Guided Sine Fitting for Latency Estimation in Virtual Reality

Jan-Philipp Stauffert* University of Würzburg
Florian Niebling† University of Würzburg
Jean-Luc Lugrin‡ University of Würzburg
Marc Erich Latoschik§ University of Würzburg

Abstract

Latency in Virtual Reality (VR) applications can lead to decreased performance and cybersickness. Multiple approaches exist to determine an average latency. Yet many scientific publications fail to report their system's latency despite the potentially detrimental impact. This paper extends Steed's sine fitting approach [13] by using a KCF tracking algorithm [2] to track the positions of physical objects in video recordings of VR systems. We provide a software for convenient usage. Our combination of sine fitting with KCF tracking allows to measure Motion-To-Photon latency of arbitrary tracking devices without any additional preparation or markers. The developed software is open source.

Index Terms: D.4.8 [Operating Systems]: Performance—Measurements; H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities

1 Introduction

Each computation causes latency, i.e., a time delay between input and output. The latency most often described for VR applications is the Motion-To-Photon (MTP) latency that describes the delay between a movement and the corresponding effect shown on the display. Delays during the input-to-output processing not only risk to annoy users but they potentially might induce more severe consequences of visually-induced motion sickness (VIMS) and cybersickness [9]. Hence, a central requirement of VR systems is to measure and finally control a VR system's latency behavior to judge its performance before negative consequences arise. Many researchers fail to report the MTP latency behavior of their applications despite the possible negative consequences. This makes it hard to determine if the findings were due to the experimental setup or a result of poor application performance. There are multiple approaches to measure MTP latency with varying degree of effort to put into.

This paper builds upon Steed's approach to measure MTP latency with sine fitting [13] to determine an estimation of the VR application performance. The original paper records a tracked pendulum and its virtual counterpart to fit their movement with a sine curve. It calculates the time difference between their movements using the sines' phase difference. We propose to use a tracking algorithm - the KCF tracker - to determine the position of the tracked objects without the need of additional preparation or markers. This further simplifies the application of sine fitting. We provide a software based on Qt, OpenCV and the Ceres Solver to lower the barrier of use. The original paper provides a Matlab implementation that requires Matlab knowledge to run.

*e-mail:jan-philipp.stauffert@uni-wuerzburg.de
†e-mail:florian.niebling@uni-wuerzburg.de
‡e-mail:jean-luc.lugrin@uni-wuerzburg.de
§e-mail:marc.latoschik@uni-wuerzburg.de

2 Related Work

Visual delay was found as a major contributing factor already in early simulators [5]. Latency causes cybersickness [3] and decreases performance [8]. Multiple approaches to measure MTP latency exist. He et. al. [7] employ manual frame counting. They record a tracked controller's movement and its virtual counterpart at the same time with a high speed camera. They count the time delay between movement discontinuities to infer the latency. Friston and Steed [6] propose to use image processing to automatically derive the latency in place of manual analysis. Steed [13] replaces the determination of discontinuities in the video with sine fitting. Steed attaches the tracked controller to a pendulum and fits the movement with a sine curve. The use of a continuous signal instead of detecting distinct events reduces the impact of inaccuracies due to limited temporal video resolution. Fitting a sine to the real controller's movement and its virtual image yields the MTP latency in the phase difference. Sine fitting allows to determine the turning point of a pendulum even if the geometric and temporal video resolution are insufficient to determine it directly from the video. The automatic video processing has in common that the tracked objects need to have distinct features like a LED attached. In contrast to the non invasive methods that rely on recording the scene from the outside, Di Luca [4] uses photodiodes attached to the Head-Mounted Display (HMD) in more involved hardware setups. Closer inspection of latency behavior shows that latency changes with time [10] and latency spikes affect user experience [12]. Description of this behavior requires more more in depth application analysis [11]. Putting more effort into the measuring produces better estimates. However, reporting a mean latency provides at least an estimate of the overall performance.

3 Sine Fitting

Sine fitting assumes a tracked object as part of the simulation to move in a sinosoidal pattern. Most often the tracked object is the motion controller the user employs as input modality to the VR application. It is attached to a string to hang from a fixed mount. Once pushed, it performs a sinosoidal movement. There is a display next to it that shows the virtual counterpart of the motion controller. Many VR applications have some representation of the motion controller so only the virtual camera needs to set up to provide a good view of the motion controller. The virtual motion controller follows the movement of the real motion controller so it too elicits a sinosoidal movement pattern. The processing time to do the tracking, simulation and displaying leads to the virtual motion controller having a temporal delay to the real motion controller. A video camera records both the real and virtual motion controller and their movements. A sine curve is fitted to the movements of both objects. The phase difference between both sine curves describes their temporal offset and therefore the MTP latency.

4 Implementation

We created a software based on Qt 5.13.2, OpenCV 4.1.1 and the Ceres solver 1.14 [1] for automatic calculation of MTP latency in VR systems, by comparing the sinosoidal movements of physical objects and their virtual counterparts in video recordings of these systems. Figure 2 shows a screenshot.

The usage is a follows: The video to be analyzed is loaded from disk. OpenCV tries to detect the video's frame rate given as frames

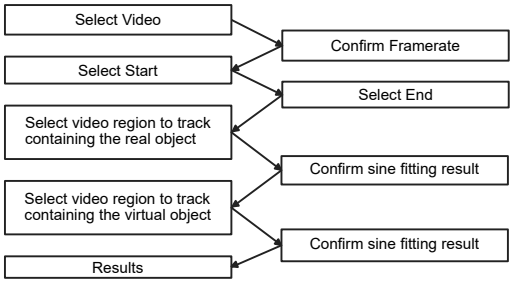

Figure 1: The application flow describing the required user interaction.

Figure 2: Application screenshot showing the selection of the real object in our synthetic test video.

per second (fps). The user can change the fps used for the calculation to account for inaccurate detection. The start and end time of the video is selected. Videos often show the experimenter's hands in the beginning and end to operate the setup which needs to get cut away. The user then selects a rectangular region in the video that shows the real object. A larger region results in a longer computation time for the tracking algorithm but provides better results. The user is asked to change the selection if the tracker looses the tracked object before all video frames are processed. A principle component analysis (PCA) converts the resulting 2d points to 1d values. The Ceres solver in turn fits a sine wave described by $e^{-dt} \cdot \cos(2\pi ft + \varphi)$ to the data using gradient descend. The decay term e^{-dt} serves as adjustment if the pendulum slows down over time. The frequency f is estimated beforehand by taking the largest frequency in the DFT (discrete fourier transform) transformed data and refined by Ceres. The fitted curve is overlayed over the tracked positions for visual inspection. Some times, the fit shows visible artifacts which necessitate choosing a different region to track and a repeat of the curve fitting. Once the real object's movement is processed, the user selects its virtual counterpart which undergoes the same procedure. The latency between the real object's movement and its virtual counterpart's movement results from the phase difference $\hat{\varphi}$ of both fitted sine waves as latency $= \frac{\hat{\varphi}}{2\pi f \cdot \text{fps}}$. Figure 1 shows a step by step overview.

5 Validation

First validations both with synthetic videos where two rectangles oscillated with a known time difference and two videos of Friston and Steed's mechanical latency simulator [6] that are available online show average estimation errors of 2.5 ms. This is similar to errors described for other analyses of the online videos.

6 Discussion

Results may vary depending on the video and tracking quality. We advice to repeat the analysis multiple times to not fall prey to outliers due to an inopportune choice of the region to track.

Sine fitting necessitates that the movement of a real object and its virtual counterpart are observable at the same time. Our approach is directly applicable to screen-based or projection-based VR setups, where target objects and their graphical counterpart are simultaneously visible. However, typical HMDs require an additional step. First, our approach has to be applied using a proxy display, e.g., a standard monitor. Then we have to calculate the latency offset between the target HMD and the proxy display, e.g., using a high speed camera that records both the monitor and the HMD screen. A solid color is displayed on both, then changed to another color. The camera detects the time difference between the second color shown on the respective screen. The time difference allows to derive the MTP latency from the real object to an HMD's screen. This approach needs the possession of a high speed camera while the sine fitting approach itself works with lower frame rates. The reason is a tradeoff in camera abilities between geometric and temporal resolution. Color changes need only little geometric resolution but high temporal resolution to determine time differences between two regions in a video. Capturing movement of a real object requires high geometric resolution which makes high framerates costly or impossible.

7 Conclusion

We suggest an improvement in usability of the sine fitting latency estimation for VR applications by using a KCF tracker. We provide the implementation as a downloadable Windows binary and provide full source access (`https://go.uniwue.de/autosine`).

References

[1] S. Agarwal, K. Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

[2] M. Danelljan, F. S. Khan, M. Felsberg, and J. v. d. Weijer. Adaptive color attributes for real-time visual tracking. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1090–1097. IEEE. doi: 10.1109/CVPR.2014.143

[3] S. Davis, K. Nesbitt, and E. Nalivaiko. A systematic review of cybersickness. pp. 1–9. ACM Press. doi: 10.1145/2677758.2677780

[4] M. Di Luca. New method to measure end-to-end delay of virtual reality. *Presence: Teleoperators and Virtual Environments*, 19(6):569–584, 2010. doi: 10.1162/pres_a_00023

[5] L. H. Frank, J. G. Casali, and W. W. Wierwille. Effects of visual display and motion system delays on operator performance and uneasiness in a driving simulator. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 30(2):201–217, 1988.

[6] S. Friston and A. Steed. Measuring latency in virtual environments. 20(4):616–625. doi: 10.1109/TVCG.2014.30

[7] D. He, F. Liu, D. Pape, G. Dawe, and D. Sandin. Video-based measurement of system latency. In *International Immersive Projection Technology Workshop*, p. 111, 2000.

[8] Z. Ivkovic, I. Stavness, C. Gutwin, and S. Sutcliffe. Quantifying and Mitigating the Negative Effects of Local Latencies on Aiming in 3d Shooter Games. pp. 135–144. ACM Press, 2015. doi: 10.1145/2702123.2702432

[9] L. Rebenitsch and C. Owen. Review on cybersickness in applications and visual displays. *Virtual Reality*, 20(2):101–125, 2016.

[10] J.-P. Stauffert, F. Niebling, and M. E. Latoschik. Reducing application-stage latencies of interprocess communication techniques for real-time interactive systems. In *Virtual Reality (VR), 2016 IEEE*, pp. 287–288. IEEE, 2016. doi: 10.1109/VR.2016.7504766

[11] J.-P. Stauffert, F. Niebling, and M. E. Latoschik. Towards comparable evaluation methods and measures for timing behaviour of virtual reality systems. In *Proceeding of the 22nd ACM Symposium on Virtual Reality Software and Technology (VRST)*, 2016.

[12] J.-P. Stauffert, F. Niebling, and M. E. Latoschik. Effects of latency jitter on simulator sickness in a search task. In *Proceedings of the 25th IEEE Virtual Reality (VR) conference*, 2018.

[13] A. Steed. A simple method for estimating the latency of interactive, real-time graphics simulations. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology*, VRST '08, pp. 123–129. ACM. doi: 10.1145/1450579.1450606