SmartSimVR: An Architecture Integrating Machine Learning and Virtual Environment for Real-Time Simulation Adaptation

Azadeh Hadadi*[1,2], Jean-Rémy Chardonnet[2], Christophe Guillet[3], and Jivka Ovtcharova[1]
[1]Institute for Information Management in Engineering, Karlsruhe Institute of Technology, Karlsruhe, Germany
azadeh.hadadi@kit.edu, jivka.ovtcharova@kit.edu
[2]Arts et Metiers Institute of Technology, LISPEN, HESAM Université, UBFC, Chalon-Sur-Saône, France
azadeh.hadadi@ensam.eu, jean-remy.chardonnet@ensam.eu
[3]Université de Bourgogne, LISPEN, UBFC, Chalon-Sur-Saône, France
christophe.guillet@u-bourgogne.fr

***Abstract*—This paper introduces SmartSimVR, a groundbreaking research initiative focused on the development of a user-specific intelligent architecture for immersive virtual environments. The primary objective of this architecture is to address real-time artificial intelligence training and adapt the virtual environment based on the user's state or external parameters. In a case study centered around the detection of cybersickness, an undesirable side effect in immersive virtual environments, we employed this architecture in a driving simulator application. Leveraging the capabilities of this architecture enables the optimization of virtual reality experiences for individual users, resulting in increased comfort.**

***Index Terms*—Auto-Adaptation, Machine Learning, Simulation, Virtual Systems**

I. Introduction

Virtual reality (VR) is a transformative technology that offers immersive experiences by simulating realistic environments. Its applications span various domains, including gaming, training simulations, virtual tourism, and telepresence. In these immersive virtual environments, adaptation is indispensable for meeting the diverse needs of users, enhancing user experience, and optimizing immersion and satisfaction.

However, real-time data processing and auto-adaptation capabilities pose significant challenges for some of the virtual applications like driving simulators. These simulators require the ability to render detailed environments, simulate accurate vehicle dynamics, and process user inputs in real-time. Such computational tasks can strain the CPU. While hardware advancements have partially addressed high CPU usage by introducing more powerful CPUs and GPUs tailored for simulation and gaming, these resources may not be accessible to everyone. Therefore, the development of optimized software and platforms becomes crucial in maximizing the utilization of available equipment, ensuring efficient performance even with less powerful hardware configurations.

The advent of Artificial Intelligence (AI) in recent years and its integration into VR applications has opened up new possibilities and enhanced the overall experience for users. By leveraging AI algorithms and techniques [1], VR applications can intelligently analyze and interpret user inputs, enabling more advanced and dynamic interactions within virtual environments.

Based on this analysis, the VR application can dynamically adapt various aspects of the virtual environment to match the user's preferences and state continually refining and enhancing the personalized experience [2] [3] [4] [5] [6].

However, AI training for personalizing the VR experience typically occurs offline, before the deployment of the VR application. The training phase involves feeding the model with historical user data to learn patterns and relationships. Once the trained model is integrated into the VR application, the adaptation and feedback process can happen in real-time within a closed-loop system.

An important drawback of offline training is the inability to learn and adapt in real-time based on immediate user interactions. This limitation hinders the model's ability to capture dynamic changes and evolving user preferences, leading to suboptimal and generalized recommendations. Additionally, offline-trained models may struggle with novel data patterns or user behaviors not present in the training dataset, potentially resulting in inaccurate or inappropriate adaptations.

A. Contribution

Our research introduces an innovative intelligent auto-adapted VR architecture that seamlessly integrates multiple concurrent processes with AI technology as demonstrated in Figure 1. It analyzes real-time data and enables continuous self-training of the AI system. By minimizing data collection and utilizing personalized training, the architecture enhances efficiency and addresses specific challenges in VR applications. Overall, our architecture contributes to concurrent data processing, AI integration, and auto-adaptation in VR without imposing heavy computational resource requirements.

This work was supported by a grant from the French-German University CDFA 03-19.

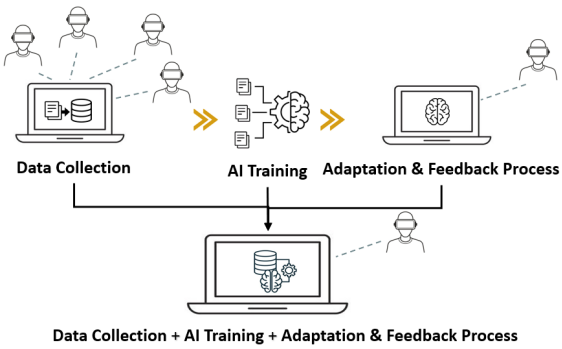

Fig. 1. Overview of the intelligent auto-adaptive VR application. Integrating three distinct stages into a real-time closed loop.

II. Related Works

The majority of studies on adaptive VR applications have commonly utilized supervised learning algorithms, such as deep neural network (DNN) [4] or random forest [3], as the adaptive logic. In supervised learning creating a suitable training dataset can be costly or unfeasible in certain domains, such as rehabilitation. To address this challenge, reinforcement learning (RL) and deep reinforcement learning (DRL) have been employed. For instance, Tsiakas et al. [5] utilized RL for adaptive VR training in rehabilitation scenarios. Mao et al. [6] used DRL to plan and execute disassembly sequences for the VR maintenance training system. Nevertheless, RL and DRL training can be computationally demanding and time-consuming, particularly when dealing with complex environments and large state or action spaces. Real-time training may require substantial computational resources to process and learn from interactions with the environment.

III. Key Features of SmartSimVR

SmartSimVR bridges the existing gap in the development of such a closed-loop architecture as showcased in Figure 2. This architecture is built upon four key components:

- **Distribution:** The integration of the architecture involves distributing its modules across multiple systems, with the VR application on one system and the AI module on another. This distribution can be achieved through various methods like TCP/IP, HTTP requests, message queues, or distributed computing architectures such as Apache Kafka or RabbitMQ. These protocols and technologies facilitate smooth data and message transfer between the VR application and the AI module, ensuring efficient and reliable information exchange.
- **Concurrent Processes** Given that the auto-adaptation system operates in real-time, it necessitates a multi-threaded architecture to effectively process data. This architecture incorporates multiple independent processes that work concurrently.
- **Shared Virtual Memory System:** To ensure smooth transitions and seamless data transfer among all concurrent processes, including the AI module, we have used the shared virtual memory system. This mechanism enables efficient and synchronized data sharing and collaboration between independent processes.
- **Stream Learning as the Adaptive Logic:** Our architecture incorporates a Stream learning approach, also referred to as Online learning or Incremental learning [7]. Instead of other supervised learning methods that rely on training general datasets and substantial computational resources, our system leverages the benefits of stream learning. Unlike traditional batch learning techniques, which process the entire dataset at once, stream learning trains the model incrementally on a continuous data stream. This methodology enables the model to learn from individual observations or small groups of observations sequentially. The use of stream learning is particularly advantageous for real-time applications that encounter rapid changes and have limited computing resources. By adapting to new data as it arrives, our architecture ensures the model can continuously update its knowledge and make timely decisions in dynamic environments. This approach proves highly effective for scenarios where real-time adaptation and efficient resource utilization are essential.

IV. Application of SmartSimVR: A Case Study

To investigate the intelligent auto-adaptation of VR, a driving simulator was developed as the application of interest. The integration of AI and the VR application is depicted in Figure 3, outlining the general concept. The virtual driving simulation was meticulously crafted within a city scene, as illustrated in Figure 4. Participants actively engaged in driving through the streets of this virtual environment, which was thoughtfully designed to form a continuous loop. We used this system to adapt the virtual environment based on the user state to solve the cybersickness problem, also referred to as visually induced motion sickness (VIMS) [8] or simulation sickness. Cybersickness is akin to motion sickness and characterized by symptoms such as nausea, discomfort in eye movements, and a sense of disorientation [9].

To mitigate the problem of cybersickness, we employed SmartSimVR for real-time adaptation of the virtual environment, utilizing user state as a determining factor. The evaluation of cybersickness involves both subjective and objective measures employed by researchers [10]. Subjective evaluation includes participants completing questionnaires like the motion sickness questionnaire (MSQ) [11], Simulator Sickness Questionnaire (SSQ) [12], Fast Motion Sickness Scale (FMS) [13], and VR Sickness Questionnaire (VRSQ) [14] to capture their subjective impressions. On the other hand, objective evaluation entails monitoring participants' physiological responses during their engagement in virtual environments. Measurements such as postural sway [15], electrodermal activity (EDA) [16] [17], electroencephalogram (EEG) [18], and electrocardiogram (ECG) [19] are recorded in real-time to analyze the occurrence and severity of cybersickness.

In our experiment, we used objective measures including physiological indicators and behavioral measurements. For be-

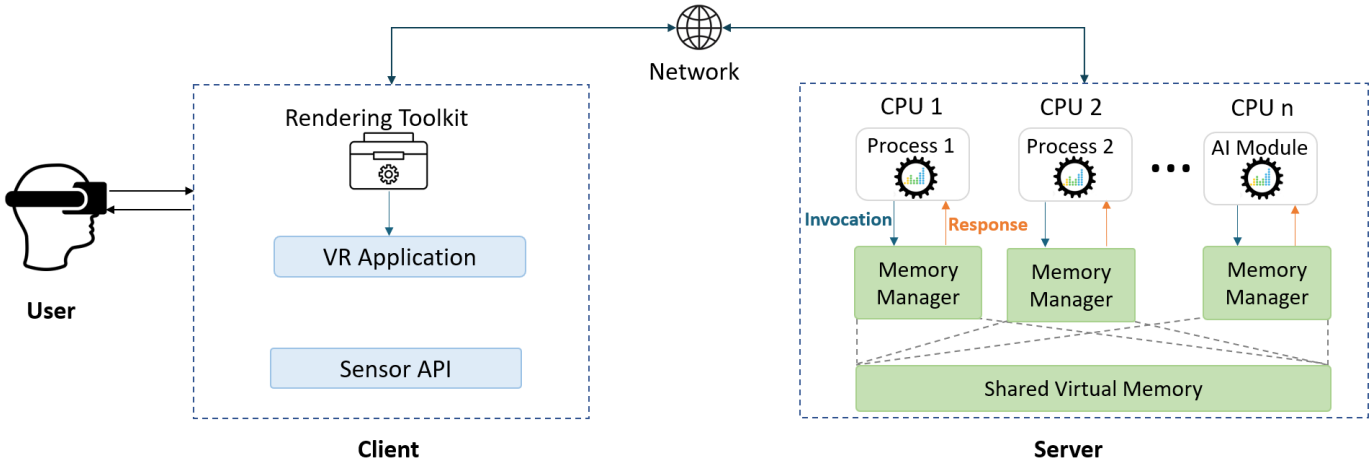

Fig. 2. Overview of SmartSimVR architecture.

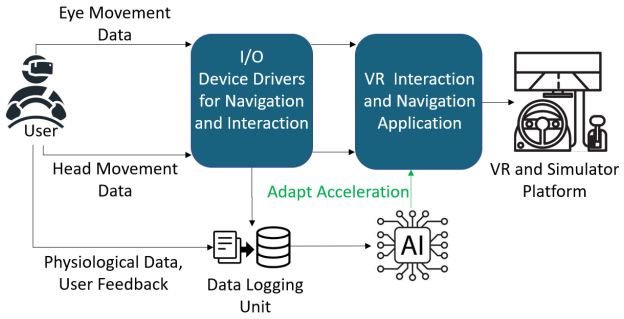

Fig. 3. Overview of the driving simulation experiment based on auto-adaptation.

Fig. 4. Sample frame from the driving simulation along with the city scene (top right corner).

havioral measurements, we recorded participants' head movements and collected eye tracker data using a Meta Quest Pro head-mounted display (HMD) equipped with an eye tracker.

To expand the range of physiological indicators, we utilized an Empatica E4 wristband worn by participants. This wristband has sensors that capture various physiological parameters, with a focus on electrodermal activity (EDA). The data was transmitted to a server computer via Bluetooth during the navigation experiment. The sensors included Galvanic Skin Response (GSR), blood volume pressure (BVP), heart rate (HR), temperature (TEM), and 3-axis accelerometer sensors. These sensors recorded participants' physiological responses while they performed the navigation task. Participants were prompted with an audio cue, "What is your score?", at one-minute intervals. They verbally expressed their level of sickness based on predefined definitions: 0 for no sickness, 1 for initial symptoms, 2 for moderate symptoms, and 3 for severe symptoms. Before the experiment, participants received training on the predefined symptom-based score range.

V. Implementation Details

Figure 5 provides a comprehensive overview of the customized SmartSimVR, illustrating its components and functionalities. As discussed in the previous section, this architecture enables simultaneous data recording and processing by incorporating multiple independent processes that operate concurrently. Each process is designed to fulfill specific purposes at pre-configured intervals, such as every minute.

A. Client Side

The client side of the system comprises two crucial components.

- **VR Application:** The VR Application is the driving simulation application at the core of the system. It provides an immersive virtual reality experience and collects important data, including eye tracker and head movement data. These captured data points offer valuable insights into users' visual behavior and physical responses in the virtual environment.
- **E4 Streaming Server:** The E4 Streaming Server is used to work in conjunction with the Emaptica E4. This server component establishes a Bluetooth connection with the E4 wristband worn by participants. Its real-time streaming capability enables continuous monitoring and analysis of participants' physiological responses throughout the VR experiment.

Together, these client-side components synergistically contribute to the effective operation and comprehensive data collection within the system.

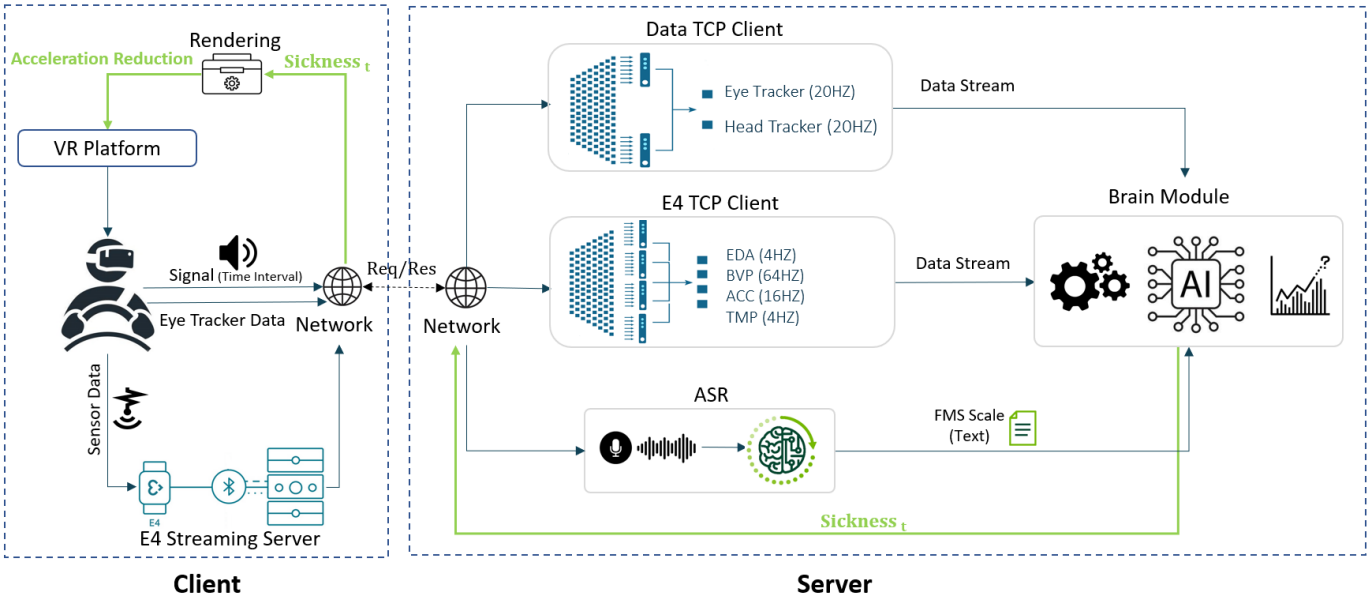

Fig. 5. Architecture design and data flow diagram of the implemented architecture, a distributed system implemented across Client and Server.

B. Server Side

The server side of this implementation of SmartSimVR incorporates multiple essential components.

- **Automatic Speech Recognition (ASR):** The ASR component plays a vital role in facilitating voice-based interactions and inputs within the system [20] [21]. By activating the microphone, it receives the user's voice and converts speech into text, enabling seamless communication.
- **Data TCP Client:** This component is responsible for receiving crucial eye tracker data and head movement data from the VR application. It establishes a TCP connection and ensures the seamless transmission of these important eye movement data and behavioral measurements to the Brain module for further analysis.
- **E4 TCP Client:** The E4 TCP Client component, on the other hand, focuses on receiving E4 data from the E4 Streaming Server. It establishes a TCP connection and efficiently retrieves the physiological data captured by the E4 wristband. This acquired data is then made available for processing and integration into the system's analysis pipeline.
- **Brain Module** The Brain Module serves as the central component on the server side, encompassing three submodules:
  - **Pre-Processing Module:** This module plays a crucial role in fusing and synchronizing the time series data collected from various sensors as is shown in Figure 5. The Empatica E4 wristband incorporates sensors with different sampling frequencies, including a 4 Hz EDA sensor, a 64 Hz PPG sensor for blood volume pressure (BVP), a 4 Hz infrared thermopile (TEM) for temperature measurement, a 32 Hz 3-axis accelerometer sensor, and eye-tracking data with head movement managed within the VR application at a frequency of 20 Hz. To ensure consistency and coherency, a dedicated pre-processing module has been developed. It synchronizes the collected data to a unified frequency of 4 Hz, effectively merging data from different sources for subsequent analysis.
  - **AI Module:**
    1) **Model:** As discussed in section III, our architecture incorporates a Stream learning approach. We have established a pipeline that integrates a scaler transformer with a binary classifier based on a linear logistic regression model [22]. To optimize the performance of the model, we have chosen stochastic gradient descent (SGD) [23] as the algorithm, with a learning rate set to 0.01.
    2) **Classification Indicator:** We used the self-reported FMS scale to classify the data into sick (scale $>=$ 1) and non-sick (scale $=$ 0) observations. Users verbally provided these indicators, which were then converted to text using the ASR module. To match participants' reporting capabilities, we modified the original FMS scale from 1 to 20 to a simplified scale of 0 to 3. This adjustment was necessary as participants were unable to report their status with high-resolution accuracy.
  - **Detection/Adaptation Module:**After sufficient training according to the prescribed protocol (three minutes of reporting "sick"), the AI model receives data from both the Data TCP client and the E4 TCP Client at a pre-set interval, usually 1 minute. If the trained model detects sickness in a record, it stops validating the remaining records in that data packet.

- **Adaptive Variable:** To mitigate cybersickness, we employed linear and rotational accelerations as adaptive variables [16]. This led to a deceleration of the virtual car in the VR program. The deceleration was accomplished by reducing the engine power by 70% in each detection time, directly impacting the primary parameter controlling the car's engine torque.

VI. Discussion

The SmartSimVR architecture is a distributed architecture in VR applications that offers numerous benefits. It incorporates concurrent processes communicating through a shared memory system, enabling efficient data exchange and preparation for the AI module.

One key advantage of SmartSimVR is stream learning, which allows real-time adaptation of the virtual environment based on the user's context. Stream learning is preferred over batch learning models like reinforcement learning due to its efficiency in terms of time, resources, and data utilization. This eliminates the need for extensive offline training with large datasets, as the AI module trains itself using user-specific data for immediate personalization.

The distributed nature of the SmartSimVR architecture optimizes resource utilization by distributing the workload across multiple systems. This effectively handles high-load processes in VR applications, leading to improved performance, reduced latency, and enhanced scalability. The result is a seamless and immersive user experience.

VII. Conclusion and Future Work

In this paper, we introduced SmartSimVR, a user-specific intelligent architecture for immersive virtual environments. The architecture addresses real-time AI training and adapts the virtual environment based on the user's state or external parameters. Through a case study on cybersickness, we successfully trained an AI model in real time and personalized it for individual users in a driving simulator application.

Future work includes a comprehensive analysis of SmartSimVR's performance, evaluating metrics like response time, scalability, memory usage, CPU utilization, and network latency. By quantitatively assessing its performance, we aim to gain deeper insights and identify areas for improvement.

Furthermore, we envision expanding SmartSimVR beyond cybersickness mitigation. The architecture has potential applications in adaptive training simulations, personalized gaming experiences, and real-time user feedback systems, enhancing various aspects of virtual reality experiences.

References

[1] M. A. Mirzaei, S. Prianto, J.-R. Chardonnet, C. Pere, and F. Mérienne, "New motherwavelet for pattern detection in ir image," in *2013 Visual Communications and Image Processing (VCIP)*. IEEE, 2013, pp. 1–6.

[2] A. Bouatrous, A. Meziane, N. Zenati, and C. Hamitouche, "A new adaptive vr-based exergame for hand rehabilitation after stroke," *Multimedia Systems*, vol. 29, no. 6, pp. 3385–3402, 2023.

[3] D. Bian, J. Wade, A. Swanson, A. Weitlauf, Z. Warren, and N. Sarkar, "Design of a physiology-based adaptive virtual reality driving platform for individuals with asd," *ACM Transactions on Accessible Computing (TACCESS)*, vol. 12, no. 1, pp. 1–24, 2019.

[4] R. Islam, S. Ang, and J. Quarles, "Cybersense: A closed-loop framework to detect cybersickness severity and adaptively apply reduction techniques," in *2021 IEEE Conference on virtual reality and 3d user interfaces abstracts and workshops (VRW)*. IEEE, 2021, pp. 148–155.

[5] K. Tsiakas, M. Huber, and F. Makedon, "A multimodal adaptive session manager for physical rehabilitation exercising," in *proceedings of the 8th ACM international conference on pervasive technologies related to assistive environments*, 2015, pp. 1–8.

[6] H. Mao, Z. Liu, and C. Qiu, "Adaptive disassembly sequence planning for vr maintenance training via deep reinforcement learning," *The International Journal of Advanced Manufacturing Technology*, pp. 1–10, 2021.

[7] A. Bifet, R. Gavalda, G. Holmes, and B. Pfahringer, *Machine learning for data streams: with practical examples in MOA*. MIT press, 2023.

[8] M. A. Mirzaei, "Influence of interaction techniques on vims in virtual environments: estimation et prédiction," Ph.D. dissertation, Ecole nationale supérieure d'arts et métiers-ENSAM, 2014.

[9] N. Dużmańska, P. Strojny, and A. Strojny, "Can simulator sickness be avoided? a review on temporal aspects of simulator sickness," *Frontiers in psychology*, vol. 9, p. 2132, 2018.

[10] Y. Niu, D. Wang, Z. Wang, F. Sun, K. Yue, and N. Zheng, "User experience evaluation in virtual reality based on subjective feelings and physiological signals," *Journal of Imaging Science and Technology*, vol. 63, no. 6, pp. 60 413–1, 2019.

[11] L. Frank, R. S. Kennedy, R. S. Kellogg, M. E. McCauley, and E. C. O. FL, "Simulator sickness: A reaction to a transformed perceptual world. 1. scope of the problem," in *Proceedings of the Second Symposium of Aviation Psychology, Ohio State University, Columbus OH*, 1983, pp. 25–28.

[12] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal, "Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness," *The international journal of aviation psychology*, vol. 3, no. 3, pp. 203–220, 1993.

[13] B. Keshavarz and H. Hecht, "Validating an efficient method to quantify motion sickness," *Human factors*, vol. 53, no. 4, pp. 415–426, 2011.

[14] H. K. Kim, J. Park, Y. Choi, and M. Choe, "Virtual reality sickness questionnaire (vrsq): Motion sickness measurement index in a virtual reality environment," *Applied ergonomics*, vol. 69, pp. 66–73, 2018.

[15] J.-R. Chardonnet, M. A. Mirzaei, and F. Mérienne, "Features of the postural sway signal as indicators to estimate and predict visually induced motion sickness in virtual reality," *International Journal of Human–Computer Interaction*, vol. 33, no. 10, pp. 771–785, 2017.

[16] J. Plouzeau, J.-R. Chardonnet, and F. Merienne, "Using cybersickness indicators to adapt navigation in virtual reality: a pre-study," in *2018 IEEE conference on virtual reality and 3D user interfaces (VR)*. IEEE, 2018, pp. 661–662.

[17] A. Hadadi, C. Guillet, J.-R. Chardonnet, M. Langovoy, Y. Wang, and J. Ovtcharova, "Prediction of cybersickness in virtual environments using topological data analysis and machine learning," *Frontiers in Virtual Reality*, vol. 3, p. 973236, 2022.

[18] J. Kim, W. Kim, H. Oh, S. Lee, and S. Lee, "A deep cybersickness predictor based on brain signal analysis for virtual reality contents," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 10 580–10 589.

[19] A. Garcia-Agundez, C. Reuter, H. Becker, R. Konrad, P. Caserman, A. Miede, and S. Göbel, "Development of a classifier to determine factors causing cybersickness in virtual reality environments," *Games for health journal*, vol. 8, no. 6, pp. 439–444, 2019.

[20] M. A. Mirzaei, F. Merienne, and J. H. Oliver, "New wireless connection between user and ve using speech processing," *Virtual Reality*, vol. 18, pp. 235–243, 2014.

[21] M. A. Mirzaei, J.-R. Chardonnet, C. Père, and F. Mérienne, "Sensor fusion for interactive real-scale modeling and simulation systems," in *Proceedings of CGAMES'2013 USA*. IEEE, 2013, pp. 149–153.

[22] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *Journal of biomedical informatics*, vol. 35, no. 5-6, pp. 352–359, 2002.

[23] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer, 2010, pp. 177–186.