47$ms. This results in a difference of ≈ 12.9ms between the centers of the two components.

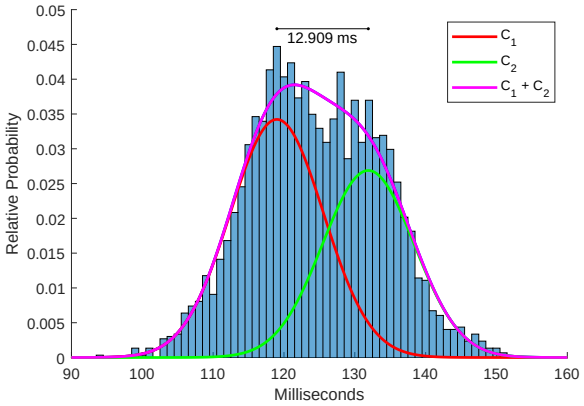

Figure 4: Histogram of the recorded data with an average of $\approx$ 124.62ms and fitted with a Gaussian Mixture Model with $k = 2$, $\mu_1 \approx$ 119ms, $\sigma_1 \approx 6.58$ms, $\mu_2 \approx 131.9$ms, $\sigma_2 \approx 6.47$ms to approximate the data.

6 Discussion

The comparison of the measurements to a relatively high-speed video was done to ensure the results to be reasonable and to exclude deviations that are due to implementation errors of the device. From the timings extracted from the hand counted measurements the device seems to measure the same as one would manually do, but needs to be verified further.

The measured average motion-to-photon latency was higher than expected. As the many components like Tracking, Tracking Server, Network, Application and Display-Output are present in our specific setup, this repeatable measurement allows us to iterate on the performance of each of the components.

A Gaussian Mixture Model was fit to the results in Figure 4. For this fit, two components were chosen due to the appearance of the histogram. We hypothesize that the two peaked appearance reflects the timing of two frames of our projection system. This hypothesis is underlined by the time between the two peaks that corresponds roughly to the period of the frame rate with $90\text{Hz} = 11.11\text{ms} \approx 12.9\text{ms}$. This could be explained by the start of the measurement, which is unsynchronized to the frame rate of the application and ends in a discretized time step due to the screens refresh rate.
The fitted Gaussian Mixture Model as well as the timing difference between the peaks will be subject to further research and so far show just one possible interpretation of the results.

7 Conclusion

The device concept presented in this paper shows a low cost device that is easy to build and use. With the measurement process in mind, the device was built wireless to make the setup and usage as easy as possible. This would allow for fast measurements on a regular basis to quantify changes in hardware and software setups. The concept can easily be adapted to work with other virtual reality systems than the tested CAVE setup, making it universal and to a certain degree future proof. The resulting measurements gave us some inside into our system and the opportunity to start improving the software stack to minimize the latency.

7.1 Future Work

In the future, more tests will be conducted to test our hypothetical interpretation of the Gaussian Mixture Model from Section 6. This will be done by introducing known delays into the application, which should allow us to visualize more than 2 peaks, by stretching the response times predictably.
To verify our measurements externally, we would like to employ a high-speed camera with a much higher temporal resolution for a few measurements. Additionally, the accuracy of the device will be analyzed by another device with a deterministic delay.
While the hardware so far proved to be working, a few modifications could be done by introducing more costly alternatives to the actuating servo or the light sensor. We dont expect this to introduce any changes to the measurement results. To make the metallic contact point more resilient to corrosion or dirt, the screws could be exchanged with gold-plated contacts or an optical sensor.

References

[1] I. Assenmacher and T. W. Kuhlen. The ViSTA Virtual Reality Toolkit. *Proceedings of IEEE VR workshop SEARIS. Reno, NV*, pp. 1–4, 2008.

[2] S.-W. Choi, M.-W. Seo, S.-L. Lee, J.-H. Park, E.-Y. Oh, J.-S. Baek, and S.-J. Kang. Head Position Model-based Latency Measurement System for Virtual Reality Head Mounted Display. *SID Symposium Digest of Technical Papers*, 47(1):1381–1384, may 2016. doi: 10.1002/sdtp.10930

[3] M. Di Luca. New Method to Measure End-to-End Delay of Virtual Reality. *Presence: Teleoperators and Virtual Environments*, 19(6):569–584, dec 2010. doi: 10.1162/pres_a_00023

[4] D. He, L. Fuhu, D. Pape, G. Dawe, and D. Sandin. Video-Based Measurement of System Latency. *International Immersive Projection Technology Workshop*, 2000.

[5] D. Miller and G. Bishop. Latency Meter: A device to measure end-to-end latency of VE systems. *Stereoscopic Displays and Virtual Reality Systems IX*, 4660(919):458–464, 2002. doi: 10.1117/12.468062

[6] M. Mine. Characterization of end-to-end delays in head-mounted display systems. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, United States, 1993. doi: 10.5555/897856

[7] G. Papadakis, K. Mania, and E. Koutroulis. A System to measure, control and minimize end-to-end head tracking latency in immersive simulations. In *Proceedings of VRCAI 2011: ACM SIGGRAPH Conference on Virtual-Reality Continuum and its Applications to Industry*, pp. 581–584, 2011. doi: 10.1145/2087756.2087869

[8] M.-W. Seo, S.-W. Choi, S.-L. Lee, E.-Y. Oh, J.-S. Baek, and S.-J. Kang. Photosensor-Based Latency Measurement System for Head-Mounted Displays. *Sensors MDPI*, 17(5):1112, may 2017. doi: 10.3390/s17051112

[9] A. Steed. A simple method for estimating the latency of interactive, real-time graphics simulations. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST*, pp. 123–129, 2008. doi: 10.1145/1450579.1450606

[10] J. Zhao, R. S. Allison, M. Vinnikov, and S. Jennings. Estimating the motion-to-photon latency in head mounted displays. In *2017 IEEE Virtual Reality (VR)*, pp. 313–314, 2017. doi: 10.1109/VR.2017.7892302