Public Review for

VRComm: An End-to-End Web System for Real-time Photo-realistic Social VR Communication

S. Gunkel, R. Hindriks, K. Assal, H. Stokking, S. Dijkstra-Soudarissanane, F. Haar, O. Niamut

This paper presents VRComm, a web-based social VR framework for enabling remote communications via video conferencing. Unlike existing social VR platforms such as Facebook Horizon and Mozilla Hubs, VRComm supports real-time photorealistic representation of users by transmitting depth data captured by the RGBD camera as 2D gray-scale images. It offers an end-to-end pipeline including capturing, processing, transmission, and rendering. To improve the scalability of the system, the authors proposed to use a multi-point control unit (MCU). The performance of VRComm was evaluated using both simulations and realistic user experiments.

The strengths are as follows. With the increasing use of video conferencing in people's daily lives, it is important to study how to effectively enable social VR and spatial computing. This paper presents a thorough description of the design and implementation of different components of such a system. The authors implemented a prototype system using off-the-shelf hardware and web-based client software that can support low-cost VR video conferencing. The authors also conducted extensive evaluation based on simulation studies and real-world experiments to show the proposed system is promising and realistic.

Besides the strengths, the paper also raised questions regarding the challenges and durability of the proposed solution. It would have been useful to present the main challenges for designing and implementing such types of social VR systems. Also, VRComm relies on several open-source software packages, and the durability of such software components impacts that of VRComm.

Public review written by
Yao Liu
State University of New York, Binghamton, USA

VRComm: An end-to-end web system for real-time photorealistic social VR communication

Simon N.B. Gunkel
TNO, Den Haag, Netherlands
simon.gunkel@tno.nl

Rick Hindriks
TNO, Den Haag, Netherlands
rick.hindriks@tno.nl

Karim M. El Assal
TNO, Den Haag, Netherlands
karim.elassal@tno.nl

Hans M. Stokking
TNO, Den Haag, Netherlands
hans.stokking@tno.nl

Sylvie Dijkstra-Soudarissanane
TNO, Den Haag, Netherlands
sylvie.dijkstra@tno.nl

Frank ter Haar
TNO, Den Haag, Netherlands
frank.terhaar@tno.nl

Omar Niamut
TNO, Den Haag, Netherlands
omar.niamut@tno.nl

Figure 1: 16 user streams in a Virtual Experience (left; anonymous; 3D Background Model "Olam Conference Room" by Gideon Abochie licensed under CC Attribution) and RGBD user transmission of RGB-part (middle) and depth-part (right)

ABSTRACT

Tools and platforms that enable remote communication and collaboration provide a strong contribution to societal challenges. Virtual meetings and conferencing, in particular, can help to reduce commutes and lower our ecological footprint, and can alleviate physical distancing measures in case of global pandemics. In this paper, we outline how to bridge the gap between common video conferencing systems and emerging social VR platforms to allow immersive communication in Virtual Reality (VR). We present a novel VR communication framework that enables remote communication in virtual environments with real-time photorealistic user representation based on colour-and-depth (RGBD) cameras and web browser clients, deployed on common off-the-shelf hardware devices. The paper's main contribution is threefold: (a) a new VR communication framework, (b) a novel approach for real-time depth data transmitting as a 2D grayscale for 3D user representation, including a central MCU-based approach for this new format and (c) a technical evaluation of the system with respect to processing delay, CPU and GPU usage.

CCS CONCEPTS

• **Information systems** → **Web conferencing**; *Multimedia information systems*; • **Computer systems organization** → **Distributed architectures**; • **Human-centered computing** → **Virtual reality**; *Mixed / augmented reality*.

KEYWORDS

Virtual Reality, VR, Communication, Conferencing, Social VR, Immersive Media, WebVR, WebXR, WebRTC

ACM Reference Format:
Simon N.B. Gunkel, Rick Hindriks, Karim M. El Assal, Hans M. Stokking, Sylvie Dijkstra-Soudarissanane, Frank ter Haar, and Omar Niamut. 2021. VRComm: An end-to-end web system for real-time photorealistic social VR communication. In *ACM Multimedia Systems Conference (MMSys '21), September 28–October 1, 2021, Istanbul, Turkey.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3458305.3459595

MMSys '21, September 28–October 1, 2021, Istanbul, Turkey

ACM ISBN 978-1-4503-8434-6/21/09…$15.00
https://doi.org/10.1145/3458305.3459595

1 INTRODUCTION

Communication and collaboration are an important part of everyday life, both in professional and private environments. Having tools to help one communicate over distance, such as the video conferencing applications Skype, Google Hangouts, or Zoom, has rapidly become a normality in a modern and globalized world. Nowadays, these applications can be run both as stand-alone, dedicated apps and programs, as well as web applications (apps) in browsers that implement rich media tooling such as WebRTC and support advanced video decoding. However, looking at these tools today, common video conferencing applications have clear limitations regarding enabling a sense of (co-)presence and immersion. That is, "the video calling stuff breaks down beyond a few people, because you have this big grid of tiny faces." Simply by the absence of "the full range of social cues, from posture to eye gaze to facial expressions, things like head nodding and hand gesturing", that may convey crucial nonverbal information (Blair MacIntyre[1]). And even when video conferencing systems manage to convey information communicated by facial expressions and body gestures, they do so at the expense of a sense of shared space.

Not only recent events (like the global covid-19 pandemic) raise the need for better solutions that increase the feeling of togetherness while communicating remotely. One of the first steps in adding space sharing in 2D video conferencing can be seen in Microsoft Teams together mode[2] that blends the 2D webcam video of users in a simply lecture hall inspired background. Further, from other works [3, 5, 48] we know that spatial awareness, presence and immersion can be provided by communication in Virtual Environments and Virtual Reality (VR) experiences. However, in initial VR frameworks, the lack of social interaction [11, 39] prevented users to experience co-presence. With recent social VR applications and platforms for VR meetings and conferencing, social interactions have started to make their way into the virtual realm as well. These platforms typically make use of model-based Avatars, i.e. a graphical representations of participants whose movements are steered by input from the VR headset and/or controllers. In order to allow users to consume existing media streams as well (for example shared consumption of live and on-demand video streams like sport), the integration of video into these platforms is currently taking shape. One example of a social VR system that is widely accessible is Mozilla Hubs[3]. It solved many interesting aspects of social VR from a technical perspective, for example, objects and states synchronization across multiple clients. However, Moxilla Hubs currently deals with video only in a limited way (i.e., streams for presentations and static 2D webcam views).

In this paper, we seek to bridge the gap between common video conferencing systems and emerging social VR platforms. That is, we aim to reuse proven technologies and frameworks from the domain of video conferencing, and build a platform for VR communication experiences that incorporates photorealistic user representations. We mainly consider the end-to-end video processing chain and the use of a multipoint control unit (MCU) to bridge the multiple video-conferencing connections . Our main hypothesis is that by reusing these components from common video conferencing systems, we can support VR conferencing under network requirements that are similar to those for traditional video conferencing.

The motivation for this work lies in the relevance of end-to-end video processing technology to provide real-time performance in web-based social VR applications. While the importance of recent volumetric video formats to provide true 6-degrees-of-freedom VR experiences is becoming apparent [8, 49], the end-to-end workflow to process such data from capture to rendering is far from real-time. The use of video-based methods allows us to benefit from existing deployed infrastructures and interfaces(such as hardware acceleration, robust coding, and streaming) and ultimately extended support in many browser platforms. The contribution in this paper is threefold:

i.) we describe a new VR communication system that combines video conferencing technology with social VR capabilities in a new end to end pipeline from user capture, processing, transmission and rendering users into virtual environments for shared immersive experiences and communication
ii.) we report on a novel transmission scheme for grayscale based depth information, for 3D user representations, including a central MCU-based approach for the transmission of this video format
iii.) we perform an evaluation of the resulting VR communication system with respect to processing delay, CPU and GPU usage

2 RELATED WORK

2.1 Video Conferencing

The first video conferencing systems were based on one-on-one connections between two sites. Scalability for multi-person video conferences could either be achieved by a full-mesh exchange of media streams between all participants, or by using potentially available multicast mechanisms. As IP multicast technology is not widely deployed across the public internet, centralized mixing facilities called Multipoint Control Units (MCU) were developed in the 1990s. These MCUs multiplex in some form the various media streams, so only a single stream needs to be sent to each participant. With many participants, such a centralized scheme allows for improved scaling of the bandwidth requirements and can scale the video conference to a large set of simultaneous users, in particular by designing hybrid centralized forwarding architectures [19].

Various developments were made more recently to further achieve scalability without sacrificing quality. For MCUs, recent cloud developments give the opportunity to use processing on-demand, thereby allowing conferencing sessions to scale up to many hundreds or even thousands of participants [36]. For an efficient stream multiplexing without any media processing, Selective Forwarding Units (SFU) are developed, see [43] and [53]. These SFUs forward streams from one participant to all other participants, thereby alleviating the need for one participant to send out separate streams. To support bandwidth adaptation, each participant can send its stream in a few different bitrates so that an SFU can select and forward individualized streams depending on the bandwidth availability for each participant.

[1] https://spectrum.ieee.org/tech-talk/consumer-electronics/audiovideo/forget-video-conferencinghost-your-next-meeting-in-vr
[2] https://news.microsoft.com/innovation-stories/microsoft-teams-together-mode/
[3] https://hubs.mozilla.com/

More recently, MCU architectures have also improved through the use of tiling mechanisms [15]. Tiling allows for stitching together video parts in the encoded domain, creating an architecture that sits somewhere between the MCU and the SFU: all incoming streams are mixed together so that each participant receives only a single media stream to be decoded, while at the same time no decoding-mixing-encoding is required.

2.2 Social VR

In the '90s, much work went into creating high-end shared virtual environments. Various universities set up cave automatic virtual environments (CAVE) and CAVE-like systems which could be used to communicate remotely, of which [23] and [37] provide good examples, using what they call "video avatars". These environments typically used back projection and large calibrated camera rigs to produce a coherent virtual environment. Other examples such as [26] used large screens, again together with calibrated camera rigs, to also offer a sense of togetherness.

Other work from this era consists of using graphical avatars to create large shared virtual environments, of which [4, 29, 45] give some overview. In these days, the impact of avatar realism on the participants' perception was studied as well [18]. Virtual reality saw a renewed interest with the rise of high-quality but affordable AR and VR HMDs, most notably the Oculus Development Kit, which carried the promise of bringing high-quality VR to the masses. This development has led to new initiatives in shared and social VR experiences as well. Nowadays, social VR is mostly associated with graphical avatars in a graphical environment. Main examples are Facebook Horizon, AltSpaceVR, BigScreen, Glue, High Fidelity, vTime, Hubs by Mozilla, VR chat, SteamVR and Spatial.

While considering the Social VR services mentioned, all represent users as graphical avatars displayed in a shared VR environment and allow users to play games, share screens, share web browsing, watch videos, explore spaces or share other experiences together. However, very little studies exist that compare those new services with existing communication tools or compare real-time photorealistic representations with artificial avatars. One study [39] suggests that graphical avatars (Facebook Horizon, formerly Facebook Spaces[4]) have limitations in terms of (co-)presence as "the social cues that you would normally have about someone … weren't there". Another study [11] presents the results that real-time photorealistic representations show no statistical differences in terms of interaction and social connectedness compared to a face-to-face meeting, while the avatar-based system (Facebook Horizon[4]) did. In the context of collaboration, VRComm and "traditional" videoconferencing are theorized to differ in their affordability of social context cue transfer, specifically in terms of body posture, gestures, and eye-contact. In addition, the experience of presence, which is typically found in VR (e.g. [9]), may affect communication and interpersonal relationships as well.

2.3 Spatial Computing & HMD replacement

Spatial Computing, which is the ability to understand the environment, the user, and objects surrounding the user, is an essential part of AR and VR applications (good examples can be found in [14, 20]). In terms of communication, this means that user's actions should be correctly reflected into the users' representation to convey good remote interactions. One example of complex and processing intensive Spatial Computing tasks is the replacement of the HMD that by default occludes the participants face when using a VR-HMD as a display device. As facial expressions and eye gaze are important factors in communication, the HMD replacement process becomes essential for a qualitative experience. The so-called facial reconstruction is relevant for improving the user experience in (video-based) social VR. When capturing the user with a video and/or depth sensor, the captured footage will include the HMD and thus occlude parts of the face, including the eyes. Our earlier experiments have shown that it still allows natural interaction and communication with an increased feeling of co-presence [21]. Takemura [50] was one of the first researchers to describe a method that accomplishes the HMD replacement by detecting the HMD location in the video and replacing the pixels by a 3D facial model captured at an earlier process. Li [30] followed a similar approach by using strain gauges inside the HMD to measure facial expressions. Burgos-Artizzu [6] used various facial models to represent various expressions and detect expressions based on the part of the face still visible in the video recording. More recently, Thies [51] and Google [17] elaborated on this approach by combining these methods with an eye-tracking camera inside the HMD to reproduce "correct" eye direction. Furthermore, recent works aim to make the HMD-replacement more robust and flexible by using RGB-D image inpainting techniques [35].

2.4 Volumetric video capture and transmission

Volumetric video is regarded worldwide as a key technology in the context of immersive AR and VR experiences. The capture, encoding, and transmission of volumetric video formats such as point clouds and meshes is an active field of research [1, 10, 33, 40, 44] as well as industry standardization[46]. In particular, for remote telepresence and immersive communication[38, 57], volumetric videos provide increased quality of experience and social presence [8, 49]. The most recent volumetric video formats (i.e. video-based (V-PCC)[46] and geometry-based (G-PCC)[33]) point cloud coding) require significant processing resources for capturing, coding, transmitting and rendering[33, 46]. While V-PCC is currently not suitable for real-time communication (due to its encoding latency), recent work has started to optimise G-VCC for telepresence scenarios [7, 25] by data reduction and fusion. Still, many open challenges exist regarding volumetric media delivery [52]. This includes high data rates, high processing load, high encoding delays, low resolution, or low frame rate (or a combination of those). To address the current gaps of V-PCC and G-PCC our work mainly considers a video-based (RGB plus depth) transmission approach as an initial step towards full volumetric representation of users. The adaptation of standard video codecs for depth streaming was studied in [41], whereas more advanced conversion of depth information to grayscale images was considered in [32] and [13].

2.5 Tele-immersion and telepresence systems

The research direction closest to this paper can be seen in tele-immersion and telepresence. While telepresence systems offer a

[4]https://www.oculus.com/facebook-horizon/

professional video conferencing experience with dedicated hardware setups to showcase people more realistic (looking more natural, examples are Lifelike and Cisco), tele-immersion is making the step to offer communication systems in virtual environments. Multiple research efforts have been done in the last decades to address tele-immersion and telepresence [27, 28, 31, 47, 55]. One of the first examples of a tele-immersion system is TEEVE [56] which allows 3D capture and rendering of users in a complex setup with a frame rate of 4 5 frames per second. Another example of a tele-immersion system is the Roomalive toolkit[5][54] from Microsoft. One research utilizing Roomalive is Room2Room [42], limited to one-to-one interactions, it allows projection-based AR telepresence with the help of a depth sensor and projector. We can summarize the efforts of tele-immersion and telepresence systems in a complex setup that require dedicated hardware, have high performance and network requirements, or support only a very limited number of users. In this paper, we present our work to bridge the gap from low-cost simple video conferencing solutions towards volumetric user representations in VR, with a focus on off-the-shelf hardware and web-based client software for a low entry burden. Furthermore, we aim to support an capture framerate, capture resolution, and network utilisation similar to existing video conferencing solutions.

3 VRCOMM FRAMEWORK

Moving from traditional video conferencing towards VR conferencing arises new requirements on the system as well as the device setup [21]. One of the main differences between video conferencing and VR communication is that in VR, we work on a geometry-based 3D environment rather than a window-like 2D arrangement for video conferencing. VR conferencing entails multiple aspects of supporting new media formats, interaction paradigms, as well as ways to orchestrate and synchronise media in the virtual environment. In particular, new ways to capture and blend users into the geometrical space are crucial for higher immersion and presence for natural communication [21], which includes 3D user representation, enabling self-representation and maintaining eye gaze with others (i.e. not being restricted by wearing a HMD). In addition to supporting novel user representation formats, VR conferencing systems should scale to support numbers of users that are similar to those of traditional 2D video conferencing applications. Figure 2, shows the VRComm system with the different components and technology aspects being explained in the following subsections.

3.1 Architecture and media orchestration

VRComm is a web-based framework to build and consume shared and social VR experiences. Our main motivation to utilize Web-based technology is to cater for an easy and widespread deployment and low entry burden for end users and developers. In this way, we currently only make use of off-the-shelf hardware and currently available web technologies.

Figure 2, shows the overall framework architecture of VRComm. To initiate a web client instance, first the client JavaScript code and any multimedia files are downloaded from a web server, secondly the web client will register at the Media Orchestrator by selecting and exchanging session metadata and finally will negotiate WebRTC stream connections with the help of the Signal Master (either as peer-to-peer or MCU transmission scheme). This results in the following video transmission modules and processing steps:

(1) Capture of raw sensory data (see Section 3.3)
(2) Capture Processing (see Section 3.3):
  (a) Video Background Removal
  (b) (optional) Camera calibration
  (c) (optional) HMD Replacement
(3) RGBD grayscale conversion (see Section 3.4)
(4) Web client ingest (see Section 3.2)
(5) (opt.) local rendering of self-representation (see Section 3.2)
(6) WebRTC Transmission (see Section 3.4), either
  (a) Peer-to-peer (p2p), or
  (b) (optional) central server / MCU-based (see Section 3.5)
(7) rendering of remote user(s) (see Section 3.2)

One central component of VRComm is a media orchestration server, which manages communication sessions that users can join to communicate with each other. For example, this also includes the calibration data of the user capture to allow 3D reconstruction of the user representation. Furthermore, the orchestration server maintains all metadata to synchronize and modify the virtual environments of each client at run-time. The orchestration server powers monitoring and modifying all client properties relevant to the VR experience in real-time and thus facilitates complex interactive VR multimedia experiences. Particularly, the content that is displayed can be modified, e.g., a game or movie, and the placement of objects and users.

To facilitate the calibration and design of the VR rooms, we have developed a metadata format to position users and immersive media objects into the virtual space. The schema of the rooms metadata is shown in Figure 3. In the centre is the VRoom which can cover different objects related to users, the virtual scene (or environment), and different media objects (like video panes or interaction elements). We can currently support many different media objects like 2D video (including DASH), images, 360-degree content, 3D models (e.g., in the OBJ or GLTF format) and our own 2D/3D RGBD format (see Section 3.2]). Furthermore, the metadata-based virtual scene creation and media object allocation also serves as a global coordinate system and simplifies the synchronisation, interaction and remote configuration of different user client states in the system. The media orchestration provides an admin console that grants a fine-grained control of all metadata properties in real-time. This remote control is designed with the main aim to support any user experience research.

3.2 Web Client

The entry point for the VRComm web client is offered by a web server back-end. WebXR-enabled web browsers can connect to this server back-end to obtain the client application, which is based on open source software JavaScript frameworks Node.js, React, SimpleWebRTC and A-Frame[6]. This allows any modern browser to display the VR content on a screen or any Open VR hardware enabled VR-HMD (e.g., Oculus Rift or Windows MR). Furthermore, the client can access the image produced from the RGBD capture module (see Section 3.3) to be displayed as self-representation or

[5]https://www.microsoft.com/en-us/research/project/roomalive-toolkit/

[6]https://aframe.io/

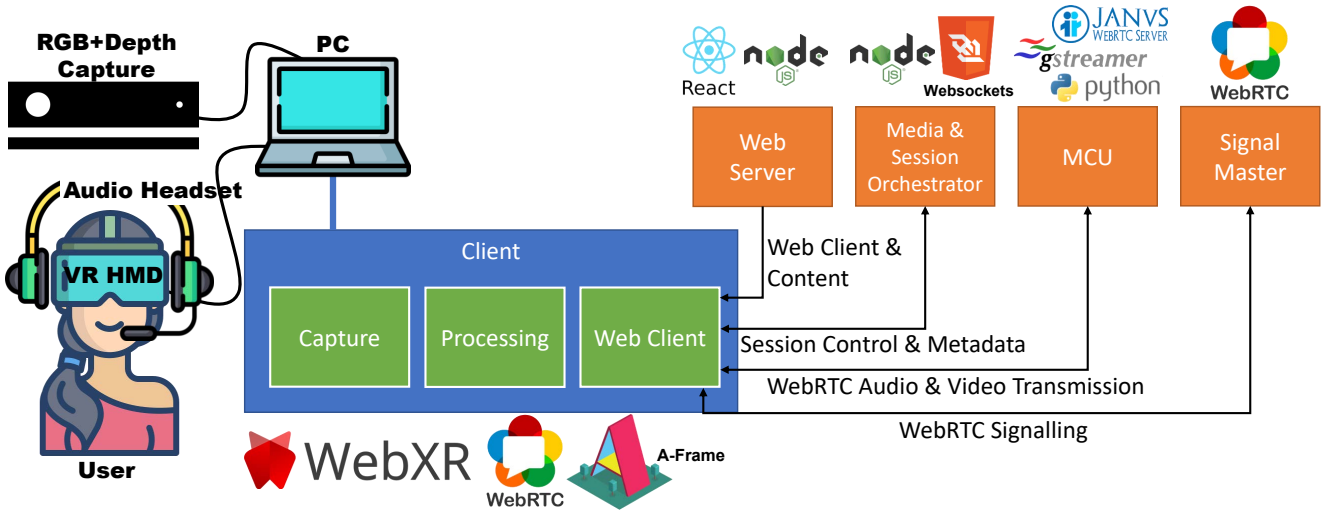

Figure 2: VRComm System components and user setup

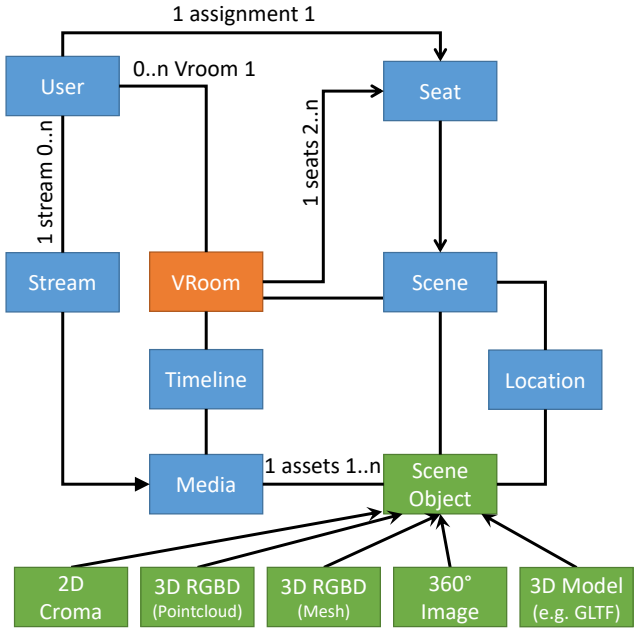

Figure 3: Virtual Scene Description Metadata Schema

to be sent via WebRTC to one or multiple other clients. To support such a multi-user connection, the client is connected to a signalling server that handles streaming orchestration.

The rendering of users in the VR environment is done via custom WebGL shaders that alpha-blend user representations into the virtual environment for a natural visual representation. We first record users with a RGB-plus-depth sensor (e.g., in VRComm we currently support Kinect v2, RealSense™ and Azure Kinect) and then have two options for transmission: i) we replace the users background with a fixed "chroma" colour before transmission (over WebRTC), and after reception apply alpha-blending to remove the background, resulting in a transparent image showing just the user without his/her physical background. ii) we convert the depth values into grayscale and transmit them along the image to render the user in 3D (see section 3.4). For capture and transmission, we currently use a resolution of 540x800 pixels for RGB (with chroma background) or 1080x800 pixels for RGBD images. This resolution is matching the depth resolution of most depth sensors. However, our system is fully adaptable to any resolution. Audio is also captured, transmitted, and made spatially audible with the help of the Google Resonance API[7].

When utilizing a 360-degree VR environment, we will transmit and render users in 2D only (RGB + chroma). While for 3D geometry-based environments, we render users in such a way that they can observe themselves (as self-representation) via 3D point cloud. The point cloud is created by first converting the grayscale depth data to a depth value and then recalculating the 3D position of each point based on the calibration data of each RGB-D sensor. This is done in a dedicated WebGL (GLSL) shader and thus runs efficiently on the GPU. The following 3D mapping is used (per pixel):

$$z = depth \tag{1}$$

$$x = (position.x - outputCenter.x - 0.5) * z/focalLength \tag{2}$$

$$y = (position.y - outputCenter.y - 0.5) * z/focalLength \tag{3}$$

Where depth is the distance of the pixel in meters, position is the pixel coordinate on the RGB video, outputCenter is the centre pixel coordinate of the sensor metadata, and focalLength is the focal length of the sensor.

Similar to the self-representation, we display remote users based on the video from a WebRTC connection. To render remote users, we developed 3 types of components for 3 types of rendering:

(1) Rendering in 2D (RGB + chroma background) via a shader that alpha blends the user into the virtual scene

[7]https://resonance-audio.github.io/resonance-audio/

(2) Rendering in 3D (RGB + Depth) as point cloud (mapping the 3D points on a THREE.js Geometry[8])
(3) Rendering in 3D (RGB + Depth) as mesh (mapping the 3D points on a PlaneBufferGeometry[9])

A VRComm client experience consists of VR environments referred to as 'VRooms' (see Figure 3). For the rendering of visual and audio information (e.g., the VR Environment, objects and users), the client will utilize the A-Frame framework, which combines the WebXR API with Three.js to provide a simple scripting framework for the design and development of web-based XR experiences. This allows to easily create 360-degree and 3D volumetric VR applications while supporting many 3D scenes and models (including glTF™[10]). A-Frame has integrated support for most common consumer hardware (including any SteamVR and OpenXR enabled device), among which the Oculus Quest, Windows Mixed Reality (WMR) headsets and HTC Vive. Further A-Frame is supported by major PC and laptop browsers like Mozilla Firefox, Google Chrome and Microsoft Edge, several mobile browsers including Chrome and Firefox. As a result, easy access to VR technology has become available on the web. Internally, the client follows a completely modularized structure via a combination of React Modules and A-Frame components. This allows an easy and dynamic creation of individual VR applications with different features.

3.3 Capture

One of the main challenges in any immersive communication system is on how to capture the users. For natural interaction and true social presence, it is important that a visual representation accurately reflects the appearance and actions of each user. Therefore we focus in this work on a camera-based capture solution to capture a photorealistic representation of users in real-time. Many factors have to be considered for this capture, as an example, some of the main factors include lighting, colour, edge accuracy, and other capture artefacts. The capture in VRComm is developed in a modular architecture and currently allows the use of one or two RGBD sensors and different processing modules. In the following, we outline the capture modules:

Raw Sensor Capture. The first capture module does real-time scene and user capture by reading the raw RGBD data from the capture sensor and mapping the colour and depth images to a shared memory location on a client PC. Currently, we have modules to support three types of RGBD sensors: Kinect v2[11], RealSense™ (Intel® RealSense™ SDK 2.0[12]) and Azure Kinect[13]. However, different RGBD sensors could easily be added without affecting any of the subsequent modules.

Foreground-background removal (FGBG). The raw RGBD sensor needs some further processing in a second capture module. The main aim of this module is to improve the image quality and perform real-time foreground-background (FGBG) extraction using colour and depth images from shared memory. This is an important step as for the user capture, we are exclusively interested in the user itself rather than his or her background. Furthermore, this allows us to only transmit the captured user and blend the visual (rendered) representation of users into the virtual environment.

Multi-cam capture and calibration. Currently, we support capture with one or two depth sensors. When two cameras are used, we calibrate and align both cameras. The calibration phase concerns the alignment of the two RGB-D sensors used to capture the participant. The registering and aligning of the two sensors are done via the help of a large ArUco[14] marker ($30x30cm$) and pose matching. This results in a near 180° 3D representation of the user (front view), from the RGB-D frame pairs [16]. The calibration parameters from the rigid body transformation are sent as metadata together with the RGB-D visual data.

HMD Replacement. The HMD Replacement module consists of an open source available ArUco marker detection (implemented using OpenCV[15] in Python[16]) applied to the RGB-D image. When the marker is attached to the HMD of the subject, the HMD can be detected in real-time without assumptions on the position of the subject or the capturing device. With additional markers on all sides of the HMD, the detection also works when the user looks left or right, up or down. The 2D detection in the RGB stream is combined with the depth to acquire an accurate 3D position and orientation of the HMD. There are multiple applications possible for this 3D position and orientation:

(1) 3D head removal for self-representation, such that the view of the subject is not occluded by the scan of his/her face
(2) Auto-calibration of multiple sensors, such that when two sensors detect the same marker they can auto-calibrate
(3) Integration with FGBG removal, to only have one foreground subject in VR
(4) 3D HMD replacement and 3D face repair in RGB-D or VR

Note that HMD replacement is not a core part of this paper and will be handled in a subsequent publication.

3.4 RGBD grayscale transmission

The VRComm streaming approach relies on a web framework with a peer-to-peer (p2p) nature for delivering video-based social VR experiences to each of the participants. This web streaming framework employs WebRTC for browser-based real-time communication. All 2D video streams and users' audio are transmitted via WebRTC. Any associated metadata are transmitted via a central media orchestration server and Socket.IO . Despite newer volumetric streaming formats (like V-PCC [46] and G-PCC [33]), this allows us to reuse many existing real-time streaming components (including to benefit from full hardware acceleration).

We use the SimpleWebRTC library to support direct WebRTC-based peer-to-peer communication between users for audio and video. At this moment, voice communication is monaural and spatially positioned in the receiver client (utilizing the Google Resonance Audio SDK for Web[17]). The integration of WebRTC with

[8] https://threejs.org/docs/index.html#api/en/core/Geometry
[9] https://threejs.org/docs/index.html#api/en/geometries/PlaneBufferGeometry
[10] https://www.khronos.org/gltf/
[11] https://developer.microsoft.com/en-us/windows/kinect/
[12] https://www.intelrealsense.com/sdk-2/
[13] https://azure.microsoft.com/en-in/services/kinect-dk/
[14] http://www.uco.es/investiga/grupos/ava/node/26
[15] https://opencv.org/
[16] https://www.python.org/
[17] https://resonance-audio.github.io/resonance-audio/

a VRComm client is managed through an (Node.js) orchestration server.

Grayscale conversion. For a depth transmission in VRComm we target a simple and reliable approach that works in real-time and is applicable to be used in any modern browser. This already implies a couple of design choices, e.g., transferring of RGB and depth should be done in one frame as a separate transmission and frame accurate synchronisation is very difficult in the browser (as the underlying media APIs needed for frame accurate synchronization are not exposed in JavaScript in every browser). Similarly, many underlying WebRTC transmission and decoding APIs are also not exposed by a browser raising the need to make any colour conversion and depth mapping directly in the WebGL (GLSL) shader itself.

For transmitting the RGB-D frame data over WebRTC to VR over the internet, we convert the depth data into a grayscale image for complying with current video encoders. For this conversion we use an improved version of [22]. While [22] does not utilise the full RGB range, we convert depth values corresponding to a real-world distance of $0 - 1.5m$ into gray-colour values that are mapped to the full RGB colour space (contrary to other approaches that directly modifies the YUV values, which will not be possible to convert back to depth in a browser WebGL shader). The grayscale depth image is concatenated to the RGB image to stream it as a single RGB-D video stream. In the VR environment, the depth image is converted back into the 3D positions of individual pixels. In this paper, we will refer to our algorithm as "GrayAVG". To ensure that GrayAVG works within a depth range of 1.5 m we first subtract a fixed value (this is the minimum distance a person is away from the camera to allow full body capture and is transmitted as metadata and added in the reconstruction) and remove all values outside of the 1.5 m range. The following shows our algorithm in Python/NumPy[18]:

$$depth = depth - min\_distance \tag{4}$$

$$depth[depth > 1500] = 0 \tag{5}$$

$$r = depth/3 \tag{6}$$

$$g = (depth - r)/2 \tag{7}$$

$$b = depth - r - g \tag{8}$$

3.5 RGB(D) Multipoint control Unit

One drawback of a p2p based WebRTC approach for transmission is scalability [43], as multiple users can quickly elevate the (CPU/GPU) resource usage. To mitigate this, we can (optionally) deploy a Multipoint Control Unit (MCU), to aggregate streams centrally and reduce the processing burden on individual clients.

Figure 4 depicts the architecture of our MCU. It reuses existing open source components, such as the Janus Video Bridge [2], which is a general purpose, central WebRTC server. While Janus takes care of all WebRTC stream handling (i.e., SDP negotiation and stream forwarding, as well as any audio transmission), the MCU composes all uploaded video streams it receives from clients into a single output stream which is then published via WebRTC to all clients. This output stream resembles a video mosaic combining all user streams, as depicted in Figure 5. As a result, clients are able to retrieve all relevant streams together instead of separately.

[18]https://numpy.org/

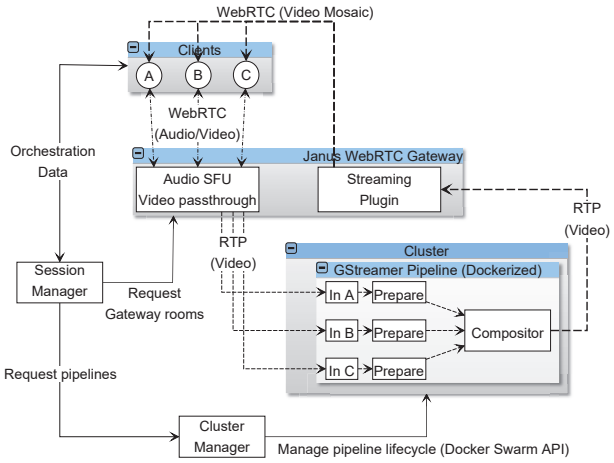

Figure 4: MCU Architecture [12]

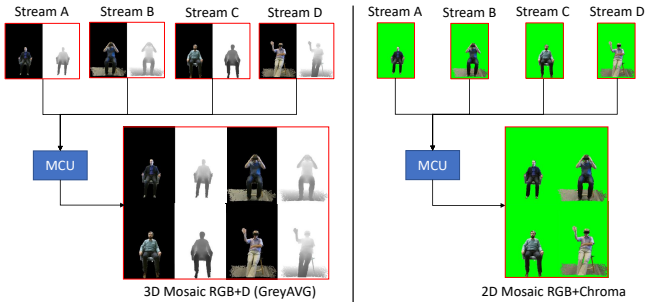

Figure 5: MCU Mosaic composition of RGBD (GreyAVG, left) and RGB+Chroma (right) input streams

This optimizes the network bandwidth due to more efficient routing (each client only sends its video stream to the MCU, and no longer to all other clients), as well as the decoding resources (clients typically have a limited amount of hardware decoders, which can result in higher CPU usage with many receiving streams).

Incoming WebRTC streams to the server are first remuxed by the Janus WebRTC Gateway into RTP streams that are sent to the MCU (based on GStreamer[19] and Python). The GStreamer media pipeline will then decode each individual user video stream into frames and convert them into NumPy[20] arrays. All NumPy images are then mapped into one complete output mosaic (see Figure 5) with efficient in-memory functions. This mosaic image is encoded into video frames and sent as a single RTP stream to Janus for distribution to an all client broadcast. The MCU system has been designed as a containerized service such that given enough hardware, it is horizontally scalable over multiple parallel sessions. These services can be managed using Docker Swarm[21] and the Media Orchestrator.

[19]https://gstreamer.freedesktop.org/
[20]https://numpy.org/
[21]https://www.docker.com/

Table 1: Capture Performance According to sensor (1000+ samples each; mean values)

RGBD Sensor	Capture (CPU in %)	Capture (GPU in %)	Processing (CPU in %)	RDA Delay (in ms)	Processing Delay (in ms)	Browser Delay (in ms)
One Kinect v2	10.53% (SD 1.13)	6.63% (SD 2.73)	19.08% (SD 2.70)	78.72 (SD 15.87)	138.19 (SD 18.71)	342.01 (SD 39.31)
One RealSense™ D415	3.94% (SD 1.21)	n/a	21.81% (SD 4.10)	79.54 (SD 15.79)	152.18 (SD 18.89)	190.16 (SD 19.56)
One Azure Kinect	4.43% (SD 2.00)	4.36% (SD 1.12)	9.64% (SD 1.90)	112.29 (SD 15.51)	155.99 (SD 16.90)	261.74 (SD 24.92)
Two RealSense™ D415	10.79% (SD 1.32)	n/a	10.36% (SD 2.32)	82.84 (SD 15.08)	147.06 (SD 16.35)	234.89 (SD 24.63)
Two Azure Kinect	9.11% (SD 2.49)	9.22% (SD 0.45)	9.52% (SD 1.62)	162.28 (SD 13.38)	306.28 (SD 29.35)	382.99 (SD 35.43)

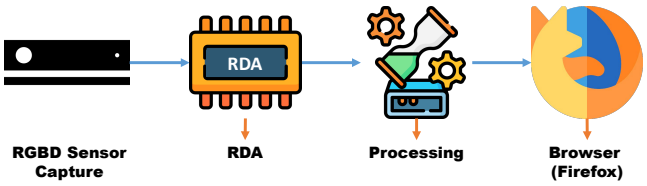

Figure 6: Measurements points of capture system

The user experience in VR Conferencing is greatly enhanced by using spatial audio. This requires to uniquely address each individual audio stream, to render it at the appropriate spatial location. Therefore, we use an SFU architecture (within the MCU) for the audio, such that clients can selectively request and retrieve individual audio streams from the MCU. The SFU part of the MCU is implemented purely using Janus (using the Videoroom plugin, see [2]), and requires no further processing.

4 SYSTEM EVALUATION

In this section, we focus on the technical evaluation of our system only. A summary of user evaluation of our system can be found in [21]. The evaluation of the core components of the system are structured into three parts, capture (Section 4.1), grayscale based depth transmission (Section 4.2) and web client evaluation based on using p2p vs. central MCU-based transmission (Section 4.3).

4.1 Capture Evaluation

We evaluate the capture performance with different sensor set-ups and at different points in our system. Figure 6 shows the different components and measure points for delay and CPU/GPU usage:

(1) Read sensor data via the sensor SDK into RDA. RDA (Remote Data Access) is a flexible infrastructure for real-time distributed data access and data acquisition. It allows easy exchange of video frame data between different software modules and allow a high flexibility for development and different hardware set-ups.
(2) Display data from RDA on the screen.
(3) Processing (i.e. FGBG and grayscale mapping)
(4) Screen capture the processed image and display it in the browser as self-representation, or transmit via WebRTC. We are currently following this procedure as no current browser implements the Media Capture Depth Stream Extensions [22] thus making it impossible to capture depth data directly in the browser.

All measurements were done on a MSI GS65-Stealth-Thin-8RF (with Intel® Core™ i7-8750H, GeForce® GTX 1070 Max-Q and 32GB RAM). The capture-to-display delays were measured with VideoLat[24][23], with at least 1000 samples each. CPU and GPU performance was measured with a modified version of the Resources Consumption Metrics (RCM) measurement tool[24] [34]. The RCM tool is a native Windows application that allows to capture CPU, GPU, memory usage per process and network statistics of the system in a 1-second interval. Each performance measure was done with a representative sample size of at least 30 minutes. Table 1 shows the results of the different measurements.

Overall, the different capture requirements and delays are all in the expected range. However, the Azure Kinect shows an overall higher delay. We also observed high CPU and GPU usage under certain conditions (i.e., all other GPU processes being idle). As the Azure Kinect is still a relatively new sensor, we expect that the performance of the sensor might still improve in the future with further updates to the SDK (we did our tests with firmware version 1.6.108079014 and SDK 1.4.0). Furthermore, the Kinect v2 and RealSense™D415 included FGBG and thus show 10% more processing loads compared to the other capture methods. In conclusion, our current approach, including RDA, proofs beneficial for testing and rapid prototyping. However, in an operational environment, we expect to decrease the delay by at least 1-2 frames (30-60ms).

4.2 Depth Transmission Evaluation

We compared our depth-based conversion (GrayAVG) under different encoding and bandwidth conditions with two other algorithms (naive/simple, HSV). The "simple" algorithm is the most simple conversion based on direct depth to RGB mapping (and thus only serves as a minimal baseline). The HSV conversion is a reimplementation of the HoloTuber Kit[25] and further explained in this presentation[26]. Thus, the different mapping functions in the following:

$$Depth(simple) = (r + g + b)/3 * 4 + min\_distance \quad (9)$$

$$Depth(HSV) = h * 4 + min\_distance \quad (10)$$

$$Depth(GrayAVG) = (r + g + b) * 2 + min\_distance \quad (11)$$

[22] https://w3c.github.io/mediacapture-depth/

[23] https://videolat.org/
[24] https://github.com/ETSE-UV/RCM-UV
[25] https://github.com/TakashiYoshinaga/HoloTuberKit-for-AzureKinect
[26] https://speakerdeck.com/takashiyoshinaga/creating-holotuber-kit-hologram-visualization-with-rgb-d-image-streaming-via-youtube

In this test, we focus on encoding formats that are widely available in modern browsers and compatible with WebRTC [27]: VP8, VP9 and H.264. For the test, we used 3 clips of 10 seconds length with 30 fps (resulting into 300 frames in total for each video). Furthermore, we analysed the videos using the mean absolute difference (MD) to indicate the motion in the video stream:

low: user sitting, no movement, MD 44,92 (SD 13,51)
med: user sitting, some movement, MD51,34 (SD 27,27)
hi: user standing, high movement, MD61,52 (SD 33,34)

For each test condition, we made a full reference analysis measured with the peak signal-to-noise ratio (PSNR) of each frame and with 30 different encoding bitrates (0.1 Mbit to 3 Mbit). We encoded the video sequences with FFmpeg[28] (libvpx, libvpx-vp9 and libx264) with real-time encoding flags (deadline="realtime" for VP8/9 and tune="zerolatency" for H264), GOP size of 6 and yuv420 pixel format. Results of the 3 depth conversion methods are shown in Figure 7. While the HSV conversion is more robust on lower bandwidth constrains (up to 1,7Mbit), our approach (GrayAVG) outperforms the HSV on higher bit-rates that are common for real-time video conferencing (2-3Mbit). However, the main benefit of our approach is that it does not require any HSV or color mapping but works with a simple RGB conversion function that can be directly implemented in a 3D rendering shader (i.e. OpenGL GLSL). This is of particular importance for a web/browser implementation.

While Figures 7(a,b,c) include the overall average of all 3 video sequences, we also compared our conversion to the 3 sequences in more detail in Figure 7 (d). For this comparison, we excluded VP8 as it significantly underperformed on all methods before (see Figure 7 a,b,c). As to be expected, the videos with a lower motion achieve a higher PSNR under the same bandwidth and VP9 achieves a slightly higher PSNR as H.264 (overall average of 44,47 VP9 vs. 42,72 H.264). Thus, similar to other analysis of VP9. However, what is not shown in this graph is that still VP9 is not on pair with H.264 in regards to real-time encoding (delay and required CPU/GPU performance). In this regard, VP9 only offers a marginal PSNR increase from H.264.

4.3 P2P vs MCU for SocialVR (Simulation)

To evaluate our system and client performance, we compare MCU vs. peer-to-peer (p2p) transmission with different numbers of simulated users and two types of streams RGB with green chroma background (rendered as flat 2D sprite) and RGB + Depth (rendered as 3D point cloud). Users were simulated with the same stream used in the grayscale evaluation (med - user siting with some movement, MD 51,34 - SD: 27,27) and pre-encoded in H.264 with 2Mbit for the RGB stream and 4 MBit for the RGBD stream. We utilize H.264 under these bit-rates as it provides the best balance of performance and stream quality (based on our results from Section 4.2). When using VP9 we observed higher CPU usage in the client and lower frame rate throughput in the MCU. Furthermore, 2/4MBit aligns with the values that are negotiated by Chrome in the p2p case.

We run the server and MCU on a Microsoft® Azure cloud instance (Standard F8s_v2) with 8 vcpus (Intel® Xeon® Platinum 8168 @ 2.70GHz) and 16GB memory. The client runs in a Chrome browser on a VR laptop, MSI GS65-Stealth-Thin-8RF (Intel® Core™ i7-8750H, GeForce® GTX 1070 Max-Q and 32GB RAM).

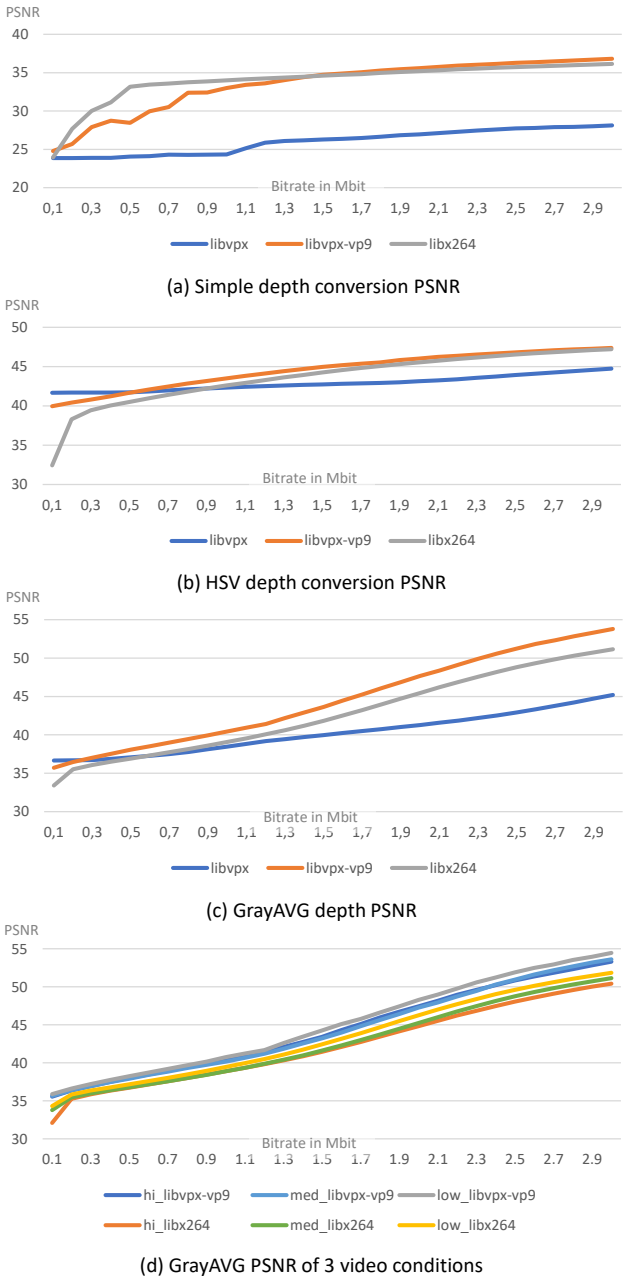

Figure 7: Different Depth conversion with VP8, VP9 and H.264 encoding and different bitrate (in MBit)

The measurement results of our tests can be seen in Figure 8a (for 2D RGB+Chroma) and Figure 8b, for 3D RGBD). Further, Figure 8c shows the performance of the MCU under the conditions tested. The p2p transmission shows a much steeper curve in terms of CPU resource usage than using an MCU, and both RGB and RGBD behave very similarly. This is as the overhead from multiple

[27] https://developer.mozilla.org/en-US/docs/Web/Media/Formats/WebRTC_codecs
[28] https://ffmpeg.org/

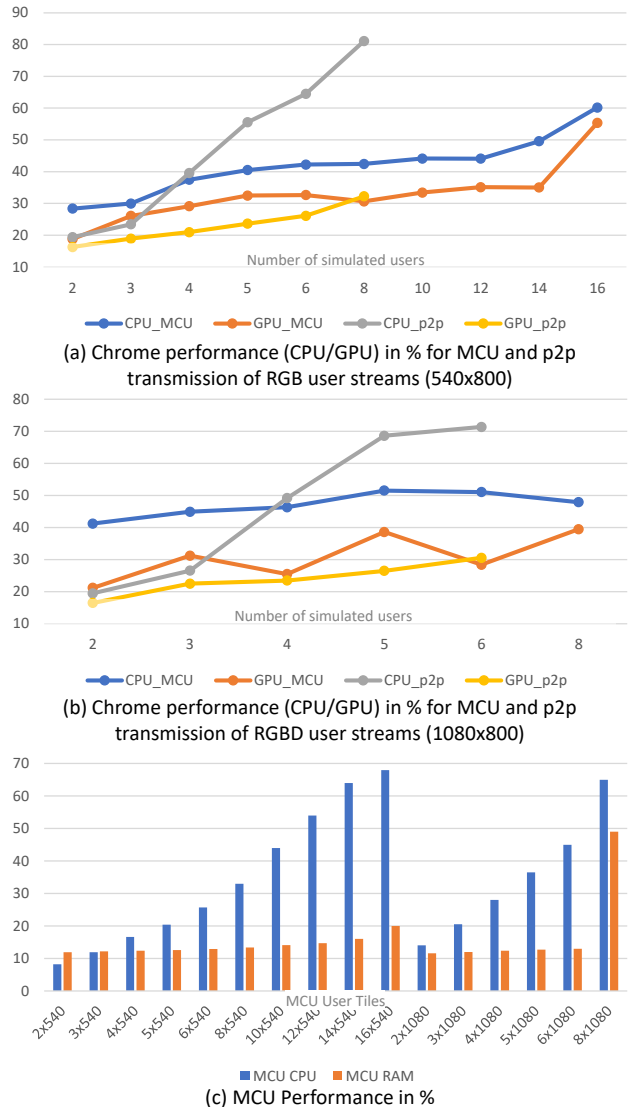

Figure 8: Web client (Chrome) and MCU performance

encoding and decoding streams is significant against an one stream upload and download in all MCU conditions. This said, p2p has clear advantages on a lower number of users, this is also to be expected as the MCU stream also transmits the uploaded stream back to each client and thus creates overhead on a lower number of users. Under all tests, the Chrome memory usage was kept in a reasonable boundary ranging from 473 MB (2ppl RGB) up to 1368 MB (16ppl RGB) on average. Thus, given our results, it is beneficial from 4 users on to follow an MCU methodology and the system becomes unstable and unusable (when adding capture modules and adding further processing needs when using an VR HMD) in p2p from 6 RGB and 5 RGBD clients. In our current implementation, however, the MCU can support a maximum of 16 RGB and 8 RGBD clients (see MCU performance in Figure 8,c). This said, as to be expected from a central transcoding entity, the improved performance on

Figure 9: 6 users in 3D performance test

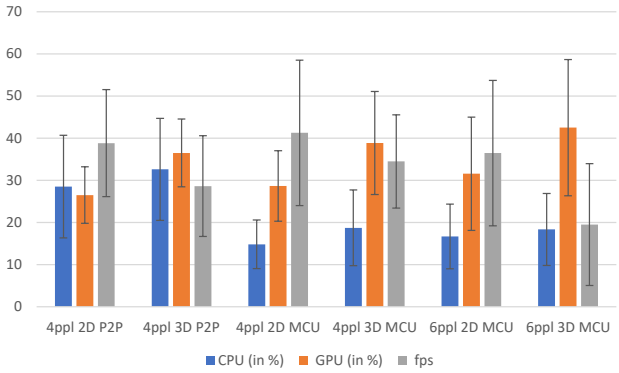

Figure 10: Chrome Browser Performance of user test

end clients comes at the price of added delay, the full end-to-end (capture to display) delays of the MCU vs p2p in the following (measured with VideoLat and >1000 samples):

RGB delay: p2p 396ms (SD 41) / MCU 564ms (SD 69)
RGBD delay: p2p 384ms (SD 44) / MCU 622ms (SD 68)

Overall, this shows that the MCU might add an significant overhead in terms of delay (but still in a considerable range of real-time communication). Further enhancements in the MCU like GPU accelerated encoding or tiled based compositions (that do not need transcoding) can further increase the number of maximum users and decrease the delay in the future.

4.4 Evaluation in realistic user setting

To further evaluate the simulation results, we conducted a set of user sessions in a realistic setting connecting 4 and 6 users (from Netherlands, France, and Germany). The details of the user end points used can be found in Table 2. We conducted 6 sessions with a duration of at least 25 minutes. An example of the user test is shown in Figure 9. Four user sessions were conducted between the nodes NL1, NL2, FR1 and FR2 in four conditions: P2P with 2D and 3D presentation, MCU with 2D and 3D representation. As the performance is not stable enough for a 6 user P2P condition, we only tested 2 conditions: MCU with 2D and 3D representation. The 6 user sessions were conducted between the nodes NL2, NL3, NL4,

Table 2: User Devices for performance evaluation

Name	CPU	GPU	Memory	Sensor	Location
NL1	Intel Core i7-8750H CPU @ 2.20GHz	NVIDIA GeForce GTX 1070 Max-Q	32 GB	Azure Kinect	NL, Amsterdam
NL2	Intel Core i7-8700 CPU @ 3.20GHz	NVIDIA GeForce RTX 2080	32 GB	Azure Kinect	NL, Katwijk
NL3	Intel Core i7-6700K CPU @ 4.00Ghz	NVIDIA GeForce GTX 980 Ti	16 GB	Kinect V2	NL, The Hague
NL4	Intel Core i7-7820HK CPU @ 2.9GHz	NVIDIA GeForce GTX 1070	24 GB	Azure Kinect	NL, Enschede
FR1	Intel Core i7-8750H CPU @ 2.20GHz	NVIDIA GeForce GTX 1070	16 GB	Kinect V2	FR, Rennes
FR2	Intel Core i7-47770K CPU @ 3.50GHz	NVIDIA GeForce GTX 1050 Ti	16 GB	Kinect V2	FR, Paris
DE1	Intel Core i7-8750H CPU @ 2.20GHz	NVIDIA GeForce GTX 1070 Max-Q	32 GB	Azure Kinect	DE, Berlin

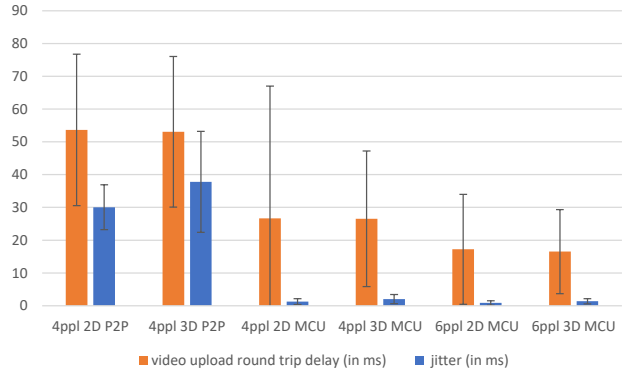

Figure 11: WebRTC video upload delay and jitter

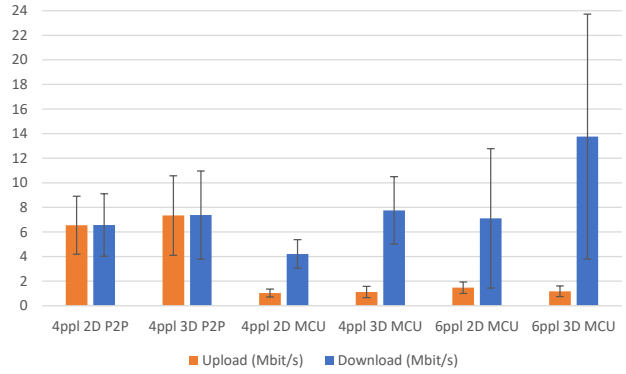

Figure 12: Video Upload & Download Traffic

FR1, FR2 and DE1. The results of these tests are presented in the following. CPU, GPU and network performance was measured with the same Resources Consumption Metrics (RCM) measurement tool [34] as in the simulation evaluation (section 4.3), the frame rate was measured via the Aframe stats and the WebRTC delay was measured via the Chrome WebRTC stats.

Figure 10, shows the overall performance average per condition (of all user end points) of the Chrome instance running the Web client in terms of CPU, GPU, and rendering frame rate. The values are slightly lower in terms of GPU/CPU usage than in the simulation due to more powerful end points with the same trend in MCU vs P2P resource usage: The MCU condition allows to reduce the CPU load at the cost of GPU usage. As CPU resources are more sparse and necessary for many more processes, this is particularly beneficial to support more simultaneous users and constant high-quality rendering. Important to note is that the frame rates are only indicative and not realistic for the rendering performance in an VR HMD. This is, we conducted the evaluation without a VR HMD to simplify the measurements and user interactions. Furthermore, the browser executes various optimisation strategies to balance the performance load with visual rendering quality. None of the users perceive stuttering or visual impact due to performance and the CPU/GPU load was low enough to allow higher frame rates in VR.

Figure 11, shows the overall average per condition (of all user end points) of the video upload round trip delay and jitter. These delays are additional to the overall delay values reported in the simulation evaluation (section 4.3). We can observe an overall higher delay and jitter for P2P transmission compared to central MCU transmission. However, one condition "4 users MCU with 2D representation" shows a high standard derivation as one client (FR1) observed higher delay values. This is to be expected in such a test (with a realistic and varying internet connection) and is in the normal boundaries of delay to expect for WebRTC transmission (and in the delay range acceptable for remote communication).

Figure 12, shows the overall average per condition (of all user end points) in network traffic. Our results reflect the main benefits of an central WebRTC approach (MCU) as the upload traffic is significantly decreased. This is as in the MCU condition the representation of a user is only uploaded once, while in the P2P condition the user representation has to be uploaded to each other end point. Overall, the MCU is capable to use network resources much more efficiently (based on the cost of central computation, see 4.3).

Overall, the user evaluation confirms the performance measures of the simulation in more realistic settings. The results show that we can achieve VR communication with similar network transfer rates and slightly more CPU/GPU resource usage compared to video conferencing solutions while being able to render users both in 2D and volumetric 3D.

5 DISCUSSION & FUTURE WORK

Our evaluation (section 4) shows that our proposed capture and depth to grayscale conversion for RGBD video data is suitable for real-time video transmission under bandwidth considerations typical for current video conferencing systems. However, for other bitrates (i.e., below 1.5 Mbit and above 3 Mbit), as well as pre-encoded

content, other solutions might result into a better visual quality. The real strength of our method is that it works on the RGB colour space and thus does not require direct access to encoding APIs (e.g., as YUV mapping would require), which makes it a suitable solution for web applications (due to its limitations in not revealing many underlying native media APIs do not allow many other RGB+Depth transmission techniques).

With VRComm, we extend common video conferencing optimisation solutions, i.e., an MCU to support RGBD video which is important to address the dedicated performance requirements of VR applications. In our current setup, we can support 16 2D (RGB + chroma) users and 8 3D (RGB + Depth) users, while the use of an MCU proves to be efficient from 4 users onwards. It is important to note here that our additions to the MCU in terms of handling our RGBD video format are fully compatible to any other MCU optimisation technique. For example, in more complex use cases one MCU might not be enough to cater for many geographical distributed users. Then a multi-MCU solution (or extended with multiple SFUs) could be deployed. Together with other optimisations like not rendering all users at the same time, this can increase scalability, visual quality and reduce delay.

Our evaluation of different capture configurations (section 4.1) shows reasonable CPU / GPU usage in all conditions. Furthermore, it offers a modular design to add different processing and image improvements like foreground background removal, HMD replacement, and image alignment calibration. One of the main bottle necks of our current approach is the connection to the browser, as this is currently done via a screen rendering and screen capture approach. In the future, this can be mitigated by direct access of the browser to the APIs of the depth sensor (i.e., via the W3C Media Capture Depth Stream Extensions) and a combination of processing within the browser client and in the network.

Our current system design and example VR experiences show that multi-user photorealistic immersive media applications are possible in real-time on the web. Allowing to use such applications without downloading and installing large software packages. However, current browser implementations still have some drawbacks regarding WebXR and other immersive media functionalities. One problem is that the performance can significantly vary between different browsers and different browser versions. Which can make it difficult to widely support your application with a constant high quality. For this paper, we only used Chrome as a browser client. We also tested a working solution of VRComm with other browsers like Firefox and Edge (i.g., utilizing VP8 or VP9 encoding for the WebRTC transmission). Currently, however, daily updates and changes in APIs can still break different aspects of the application and might make a widespread deployment cumbersome. Overall, video codec support, frame accurate synchronisation, underlying WebRTC functionality, and the connection of WebXR with different headsets (or the Steam[29] OpenXR[30] Api) still need to mature across browsers to offer a constant and high-quality user experience.

To get towards full VR and AR conferencing, the main goal is to create volumetric representation that fully blends into the AR or VR environment. For this, it is most essential to have good depth information (sending, transmission, and rendering). For now this can be done with solutions (as presented in this paper), but eventually can be done via new 3D media formats like MPEG V-PCC or G-PCC, however currently V-PCC is not real-time yet and GPCC might suffer from performance gaps (as there is no hardware acceleration for such codecs yet). Thus, the presented solution can be utilized right now for many VR and AR scenarios (as presented in [21]), while still offering multiple points for further improvement. Furthermore, the solution as presented in this paper does not exclude but adds to the current development of new 3D video formats like V-PCC. As real-time depth to 2D video mapping is also an aspect of V-PCC and solutions like the MCU can be utilized in the future to support scalability and large user groups for V-PCC.

Currently, VRComm can support up to 16 users in VR communication experiences, which is similar to direct interaction support in most video conferencing systems. To support more complex and large groups of users (100+ users, e.g., to support lectures, conferences, or galas), our future work will focus on moving more processing from the client into the cloud and edge. For example, new split rendering techniques can allow more lightweight and low powered end devices (e.g., AR glasses and mobile phones) to allow photorealistic XR applications with increased visual quality.

6 CONCLUSION

In this paper, we present a web-enabled video-based social VR framework that allows to rapidly develop, test, and evaluate photorealistic VR communication experiences. By combining video conferencing technology with social VR capabilities, we offer a new end-to-end pipeline (capture, processing, transmission, and rendering) to allow real-time shared immersive experiences and volumetric communication. Our novel transmission scheme for grayscale based depth information via 2D video proofs particularity usefully for web applications (that do not allow other depth conversion due to browser API limitations). Finally, the evaluation of our system in a simulation and realistic user setting shows that our solution utilizes processing (CPU / GPU) acceptable for modern (VR-ready) PCs and under network bandwidth constrains similar to existing video conferencing solutions. Still, more research is necessary to get all aspects of the technology ready (i.e., spatial computing, HMD replacement, enhanced quality transmission of real-time 2D/3D video data and system scalability for large sets of simultaneous users).

ACKNOWLEDGMENTS

This paper was partly funded by the European Commission as part of the H2020 program, under the grant agreement 762111 (VRTogether, http://vrtogether.eu/). In particular, we like to acknowledge the valuable contributions made by our project partner Viaccess-Orca (VO), part of the Orange Group, who provided a library to capture the WebRTC transmission stats. Furthermore, we like to thank Vincent Lepec, Jean-Baptiste Pigree and Guillaume Debeneix from VO who participated in the remote testing.

[29] https://store.steampowered.com/steamvr
[30] https://www.khronos.org/openxr/

REFERENCES

[1] D. S. Alexiadis, A. Chatzitofis, N. Zioulis, O. Zoidi, G. Louizis, D. Zarpalas, and P. Daras. 2017. An Integrated Platform for Live 3D Human Reconstruction and Motion Capturing. *IEEE Transactions on Circuits and Systems for Video Technology* 27, 4 (2017), 798–813.

[2] Alessandro Amirante, Tobia Castaldi, Lorenzo Miniero, and Simon Pietro Romano. 2015. Performance Analysis of the Janus WebRTC Gateway. In *Proceedings of the 1st Workshop on All-Web Real-Time Systems* (Bordeaux, France) *(AWeS '15)*. Association for Computing Machinery, New York, NY, USA, Article 4, 7 pages. https://doi.org/10.1145/2749215.2749223

[3] Jeremy N Bailenson and Nick Yee. 2005. Digital chameleons: Automatic assimilation of nonverbal gestures in immersive virtual environments. *Psychological science* 16, 10 (2005), 814–819.

[4] Steve Benford, Chris Greenhalgh, Tom Rodden, and James Pycock. 2001. Collaborative virtual environments. *Commun. ACM* 44, 7 (2001), 79–85.

[5] Jim Blascovich and Jeremy Bailenson. 2011. *Infinite reality: Avatars, eternal life, new worlds, and the dawn of the virtual revolution.* William Morrow & Co.

[6] Xavier P Burgos-Artizzu, Julien Fleureau, Olivier Dumas, Thierry Tapie, François LeClerc, and Nicolas Mollet. 2015. Real-time expression-sensitive hmd face reconstruction. In *SIGGRAPH Asia 2015 Technical Briefs*. 1–4.

[7] Gianluca Cernigliaro, Marc Martos, Mario Montagud, Amir Ansari, and Sergi Fernandez. 2020. PC-MCU: Point Cloud Multipoint Control Unit for Multi-User Holoconferencing Systems. In *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video* (Istanbul, Turkey) *(NOSSDAV '20)*. Association for Computing Machinery, New York, NY, USA, 47–53. https://doi.org/10.1145/3386290.3396936

[8] S. Cho, S. Kim, J. Lee, J. Ahn, and J. Han. 2020. Effects of volumetric capture avatars on social presence in immersive virtual environments. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 26–34.

[9] Carlos Coelho, JG Tichon, Trevor J Hine, GM Wallis, and Giuseppe Riva. 2006. Media presence and inner presence: the sense of presence in virtual reality technologies. In *From communication to presence: Cognition, emotions and culture towards the ultimate communicative experience*. IOS Press, Amsterdam, 25–45.

[10] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. 2015. High-Quality Streamable Free-Viewpoint Video. *ACM Trans. Graph.* 34, 4, Article 69 (July 2015), 13 pages. https://doi.org/10.1145/2766945

[11] Francesca De Simone, Jie Li, Henrique Galvan Debarba, Abdallah El Ali, Simon NB Gunkel, and Pablo Cesar. 2019. Watching videos together in social virtual reality: An experimental study on user's QoE. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 890–891.

[12] Sylvie Dijkstra-Soudarissanane, Karim El Assal, Simon Gunkel, Frank ter Haar, Rick Hindriks, Jan Willem Kleinrouweler, and Omar Niamut. 2019. Multi-sensor capture and network processing for virtual reality conferencing. In *Proceedings of the 10th ACM Multimedia Systems Conference*. 316–319.

[13] Sam Ekong, Christoph W. Borst, Jason Woodworth, and Terrence L. Chambers. 2016. Teacher-Student VR Telepresence with Networked Depth Camera Mesh and Heterogeneous Displays. In *Advances in Visual Computing*, George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Fatih Porikli, Sandra Skaff, Alireza Entezari, Jianyuan Min, Daisuke Iwai, Amela Sadagic, Carlos Scheidegger, and Tobias Isenberg (Eds.). Springer International Publishing, Cham, 246–258.

[14] Carmine Elvezio, Mengu Sukan, Ohan Oda, Steven Feiner, and Barbara Tversky. 2017. Remote Collaboration in AR and VR Using Virtual Replicas. In *ACM SIGGRAPH 2017 VR Village* (Los Angeles, California) *(SIGGRAPH '17)*. Association for Computing Machinery, New York, NY, USA, Article 13, 2 pages. https://doi.org/10.1145/3089269.3089281

[15] Christian Feldmann, Christopher Bulla, and Bastian Cellarius. 2013. Efficient stream-reassembling for video conferencing applications using tiles in HEVC. In *Proc. of International Conferences on Advances in Multimedia (MMEDIA)*. 130–135.

[16] Leonor Fermoselle, Simon Gunkel, Frank ter ter Haar, Sylvie Dijkstra-Soudarissanane, Alexander Toet, Omar Niamut, and Nanda van van der Stap. 2020. Let's Get in Touch! Adding Haptics to Social VR. In *ACM International Conference on Interactive Media Experiences* (Cornella, Barcelona, Spain) *(IMX '20)*. Association for Computing Machinery, New York, NY, USA, 174–179. https://doi.org/10.1145/3391614.3399396

[17] Christian Frueh, Avneesh Sud, and Vivek Kwatra. 2017. Headset removal for virtual and mixed reality. In *ACM SIGGRAPH 2017 Talks*. 1–2.

[18] Maia Garau, Mel Slater, Vinoba Vinayagamoorthy, Andrea Brogni, Anthony Steed, and M Angela Sasse. 2003. The impact of avatar realism and eye gaze control on perceived quality of communication in a shared immersive virtual environment. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 529–536.

[19] Juan C Granda, Pelayo Nuño, Francisco J Suárez, and Daniel F García. 2015. Overlay network based on WebRTC for interactive multimedia communications. In *2015 International Conference on computer, information and telecommunication systems (CITS)*. IEEE, 1–5.

[20] RA Grier, H Thiruvengada, SR Ellis, P Havig, KS Hale, and JG Hollands. 2012. Augmented Reality–implications toward virtual reality, human perception and performance. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 56. SAGE Publications Sage CA: Los Angeles, CA, 1351–1355.

[21] S.N.B Gunkel, Stokking H., T. De Koninck, and OA Niamut. 2019. Everyday Photo-Realistic Social VR: Communicate and Collaborate with an Enhanced Co-Presence and Immersion. In *Technical Papers International Broadcasting Convention (IBC)*.

[22] Simon NB Gunkel, Hans M Stokking, Martin J Prins, Nanda van der Stap, Frank B ter Haar, and Omar A Niamut. 2018. Virtual Reality Conferencing: Multi-user immersive VR experiences on the web. In *Proceedings of the 9th ACM Multimedia Systems Conference*. 498–501.

[23] Michitaka Hirose, Tetsuro Ogi, and Toshio Yamada. 1999. Integrating live video for immersive environments. *IEEE MultiMedia* 6, 3 (1999), 14–22.

[24] Jack Jansen. 2014. VideoLat. *Proceedings of the ACM International Conference on Multimedia - MM 14* (2014). https://doi.org/10.1145/2647868.2654891

[25] Jack Jansen, Shishir Subramanyam, Romain Bouqueau, Gianluca Cernigliaro, Marc Martos Cabré, Fernando Pérez, and Pablo Cesar. 2020. A pipeline for multiparty volumetric video conferencing: transmission of point clouds over low latency DASH. In *Proceedings of the 11th ACM Multimedia Systems Conference*. 341–344.

[26] Peter Kauff and Oliver Schreer. 2002. An immersive 3D video-conferencing system using shared virtual team user environments. In *Proceedings of the 4th international conference on Collaborative virtual environments*. 105–112.

[27] Gregorij Kurillo and Ruzena Bajcsy. 2013. 3D teleimmersion for collaboration and interaction of geographically distributed users. *Virtual Reality* 17, 1 (2013), 29–43.

[28] Jason Leigh, Thomas A DeFanti, A Johnson, Maxine Brown, and D Sandin. 1997. Global tele-immersion: Better than being there. In *Proceedings of ICAT*, Vol. 97. 3–5.

[29] Jason Leigh, Andrew E Johnson, Thomas A DeFanti, Maxine Brown, M Dastagir Ali, Stuart Bailey, Andy Banerjee, P Benerjee, Jim Chen, Kevin Curry, et al. 1999. A review of tele-immersive applications in the CAVE research network. In *Proceedings IEEE Virtual Reality (Cat. No. 99CB36316)*. IEEE, 180–187.

[30] Hao Li, Laura Trutoiu, Kyle Olszewski, Lingyu Wei, Tristan Trutna, Pei-Lun Hsieh, Aaron Nicholls, and Chongyang Ma. 2015. Facial performance sensing head-mounted display. *ACM Transactions on Graphics (ToG)* 34, 4 (2015), 1–9.

[31] Jyh-Ming Lien, Gregorij Kurillo, and Ruzena Bajcsy. 2010. Multi-camera tele-immersion system with real-time model driven data compression. *The Visual Computer* 26, 1 (2010), 3.

[32] Yunpeng Liu, Stephan Beck, Renfang Wang, Jin Li, Huixia Xu, Shijie Yao, Xiaopeng Tong, and Bernd Froehlich. 2015. Hybrid Lossless-Lossy Compression for Real-Time Depth-Sensor Streams in 3D Telepresence Applications. 442–452. https://doi.org/10.1007/978-3-319-24075-6_43

[33] R. Mekuria, K. Blom, and P. Cesar. 2017. Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video. *IEEE Transactions on Circuits and Systems for Video Technology* 27, 4 (2017), 828–842.

[34] Mario Montagud, Juan Antonio De Rus, Rafael Fayos-Jordan, Miguel Garcia-Pineda, and Jaume Segura-Garcia. 2020. Open-Source Software Tools for Measuring Resources Consumption and DASH Metrics. In *Proceedings of the 11th ACM Multimedia Systems Conference* (Istanbul, Turkey) *(MMSys '20)*. Association for Computing Machinery, New York, NY, USA, 261–266. https://doi.org/10.1145/3339825.3394931

[35] Nels Numan, Frank Haar, and Pablo Cesar. 2021. Generative RGB-D Face Completion for Head-Mounted Display Removal. In *2021 IEEE Virtual Humans and Crowds for Immersive Environments (VHCIE)*. IEEE, IEEE.

[36] Pelayo Nuño, Francisco G Bulnes, Juan C Granda, Francisco J Suárez, and Daniel F García. 2018. A Scalable WebRTC Platform based on Open Technologies. In *2018 International Conference on Computer, Information and Telecommunication Systems (CITS)*. IEEE, 1–5.

[37] Tetsuro Ogi, Toshio Yamada, Ken Tamagawa, Makoto Kano, and Michitaka Hirose. 2001. Immersive telecommunication using stereo video avatar. In *Proceedings IEEE Virtual Reality 2001*. IEEE, 45–51.

[38] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. 2016. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 741–754.

[39] J. Outlaw and B. Duckles. 2017. Why Woman Don't Like Social Virtual Reality. https://extendedmind.io/social-vr

[40] J. Park, P. A. Chou, and J. Hwang. 2019. Rate-Utility Optimized Streaming of Volumetric Media for Augmented Reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (2019), 149–162.

[41] Fabrizio Pece, Jan Kautz, and Tim Weyrich. 2011. Adapting standard video codecs for depth streaming.. In *EGVE/EuroVR*. 59–66.

[42] Tomislav Pejsa, Julian Kantor, Hrvoje Benko, Eyal Ofek, and Andrew Wilson. 2016. Room2room: Enabling life-size telepresence in a projected augmented reality environment. In *Proceedings of the 19th ACM conference on computer-supported cooperative work & social computing*. 1716–1725.

[43] Stefano Petrangeli, Dries Pauwels, Jeroen van der Hooft, Tim Wauters, Filip De Turck, and Jürgen Slowack. 2018. Improving quality and scalability of WebRTC video collaboration applications. In *Proceedings of the 9th ACM Multimedia Systems Conference*. 533–536.

[44] O. Schreer, I. Feldmann, S. Renault, M. Zepp, M. Worchel, P. Eisert, and P. Kauff. 2019. Capture and 3D Video Processing of Volumetric Video. In *2019 IEEE International Conference on Image Processing (ICIP)*. 4310–4314.

[45] Ralph Schroeder. 2012. *The social life of avatars: Presence and interaction in shared virtual environments*. Springer Science & Business Media.

[46] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko. 2019. Emerging MPEG Standards for Point Cloud Compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (2019), 133–148. https://doi.org/10.1109/JETCAS.2018.2885981

[47] Renata M. Sheppard, Mahsa Kamali, Raoul Rivas, Morihiko Tamai, Zhenyu Yang, Wanmin Wu, and Klara Nahrstedt. 2008. Advancing Interactive Collaborative Mediums through Tele-Immersive Dance (TED): A Symbiotic Creativity and Design Environment for Art and Computer Science. In *Proceedings of the 16th ACM International Conference on Multimedia* (Vancouver, British Columbia, Canada) *(MM '08)*. Association for Computing Machinery, New York, NY, USA, 579–588. https://doi.org/10.1145/1459359.1459437

[48] Mel Slater. 2018. Immersion and the illusion of presence in virtual reality. *British Journal of Psychology* 109, 3 (2018), 431–433.

[49] S. Subramanyam, J. Li, I. Viola, and P. Cesar. 2020. Comparing the Quality of Highly Realistic Digital Humans in 3DoF and 6DoF: A Volumetric Video Case Study. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 127–136.

[50] Masayuki Takemura and Yuichi Ohta. 2005. Generating High-Definition Facial Video for Shared Mixed Reality.. In *MVA*. 422–425.

[51] Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. 2017. Demo of FaceVR: real-time facial reenactment and eye gaze control in virtual reality. In *ACM SIGGRAPH 2017 Emerging Technologies*. 1–2.

[52] J. v. d. Hooft, M. T. Vega, T. Wauters, C. Timmerer, A. C. Begen, F. D. Turck, and R. Schatz. 2020. From Capturing to Rendering: Volumetric Media Delivery with Six Degrees of Freedom. *IEEE Communications Magazine* 58, 10 (2020), 49–55. https://doi.org/10.1109/MCOM.001.2000242

[53] M. Westerlund and S. Wenger. 2015. *RTP Topologies*. RFC 7667. RFC Editor. https://tools.ietf.org/html/rfc7667

[54] Andrew D. Wilson and Hrvoje Benko. 2016. Projected Augmented Reality with the RoomAlive Toolkit. In *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces* (Niagara Falls, Ontario, Canada) *(ISS '16)*. Association for Computing Machinery, New York, NY, USA, 517–520. https://doi.org/10.1145/2992154.2996362

[55] Zhenyu Yang, Bin Yu, Klara Nahrstedt, and Ruzena Bajscy. 2006. A multi-stream adaptation framework for bandwidth management in 3D tele-immersion. In *Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video*. 1–6.

[56] Zhenyu Yang, K. Nahrstedt, Yi Cui, Bin Yu, Jin Liang, Sang-hack Jung, and R. Bajscy. 2005. TEEVE: the next generation architecture for tele-immersive environments. In *Seventh IEEE International Symposium on Multimedia (ISM'05)*. 8 pp.–. https://doi.org/10.1109/ISM.2005.113

[57] N. Zioulis, D. Alexiadis, A. Doumanoglou, G. Louizis, K. Apostolakis, D. Zarpalas, and P. Daras. 2016. 3D tele-immersion platform for interactive immersive experiences between remote users. In *2016 IEEE International Conference on Image Processing (ICIP)*. 365–369.