© 2023 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

To appear in the 2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshop (VRW) proceedings.

Ubiq-Genie: Leveraging External Frameworks for Enhanced Social VR Experiences

Nels Numan* Daniele Giunchi† Benjamin Congdon‡ Anthony Steed§

University College London

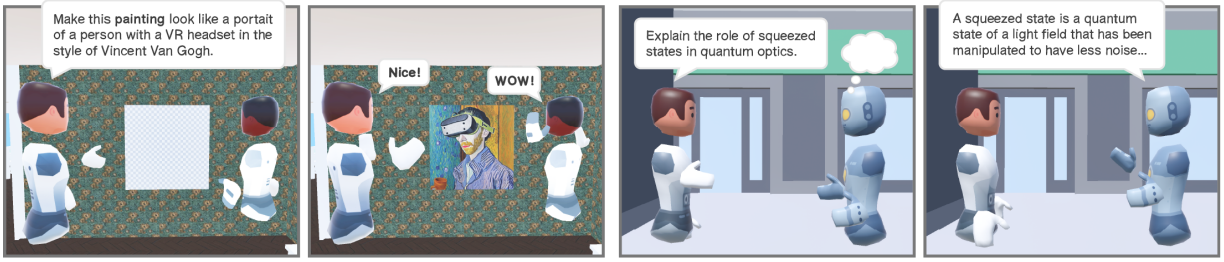

Figure 1: Overview of the two prototypes of collaborative applications presented in this paper that are based on Ubiq-Genie. Left: a voice-controlled texture generation method based on the diffusion-based image synthesis model Stable Diffusion 2.0 [21]. Right: a voice-controlled conversational agent based on the text synthesis tool ChatGPT [19].

Abstract

This paper describes the Ubiq-Genie framework for integrating external frameworks with the Ubiq social VR platform. The proposed architecture is modular, allowing for easy integration of services and providing mechanisms to offload computationally intensive processes to a server. To showcase the capabilities of the framework, we present two prototype applications: 1) a voice- and gesture-controlled texture generation method based on Stable Diffusion 2.0 and 2) an embodied conversational agent based on ChatGPT. This work aims to demonstrate the potential of integrating external frameworks into social VR for the creation of new types of collaborative experiences.

Keywords: social virtual reality, collaboration, open source, system architecture, generative artificial intelligence

Index Terms: Human-centered computing—Collaborative and social computing systems and tools;

1 Introduction

The proliferation of open-source projects in recent years has greatly impacted various fields, including machine learning and computer vision [6]. These projects have grown significantly due to advancements in technology and the availability of data. Many of these frameworks can serve a purpose within virtual reality (VR) experiences. For example, recent advances in machine learning might provide the ability to generate textual, auditory, or visual data within collaborative virtual environments (CVEs) for creative applications. Additionally, open-source visualisation techniques and machine learning algorithms might be utilised to analyse and interpret user behaviour, aiding in the evaluation of VR applications.

In this paper, we describe Ubiq-Genie, a system that enables the integration of external (open source) frameworks with the Ubiq social VR platform [4]. This platform is uniquely positioned for integration with other frameworks due to its open and flexible design on both the server and client sides. To demonstrate the potential use cases and applications enabled by the proposed system, we present two prototype applications built with generative artificial intelligence (AI) frameworks that rely on server-side processing through Ubiq-Genie (Figure 1). Specifically, (1) a voice-controlled texture generation method based on speech-to-text and the diffusion-based image synthesis model Stable Diffusion 2.0 [21]; and (2) a voice-controlled conversational agent based on speech-to-text, the text synthesis model ChatGPT [19], and text-to-speech. Our prototypes are primarily implemented with the Python programming language, which is currently one of the most common languages for open-source data science, machine learning, and computer vision projects. However, the techniques described in this paper are also applicable to frameworks written in most other programming languages.

In this paper, we introduce techniques to incorporate a wide range of services into CVEs built with the Ubiq platform. The proposed architecture for integrating external frameworks is designed to be modular, utilising centralised app controllers to communicate with one or more services which each are responsible for a specific function (e.g. image synthesis or speech-to-text). This design allows for easy integration of new services and the ability to modify or replace individual services without affecting the entire system. By providing a framework for the integration of external services and frameworks, we aim to pave the way for new and innovative VR experiences that can enhance collaboration and communication within CVEs. In addition, this work aims to inspire future work that explores the potential applications of these techniques for both users and researchers.

2 Related Work

Decoupling system components, with separate client and server sides, has been widely adopted in VR and augmented reality (AR) systems to improve scalability, flexibility, and resource optimisation. With the increasing application of computer vision and deep learning in human-centred systems, such architectures have become increasingly relevant. This is because they allow for the integration of large and complex system components, such as computationally intensive models, which can enable types of VR and AR experiences that were previously only possible for systems that are tethered to high-end desktops, or, in some cases, backpack computers [27]. To address

*e-mail: nels.numan@ucl.ac.uk
†e-mail: dgiunchi@ucl.ac.uk
‡e-mail: ben.congdon.11@ucl.ac.uk
§e-mail: asteed@ucl.ac.uk

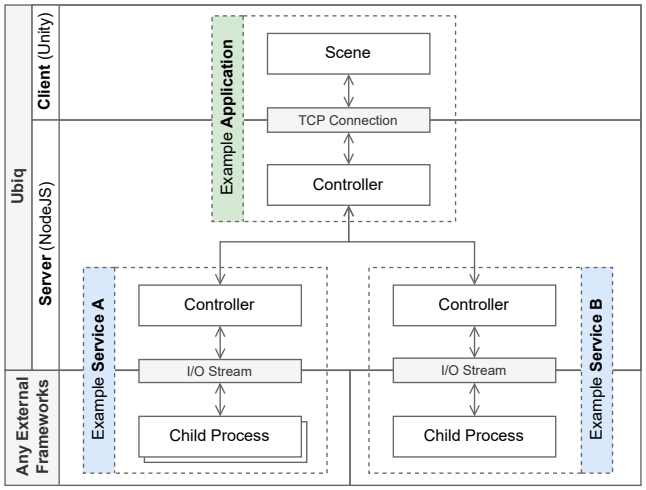

Figure 2: High-level architecture of the proposed system model, showing an application (incl. Unity scene), services, and their interactions. To demonstrate a typical implementation of an application, this diagram shows an application that communicates with two services.

this, a large body of work has explored the use of client-server architectures, which offload computation to either the cloud [8] or edge-based devices through a local network [25] (e.g. smartphones).

Client-server architectures have been widely used in various server-assisted applications, such as real-time object recognition [5,13], information extraction from scenes [2,11], rendering and encoding [28], and localisation [15]. For example, Chatzopoulos et al. [2] developed Hyperion, a wearable AR system that implements a real-time text extraction application with Google Glass. To alleviate the computational burden of the text extraction process, the proposed system partially offloads computation to a server.

Although the concept of offloading computation to servers is not a novel concept, currently there is no open-source framework that enables the development of server-assisted social VR applications. To address this gap, we developed a solution that extends the architecture of Ubiq [4], with the goal of integrating and simultaneously decoupling system components in a transparent manner. This framework allows for the easy integration of external services and frameworks by researchers and developers to enhance the functionality and capabilities of VR and AR systems and provide new opportunities for social VR research.

3 System Design

We designed a modular architecture to facilitate the integration of new services and the ability to update or replace individual services without affecting the entire system. At a high level, the architecture consists of three main components: the Unity scene, applications, and services (Figure 2). While each application should have its own scene and `ApplicationController`, services are modular and can be reused in different applications. Examples of services are speech-to-text and image synthesis (see Section 4 for all services).

The Unity scene serves as the interface for VR users and contains application-specific Unity components that communicate with a server-side `ApplicationController` through a TCP connection, using either Ubiq's `Networking` or `Logging` components. These client-side components are written in C# and ensure that outgoing and incoming data are processed and routed correctly.

The `ApplicationController` is a server-side component written in NodeJS that contains the necessary Ubiq `Networking` objects to communicate with the Unity scene and acts as the central coordination point for one or more different services. This component interacts with each service by communicating with `ServiceController` objects. Furthermore, the `ApplicationController` component implements any logic to pre-process service input or post-process service output for transmission to other services or the Unity scene.

The `ServiceController` is a server-side component written in NodeJS which is responsible for providing a specific function. This component orchestrates the data flow of one or more underlying child processes through the I/O stream of the server's operating system, which carry out the computation for the service. These child processes can be written in any programming language that can handle communication through the I/O stream (e.g. Java, Python, C++). As such, this allows for the integration of a wide range of open-source frameworks into the Ubiq platform.

The source code of Ubiq-Genie has been made available publicly[1] including documentation describing how services and applications can be implemented.

4 Services

We implemented several services to demonstrate the potential use cases and applications enabled by Ubiq-Genie. This includes a speech-to-text (STT), image synthesis, conversational text synthesis, file server, and text-to-speech (TTS) service, which each rely on child processes written in Python. As described in Section 3, service input and output are handled by the `ServiceController` component and are orchestrated by the `ApplicationController` component.

- **Speech-to-text (STT)** generates text based on an audio stream. This service is designed to transcribe user microphone audio, allowing for voice-controlled interactions within the CVE. A new child process is spawned for each peer to prevent data congestion when multiple users join the same room. This service is currently implemented using the Python client of Azure Speech Service[2]. However, alternative (open-source) solutions could be utilised such as Whisper [20] or DeepSpeech [14].

- **Text-to-speech (TTS)** generates audio based on text input. The resulting audio is streamed over the network to the Unity scene, which can be played by a companion script through a specified audio source. This service is also currently implemented using Azure Speech Service, but can similarly be replaced by alternative (open-source) solutions.

- **Image Synthesis** generates images based on text input, using the Hugging Face [10] implementation of the image synthesis model Stable Diffusion 2.0 [21]. This service is capable of generating images of a size of 768×768 pixels in a few seconds on a server with a high-end GPU (e.g. NVIDIA RTX 3080 Ti, as used in our prototypes). This service includes the option to generate seamless images, which is especially useful for the generation of tiled textures.

- **Text Synthesis** generates text that simulates a conversation based on text-based prompts. This service is based on ChatGPT, approached through a third-party Python library[3], as an official API endpoint is not available at this time.

- **File Server** makes any file directory accessible over the network using HTTP. In addition to serving files such as images, audio, or 3D models, this utility service can be used to serve HTML pages that can interact with Ubiq through its `Networking` components.

[1] `https://github.com/UCL-VR/ubiq-genie`
[2] `https://learn.microsoft.com/azure/cognitive-services/`
[3] `https://github.com/acheong08/ChatGPT`

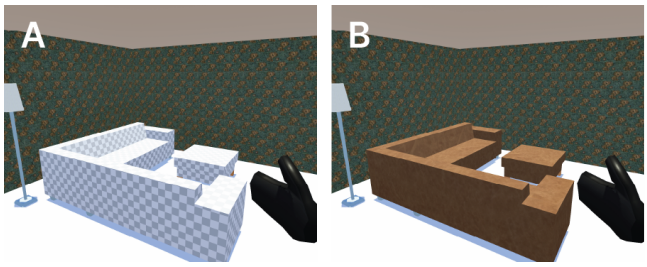

Figure 3: The texture generation prototype from a user's perspective. (A) After the user provides a voice prompt and the target object is recognised, it is assigned a chequerboard pattern; (B) Once the image synthesis process finishes, the resulting image is downloaded and applied to the target object.

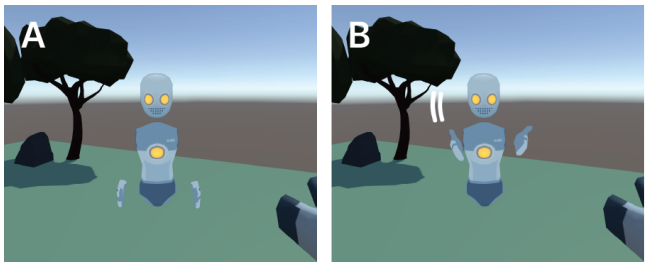

Figure 4: The conversational agent prototype from the user's perspective. (A) The agent's state while listening to the user; (B) The agent's state while responding to the user, including gesturing and a speech indicator.

5 Prototypes

We implemented two prototypes of collaborative applications to demonstrate the potential use cases enabled by Ubiq-Genie: a texture generation prototype and a conversational agent prototype. These prototypes showcase how the services described in Section 4 can be combined into a pipeline to create new and innovative VR experiences that enhance interactions within a CVE. A key aspect of these prototype applications is that they are designed to be used collaboratively, where multiple users can interact with the application at the same time.

5.1 Texture Generation

This prototype application allows users to generate textures within a CVE through voice prompts, optionally combined with ray-based selection to select the target. To demonstrate its capabilities, we applied this prototype to a simplified interior design task where objects can be textured by users. The use of a server-side diffusion-based image synthesis model allows for the generation of highly detailed and diverse textures, enabling users to create realistic and unique designs, without requiring the client device to perform any computationally intensive processing.

This prototype was built with the STT, image synthesis, and file server service. The STT service is used to transcribe user microphone audio. The text output of this service is then parsed through regular expressions, resulting in a target object identifier and a target appearance. The target object identifier is then passed to the Unity scene, which temporarily applies a chequerboard pattern to the object that is tagged with the specified identifier (e.g. "couch").

At the same time, on the server, the text describing the target appearance is passed to the image synthesis service, which generates an image using Stable Diffusion 2.0. The resulting image is then made available to the Unity scene through the file server service, of which a hyperlink is sent to the Unity client in a network message. The Unity scene then downloads the texture, generates a mipmap for the texture, and applies it to the target object (Figure 3B).

Interaction Users can specify the desired appearance of an object through voice prompts, such as "Make the *floor* look like *lava*". In addition, instead of specifying the target object verbally, users can use their controller for ray-based selection. Users can use this selection method by pointing their controller at the target object and pressing and holding the trigger button. While the trigger button is held, users can specify the desired appearance of the object verbally (e.g. "Make *this* look like *lava*"). The resulting target object identifier is then sent to the server-side `ApplicationController` as a log message. Ray-based selection is particularly useful for selecting abstract objects or submeshes of complex objects, such as the knob of a door.

5.2 Conversational Agent

This prototype application allows users to interact with a conversational agent through voice prompts in a CVE (Figure 4). The agent is represented in the CVE with a robot-like avatar. The text synthesis model that drives the agent is made aware of what is said by who, which enables a wide range of multi-party conversations, such as debates, quizzes, and discussions.

The prototype was built with the STT, text synthesis, and TTS service. After the initialisation of the text synthesis model, the model is fed with a prompt that precisely specifies the script-like syntax that should be used when receiving and sending text data. Specifically, it is told that its name is *Agent*, that all text should be prefixed with a *source* and a *target* avatar name (e.g., Agent -> Bob: ...), and that it should keep note of who is part of the conversation. This gives the conversational agent awareness of what is said by who in the CVE, enabling multi-party conversations.

The STT service is used to transcribe user microphone audio. The text output of this service is prefixed with the avatar name of the respective user and the target user. The prefixed text is then passed to the text synthesis service, which generates text that simulates a real-world dialogue. The text output generated by this service, with a prefix including the target avatar, is then passed to the TTS service, which generates an audio version of the text output. The audio output is then streamed to the Unity scene along with information specifying the target avatar, where it is played.

Interaction Users can interact with the agent through voice prompts, which can range from simple questions such as "Who was the first person to step foot on the moon?" through to more complex questions including other user's names such as "Please explain the role of squeezed states in quantum optics to *Bob*". Users can activate the transmission of audio data by pressing and holding the grip button of their controller.

The agent's audio response is spatialised based on its position within the CVE. Furthermore, when the agent speaks, it gestures its hands based on prerecorded motion and a visual speech indicator appears near its head (Figure 4B). Furthermore, the agent turns towards users who speak to indicate that it listens to what is said. It also turns towards specific users when this is specified in the output of the text synthesis service.

6 Opportunities and Future Work

The proposed architecture to integrate external frameworks into the Ubiq social VR platform presents a wide range of opportunities for future research and development. In this section, we discuss several potential opportunities for building upon this work, including the expansion and improvement of the proposed services, applications, and architecture.

6.1 Services and Applications

Expanding the services and applications offered by Ubiq-Genie could lead to novel types of VR experiences that enhance collaboration and communication within CVEs. This includes the integration of other generative models for content creation, the use of experimenter dashboards for user behaviour analysis, and the potential of online machine learning.

Generative Models Incorporating other generative models in Ubiq-Genie, beyond the employed image and text synthesis models, could lead to interesting applications. Potential models to be integrated could be capable of synthesising 3D models from text or images such as Point-E [16], personalised speech from text such as VALL-E [24], or audio from text or images such as Make-An-Audio [9] and MusicLM [1]. In addition, the currently implemented services could be expanded to build more advanced types of applications and experiences. Examples of this include the generation of additional texture maps (e.g. normal and specular maps) and the generation of entire virtual environments [3]. Furthermore, the Ubiq-Genie architecture provides straightforward techniques for communication among services, which could enable interesting opportunities such as image synthesis based on complex descriptions provided by the text synthesis service or audio synthesis based on the output of the image synthesis service.

Behaviour Analysis Another potential direction of future work is the exploration of experimenter dashboards built with Ubiq-Genie. These dashboards could allow researchers to control their experiments and analyse user behaviour in real time, providing valuable insights for the evaluation of VR applications. Real-time metrics could be included to inform researchers about the level of user activity (e.g. head rotation velocity and words spoken [17]), communication style (e.g., recognition of F-formations [22]), emotional state (e.g., sentiment analysis using natural-language processing [7]), and user experience (e.g. the objective representations of presence and co-presence [18]). This could be particularly useful in observational, pilot, and remote studies [12, 23].

6.2 Architecture

Aside from the wide range of potential applications, depending on the needs, there are several ways that the Ubiq-Genie architecture could be improved in the future. For example, in terms of scalability, latency, and input sources.

Input Sources The prototype applications currently rely on users providing input through voice prompts and basic controller-based interactions. However, other input sources could be integrated, such as hand tracking, facial expressions, or body tracking. The data provided by these input sources could not only be used to expand the user interfaces of applications but could also be used to incorporate users' expressions and emotions in the input of generative models, which could be used to adapt their output.

Furthermore, visual data from the sensors of VR or AR head-mounted displays (HMDs) may be used to enable server-assisted applications, including real-time object recognition, semantic segmentation, and 3D reconstruction, which could enable interesting data-driven interaction capabilities [26].

Latency and Scalability The Ubiq-Genie architecture and prototypes are designed to be modular and to handle situations where multiple users interact with it at the same time. However, there are opportunities for improvement in terms of reducing latency and increasing scalability. For instance, for CVEs where very large groups of people gather and interact, the architecture could be optimised to handle incoming and outgoing data in more efficient or distributed ways. For instance, the image synthesis service currently can only generate one image at a time, which could be parallelised to allow for multiple requests. This could potentially be achieved through the integration of production-grade container and orchestration systems[4,5]. Future work could formally evaluate the limits of Ubiq-Genie related to latency and scalability.

7 Conclusion

This paper presents Ubiq-Genie, a system for the integration of external frameworks with the Ubiq social VR platform. The Ubiq-Genie system architecture is designed to be modular, allowing for the easy integration of services and providing mechanisms to offload computationally intensive processes to a server. We present two prototypes of collaborative applications built using Ubiq-Genie: a voice-controlled texture generation method and a voice-controlled conversational agent.

By providing techniques for integrating external services and frameworks, we aim to pave the way for new and innovative VR experiences that can enhance collaboration and communication within CVEs. Additionally, we aim to inspire further research that explores the potential applications of the described techniques for both users and researchers.

Overall, the proposed system architecture and the implemented prototypes demonstrate the potential of integrating external (open-source) frameworks into social VR, which can enable a wide range of enhanced experiences. We hope that the Ubiq-Genie system architecture can serve as a starting point for researchers and developers to further explore the potential of integrating external frameworks and services into social VR.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 Research and Innovation program under grant agreement No 739578.

References

[1] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi, M. Sharifi, N. Zeghidour, and C. Frank. MusicLM: Generating Music From Text, Jan. 2023. arXiv:2301.11325 [cs, eess]. doi: 10.48550/arXiv.2301.11325

[2] D. Chatzopoulos, C. Bermejo, Z. Huang, A. Butabayeva, R. Zheng, M. Golkarifard, and P. Hui. Hyperion: A Wearable Augmented Reality System for Text Extraction and Manipulation in the Air. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pp. 284–295. ACM, Taipei Taiwan, June 2017. doi: 10.1145/3083187.3084017

[3] S. Fox. Stable Diffusion VR, Oct. 2022.

[4] S. J. Friston, B. J. Congdon, D. Swapp, L. Izzouzi, K. Brandstätter, D. Archer, O. Olkkonen, F. J. Thiel, and A. Steed. Ubiq: A System to Build Flexible Social Virtual Reality Experiences. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, VRST '21, pp. 1–11. Association for Computing Machinery, New York, NY, USA, Dec. 2021. doi: 10.1145/3489849.3489871

[5] S. Gammeter, A. Gassmann, L. Bossard, T. Quack, and L. Van Gool. Server-side object recognition and client-side object tracking for mobile augmented reality. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pp. 1–8, June 2010. ISSN: 2160-7516. doi: 10.1109/CVPRW.2010.5543248

[6] GitHub. Octoverse 2022: The state of open source, 2022.

[7] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. spaCy: Industrial-strength natural language processing in python. 2020. doi: 10.5281/zenodo.1212303

[8] B.-R. Huang, C. H. Lin, and C.-H. Lee. Mobile augmented reality based on cloud computing. In *and Identification Anti-counterfeiting, Security*, pp. 1–5, Aug. 2012. ISSN: 2163-5056. doi: 10.1109/ICASID.2012.6325354

[9] R. Huang, J. Huang, D. Yang, Y. Ren, L. Liu, M. Li, Z. Ye, J. Liu, X. Yin, and Z. Zhao. Make-An-Audio: Text-To-Audio Generation with

[4] Docker (`https://docker.com/`)

[5] Kubernetes (`https://kubernetes.io/`)

Prompt-Enhanced Diffusion Models, Jan. 2023. arXiv:2301.12661 [cs, eess]. doi: 10.48550/arXiv.2301.12661

[10] Hugging Face. huggingface/diffusers, Jan. 2023. original-date: 2022-05-30T16:04:02Z.

[11] A. Khurshid, S. Cleger, and R. Grunitzki. A Scene Classification Approach for Augmented Reality Devices. In C. Stephanidis, J. Y. C. Chen, and G. Fragomeni, eds., *HCI International 2020 – Late Breaking Papers: Virtual and Augmented Reality*, Lecture Notes in Computer Science, pp. 164–177. Springer International Publishing, Cham, 2020. doi: 10.1007/978-3-030-59990-4_14

[12] J. Lee, R. Natarrajan, S. S. Rodriguez, P. Panda, and E. Ofek. RemoteLab: A VR Remote Study Toolkit. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, UIST '22, pp. 1–9. Association for Computing Machinery, New York, NY, USA, Oct. 2022. doi: 10.1145/3526113.3545679

[13] L. Liu, H. Li, and M. Gruteser. Edge Assisted Real-time Object Detection for Mobile Augmented Reality. In *The 25th Annual International Conference on Mobile Computing and Networking*, pp. 1–16. ACM, Los Cabos Mexico, Aug. 2019. doi: 10.1145/3300061.3300116

[14] Mozilla. Project DeepSpeech, 2021. original-date: 2016-06-02T15:04:53Z.

[15] Niantic. Lightship VPS, Sept. 2022.

[16] A. Nichol, H. Jun, P. Dhariwal, P. Mishkin, and M. Chen. Point-E: A System for Generating 3D Point Clouds from Complex Prompts, Dec. 2022. arXiv:2212.08751 [cs]. doi: 10.48550/arXiv.2212.08751

[17] N. Numan and A. Steed. Exploring User Behaviour in Asymmetric Collaborative Mixed Reality. In *Proceedings of the 28th ACM Symposium on Virtual Reality Software and Technology*, p. 11. ACM, Tsukuba, Japan, 2022. doi: 10.1145/3562939.3565630

[18] M. Ochs, S. Jain, and P. Blache. Toward an Automatic Prediction of the Sense of Presence in Virtual Reality Environment. In *6th International Conference on Human-Agent Interaction (HAI-2018)*. Southampton, United Kingdom, Dec. 2018.

[19] OpenAI. ChatGPT: Optimizing Language Models for Dialogue, Nov. 2022.

[20] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever. Robust Speech Recognition via Large-Scale Weak Supervision, Dec. 2022. arXiv:2212.04356 [cs, eess]. doi: 10.48550/arXiv.2212.04356

[21] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685, June 2022. ISSN: 2575-7075. doi: 10.1109/CVPR52688.2022.01042

[22] F. Setti, C. Russell, C. Bassetti, and M. Cristani. F-Formation Detection: Individuating Free-Standing Conversational Groups in Images. *PLoS ONE*, 10(5):e0123783, May 2015. doi: 10.1371/journal.pone.0123783

[23] A. Steed, D. Archer, K. Brandstätter, B. J. Congdon, S. Friston, P. Ganapathi, D. Giunchi, L. Izzouzi, G. W. W. Park, D. Swapp, and F. J. Thiel. Lessons learnt running distributed and remote mixed reality experiments. *Frontiers in Computer Science*, 4, 2023.

[24] C. Wang, S. Chen, Y. Wu, Z. Zhang, L. Zhou, S. Liu, Z. Chen, Y. Liu, H. Wang, J. Li, L. He, S. Zhao, and F. Wei. Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers, Jan. 2023. arXiv:2301.02111 [cs, eess]. doi: 10.48550/arXiv.2301.02111

[25] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser. Service Entity Placement for Social Virtual Reality Applications in Edge Computing. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pp. 468–476, Apr. 2018. doi: 10.1109/INFOCOM.2018.8486411

[26] W. Willett, B. A. Aseniero, S. Carpendale, P. Dragicevic, Y. Jansen, L. Oehlberg, and P. Isenberg. Perception! Immersion! Empowerment! Superpowers as Inspiration for Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):22–32, Jan. 2022. doi: 10.1109/TVCG.2021.3114844

[27] J. J. Yang, C. Holz, E. Ofek, and A. D. Wilson. DreamWalker: Substituting Real-World Walking Experiences with a Virtual Reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, pp. 1093–1107. Association for Computing Machinery, New York, NY, USA, Oct. 2019. doi: 10.1145/3332165.3347875

[28] L. Zhang, A. Sun, R. Shea, J. Liu, and M. Zhang. Rendering multiparty mobile augmented reality from edge. In *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 67–72. ACM, Amherst Massachusetts, June 2019. doi: 10.1145/3304112.3325612