International Conference on Artificial Reality and Telexistence
Eurographics Symposium on Virtual Environments (2018)
G. Bruder, S. Cobb, and S. Yoshimoto (Editors)

HTC Vive Pro time performance benchmark for scientific research

M. Le Chénéchal[1] and J. Chatel-Goldman[1]

[1]Open Mind Innovation

Abstract

Widespread availability of consumer-level virtual reality (VR) devices creates a venue for their massive use in psychology and neuroscience research. The application of VR to scientific research however poses significant constraints on system performance and stability. In particular, studies with multimodal measurement of human behavior and physiology require precise hardware-software synchronization with precise event labeling (within 10 milliseconds). Previous works investigating suitability of VR systems for research have mainly focused on benchmarking performance in spatial tracking. Therefore, it remains unclear if timing parameters such as latency or jitter in VR motion capture and VR audiovisual stimulation allow for carrying out science under strong time constraints. Here we present the first quantitative test of time performance in VR input and VR feedback of the current state-of-the-art HTC Vive Pro system. Using both low-level Python-based API and a high-level game engine (Unity), our multilevel testing procedure allows us to isolate software influence on observed results. We report that, in both test conditions, latencies are non-negligible considering fine synchronization with multimodal measurements; however, jitters are stable and low, which allows to counter-balance the effect of latency by using constant offsets to re-synchronize multimodal data. Finally, we plan to share our testing hardware setup as an open-source and low-cost benchmark toolkit, allowing objective testing to be easily reproduced by the community in an open collaborative framework.

CCS Concepts

• ***Computing methodologies*** → ***Virtual reality;*** • ***Hardware*** → ***Board- and system-level test;*** *Sensors and actuators;*

1 Introduction

The recent introduction of consumer-level, relatively low-cost virtual reality (VR) hardware systems fosters their widespread use for psychology and neuroscience research [WMK*18]. This holds the promise of pushing further what can be explored in terms of user action (VR input) and user perception (VR feedback) in various experimental settings, from lab-based studies to crowd-sourced experiments [BAB11, MCP*18]. However, beyond entertainment, the use of VR as a research tool poses significant constraints on system performance and stability. In the spatial domain, good accuracy and precision of position and orientation tracking is necessary for research demanding precise body motion capture from the head mounted display (HMD) and controller sensors, such as in sports science. Perhaps more crucially in the time domain, a sufficient sampling rate as well as low measurement latency and jitter [TPSM09] are of prime importance for experiments involving synchronized multimodal measurement of human behavior and physiology. In brain-computer interface (BCI) research, for instance, one must synchronize electrophysiological measurements with audiovisual stimulations at the <10ms level in order to adequately analyze the event-related potentials [ICM*14]. In experiments where biosignals must be used in real-time, such as in BCI or during biofeedback protocols, post-hoc synchronization of various data streams is inadequate, as overall system timing performance must be characterized and corrected beforehand when necessary. While previous studies have extensively assessed HTC Vive system tracking accuracy in the spatial domain (e.g., [NLL17]), a thorough testing of latency parameters is still required to evaluate to what extent this popular system is suitable for research implying strong time constraints. In addition, most VR content today is produced using high-level game engines, whose specific influence on time performance must be assessed separately. Finally, as benchmarking is a time consuming process and hardware evolves quickly, it is of general interest to make these tests easily reproducible by the community. An open collaborative framework is therefore desirable. Our contribution in this paper is threefold: 1) we present the first quantitative test of time performance in VR input and VR feedback for the current state-of-the-art HTC Vive Pro system; 2) in order to isolate the software bias we have run a multi-level time benchmark on both low-level python-based API and one high-level game engine (Unity); and 3) we have conceived our testing hardware setup as a low-cost benchmark toolkit and we plan to share its design and code as open-source.

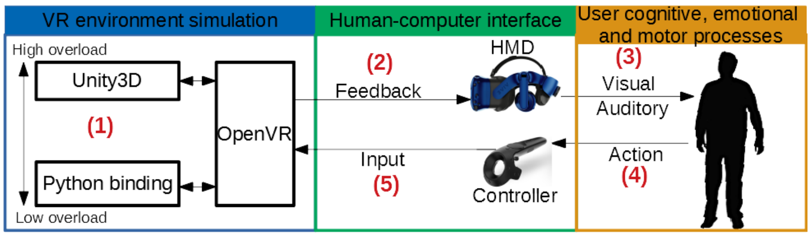
2 Methods overview: interactive system loop and latencies

Classically, an interactive system can be defined as a continuous discrete 3-step loop embracing perception, decision, and action [Dix96]. Delays can add up at each stage with more or less severe impact on the user experience, depending on their duration and their jitter. As an example, let us consider the measure of a user action in response to a visual stimulation and let us categorize various additive system latencies as it follows (see illustration in Figure 1a):

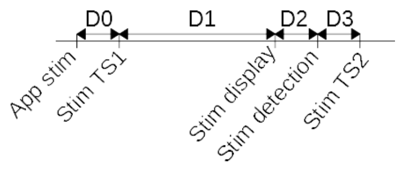
1. VR environment simulation (e.g., visual): software latency
2. Stimulation-to-display (VR feedback): hardware latency
3. Display-to-user perception: physical propagation
4. Decision-to-action: timing of cognitive processes
5. Action-to-simulation (VR input): hardware latency

Latencies introduced at each of these steps do not have equivalent relevance and order of magnitude. In particular, display-to-user perception latency (3) can be neglected as light and sound waves prop-

DOI: 10.2312/egve.20181318



(a)



(b)

Figure 1: *(a) Illustration of the interactive system loop showing the different steps involved in latency introduction. (b) Timeline describing different latencies involved in our measurement setup.*

agate almost instantaneously from HMD screen and headphone to user eyes and ears. Timing of cognitive processes in the decision-to-action step (4), considered to be generally less than 100ms [Car81], is a subject of study by itself in neuroscience experiments. The VR environment simulation step (1) is typically bounded to run at 90 frames per second (FPS), which adds a maximum of 11ms latency if computations are performed within one frame. **The benchmark toolkit we propose aims at evaluating hardware latencies introduced during VR feedback (stimulation-to-display, step 2) and during VR input (action-to-simulation, step 5).**

We aim at measuring intrinsic latencies of the HTC Vive Pro system. Our testing app uses a low-level interface for a direct communication with the HMD through its native OpenVR API wrapped into a Python library. This approach allows us to evaluate latency parameters with a minimal overload generated by the testing app itself. However, in more realistic cases, researchers and developers use high-level game engine to ease and accelerate development of their apps. Therefore we also ran the same tests using a Unity3D-based app, which is one of the most popular game engine. This lets us evaluate practical latencies of the whole system when used in realistic scenarios and determine non-negligible overloads introduced by the game engine.

For this evaluation we have substituted the human user with a microcontroller assembled with simple sensors and actuators to allow precise timing of visual/auditory feedback and controller input. We used an Arduino-Uno board directly wired through USB to the computer in order to minimize communication latency between the board and the benchmark process running on the computer. Different setups used for testing visual / auditory stimulation and testing controller input are presented in each section of the paper. All our tests have been run on a PC embedding an Intel core i7-6700K CPU @ 4.00GHz and a Nvidia GeForce GTX 1080.

We used a timestamp (TS)-based system to run our tests and evaluate the different time-based parameters of the HTC Vive Pro system. Timeline in Figure 1b illustrates the different latencies involved in our measurement setup for a visual stimulation (similar timelines describe auditory stimulation and controller input). D0 is the duration between the draw call and the timestamp call (TS1). D1 is the duration between the stimulus timestamp and the actual stimulus display. D2 considers the delay between screen display and actual stimulus detection by the sensor. Finally, D3 is the delay between this detection and its associated timestamps (TS2). We use sensors to detect the actual stimulus into the HMD, which means that we actually measure $D1+D2+D3$ (i.e $TS2-TS1$). However, several durations can be considered negligible, i.e., less than 1ms (arbitrary threshold at $1/10$ of usual VR framerate). First, D0 can be neglected because, in our testing apps, the stimulus is timestamped right after it is generated. This was assessed by measuring the time difference between two timestamps generated just before and just after the stimulus call ($deltamean = 0.5ms, std < 0.1ms$). Then, we can consider D2 as being negligible due to the very small reaction time of the photodiode and microphone sensors used ($<< 1ms$), as documented in their electronic datasheet[†]. Lastly, D3 can be considered negligible because our entire experimental setup is wired, thus this delay depends only on the PC Arduino serial communication thread, which runs faster than 1000Hz ($< 1ms$ latency).

3 Evaluation of app-to-display latency

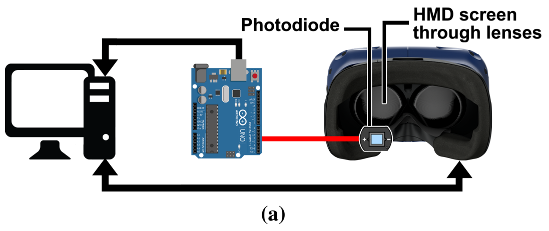
To assess the exact timing from app timestamping to actual HMD display, our setup uses a photodiode positioned in front of the HMD display panel (Figure 2a). Stimulation consists in flashing the entire HMD screen from black to white at 5Hz. It is sent from Unity/Python apps along with initial timestamp TS1. In turn, timestamp TS2 is triggered with the Arduino board when a significant change in screen luminance is measured by the photodiode.

Results Figure 3 presents our results for 600 test samples.

- Python app: $mean = 18.35ms, std = 0.96ms$
- Unity app: $mean = 31.33ms, std = 1.41ms$

The results we obtain using Python binding are in line with what has been previously described using third party commercial solution such as VRTrek by Basemark [Bas18]; using an equivalent setup with similar computer configuration and no game engine, a 20ms *"app-to-photon"* latency for the HTC Vive system was found. The Unity3D engine approximately adds 50% overload compared with the native Python-app, which is partly due to the computation introduced by its high-end rendering pipeline. Note that we ran these tests within Unity using an empty scene and a scene containing more than 1 million polygons; the two scenes provides equivalent results. Overall, latency parameters are stable over the 2 min testing period, indicating that they can be reliably extrapolated for longer experimental sessions.

[†] https://www.vishay.com/docs/81521/bpw34.pdf, https://www.mouser.fr/datasheet/2/737/MAX4465-MAX4469-932825.pdf



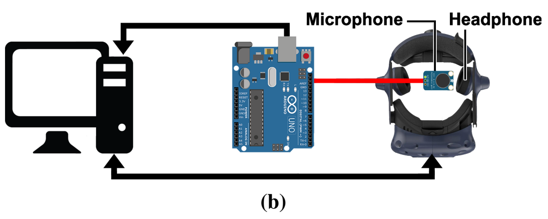


Figure 2: *Schematic view of the setup used to evaluate (a) the app-to-display and (b) the app-to-headphone latencies*

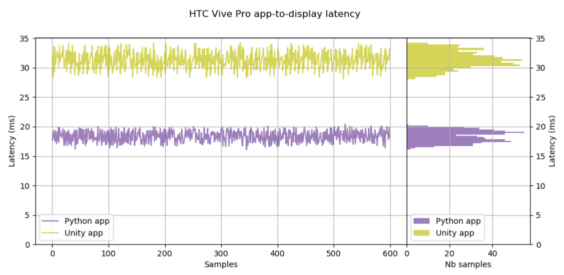


Figure 3: *App-to-display latency: sample time series and delays distribution.*

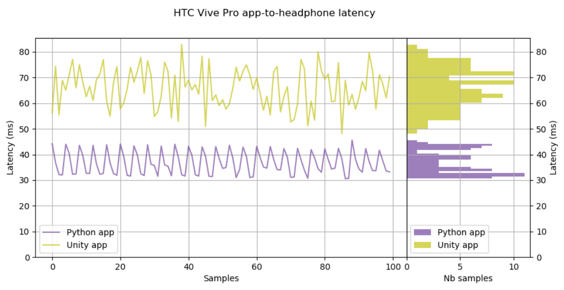


Figure 4: *App-to-headphone latency: sample time series and delays distribution.*

4 Evaluation of app-to-headphone latency

To assess the exact timing from app timestamping to actual HMD sound emission, our setup uses a microphone positioned in front of the HMD integrated headphones (Figure 2b). Stimulation consists in producing a serie of audio bips with sharp onset/offset. For the python-app, we used pyAudio to play the sound, which is a binding of PortAudio, a cross-platform low-level audio library. For the Unity-app, we chose to use a trade-off setting that balances latency and performance by setting the DSP buffer size to 2x512 samples. Timestamp TS2 is triggered when a significant change in the audio signal is measured from the microphone.

Results Figure 4 presents our results for 100 test samples.

- Python app: $mean = 36.75ms, std = 4.53ms$
- Unity app: $mean = 65.93ms, std = 7.75ms$

We observe that the Unity-app adds a 100% overload compared to the Python-app. Moreover, it seems that both mean and std values are much higher than the ones found for visual feedback. This is partly due to the buffering introduced in the audio processing pipeline, which may add variable latencies, but which also im-

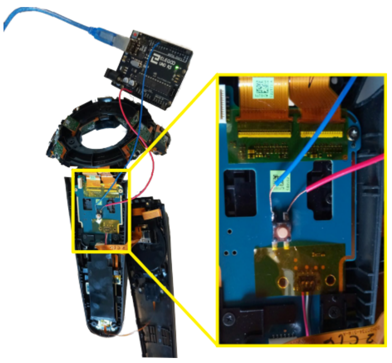


Figure 5: *Setup used to evaluate input latency of Vive controller buttons. The Arduino board is wired directly to controller board pins to measure button push at the lowest hardware level we can access.*

proves perceived sound quality. Furthermore, audio performances are very OS- and drivers-dependent [WSR10], thus this test should be run for the hardware configuration used at hand.

5 Evaluation of controller button input

For the evaluation of the HTC Vive Pro input latency parameters, we used the main pad as testing button and we hypothesize that other buttons would show similar timing performance. Indeed, maximal information transmission rate is presumably imposed by the bluetooth communication between controllers and HMD, which presumably set the upper bound in timing performances for all buttons. As we needed to access the lowest-level of the controller input, we directly wired the Arduino board to the physical button pins into the controller, as shown in Figure 5. This setup allows reading the button state through the Arduino pins directly. This way, we trigger a *"ground truth"* timestamp TS1 at the exact moment of the user button press. TS2 is generated upon button press software reading in the Python OpenVR thread and the Unity main thread, respectively. This controller *"hacking"* procedure allows bypassing the bluetooth communication overload and potential intrinsic latencies introduced by the Vive system, which can now be characterized by observing $TS2 - TS1$ differential delays.

Latency/jitter Figure 6 presents our results for 200 button push samples using Python and Unity-based apps.

- Python app: $mean = 7.18ms, std = 3.01ms$
- Unity app: $mean = 13.63ms, std = 4.80ms$

As for VR feedback (display and audio) latencies, VR input latencies show relatively small jitter ($std < 5ms$) with a stable behavior throughout testing. Average time delay between actual button presses and their software timestamping are almost doubled when observed using Unity app as compared to using the native OpenVR API through Python wrapper app. This can be explained by the fact

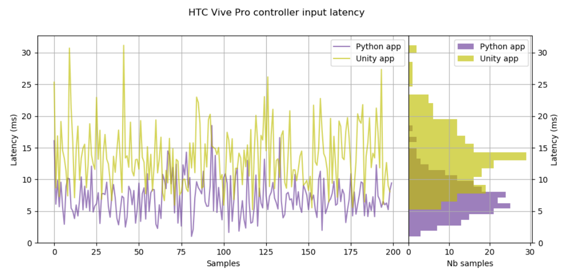


Figure 6: *Controller button latency: sample time series and delays distribution.*

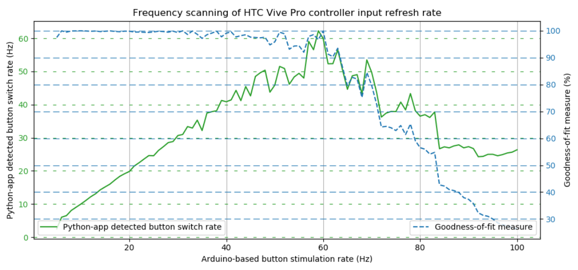


Figure 7: *Controller button refresh rate frequency scanning test.*

that input detection is performed within Unity main loop, which is bounded to run at 90FPS when simulating VR environments. This leads to an approximate $1/180 = 5.5ms$ additional delay on average, which is compatible with our observations.

Refresh rate Manual pushing of the button is a tedious task. Furthermore this approach does not allow to test the maximum refresh rate of the system. Thus, we took advantage of our setup to stimulate the button electronically; indeed, thanks to our direct wiring onto the button pins we are able to use the Arduino board to trigger button push on demand by applying a potential difference between the two button pins. This way we can benchmark the controller input refresh rate by swapping over testing frequencies, and thus evaluate its upper limit. To do so, we first run a frequency scanning test from 5 to 100Hz. These observations are shown in Figure 7. The signal variance increases monotonically up to 60Hz, showing increased button switch detection noise, after which detection performance drops with a definite bias and missed values. Therefore the system upper bound lies at a maximum refresh rate of approximately 50-60Hz. Above these values the stimulation rate might exceed functioning limits in some internal controller chips, with possible saturation effects that lead to decreased performance in button push detection or in its information transfer. To assess the consistency of the results obtained with the scanning procedure we stimulated at 60Hz over 1000 trials; we obtained $mean = 52.80Hz, std = 1.77Hz$.

6 Conclusion

In this paper we presented the first quantitative test of time performance in VR input and VR feedback of the current state-of-the-art HTC Vive Pro system. Our objective was to evaluate whether this popular system is suitable for research under strong time constraints, such as in neuroscience or biofeedback studies requiring precise synchronization between multimodal measurements. Using both low-level python-based API and a high-level game engine (Unity), our multi-level testing procedure allowed us to isolate software influence from the observed results. Our results show that, in both test conditions, latencies are non-negligible if a synchronization latency of less than 10ms is needed during the VR simulation (with minimum observed delays $> 15ms$ for VR visual feedback, $> 30ms$ for VR audio feedback, $> 5ms$ for VR input from controller button push). However jitter (latency variance) is relatively low and stable, which suggest the possibility to apply a simple correction procedure. This consists in simply counter-balancing mean observed delays by using constant offsets to re-synchronize multimodal data. Finally, we evaluated the overload introduced by the Unity game engine, which is commonly used in scientific investigations involving VR. This high-level simulation environment introduces additional latency with increased jitter in the overall system performance. Observed delays however stay in a reasonable range with a maximum values of 80ms delay during app-to-headphone stimulation. Depending on the experimental demand, research engineers can rely on Unity game engine and possibly apply some simple time compensation to correct for the most important delays. As the timing constraints increase in studies requiring more precise synchronization to other measurement devices and to actual action/perception at the user level, it might become necessary to tap into lower-level VR simulation input/output in the native OpenVR API, using some additional layers such as our Python wrapper apps. Our benchmark has been designed to be easy to setup. It is based on simple, low-cost hardware. This is meant to allow replications and extensions of our tests to other VR headsets, especially OpenVR-compatible HMDs, which currently cover all major commercial systems.

Acknowledgments
We thank Deborah Varoqui and Marco Congedo for their help in reviewing this paper, as well as Antoine Semblat for his graphical contributions.

References

[BAB11] Bohil C. J., Alicea B., Biocca F. A.: Virtual reality in neuroscience research and therapy. *Nature reviews neuroscience 12*, 12 (2011), 752. 1

[Bas18] Basemark: VRTrek by Basemark, 2018. URL: https://www.basemark.com/products/vrtrek-library/. 2

[Car81] Card S. K.: The Model Human Processor: A Model for Making Engineering Calculations of Human Performance. *Proceedings of the Human Factors Society Annual Meeting 25*, 1 (1981), 301–305. 2

[Dix96] Dix A.: Closing the loop: modelling action, perception and information. In *Proceedings of the workshop on Advanced visual interfaces* (1996), ACM, pp. 20–28. 1

[ICM*14] Iturrate I., Chavarriaga R., Montesano L., Minguez J., Millán J.: Latency correction of event-related potentials between different experimental protocols. *Journal of neural engineering 11*, 3 (2014), 036005. 1

[MCP*18] Ma X., Cackett M., Park L., Chien E., Naaman M.: Web-Based VR Experiments Powered by the Crowd. *arXiv preprint arXiv:1802.08345* (2018). 1

[NLL17] Niehorster D. C., Li L., Lappe M.: The accuracy and precision of position and orientation tracking in the HTC vive virtual reality system for scientific research. *i-Perception 8*, 3 (2017), 2041669517708205. 1

[TPSM09] Teather R. J., Pavlovych A., Stuerzlinger W., MacKenzie I. S.: Effects of tracking technology, latency, and spatial jitter on object movement. In *3D User Interfaces, 2009. 3DUI 2009. IEEE Symposium on* (2009), IEEE, pp. 43–50. 1

[WMK*18] Wade A., McCall C., Karapanagiotidis T., Schofield G., Preston C., Hartley T., Kaestner M., Horner A., Maloney R., Smallwood J., others: A neuroscientific approach to exploring fundamental questions in VR. *Electronic Imaging 2018*, 3 (2018), 435–1. 1

[WSR10] Wang Y., Stables R., Reiss J.: Audio latency measurement for desktop operating systems with onboard soundcards. In *Audio Engineering Society Convention 128* (2010), Audio Engineering Society. 3