Machine Learning Application for Real-Time Simulator

Azadeh Hadadi*
Institute for Information Management in Engineering,
Karlsruhe Institute of Technology
Karlsruhe, Germany
Arts et Metiers Institute of Technology, LISPEN, HESAM
Université, UBFC
Chalon-Sur-Saône, France
azadeh.hadadi@ensam.eu

Jean-Rémy Chardonnet
Arts et Metiers Institute of Technology, LISPEN, HESAM
Université, UBFC
Chalon-Sur-Saône, France
jean-remy.chardonnet@ensam.eu

Christophe Guillet
Université de Bourgogne, LISPEN, UBFC
Chalon-Sur-Saône, France
christophe.guillet@u-bourgogne.fr

Jivka Ovtcharova
Institute for Information Management in Engineering,
Karlsruhe Institute of Technology
Karlsruhe, Germany
jivka.ovtcharova@kit.edu

ABSTRACT

This paper presents a groundbreaking research initiative that focuses on the development of an intelligent architecture for Adaptive Virtual Reality Systems (AVRS) in immersive virtual environments. The primary objective of this architecture is to enable real-time artificial intelligence training and adapt the virtual environment based on user states or external parameters. In a case study focused on detecting cybersickness, an undesired side effect in immersive virtual environments, we utilized this architecture to train an artificial intelligence model and personalize it for individual users in a driving simulator application. By leveraging the capabilities of this architecture, we can optimize virtual reality experiences for individual users, leading to increased comfort. We evaluated the system's performance in terms of memory usage, CPU and GPU usage, temperature monitoring, frame rate, and network performance, and our results demonstrated the efficiency of our proposed architecture.

CCS CONCEPTS

• **Computer systems organization → Real-time system architecture**.

KEYWORDS

artificial intelligence, auto-adaptive systems, real-time systems

ACM Reference Format:
Azadeh Hadadi, Jean-Rémy Chardonnet, Christophe Guillet, and Jivka Ovtcharova. 2024. Machine Learning Application for Real-Time Simulator. In *2024 9th International Conference on Machine Learning Technologies (ICMLT) (ICMLT 2024), May 24–26, 2024, Oslo, Norway*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3674029.3674030

ICMLT 2024, May 24–26, 2024, Oslo, Norway

ACM ISBN 979-8-4007-1637-9/24/05
https://doi.org/10.1145/3674029.3674030

1 INTRODUCTION

Advanced technologies such as Adaptive Virtual Reality Systems (AVRS) have revolutionized the Virtual Reality (VR) experience by dynamically tailoring it to individual preferences and requirements. This adaptive process optimizes immersion and satisfaction, promoting accessibility and accommodating diverse user profiles in real-time.

The integration of Artificial Intelligence (AI) into VR applications has contributed significantly to enriching the overall user experience. Leveraging AI algorithms and techniques [16], VR applications can intelligently analyze and interpret user inputs, enabling more sophisticated and dynamic interactions within virtual environments. AI, acting as adaptive logic, facilitates real-time adaptation and personalization of the VR experience.

However, AI training for personalizing the VR experience often happens offline before deploying the VR application. The model learns from historical user data and can adapt in real-time using user data collected during interactions. However, this approach lacks real-time learning and adaptation within a closed-loop system. The offline-trained model may not capture dynamic changes or evolving user preferences, leading to suboptimal recommendations. It may also struggle with novel data patterns or user behaviors, potentially resulting in inaccurate adaptations.

The majority of studies on AVRS have commonly utilized supervised learning algorithms, such as Deep Neural Network (DNN) [9] or random forest [1], as the adaptive logic. With supervised learning, the algorithm is initially trained on a dataset, known as the training data, to learn relationships between input and output values. It can then predict output values, such as workload level, scenario difficulty, or performance, based on the learned patterns from the training data. However, 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. [19] utilized RL for adaptive VR training in rehabilitation scenarios. Mao et al. [14] 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.

To address these challenges, we have developed and implemented an AVRS architecture called SmartSimVR. This architecture effectively integrates multiple concurrent processes with AI technology, enabling real-time analysis of data and continuous self-training of the AI system. By minimizing data collection and leveraging personalized training, our architecture significantly enhances efficiency and effectively addresses specific challenges in VR applications.

2 ARCHITECTURE KEY FEATURES

To bridge the gap between intelligent VR application training and real-time adaptation our team developed an advanced architecture based on four fundamental components.

- `Distribution`: The integration involves distributing the architecture's modules across multiple systems, with the VR application on one system and the AI module on another. Various methods, such as TCP/IP, HTTP requests, and message queues, enable smooth data transfer between the two, ensuring efficient and reliable communication.
- `Concurrent Processes`: The real-time auto-adaptation system relies on a well-designed multi-threaded architecture for efficient data processing. This architecture utilizes concurrent processes to handle dynamic adaptive VR applications, ensuring real-time adaptation based on user inputs and environmental changes.
- `Shared Virtual Memory System`: Shared virtual memory system is employed to enable smooth transitions and synchronized data sharing among concurrent processes, including the AI module. This mechanism facilitates efficient coordination and collaboration between components, improving overall performance by reducing inter-process communication overhead.
- `Stream Learning as the Adaptive Logic`: Our architecture utilizes a Stream learning [2] approach, which involves incremental learning from a continuous data stream. Unlike batch learning methods, which process the entire dataset at once, stream learning trains the model sequentially on individual or small groups of observations (see Figure1). This approach is advantageous for real-time applications with limited computing resources and rapid changes. By continuously updating its knowledge, our architecture ensures timely decision-making in dynamic environments, making it highly effective for real-time adaptation and efficient resource utilization.

3 CASE STUDY

To delve into the realm of intelligent auto-adaptation in VR, we conducted a research study focused on implementing a driving simulator as the target application of interest. The integration of AI

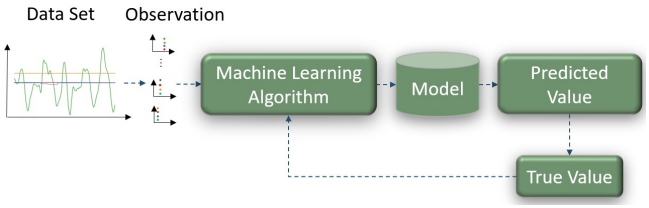

Figure 1: Stream Machine Learning

technology with the VR application is visually depicted in Figure 2, providing an overview of the underlying concept. Our team meticulously crafted a virtual driving simulation set within a city scene. Participants actively immersed themselves in the virtual environment, maneuvering through the carefully designed streets. This virtual environment was thoughtfully constructed to form a continuous loop, ensuring a seamless and immersive driving experience.

The primary objective of our research was to address the challenge of cybersickness, also known as visually induced motion sickness (VIMS) [15] or simulation sickness. Cybersickness is a prevalent issue encountered during VR navigation and bears a resemblance to traditional motion sickness. It manifests through various symptoms, including nausea, discomfort in eye movements, and a sense of disorientation [4]. To mitigate the problem of cyber-

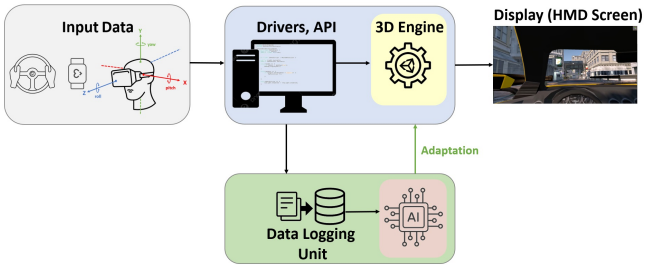

Figure 2: Overview of the driving simulation experiment based on auto-adaptation

sickness, we implemented our proposed architecture 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 [17]. Subjective evaluation entails participants completing a questionnaire regarding their experiences with VR tasks. Questionnaires such as the motion sickness questionnaire (MSQ) [5], Simulator Sickness Questionnaire (SSQ) [10], Fast Motion Sickness Scale (FMS) [11], and VR sickness questionnaire (VRSQ) [12] are utilized to capture their subjective impressions of cybersickness. On the other hand, objective evaluation entails monitoring participants' physiological responses during their engagement in virtual environments. Measurements such as postural sway [3], electrodermal activity (EDA) [18][8], electroencephalogram (EEG) [13], and electrocardiogram (ECG) [6] are recorded to analyze the occurrence and severity of cybersickness.

In our experiment, we incorporated objective measures that encompassed both physiological indicators and behavioral measurements. As part of the behavioral measurements, we recorded

participants' head movements to capture their physical responses during the experiment. To gain further physiological insights into participants' eye movements, we collected eye tracker data. Participants wore a Meta Quest Pro head-mounted display (HMD) equipped with an eye tracker for this purpose as is shown in Figure 3. In our study, we expanded the range of physiological indicators

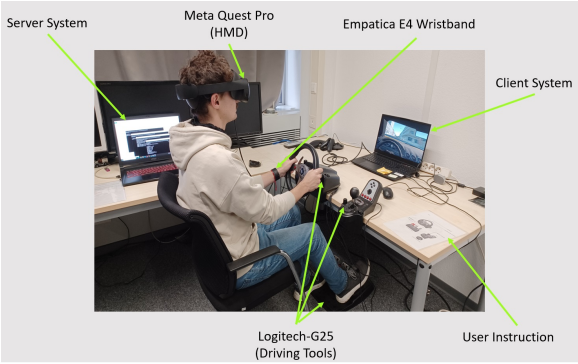

Figure 3: Experimental setup for the study with Meta Quest Pro head-mounted display (HMD) and Empatica E4 wristband attached to a participant.

by utilizing an Empatica E4 wristband worn by participants. This wristband is equipped with high-quality sensors, including Galvanic Skin Response (GSR), blood volume pressure (BVP), heart rate (HR), and temperature (TEM) sensors. Additionally, we computed longitudinal (LG) and rotational (RT) accelerations based on the recorded navigation speed. The data from the wristband was transmitted to a server computer via Bluetooth. To assess participants' level of sickness, they were prompted with an automated audio cue asking for their score at one-minute intervals (FMS Scale). Participants verbally expressed their sickness level on a predefined scale, which they were trained on before the experiment.

4 IMPLEMENTATION

Figure 4 provides a comprehensive overview of the details of customized SmartSimVR and showcases the hardware and software components. This software operates as a distributed application, utilizing a TCP/IP socket for efficient communication and data exchange between the client and server components. For a more comprehensive understanding of the system's modules and the flow of data within, it is recommended to refer to the detailed documentation provided in [7]. This resource delves into the intricacies of our system, elucidating the functionalities and interdependencies of its various components.

- `Client Side`: The client side consists of two important components: the VR Application and the E4 Streaming Server. The VR Application provides an immersive virtual reality experience and captures eye tracker data and head movement data. The E4 Streaming Server establishes a Bluetooth connection with the E4 wristband worn by participants to stream physiological data in real time.
- `Server Side`: The server side incorporates several components: Automatic Speech Recognition (ASR) for voice-based interactions, a DataTCPClient for receiving eye tracker and head movement data, an E4TCPClient for receiving data from the E4 Streaming Server, a PreProcessing Module that synchronizes data from different sensors, an AIModule with a binary classifier for classification based on the FMS scale, and a Adaptation Module that halts validation when sickness is detected. Adaptive variables such as accelerations are used to mitigate cybersickness.

5 EXPERIMENTAL SETUP

To evaluate the real-time performance of our distributed application, an experimental setup was established with specific hardware and software configurations. The client side, developed using Unity3D, was deployed on a high-performance workstation running Windows 10 Pro 64bit version 22h2. The workstation was equipped with an 11th Gen Intel Core i7-11800H processor, 32GB of RAM, and an NVIDIA GeForce RTX 3080 Laptop GPU graphics card. These hardware specifications provided sufficient computing power and graphics capabilities to render the immersive driving simulator environment in real-time.

On the server side, Python 3.8 was utilized as the backend. The server ran on a dedicated machine powered by an Intel Core i7-7700HQ processor, 16GB of RAM, and a high-speed solid-state drive. The server machine was connected to the client workstation via a local area network (LAN) with 543 MB Ethernet connections to ensure fast and reliable communication.

6 RESULT AND DISCUSSION

The performance evaluation of our system yielded valuable insights into its efficiency. As illustrated in Table 1, we conducted an extensive analysis of various performance metrics on both the client and server sides to assess the application's overall performance. The metrics measured during the 10 minutes of AI training included Memory Usage (MB), CPU Usage, GPU Usage, CPU temperature, Frames Per Second (FPS) for the client application, as well as the amount of Data Sent (MB) and Data Received (MB). The reported results in the table represent the average values obtained from the 8 cores in the server system and 16 cores in the client system.

Additionally, we conducted an in-depth investigation into the response times of various server-side artifacts involved in the AI training cycle, spanning from the moment of voice reception to the completion of training, throughout the 10-minute AI training phase. Figure 5 exclusively showcases the response time of the most time-consuming module, namely the ASR module, alongside the total AI training response time. This analysis provides valuable insights into the application's efficiency and responsiveness across different scenarios.

The memory and CPU usage analysis reveals interesting patterns on both the server and client sides. On the server side, the ASR.py module has low memory usage (28.58 MB) and minimal CPU utilization (0.05%). DataTCPClient.py and E4TCPClient.py show moderate memory usage (80.35 MB and 34.40 MB, respectively) with low CPU usage (0.22% and 12.05%, respectively). However, the AIModule.py stands out with higher memory usage (226.15 MB) and CPU usage (12%). On the client side, the IntelligentDS.exe application exhibits higher memory usage (280.83 MB) and CPU utilization (18.99%). It

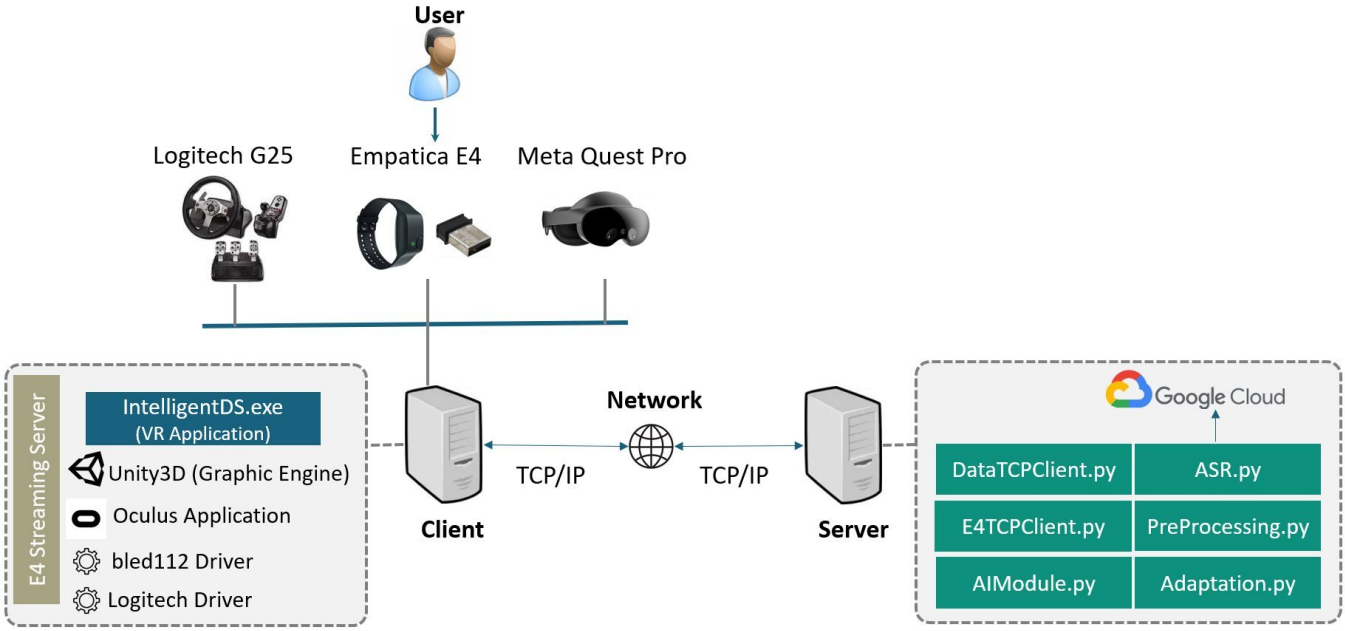

Figure 4: Hardware and software components of the implemented distributed system architecture.

Table 1: Performance Analysis

	Artifact	Memory Usage (MB)	CPU Usage	GPU Usage	FPS	Data Sent (MB)	Data Received (MB)	CPU Temperature
Server	ASR.py	28.58	0.05%	0	-	884.42	3073.93	85.8°C
	DataTCPClient.py	80.35	0.22%	0	-	891.11	3108.16	
	E4TCPClient.py	34.40	12.05%	0	-	891.14	3107.16	
	AIModule.py	226.15	12%	0	-	888.76	3094.20	
Client	IntelligentDS.exe	280.83	18.99%	40.57%	78	4925.22	6009.92	64.41°C

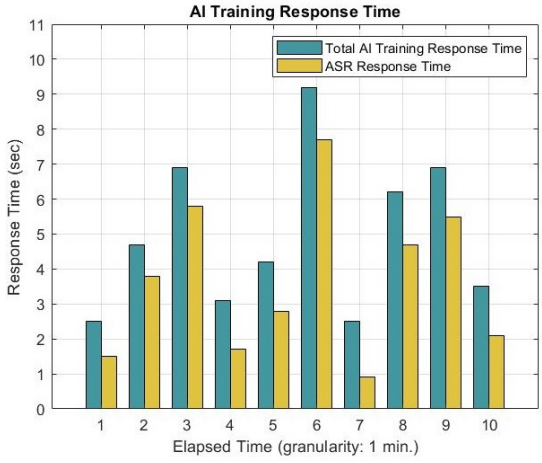

Figure 5: Server-side Response Time during 10 Minutes of AI Training: Total Response Time and ASR Response Time

also utilizes the GPU significantly, with a reported usage of 40.57%. These findings suggest that the server-side modules have efficient memory and CPU usage, while the client-side application requires relatively higher resources, including GPU usage, which is expected as the VR application runs on the client system.

The temperature monitoring results provide insights into the thermal performance of the system. The server system operates at a relatively high temperature of approximately 85.8°C, indicating potential thermal stress. In contrast, the client system maintains a lower CPU temperature of 64.41°C. It is essential to monitor and manage the temperature of the server system to ensure system stability and prevent thermal throttling or performance degradation.

The analysis of network performance, including data sent and data received shows consistent values across the server-side artifacts. The ASR.py, DataTCPClient.py, E4TCPClient.py, and AIModule.py demonstrate similar levels of data transfer with minor variations in the range of a few megabytes. On the client side, the IntelligentDS.exe application exhibits higher amounts of data sent (4925.22 MB) and data received (6009.92 MB). These findings suggest

that the client-side application requires more extensive communication with the server-side modules, likely due to AI training and adaptation processes.

The analysis of response time reveals that the ASR.py module is the most time-consuming aspect (Figure 5). The response time of ASR.py is heavily influenced by the Internet speed as it relies on the Google ASR service. The process involves uploading voice data to Google Cloud Storage, sending requests to the Google ASR service endpoint, receiving and processing responses, error handling, repeating the cycle in case of failures, and performing post-processing.

In general, the complete cycle of AI training, which includes voice reception, data collection from eye and head trackers, physiological data acquisition through the Empatica E4 wristband, dataset creation, pre-processing, and AI training, has a response time ranging from 2.5 seconds to 9.2 seconds, which is within the acceptable range of less than 1 minute in our system.

The findings emphasize the importance of optimizing the response time of the ASR.py module to improve the overall efficiency of the AI training cycle. Improving the Internet speed can potentially reduce the response time associated with ASR.py. Additionally, optimizing the data handling and processing steps within the module can lead to more efficient execution and further reduction in response time.

7 CONCLUSION AND FUTURE WORK

In this groundbreaking research initiative, we have developed an intelligent architecture for immersive virtual environments. The architecture addresses real-time AI training and adapts the virtual environment based on user state and external parameters. Through a case study focused on cybersickness detection and mitigation, we successfully trained an AI model in real-time and personalized it for individual users in a driving simulator application.

The distributed architecture offers benefits such as efficient data exchange, real-time adaptation, and optimized resource utilization. Our analysis of system efficiency and performance characteristics, including memory, CPU, and GPU usage, temperature monitoring, FPS, network performance, and response time provides valuable insights.

Future work involves optimizing the ASR.py module for improved efficiency and timely AI training. Additionally, we plan to analyze the scalability of our system. Furthermore, the proposed architecture holds potential for enhancing various aspects of virtual reality experiences beyond cybersickness mitigation. Future research can explore applications in adaptive training simulations, personalized gaming experiences, and real-time user feedback systems.

ACKNOWLEDGMENTS

This work was supported by a grant from Institute for Information Management in Engineering (IMI) and the French-German University CDFA 03-19.

REFERENCES

[1] Dayi Bian, Joshua Wade, Amy Swanson, Amy Weitlauf, Zachary Warren, and Nilanjan Sarkar. 2019. Design of a physiology-based adaptive virtual reality driving platform for individuals with ASD. *ACM Transactions on Accessible Computing (TACCESS)* 12, 1 (2019), 1–24.

[2] Albert Bifet, Ricard Gavalda, Geoffrey Holmes, and Bernhard Pfahringer. 2023. *Machine learning for data streams: with practical examples in MOA*. MIT press.

[3] Jean-Rémy Chardonnet, Mohammad Ali Mirzaei, and Frédéric Mérienne. 2017. 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* 33, 10 (2017), 771–785.

[4] Natalia Dużmańska, Paweł Strojny, and Agnieszka Strojny. 2018. Can simulator sickness be avoided? A review on temporal aspects of simulator sickness. *Frontiers in psychology* 9 (2018), 2132.

[5] Lawrence Frank, Robert S Kennedy, Robert S Kellogg, Michael E McCauley, and ESSEX CORP ORLANDO FL. 1983. 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*. 25–28.

[6] Augusto Garcia-Agundez, Christian Reuter, Hagen Becker, Robert Konrad, Polona Caserman, André Miede, and Stefan Göbel. 2019. Development of a classifier to determine factors causing cybersickness in virtual reality environments. *Games for health journal* 8, 6 (2019), 439–444.

[7] Azadeh Hadadi, Jean-Rémy Chardonnet, Christophe Guillet, and Jivka Ovtcharova. 2024. Intelligent Virtual Platform for Real-time Cybersickness Detection and Adaptation. In *2024 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR)*. IEEE, 231–235.

[8] Azadeh Hadadi, Christophe Guillet, Jean-Rémy Chardonnet, Mikhail Langovoy, Yuyang Wang, and Jivka Ovtcharova. 2022. Prediction of cybersickness in virtual environments using topological data analysis and machine learning. *Frontiers in Virtual Reality* 3 (2022), 973236.

[9] Rifatul Islam, Samuel Ang, and John Quarles. 2021. 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, 148–155.

[10] Robert S Kennedy, Norman E Lane, Kevin S Berbaum, and Michael G Lilienthal. 1993. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The international journal of aviation psychology* 3, 3 (1993), 203–220.

[11] Behrang Keshavarz and Heiko Hecht. 2011. Validating an efficient method to quantify motion sickness. *Human factors* 53, 4 (2011), 415–426.

[12] Hyun K Kim, Jaehyun Park, Yeongcheol Choi, and Mungyeong Choe. 2018. Virtual reality sickness questionnaire (VRSQ): Motion sickness measurement index in a virtual reality environment. *Applied ergonomics* 69 (2018), 66–73.

[13] Jinwoo Kim, Woojae Kim, Heeseok Oh, Seongmin Lee, and Sanghoon Lee. 2019. A deep cybersickness predictor based on brain signal analysis for virtual reality contents. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10580–10589.

[14] Haoyang Mao, Zhenyu Liu, and Chan Qiu. 2021. Adaptive disassembly sequence planning for VR maintenance training via deep reinforcement learning. *The International Journal of Advanced Manufacturing Technology* (2021), 1–10.

[15] Mohammad Ali Mirzaei. 2014. *Influence of interaction techniques on vims in virtual environments: estimation et prédiction*. Ph. D. Dissertation. Ecole nationale supérieure d'arts et métiers-ENSAM.

[16] M Ali Mirzaei, Sugeng Prianto, Jean-Rémy Chardonnet, Christian Pere, and Frédéric Mérienne. 2013. New motherwavelet for pattern detection in IR image. In *2013 Visual Communications and Image Processing (VCIP)*. IEEE, 1–6.

[17] Yunfang Niu, Danli Wang, Ziwei Wang, Fan Sun, Kang Yue, and Nan Zheng. 2019. User experience evaluation in virtual reality based on subjective feelings and physiological signals. *Journal of Imaging Science and Technology* 63, 6 (2019), 60413–1.

[18] Jérémy Plouzeau, Jean-Rémy Chardonnet, and Frédéric Merienne. 2018. 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, 661–662.

[19] Konstantinos Tsiakas, Manfred Huber, and Fillia Makedon. 2015. A multimodal adaptive session manager for physical rehabilitation exercising. In *proceedings of the 8th ACM international conference on pervasive technologies related to assistive environments*. 1–8.