Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

Expanding immersion in virtual reality

Bachelor Thesis

2024
Tomáš Búcsi

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Expanding immersion in virtual reality

Bachelor Thesis

Study Programme:	Applied Informatics
Field of Study:	Computer Science
Department:	Department of Applied Informatics
Supervisor:	prof. Ing. Igor Farkaš, Dr.
Consultant:	Mgr. Iveta Bečková

Bratislava, 2024
Tomáš Búcsi

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:
Študijný program:

Študijný odbor:
Typ záverečnej práce:
Jazyk záverečnej práce:
Sekundárny jazyk:

Názov:

Anotácia:

Vedúci:
Katedra:
Vedúci katedry:

Dátum zadania:

Dátum schválenia:

garant študijného programu

.....

študent

.....

vedúci práce

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:
Study programme:

Field of Study:
Type of Thesis:
Language of Thesis:
Secondary language:

Title:

Annotation:

Supervisor:
Department:
Head of department:

Assigned:

Approved:

Guarantor of Study Programme

.....
Student

.....
Supervisor

Contents

List of Figures

Chapter 1

Background

1.1 Introduction

Virtual reality (VR) has long existed as a visionary concept confined to the realms of science fiction literature, evoking imaginations of a distant future. However, in recent years, this concept has transformed into a tangible technology that has revolutionized the way we perceive and interact with computer-generated environments. Its wide availability in general retail stores and dropping price over the years has started to make virtual reality more accessible to consumers. These are just some factors that started to create a greater interest in the technology itself and the possible applications it may have in different areas of life such as leisure, medicine, research, education and others.

For virtual reality to be useful in the future it needs to become advanced enough for humans to not be able to differentiate between traditional reality and the man-made world virtual reality is trying to present. To achieve such a feat, it is essential for us to comprehend the nature of what we seek to replicate within the virtual setting. Common physical reality refers to the tangible interactions confined within the spatial, temporal, and material constraints of the physical world. On the other hand, virtual reality involves the creation of artificial simulations, typically recreating real-life environments, to enhance the perception of an imagined reality or situation (Tham et al., 2018). To blur the barrier of reality we need to increase the users presence or immersion, ideally both. While both these terms have been somewhat convoluted and at times, even used interchangeably (Wilkinson et al., 2021).

1.1.1 Presence vs Immersion

As mentioned, these terms are often used synonymously even though in the context of VR they hold distinct meanings. Immersion was defined as follows by Kim et al. (2017):

> "Immersion, leveraging the user's five senses, offers a remarkable ability to transport individuals into a virtual realm where they can perceive their surroundings, interact with others, and engage in activities as if they were genuine experiences. It encompasses the depth of engagement a user feels, evoking a range of emotions within the virtual space and manifesting them in relation to the simulated environment. In contrast, the concept of presence revolves around the phenomenon where users genuinely believe and feel as though they are physically present in the computer-generated world presented by the display technology. Consequently, immersion serves as the term employed to describe the technology that enables the emergence of presence, providing users with an immersive and captivating virtual experience."

Though the increase of immersion is desirable, miss-handling or incorrect implementation may lead to unfortunate side effects such as nausea, headaches and vertigo, also called VR sickness, directly caused by the discrepancy between visual information and the vestibular information received from the VR simulating movement (Tanaka and Takagi, 2004). As we now have an understanding of what immersion is, the next section will contain how it can be improved.

1.2 Improving Immersion

In a case study participants reported, that a VR environment which provided them a rich sensory stimulation, such as visual, auditory and tactile feedback increased their sense of presence (Tham et al., 2018). Their findings also suggest that higher VR fidelity contributes to a greater sense of embodiment among participants. In other words, when the visual and auditory cues in the virtual environment are of higher quality and realism, individuals tend to feel more fully present and engaged within that virtual world. This enhanced fidelity strengthens the connection between the user's physical self and their virtual representation, resulting in a more immersive and convincing VR experience. To them, reality in the context of VR is defined by how effectively the VR hardware convinces their senses in any given scenario (Tham et al., 2018).

As immersion is inherently tied to technology that directly enhances presence, as a result we can enhance immersion by improving the realism of the VR environment through advancements in graphics and sound quality. However, immersion can also be further enhanced by augmenting the quantity and quality of sensory feedback that users receive during their VR experience. By providing users with more comprehensive and realistic sensory inputs, such as tactile feedback, haptic sensations, and spatial

audio cues, we can deepen the level of immersion (Hecht et al., 2008).

Most modern commercially available VR systems come with a Head-Mounted Display (HDM), controllers, some variation of tracking sensors and may provide integrated or separate audio systems. This means that most devices mainly focus on the visual and audio aspects of VR, neglecting other senses such as touch. That's why we will first focus on audio and visual immersion.

1.3 Visual Immersion

One of the most sought after aspects of VR is its ability to be able to see, what was once on a screen, before us, in 3D, therefore visual immersion plays a critical role in VR systems. There are multiple aspects with which we can improve visual immersion, since we need to trick ourselves to believe we really are a part of the computer generated world. The most important aspects are: realistic rendering, improved graphics and an appropriate field of view (FoV) (Bowman and McMahan, 2007).

1.3.1 Rendering

To ensure a realistic depiction of the environment, it is important, that the user continually sees their environment without interruption. For a seamless experience a high enough frame rate is needed to ensure adequate immersion. For this to be viable, a rendering engine with sufficient power is essential. This ensures that with the image changes swiftly in response to the users head movement, maintaining the illusion of real-time interaction (Regan and Pose, 1994).

VR systems may use stereoscopic rendering, to accurately simulate how the human eyes see. The simple reason behind this being, that by stereoscopically rendering each eye, we receive separately rendered images, thus creating the illusion of depth and 3D effect (Forlim et al., 2019).

1.3.2 Graphics

In contrast to older graphics cards that render 3D images using polygonal structures with shading, modern graphics cards like the NVIDIA RTX line, employ advanced techniques to simulate the behavior of light. These cards trace the path that light would naturally take from the human eye through the virtual environment. This capability enables them to generate realistic shadows and refractions, enhancing the visual fidelity of the rendered scenes (Wilkinson et al., 2021).

1.3.3 Field of View

While FoV might not seem like an important aspect of visual immersion, its incorrect implementation may lead to the users experiencing VR sickness. This of course leads to lessened immersion and an overall worse experience with VR. One of the ways found to mitigate VR sickness without influencing the persons experiences inside of VR is dynamically changing the FoV of the user. When the user is in a stationary position the users have regular, unrestricted FoV. The change occurs when the users are in motion because their FoV contracts drastically. This blocks their peripheral visual motion preventing a disconnect between what they see, and what they are doing (Teixeira and Palmisano, 2021).

1.4 Auditory Immersion

Sound is paramount in crafting an environment that exudes vibrancy and authenticity, ultimately contributing to a heightened sense of realism and immersion. There are multiple factors that contribute to the creation of an immersive environment such as spatial audio, binaural rendering and realistic sound effects.

1.4.1 Spatial Audio and Binaural Rendering

As the name suggests, spatial audio gives the user the remarkable sensation that sound emanates from distinct and specific locations within the 3D space, which helps create a dynamic and captivating environment that envelops the end user in a truly immersive experience (Schissler et al., 2016).

An important part of spatial audio is the shaping of the head-related transfer functions (HRTF), which describes how a listener and its geometry affects the sound that he is exposed to. It can be described as a filter which maps incoming sounds to the head of the user, and then delivers the necessary sounds to the left or right ear (Schissler et al., 2016). The audio also includes cues for the human ears to experience certain sounds from different directions, distance, which is created through binaural rendering (Zaunschirm et al., 2020).

1.4.2 Realistic Sound

The utilization of 3D sound technology, based on HRTF, faces a limitation in accurately reproducing realistic sound. That is due to its inability to reflect the physical impact on object materials and the surrounding environment within the virtual space. This limitation can be overcome through sound rendering. This is a technique which involves simulating the physical attributes of sound that occur between the sound source and

the listener within the virtual space, thereby addressing this constraint (Hong et al., 2017).

1.5 Haptic feedback

While visual and auditory immersion undeniably form significant pillars of the VR experience, achieving true immersion necessitates the development of precise and lifelike methods for interacting with objects within the computer-generated environment (Kim et al., 2017).

Haptic feedback can be categorized into passive haptic feedback and active haptic feedback. Passive haptic feedback is generally a physical object that is simultaneously generated in the VR, which simulates the physical properties of the object, such as it's weight, texture and shape (Viciana-Abad et al., 2010). Active haptic feedback on the other hand, is haptic feedback that can be actively programmed and felt through a haptic display such as vibrations and heat (Nilsson et al., 2018).

The majority, if not all, of commercially available VR systems currently offer limited or no haptic feedback. This limitation arises from the inherent challenge of accommodating the diverse physical attributes of individual users while maintaining a balance between wearability, lightweight design, and the inclusion of adequate hardware to deliver a convincingly realistic sense of touch (Perret and Vander Poorten, 2018).

1.6 Walking in Virtual Reality

Among the routine tasks we engage in on a daily basis, walking stands out as one of the most formidable challenges to authentically simulate in virtual reality, aiming to capture the essence of natural movement with a genuine sense of reality.

The primary hurdle in achieving a natural walking experience lies in the challenge of crafting expansive and immersive virtual worlds within the confines of our homes. The task at hand involves ingeniously navigating the limitations of physical space while delivering captivating environments that invite exploration and evoke a genuine sense of presence (Nilsson et al., 2018). Another concern may be, how to accurately simulate all the outside stimuli our bodies receive with each step, such as sight, sound and tactile feedback (Nilsson et al., 2018).

Virtual travel can be classified based on various factors, providing us with different categories to explore. Firstly, we can differentiate between mobile and stationary travel techniques, depending on whether the user physically moves or remains in a fixed position throughout the experience. In the realm of mundane travel techniques, users are bound by the limitations of physics, biology, and current technological constraints,

while in the realm of magic travel techniques, such as using portals, teleporting, these constraints are transcended. Additionally, we can discern between vehicular and body-centric travel techniques. Body-centric travel techniques simulate movement by incorporating human motions such as walking or running, whereas vehicular techniques allow users to control a simulated vehicle without the need for physical movement (Nilsson et al., 2018).

Although simulating natural walking in VR poses challenges, there exists a multitude of solutions that can be classified into three distinct walking techniques.

1.6.1 Repositioning Systems

Repositioning systems make sure, that any movement done by the user is nullified, so that while they moved in the virtual world, they are still in the same position in the reality. These systems can be categorized into two types: active repositioning and passive repositioning. Active repositioning involves intricate mechanisms, such as omnidirectional treadmills or motorized floor tiles, which actively counteract the user's movements. On the other hand, passive repositioning offers a more cost-effective approach by utilizing friction-free platforms that effectively negate the forward movement of the user without the need for complex mechanisms (Nilsson et al., 2018).

1.6.2 Proxy Gestures

Proxy gestures rely on a specific movement which replace the act of taking a step. While you can categorize them depending, on what body parts movement replaces the actual step, most such proxy movements are closely related or a part of the movement, that taking a step is. These movements are called walking-in-place techniques (Nilsson et al., 2018).

1.6.3 Redirected Walking

This specific walking technique entails the user physically walking while actively manipulating the virtual environment or adjusting their perspective to navigate in the desired direction, effectively directing their movement exclusively within the virtual realm. This technique relies on the manipulation of perspective, achieved by altering the user's visual or auditory point of view. By magnifying the distance traveled within the virtual world compared to the corresponding movement in the real world, this approach creates an illusion of extended travel and effectively enhances the user's virtual experience in a limited space (Nilsson et al., 2018).

1.7 Relevant software

1.7.1 Unity

Unity is a real-time development platform, which can be used to create 2D or 3D projects, ranging from simple games to immersive VR experiences. Its popularity is thanks to its widespread availability on all major platforms, including Linux, Apple and Windows. Renowned for its ease of use, flexibility and its strong feature set, Unity caters not only to beginner, but also to experienced developers creating high-performance products (Hussain et al., 2020).

Unity, equipped with its rendering and physics engine and a graphical user interface, the Unity Editor, allows for rapid prototyping in development. Within each Unity project, various components, termed Assets, form the foundation. The main building block is an Asset called a Scene, which is the primary element of the project. It contains the hierarchy of GameObjects that constitute the scene's contents. All projects come with preexisting objects, such as Cameras, Meshes and others, but additional Gameobjects and behaviour can be easily integrated through scripts written in C# (Juliani et al., 2018).

1.7.2 Blender

Blender is a versatile and accessible 3D creation software, freely available and open-source. It provides support the whole creation process, tasks such as modeling modeling, rigging, animation, simulation, rendering, composition and motion tracking. One of its major features is its flexibility, allowing any user to use its API to create custom tools and functionalities for their specific need. These customized additions are often added to the software at a later release. Blender is also cross-platform compatible, making sure users on all major platforms may use its features (ble, 2024).

1.8 Reinforcement Learning

Reinforcement learning (RL) is a type of machine learning, where a task is gradually learned through a process of trial and error, while also trying to uncover such a sequence of actions, which yield the highest amount of cumulative reward. Unlike supervised learning, which relies on external labeled data, and unsupervised learning, focused on uncovering latent data structures, RL places its agent within an unknown environment, permitting it to explore various action choices autonomously (Sutton and Barto, 2018).

The agent initially perceives the state of his surroundings and can subsequently perform a number of actions which change it. This iterative process continues until the agents successfully preforms its given task, or reaches a state, where no additional

actions can be taken. Throughout this process, each action is given a reward value, which is derived from the newly created state. It informs the agent, how impactful the given action was, to the completion of his end goal. This reward is calculated by the reward function, who through these calculations defines what the goal is for the agent, and also decides what states and or actions are objectively good or bad. Its objectiveness is protected, because the agent has no way of influencing or altering the outcome of the function (Sutton and Barto, 2018).

While the reward function shows the immediate benefit of an action the value function determines the positives of a decision in the long run. It promotes choosing an action with lower immediate rewards if it promises greater profitability in the future, over an action with higher immediate rewards but lower long-term yield. While the reward function lays the groundwork, the value function has a higher priority in the decision making, as it prioritizes maximizing overall value rather than immediate rewards (Sutton and Barto, 2018).

Subsequent actions are always chosen based on the policy the agent has. This function links actions to the possible result states that the given action will lead to. The policy chooses the next best action either through deterministic or stochastic means, always trying to find the sequence of actions that promises the highest cumulative reward over time. Basically, it defines how an agent behaves in any situation. A lot of algorithms keep refining their policy during the learning process. This practice is predominantly used to encourage exploration. That is because the agent is incentivised to favor routes with the highest established rewards amongst the ones he knows, which may block him from discovering superior alternatives. That is why most algorithms must always incorporate some trade-off between using what the agent knows, and options not yet explored. This prevents the agent from becoming trapped in a local maxima (Sutton and Barto, 2018).

1.9 Machine Learning Agents Toolkit

The Unity Machine Learning Agents Toolkit (Ml-Agents Toolkit) stands as an open-source project that enables developers and researchers to construct learning environments tailored for training intelligent agents. It also acts as a central platform where new algorithms and other advances in the AI field can be evaluated and made available for the wider audience of developers. These advancements can be used for research purposes or to be integrated into new game titles, providing developers with access to the latest innovations to elevate their projects (Juliani et al., 2018).

The Ml-Agents Toolkit is made up of four primary high level components, the Learning Environment, the Python Low-Level API, the External Communicator, and

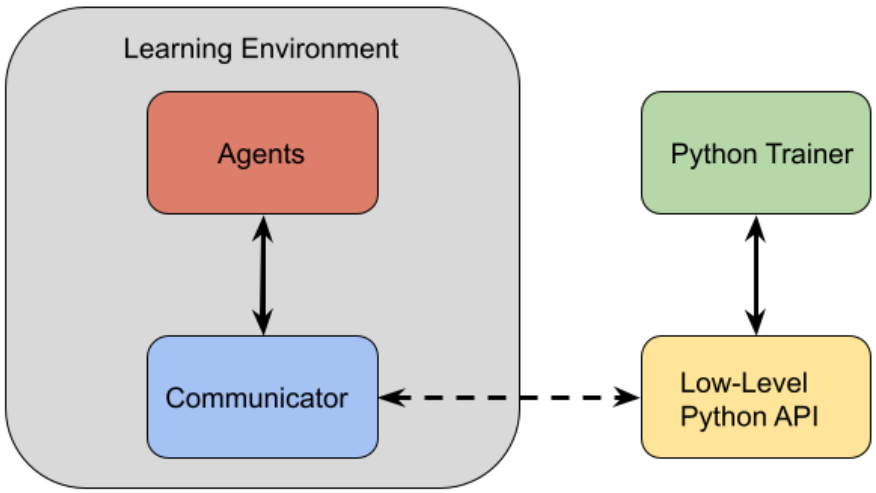

Figure 1.1: Abstract representation of the key components of the ML-Agents Toolkit (Metz, 2020; Juliani et al., 2018)

the Python Trainers.

The Learning Environment describes the Unity scene as well as all the objects inside the scene. This is where the agent observes its surroundings, performs actions, and is taught through reward signals. It's crucial to design the learning environment in a manner that is beneficial to learning a given task. If a task is very limited a more general learning environment may be enough. But for more complex tasks a more customized environment is needed. For the creation of custom environments, the Toolkit offers a specialized ML-Agents Unity SDK capable of converting any Unity scene into a learning environment (Juliani et al., 2018).

The Python Low-Level API serves as a Python interface, providing low-level functionality for modifying and interacting with the Learning Environment. Unlike the environment itself, this interface operates external, meaning it can communicate with the environment only through the External Communicator. Its main role is to communicate and manage the Academy class, although it can also be used to control Unity as a simulation engine for custom algorithms (Juliani et al., 2018).

The External Communicator serves as the bridge between the Learning Environment and the Python Low-Level API, making essential communication between these two vital parts possible. It is contained inside of the Learning Environment making it a part of it (Juliani et al., 2018).

The Python Trainers house all the algorithms used for the training of agents. As the name suggest all algorithms are implemented in Python, and this separate package solely communicates with the Low-Level API (Juliani et al., 2018).

1.9.1 ML-Agents SDK

As previously stated, any Unity scene may be converted into a learning environment. For it to happen, we need to add three scripts to an existing project:

- Agent script, a Unity script which defines what GameObject is going to be the agent that is going to be trained (Metz, 2020).
- Behaviour Parameters script, which acts as the control center for the agent, which communicates the observations and the rewards between the Agent and the Training API, and also decides which actions to take (Metz, 2020).
- Decision Requester, which is responsible for the communication between the Agent and the Academy (Metz, 2020).

In an environment created through the ML-Agent SDK, the fundamental components are the Sensors, Agents and the Academy. The Sensors provide observations of the environment to the Agents. These Sensors may relay different kinds of information, based on the type of sensor, for instance they relay rendered images or length vectors. Each Agent is a designated GameObject, with each containing a policy labeled with a *behaviour name*. Multiple Agents may share the same policy with identical *behaviour name* values. All of the Agents will act upon the same policy and all of them will contribute data to the refining of the policy. Any number of policies may be added to a scene, making it easy to create multi-agent environments, where each agent may act upon a distinct policy (Juliani et al., 2018).

The reward function of an agent may be modified dynamically during the simulation at any point by the Unity scripting system. Similarly, the state of the agent, or the entire environment can transition into a state of "done". This change occurs either when the agents have reached their objective, defined by a call from a Unity script, or when they have taken the predefined maximum number of steps. This state signals the end of an episode of the simulation (Juliani et al., 2018).

The Academy refers to a singleton class that oversees the entire simulation, maintaining control over its progression. It tracks the amount of steps taken within the simulation, and is responsible for the management of the agent. The Academy has the capability to shape the learning environment of the agent by introducing new objects or altering physical properties when the agent reaches a level of proficiency. This is done for the goal of further development of the Agent. Also it can be used to reuse the same environment for both training and testing purposes with minimal modifications (Juliani et al., 2018).

The Agents can undergo training using state-of-the-art algorithms drawn from a diverse array of machine learning methods, including reinforcement learning, imitation

learning, neuroevolution and more. However, for the scope of this thesis, we will only be focusing on the reinforcement learning methods. As such we will discuss the two algorithms this toolkit provides, which are Proximal Policy Optimization and Soft Actor Critic (Juliani et al., 2018).

1.9.2 Proximal Policy Optimization

The Proximal Policy Optimization (PPO) is a RL algorithm, which focuses on stabilizing the training of the policy. It achieves this by limiting the magnitude the policy may change at the end of each epoch, preventing excessively large updates that could disrupt the training stability (Simonini, 2022; Schulman et al., 2017).

A more stable training of the policy is preferable, because smaller policy updates tend to converge more reliably towards the optimal solution. Another reason to employ this algorithm is, mitigating the risk of large policy change potentially making the policy ineffective, prolonging training time, or failing to recover altogether (Simonini, 2022; Schulman et al., 2017).

To ensure the policy hasn't undergone excessive changes, we need to calculate the ratio of how much the current policy has changed relative to the alteration made to the former one. This is accomplished by determining the ratio between the current and former policy. We adjust this ratio so that it falls within the range of $[1-\epsilon, 1+\epsilon]$, which prevents excessive alteration of the current policy when compared to the previous one (Simonini, 2022; Schulman et al., 2017).

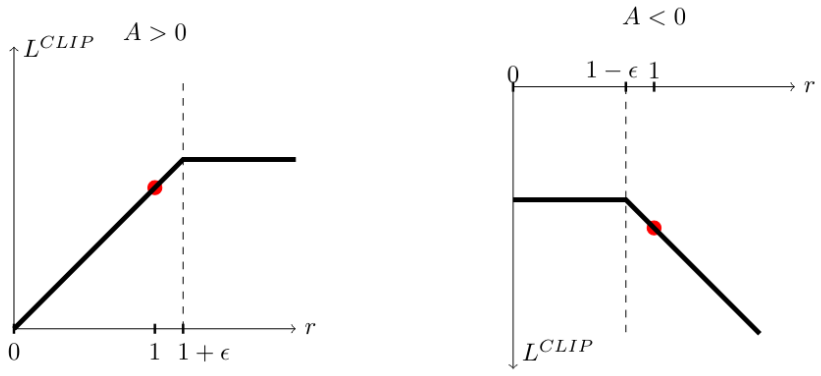

Figure 1.2: Two functions depicting one time step of the function L^{CLIP}, when the ratio function r = 1. The left one shows when the advantage is positive, while the right shows when the advantage is negative (Schulman et al., 2017).

The PPO algorithm is carried out through the function:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\Big[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big] \tag{1.1}$$

Where $r_t(\theta)$ is the ratio function:

$$r_\theta = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta\text{old}}(a_t|s_t)} \tag{1.2}$$

In 1.2 $a_t|s_t$ denotes the probability of taking action a_t in the s_t state. If the value of $r_t(\theta) > 1$, then we know that this action in the current state is more likely to happen in the new policy than the old one. And if $0 < r_t(\theta) < 1$, then it is more likely for this action/state pair to happen in the older policy rather than the new one. This is the easiest way to asses the difference between the newer and older policy.

The main term in 1.1 is the minimum of two probability ratios, the clipped and unclipped one. That is how we make sure the policy only changes inside the range of $[1 - \epsilon, 1 + \epsilon]$, and not more drastically (Simonini, 2022; Schulman et al., 2017).

1.9.3 Soft Actor Critic

An Actor Critic is an RL architecture characterized by the fact that it simultaneously trains both a policy (actor) and an estimated value function (critic) at the same time. In this architecture, the actor chooses a action to take based on the policy. This action is then given feedback by the critic, regarding its long-term quality. Based on this evaluation the policy is updated by the actor (Ezzeddine et al., 2023).

One of the advantages of AC algorithms is their ability to leverage the strengths of both policy-based and value-based RL methods. This integration allows the algorithm to converge on an optimal solution more rapidly and learn more efficiently. Additionally, these algorithms have a higher proficiency in navigating complex and high dimensional action spaces, rendering them well-suited for complex input representation (Ezzeddine et al., 2023).

Despite their numerous advantages, these algorithms also present limitations. One such drawback is the need for a substantial amount of samples for the value function to create a more accurate estimate. Another negative is the need to regulate the entropy of the policies; which if not properly managed may lead the model to converge to a deterministic policy, which results in less exploration and potentially yielding sub-optimal result (Ezzeddine et al., 2023).

To resolves such issues a solution was introduced in the form of the Soft Actor Critic (SAC) architecture. SAC not only tries to maximize the expected reward value but also to maximize the entropy of the action taken. This is to ensure that there always remains a certain level of uncertainty regarding which action will be chosen. This ensures that there remains a balance between exploration and exploitation, making sure the policy doesn't remain sub-optimal (Ezzeddine et al., 2023).

The SAC algorithm is made of three different networks: a state value function, a soft Q-function and a policy function. The estimation provided by the state value and

the soft Q-function contributes to the convergence of the algorithm (Ezzeddine et al., 2023).

Bibliography

(2024). Blender foundation - blender.org. `https://www.blender.org`. Accessed: 6.5.2024.

Bowman, D. A. and McMahan, R. P. (2007). Virtual reality: how much immersion is enough? *Computer*, 40(7):36–43.

Ezzeddine, F., Ayoub, O., Andreoletti, D., and Giordano, S. (2023). SAC-FACT: Soft actor-critic reinforcement learning for counterfactual explanations. In *World Conference on Explainable Artificial Intelligence*, pages 195–216. Springer.

Forlim, C. G., Bittner, L., Mostajeran, F., Steinicke, F., Gallinat, J., and Kühn, S. (2019). Stereoscopic rendering via goggles elicits higher functional connectivity during virtual reality gaming. *Frontiers in Human Neuroscience*, 13:365.

Hecht, D., Reiner, M., and Karni, A. (2008). Enhancement of response times to bi-and tri-modal sensory stimuli during active movements. *Experimental Brain Research*, 185:655–665.

Hong, D., Lee, T.-H., Joo, Y., and Park, W.-C. (2017). Real-time sound propagation hardware accelerator for immersive virtual reality 3D audio. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 1–2.

Hussain, A., Shakeel, H., Hussain, F., Uddin, N., and Ghouri, T. L. (2020). Unity game development engine: A technical survey. *Univ. Sindh J. Inf. Commun. Technol*, 4(2):73–81.

Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., et al. (2018). Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*.

Kim, M., Jeon, C., and Kim, J. (2017). A study on immersion and presence of a portable hand haptic system for immersive virtual reality. *Sensors*, 17(5):1141.

Metz, L. A. E. P. (2020). *An evaluation of unity ML-Agents toolkit for learning boss strategies*. PhD thesis, Rekjavík University.

Nilsson, N. C., Serafin, S., Steinicke, F., and Nordahl, R. (2018). Natural walking in virtual reality: A review. *Computers in Entertainment (CIE)*, 16(2):1–22.

Perret, J. and Vander Poorten, E. (2018). Touching virtual reality: a review of haptic gloves. In *ACTUATOR 2018; 16th International Conference on New Actuators*, pages 1–5. VDE.

Regan, M. and Pose, R. (1994). Priority rendering with a virtual reality address recalculation pipeline. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 155–162.

Schissler, C., Nicholls, A., and Mehra, R. (2016). Efficient HRTF-based spatial audio for area and volumetric sources. *IEEE Transactions on Visualization and Computer Graphics*, 22(4):1356–1366.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Simonini, T. (2022). Hugging face. `https://huggingface.co/blog/deep-rl-ppo`. Accessed: 2.5.2024.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tanaka, N. and Takagi, H. (2004). Virtual reality environment design of managing both presence and virtual reality sickness. *Journal of Physiological Anthropology and Applied Human Science*, 23(6):313–317.

Teixeira, J. and Palmisano, S. (2021). Effects of dynamic field-of-view restriction on cybersickness and presence in HMD-based virtual reality. *Virtual Reality*, 25(2):433–445.

Tham, J., Duin, A. H., Gee, L., Ernst, N., Abdelqader, B., and McGrath, M. (2018). Understanding virtual reality: Presence, embodiment, and professional practice. *IEEE Transactions on Professional Communication*, 61(2):178–195.

Viciana-Abad, R., Lecuona, A. R., and Poyade, M. (2010). The influence of passive haptic feedback and difference interaction metaphors on presence and task performance. *Presence*, 19(3):197–212.

Wilkinson, M., Brantley, S., and Feng, J. (2021). A mini review of presence and immersion in virtual reality. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 65, pages 1099–1103. SAGE Publications Sage CA: Los Angeles, CA.

Zaunschirm, M., Frank, M., and Zotter, F. (2020). Binaural rendering with measured room responses: First-order ambisonic microphone vs. dummy head. *Applied Sciences*, 10(5):1631.