frontiers | Frontiers in Signal Processing

TYPE Original Research
PUBLISHED 22 May 2023
DOI 10.3389/frsip.2023.1139897

OPEN ACCESS

EDITED BY
Irene Viola,
Centrum Wiskunde and Informatica, Netherlands

REVIEWED BY
Mihai Mitrea,
Télécom SudParis, France
Jesús Gutiérrez,
Universidad Politécnica de Madrid, Spain

*CORRESPONDENCE
Simon N. B. Gunkel,
simon.gunkel@tno.nl

RECEIVED 07 January 2023
ACCEPTED 17 April 2023
PUBLISHED 22 May 2023

CITATION
Gunkel SNB, Dijkstra-Soudarissanane S, Stokking HM and Niamut OA (2023), From 2D to 3D video conferencing: modular RGB-D capture and reconstruction for interactive natural user representations in immersive extended reality (XR) communication.
Front. Sig. Proc. 3:1139897.
doi: 10.3389/frsip.2023.1139897

From 2D to 3D video conferencing: modular RGB-D capture and reconstruction for interactive natural user representations in immersive extended reality (XR) communication

Simon N. B. Gunkel*, Sylvie Dijkstra-Soudarissanane, Hans M. Stokking and Omar A. Niamut

The Netherlands Organisation for Applied Scientific Research (TNO), Den Haag, Netherlands

With recent advancements in Virtual Reality (VR) and Augmented Reality (AR) hardware, many new immersive Extended Reality (XR) applications and services arose. One challenge that remains is to solve the social isolation often felt in these extended reality experiences and to enable a natural multi-user communication with high Social Presence. While a multitude of solutions exist to address this issue with computer-generated "artificial" avatars (based on pre-rendered 3D models), this form of user representation might not be sufficient for conveying a sense of co-presence for many use cases. In particular, for personal communication (for example, with family, doctor, or sales representatives) or for applications requiring photorealistic rendering. One alternative solution is to capture users (and objects) with the help of RGBD sensors to allow real-time photorealistic representations of users. In this paper, we present a complete and modular RGBD capture application and outline the different steps needed to utilize RGBD as means of photorealistic 3D user representations. We outline different capture modalities, as well as individual functional processing blocks, with its advantages and disadvantages. We evaluate our approach in two ways, a technical evaluation of the operation of the different modules and two small-scale user evaluations within integrated applications. The integrated applications present the use of the modular RGBD capture in both augmented reality and virtual reality communication application use cases, tested in realistic real-world settings. Our examples show that the proposed modular capture and reconstruction pipeline allows for easy evaluation and extension of each step of the processing pipeline. Furthermore, it allows parallel code execution, keeping performance overhead and delay low. Finally, our proposed methods show that an integration of 3D photorealistic user representations into existing video communication transmission systems is feasible and allows for new immersive extended reality applications.

KEYWORDS

immersive communication, augmented reality, virtual reality, extended reality, XR, RGBD, human 3D representation

1 Introduction and methodology

Although 2D video conferencing has gained increased popularity for everyday communication, it may not be ideal for all use cases and may lead to exhaustion, as well as other problems related to "zoom fatigue" (Nesher Shoshan and Wehrt, 2022). One possible solution to improve remote communication systems is to introduce 3D spatial cues in immersive extended reality (XR) applications. Immersive communication applications built on XR technology have the promise of high immersion, presence, and more natural user interactions (Stanney et al., 2021). Eventually, the goal of immersive communication systems is to maximize social presence between users in remote communication. As described by Lombard and Ditton (Lombard and Ditton, 1997), users ultimately feel the illusion of non-mediation. This is achieved in part by transporting a user virtually to the other user. We firmly believe that at least one other part of it is giving users a feeling of 'the same place,' of local togetherness, of physical proximity. Although current virtual reality (VR) communication systems are maturing, they still face significant drawbacks. One main complication is the capture and rendering of users in photorealistic quality. Photorealistic representations can be a contributing factor in allowing the high level of immersion and social presence and might be required in any photorealistic XR environment, simply to blend users into the virtual or augmented world. However, photorealistic capture is complex due to stringent real-time requirements and significant computational demands. Furthermore, the advantages and disadvantages of different processing and analysis strategies are not always clear. This raises the need for a better understanding of the technical steps and their impact on any resource usage in the system and the user's device.

To address this gap, this work outlines the main technical building blocks in a modular capture application to enable immersive communication via RGBD (color and depth) user capture and rendering. Our approach is based on existing real-time streaming components and infrastructure and was integrated into two XR communication systems. Thus, we propose a modular capture application, paired with multiple 3D rendering strategies, and integrated into a commercial communication toolkit. We evaluated our approach with users in their daily work environment. This evaluation assesses the practical usability of the system, our technical approach, and the quality of communication/interaction (i.e., Social Presence). The research in this paper was based on the following research questions and hypotheses:

RQ1. What are the advantages and disadvantages of different RGBD capture modalities, individual processing steps, and their resource needs?
H1.1 RGBD capture and processing is possible within a latency that is acceptable for real-time communication (< 500 ms glass-to-glass).
H1.2 RGBD capture and processing is possible with a performance overhead that is acceptable for any modern PC/Laptop.
RQ2. How can RGBD data be utilized for 3D photorealistic user representations, and what is its rendering resource overhead?
H2.1 RGBD data allow 3D rendering of user representations on different devices and with a performance overhead acceptable for low powered stand alone hardware.
H2.2 Different GPU shader optimizations allow for higher rendering quality without significant processing overhead.
RQ3. Do RGBD-based user representations result in a high social presence for remote communication?

While previous work outlined the system and transmission aspects of RGBD-based 3D conferencing (Gunkel et al., 2021) this paper's focus is on the capture and rendering technologies and the evaluation thereof. In order to address our research questions, we followed an experimental design under controlled conditions for RQ1 and RQ2, which offers technical guidance of different RGBD capture and processing approaches and evaluate them in detail with technical measures. Additionally, we conducted two small-scale case studies (based on RQ3) within two realistic deployments of the capture application into complete immersive communication experiences. The paper offers the following contributions:

C1. Design considerations and implementation aspects for real-time modular capture of RGBD-based photorealistic 3D user representation into real-world systems and hardware.
C2. Technical evaluation of individual modules and strategies for RGBD-based capture and rendering.
C3. Small-scale user evaluation of both an AR and a VR communication application that integrates our modular capture approach.

2 Related work

Volumetric video is recognized as a crucial technique for creating immersive AR and VR experiences. An ongoing area of research is the capture, encoding and transmission of volumetric video formats such as point clouds and models (Collet et al., 2015; Alexiadis et al., 2017; Mekuria et al., 2017; Park et al., 2019; Schreer et al., 2019). Volumetric videos, in particular, improve the quality of experience and social presence (Cho et al., 2020; Subramanyam et al., 2020) for remote telepresence and immersive communication (Orts-Escolano et al., 2016; Zioulis et al., 2016). However, most current volumetric video formats, such as video-based point cloud coding (V-PCC) (Schwarz et al., 2019) and geometry-based point cloud coding (G-PCC) (Mekuria et al., 2017)) require a lot of computing power, making them currently difficult to implement in realistic real-time pipelines. One alternative is to convert the depth information into a 2D image format that can be encoded and transmitted via existing image/video compression techniques. Pece et al. (2011) introduced a simple video codec for depth streaming, while (Liu et al., 2015; Ekong et al., 2016; Gunkel et al., 2021) considered a more sophisticated conversion of depth information into grayscale images. Furthermore, the use of RGBD images for 3D user rendering is a well-established concept in depth-image-based rendering (DIBR). DIBR was introduced for 3D TV (Fehn, 2004) but can also be applied for complex texture synthesis (Ndjiki-Nya et al., 2011). However, 3D TV was never widely deployed and, to our knowledge, DIBR was not applied in the context of a communication system. Furthermore, AR and VR applications require spatial computing, which is the ability to recognize the environment, the user, and the things surrounding the user (see

(Grier et al., 2012; Elvezio et al., 2017)). In terms of communication, this means that in order to express effective remote interactions, the user's actions must be accurately mirrored in the user's representation. Spatial processing tasks are complex and complicated to test; thus, the modular capture application presented in this paper is designed for easy extension and reconfiguration to simplify spatial processing prototyping and testing, as well as to form a basic building block for immersive communication applications.

There are some solutions to allow immersive communication through Free Viewpoint Video (FVV) (Carballeira et al., 2021; Rasmuson et al., 2021). These approaches run on consumer-grade hardware and existing encoding strategies. The ease of deployment is still hindered by complex calibration and preprocessing steps. There are further solutions based on photorealistic social VR communication (Gunkel et al., 2021; Langa et al., 2021; Reimat et al., 2022). For example, (Prins et al., 2018), introduces a web-based framework for communicative and Social VR experiences. Further, (Gunkel et al., 2019b), outlines multiple use cases of Social VR and the benefit to user experience. These experiences are also considered suitable for real-world situations, such as stand-up meetings (Gunkel et al., 2019a). In a different study, Francesca et al. (De Simone et al., 2019) found that face-to-face and photorealistic Social VR show no statistical differences in terms of interaction and social connectivity. Furthermore, immersive communication compared to 2D video conferencing is theorized to differ in its ability to provide social context-related cue transfer, particularly involving body posture, gestures, movement, and eye contact. Furthermore, the way in which presence is typically experienced in VR (Coelho et al., 2006) can also affect communication and interpersonal relationships.

Engaging in remote collaboration benefits organizations, as well as their employees, by offering an advantage over less flexible competitors and supporting employee wellbeing (Pinsonneault and Boisvert, 2001). However, virtual teams have also been shown to be prone to various problems. Thompson and Coovert (Thompson et al., 2006) propose a categorization of the main issues faced by virtual teams, consisting of: 1) decreased communication quality; 2) ineffective interpersonal relationships; and 3) lack of awareness of the work of each other. An important determinant of the decrease in communication quality and the ineffective development of interpersonal relationships appears to be the generally restricted flow of social information between virtual team members. Social information in this context is commonly understood as social context cues and refers to non-verbal cues (e.g., gestures), paraverbal cues (e.g., tone of voice), status and interpersonal cues (e.g., age), and characteristics of physical surroundings (e.g., office size) (Straus, 1997). In fact, research shows that social context cues help regulate interpersonal interaction and information exchange (Straus, 1997), and the development of trust in teams (Cascio, 2000). In turn, this supports communication and cooperation (Aubert and Kelsey, 2003; Priest et al., 2006), as well as the development of interpersonal relationships (Wilson et al., 2006). In addition, social context cues show positive correlations with levels of efficiency (Boyle et al., 1994), perceived communication quality, and satisfaction (Sellen, 1995). As such, it can be argued that the more a medium affords the transfer of social context cues, the more it supports the quality of communication and the development of effective interpersonal relationships.

3 Materials and method–RGBD-based 3D representations

In this section, we outline the technical pipeline and the generic modular capture application (see Figure 1). Most immersive communication systems can be simplified into three components: capture, client, and central system components (such as data transmission). It is important to note here that there is a causal relationship between any applied capture method and the reconstruction or rendering of a user. With a focus on RGBD as a format, this section explains the main aspects and modules of the full pipeline to allow capture and reconstruction of users in 3D. Further examples of actual application implementations are presented in Section 4.2.

3.1 Design considerations

One of our main requirements is to have a capture and render pipeline that is completely modular to make different individual process blocks easy to measure and to extend. This includes an easy extension to use different capture devices and an open API to access the final capture in any application. All with the main purpose to allow photorealistic 3D user representations in real-time communication use cases. Therefore, we consider the following design criteria (based on our four hypotheses):

3.1.1 Capture latency (H1.1)

One of the most important factors for any real-time communication system is delay. The acceptable end-to-end (capture-to-display) for video conferencing is still debated; however, an acceptable value was identified as less than 500 ms for group discussions (Schmitt et al., 2014). Thus a latency of up to 500 ms should be acceptable for most people. Furthermore, latency values between 300 and 500 ms are in line with many existing video conferencing systems (Yu et al., 2014) and immersive communication systems (Roberts et al., 2009; Gunkel et al., 2021). To the best of our knowledge, no research has been conducted on the user and communication perception of latency in photorealistic 3D immersive communication. Therefore, in this article, we will focus on technical values and expect more research on user-side aspects in the future.

3.1.2 Capture performance (H1.2)

Essentially, the capture system has to be accessible and easy to deploy. Therefore, it should operate under performance constraints (CPU, GPU, and memory utilization) that are acceptable for any modern laptop/PC.

3.1.3 Render interoperability (H2.1)

The RGBD-based approach presented in the paper presents challenges not only in the capture of such data but also in the reconstruction and rendering. Any rendering approach presented

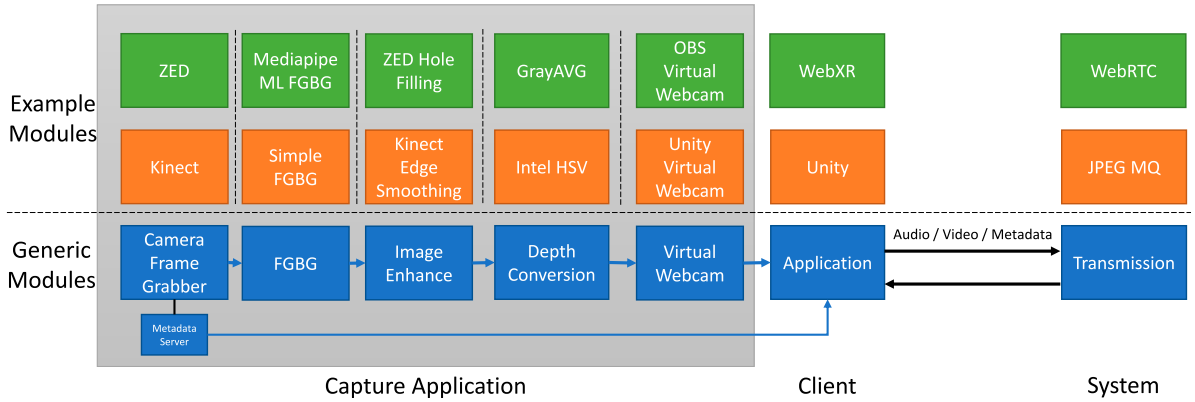

FIGURE 1
Modular capture architecture.

has to be adoptable for different render engines (e.g., WebGL and Unity) and under performance constraints that allow low-powered devices (i.e., AR glasses) to reconstruct and render the 3D user representations.

3.1.4 Render optimizations (H2.2)

When it comes to the quality of the user representation, the capture quality is a very important factor. However, due to expected lossy compression (otherwise, the data rates would not be suitable for any internet-based system), many more artifacts and distortions may be injected into the representation. Thus creating a need to optimize and clean the image not only at the capturing stage but also during receiving or rendering. Furthermore, any performance overhead resulting from such optimizations should have minimal impact on the rendering device.

3.2 Modular capture and processing

In this paper, we introduce a modular capture and processing application that was fully developed in Python. Python was chosen because it offers easy code development and offers a rich set of modules for image processing and machine learning. Furthermore, most RGBD sensors SDKs offer simple wrappers in Python (like the Kinect Azure and ZED used as exampled in this paper), which offer similar performance as the native C solutions. Also, utilizing OpenCV[1] and Numba[2], many image processing tasks are implemented with hardware acceleration and underlying C bindings, thus keeping the performance overhead low. Although in a production environment it would be more desirable to have a complete native solution, our approach offers an excellent entry point for prototyping, experimentation, and research. This module can be easily replaced for new research needs or extended with new processing functions such as complex spatial user and environment analysis. Finally, all modules are fully interchangeable with the ZED 2i, K4A, and any future sensor capture input, but may lead to undesirable or suboptimal image quality.

The capture application is divided into five generic modules (see Figure 1). For optimal performance and parallel processing, each module runs in its own thread loop. Information between modules is exchanged via simple queues that ensure thread safety and image ordering. All modules are glued together via a central governing thread that also is the main tread to interface with OpenCV image rendering (displaying any required final or debug output in a window). In addition to the five modules, a Metadata Server may offer additional metadata information from the capture module to applications on the same computer system. The five generic capture modules, as shown in Figure 1, are explained in more detail (including example implementations of these modules) below.

3.2.1 Camera frame grabber

The first module in the capture chain is a frame grabber to allow simple access to different capture sensors. The main function of this module is to extrapolate any dedicated capture API and to supply a steady real-time flow of synchronized RGB color and depth image frames. Currently, we implemented two examples of this module to support the Kinect Azure sensor (based on[3]) and ZED 2i (based on[4]). In addition to images, this module also offers generic metadata of the camera intrinsic (i.e., resolution and focal length), which are important for the correct rendering of the image into the 3D plane. The implementation of the two sensor modules currently offers a high- and low-capture mode with an output resolution of 1024 × 1024 pixels (high) and 512 × 512 pixels (low). The image capture is internally cropped, and all metadata is adjusted accordingly. This is, however, not a limitation of our approach but a deliberate decision on order to optimize different parts of the end-to-end chain, as an image with a resolution power of 2 can be handled more performantly in most image/video compression and

1 https://opencv.org/.
2 https://numba.pydata.org/.
3 https://github.com/etiennedub/pyk4a.
4 https://www.stereolabs.com/docs/app-development/python/install/.

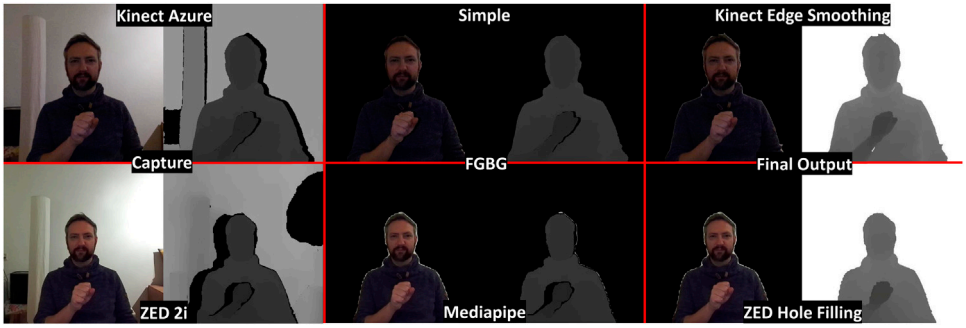

FIGURE 2
Capture example–Capture and FGBG and final images (TOP row = Kinect; Bottom row ZED).

rendering engines. This is specifically the case for the image mapping in WebGL rendering (as internally any image will be scaled to a power of two for texture-to-shader mapping). Furthermore, the capture rate can be fully adjusted (based on sensor capabilities). For the remainder of the paper, we refer to the capture rate of 15fps. In the following, we explain the main differences between the two sensors.

3.2.1.1 Kinect azure (K4A)[5]

The K4A sensor is a Time-of-Flight (ToF) sensor. In addition to a hi-resolution color sensor, it utilizes two near-infrared (NIR) laser diodes enabling near and wide field-of-view (FoV) depth by measuring the time the (modulated) infrared laser beam takes from sending to sensor reading. This gives accurate depth approximations with a limited resolution (640 × 576 px in the near field of view, NFOV, mode). This resolution is still suitable for human capture (in a range of 3 m). It is important to note that the depth image is also significant scaled with a complex mapping to the color pixels, depending on the color capture resolution. More details on the accuracy of the Kinect Azure are presented in (Kurillo et al., 2022). Although the depth capture quality is generally high, the raw capture may still include different types of distortion. An example of a raw human capture can be seen in Figure 2 (image on top left). For example, misalignment between the infrared (IR) sensor (to measure depth) and the color image can create a halo effect, resulting in wrong or no depth values in certain sections of the image. This problem is particularly undesirable around the hand of the user (i.e., in the raw image part of the body, even though color information is present, would be ignored in the 3D rendering). One potential disadvantage of the K4A is that it may cause IR interference with any other devices that may rely on IR device tracking (like the HoloLens 2 AR glasses or the SteamVR Lighthouse tracking system).

3.2.1.2 ZED 2i[6]

The ZED 2i sensor is a stereo camera with two high resolution color cameras in a predefined fixed setup. With the disparity of the 2 synchronized video streams and (CUDA, hardware accelerated) stereo matching, the sensor provides both color and depth information. As the full image is used for the depth estimation step, the depth image completely resembles the resolutions of the color image. This can also be accounted for in case of high resulting capture (e.g., 2 k or 4 k) even though some depth estimation steps might be internally scaled by the SDK (to allow real-time capture with low performance overhead). More details on the accuracy of the ZED 2i sensor (and similar stereo matching depth sensor devices) are presented in (Tadic et al., 2022). Furthermore, the ZED SDK offers two other important parameters related to depth capture: depth_mode and sensing_mode. In our current implementation, we use depth_mode = DEPTH_MODE.QUALITY. This was determined to be the best option. The alternative mode DEPTH_MODE.ULTRA seems quite pixelated and has a less consistent depth granularity. As sensing_mode we use SENSING_MODE.STANDARD. Compared to the alternative SENSING_MODE.FILL, standard sensing produces a more consistent depth image. The SENSING_MODE.FILL option's main drawback is that it generates a lot of overlap and causes objects and body parts to blend into the backdrop, leading to inconsistent depth sensing findings. Both depth_mode and sensing_mode are fully configurable as parameters in the ZED capture module. An example of a raw human capture can be seen in Figure 2 (bottom left image). Compared to the K4A the ZED offers a more uniform depth image but may result into multiple distortions based on the limitations of the stereo matching approach. First of all, the ZED depth image quality often appears more noisy (compared to K4A). Furthermore, uniformly colored surfaces cannot be correctly matched and cannot be represented as depth (see the black blob on the top right of the raw ZED capture, Figure 2). This is not necessarily an issue for humans (as body parts or cloth is rather uniformly colored, particularly in common lighting conditions). Finally, we can observe similar image misalignment distortions (see users hand, Figure 2) in the ZED capture as in the K4A capture. In addition, these misalignment distortions often lead to distortions in the outline of the person as well. However, one clear benefit of the ZED sensor is that it does not produce any IR interference with any other devices that may rely on IR device tracking (such as the HoloLens 2 AR glasses or the SteamVR Lighthouse tracking system). Furthermore, please note that

5 https://azure.microsoft.com/en-us/products/kinect-dk/.

6 https://www.stereolabs.com/zed-2i/.

currently due to CUDA processing requirements the ZED 2i sensor is only supported on a PC with an NVIDIA graphics card.

3.2.2 Foreground-background extraction (FGBG)

This module's main goal is to enhance the quality of the image while doing real-time foreground-background extraction (FGBG). This is a crucial step, since, when capturing users, we are only concerned with the user's representation and not their background. Additionally, doing so enables us to visually render the users' representation into the XR environment (i.e., virtual environment for VR or augmented into the real environment for AR).

For the K4A image capture we implemented a simple threshold based FGBG method, removing any pixels out of the threshold. This is done by simply overwriting any values out of the threshold with 0 both for the depth and color. This is we currently allow pixels with a depth value in the range of 500–2,000 mm away from the camera. This is based on the following assumptions:

- user will be at least 50 cm away from the camera
- main body is around 1.5 m away from the sensor
- we only encode a depth range of (approx.) 1.5 m

As we do not consider the depth images captured by the ZED 2i sensor as reliable as from the K4A, a simple FGBG threshold might not be good enough to provide a high quality reconstruction. Therefore we implemented an enhanced FGBG module using a machine learning approach for human body segmentation (based on Mediapipe[7]). This offers a fine grain body segmentation map (with reasonable performance overhead due to Mediapipe's hardware acceleration, running on the GPU). Example images of the FGBG modules can be seen in Figure 2, simple FGBG based on K4A capture (top middle) and enhanced Mediapipe FGBG based on ZED 2i capture (bottom middle).

3.2.3 Other image enhancements

After the FGBG processing further image enhancements are desirable to for a high quality user representation. This is we like to clean the image from any distortions and holes. As these holes and image imperfections are rather different for the raw images of the two examples sensor implementations presented in this paper (K4A and ZED) we implemented two very distinctive enhancement modules. Enhancements and further spatial processing blocks could easily be introduced in the future.

3.2.3.1 ZED hole filling

This module has the main aim to improve the depth image; that is: A. fill small holes and remove imperfections and B. fill any big holes that might be present due do capture misalignment. This image enhancements rely on the (Mediapipe) image segmentation of the previous FGBG module and encompass the following steps:

- Step 0: As a basis for this image enhancement the Mediapipe segmentation from the FGBG module already removed unnecessary pixels from the color and depth image.
- Step 1: clean the edges with cv2.dialate followed by cv2.erode (both with kernel 5 and 2 iterations)
- Step 2: build a mask for all holes (empty depth values that in the segmentation are detected as person)
- Step 3: build a temporary "filler depth image" by copying the depth image and applying cv2.erode (kernel size 8 and 4 iterations) followed by cv2.dilate (kernel size 8 and 8 iterations), this will fill all big holes but also fill across the segmentation map
- Step 4: finally we combine the depth image with the "filler depth image" (from the previous step) by only replacing missing values in the depth image

The resulting depth image fills the complete segmentation and thus offers a value for each color pixel. An example output can be seen in Figure 2 (bottom right). Note that the Mediapipe segmentation may result into a halo around the cutout (part of the background is detected as person). This might result into less sharp edges than the final image of the K4A sensor but often result into better rendering of hair that might be cutout in the K4A image. Further, note that the edge of the image will be further optimized in the rendering of the image.

3.2.3.2 Kinect edge smoothing

This module has the main aim to improve the misalignment of the K4A image on the edges and of any foreground objects (including the users hand) in particular. This step is based on the simple FGBG and thus only relies on OpenCV[1] functions. The different enhancement steps are explained in the following:

- Step 1: first we need to resize the image in case of high resolution (this means the optimization will always we done in 512 pixel to ensure real-time low performance execution)
- Step 2: create image outline with cv2.Canny
- Step 3: match edges from cv2.Canny algorithm with depth values
- Step 4: fill edges and create contours with cv2.findContours
- Step 5: find biggest contour (this is our person/including holes)
- Step 6: identify a depth background threshold averaging the depth image - 10 cm (we asume that in K4A the foreground object, like a hand, will result into holes in the back of the image, so we like to will the background with background depth values)
- Step 7: create a temporary background depth image by coping the depth image and removing all depth values smaller than the background threshold
- Step 8: now we loop 5 times, creating new depth values with a cv2.dilate (kernel 5) and only replace the new values into the temporary background depth image (Note: this works significantly different than cv2.dilate iterations would work, as it would also alter the depth information of pixels that we consider as "correct")
- Step 9: finally we replace any missing values in our original depth image with values from the temporary background depth image (based on the detected biggest contour)

The final image successfully closes any gaps in the depth image edges and offers a sharp and coherent matching of the depth values to its color counter pixel. An example output is shown in Figure 2 (top right).

7 https://google.github.io/mediapipe/.

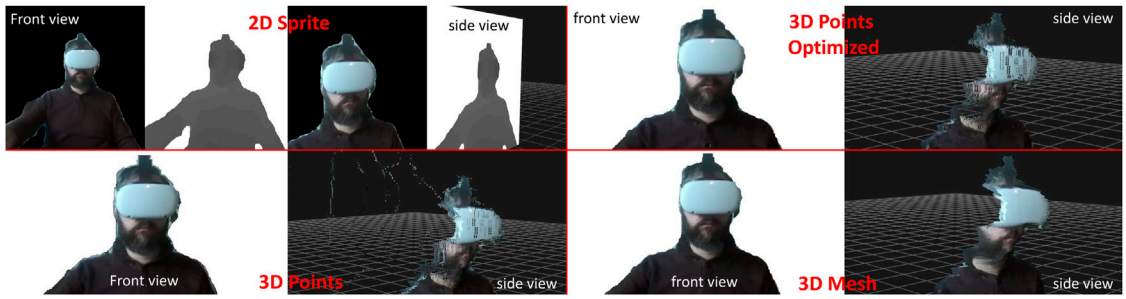

FIGURE 3
Render Example (RGBD Capture with ZED sensor).

3.2.4 Depth conversion

In order to transmit depth over existing video/image compression formats, the 16-bit depth data is converted into an 8-bit RGB color format, with a window of 1,530 values. Currently we offer two flavors of this conversion GrayAVG, presented in (Gunkel et al., 2021), and Intel HSV[8], both implemented in Numba[2] to offer maximum hardware acceleration with Python.

3.2.4.1 GrayAVG

We use the GrayAVG depth conversion algorithm as presented in (Gunkel et al., 2021). GrayAVG maps depth values with a range of 0–1.5 m (i.e., 1,530 values) into gray-color values on the entire RGB color space. By adding a minimum distance (distance the user is minimally away from the sensor) as metadata, this offers a flexible distance. We usually assume a minimal distance of 50cm, thus transmitting a depth range of 500–2,030 mm. To stream it as a single RGB-D video stream, the RGB image and the grayscale depth image are concatenated.

3.2.4.2 Intel HSV

We use our own implementation of[8], in Python and Numba[2]. This algorithm also converts the 16 bit depth data into 8-bit RGB color by utilizing the Hue color. The hue-color space offers 1,529 discrete levels, or approximately 10.5 bits, and 6 gradations in the R, G, and B up and down directions. Furthermore, the image never gets too dark because one of the rgb colors is always 255. This has the advantage of making sure that some lossy image/video compression does not lose details.

3.2.5 Output

Besides rendering the output on the screen, the output can be written into different virtual webcam drivers based on pyvirtualwebcam[9]. Currently, this supports two different virtual webcam drivers: OBS[10] and Unity Capture[11]. The main difference between these two drivers is that the OBS virtual driver operates in RGB format, while the Unity Capture virtual driver operates in RGBA format. Thus, choosing the right driver is important depending on the application accessing the data. For example, Unity-based applications generally expect RGBA format from webcam device, while for a browser the alpha channel is usually ignored. The output module will simple copy the final frame buffer (with or without adding an alpha) to the virtual device.

3.3 Transmission

The concept of our approach is based on converting any depth information into a 2D RGB image format (see Section 3.2.4) that can be transmitted via existing 2D encoding and distribution technology. This concept was previously reported in (Gunkel et al., 2018; Gunkel et al., 2021). Compared to newer volumetric streaming formats (such as V-PCC (Schwarz et al., 2019) and G-PCC (Mekuria et al., 2017)) this allows the use of reliable and established real-time streaming components (including full hardware accelerating), allowing high quality stream decoding even on low-powered end devices (like the Microsoft HoloLens 2[12] and Oculus Quest 2[13]). The actual transmission is encoding agnostic as long as the system supports any type of 2D RGB video format. Examples of applications transmitting the RGBD data are presented in Section 4.2.

3.4 RGBD 3D rendering

In this section we outline our approach on how to render the 2D RGB stream in 3D and what optimizations we can apply in the rendering shader to increase the visual quality. One important aspect of our approach is that the reconstruction of the RGBD image into 3D space does not require any additional processing steps but is directly executed in the shader code on the GPU and thus increases reliability and reduces resource overhead. Further, this allows us to apply pixel optimizations within the shader and different rendering techniques (i.e., as points or meshes) as explained in the following (in the example of WebGL).

8 dev.intelrealsense.com/docs/depth-image-compression-by-colorization-for-intel-realsense-depth-cameras.

9 https://github.com/letmaik/pyvirtualcam.

10 https://obsproject.com/.

11 https://github.com/schellingb/UnityCapture.

12 https://www.microsoft.com/en-us/HoloLens/.

13 https://www.meta.com/nl/en/quest/products/quest-2/.

3.4.1 Depth based rendering

Rendering the RGBD values back into 3D space is a simple geometric function based on the depth value and the intrinsic value of the camera (center pixel and focal length). Thus in order to render points correctly first the depth value needs to be reconstructed from the 2D image (HSV or grayscale) to a z-value in meter followed by the back projection.

For example, for GrayAVG (depth grayscale), the depth-to-z conversion can be done by combining the grayscale value and multiplying it by 1.53 (so that it represents a range of 0 – 1.53 m) and adding the fixed minimal user distance. In a WebGL shader this looks as follows:

$$float\ z = ((color.r + color.g + color.b)/3. * 1.53) + minDepth; \tag{1}$$

The back projection will look as follows in a WebGL shader. Note that the camera_center and camera_focalLength is the metadata coming from the capture frame grabber module of the user to be rendered (e.g., the sending application).

$$vec4\ position = projectionMatrix * modelViewMatrix * userTransformationMatrix * vec4 \tag{2}$$

$$((pixel\_position.x) - camera\_center.x) * z/camera\_focal\_length.x, \tag{3}$$

$$((pixel\_position.y) - camera\_center.y) * z/camera\_focal\_length.y, \tag{4}$$

$$z, 1.0); \tag{5}$$

The above examples are in WebGL, to render points in space (effectively a point cloud), the shader will write the values in a preallocated array of geometry point vertices[14] (an example is shown in Figure 3 3D Points). Changing this preallocated array into a PlaneBufferGeometry[15] also allows rendering the user with all points connected to each other (thus effectively as a Mesh; an example is shown in Figure 3 3D Mesh). The actual rendering is application dependent and can easily adopted, for example, into the different Unity render pipelines.

3.4.2 Edge smoothing

Simply rendering the RGBD image in 3D may lead to different distortions and imperfections. This can particularity affect the distortions in the edges of the image, based on errors in the capture, FGBG or image compression (i.e., YUV conversion can also inject errors in the corners of the image as encoding always combines information of 4 adjacent pixels). An example of this problem can be seen in Figure 3 (3D Points, bottom left), a halo around the user is stretched into space. To counter this effect and increase the rendering quality, we apply a simple edge smoothing. The edge smoothing we apply is based on the assumption that every adjunct pixels are connected to each other in space (different parts of a users body are connected). This means that any pixel that is adjunct but distant in space exceeding a specific threshold can be ignored. Thus, for each pixel, we calculate its distance to each neighboring pixel (x and y ± one) and apply two thresholds smoothing_threshold and cutoff_threshold. For the cutoff_threshold if any neighboring pixel is over that distance, we will not render this pixel. For the smoothing_threshold if any neighboring pixel is under that distance we average z value with that pixel to create a more smooth surface. Our best practice showed an ideal smoothing_threshold of 3 cm and a cutoff_threshold of 10 cm. An example of the edge smoothing as WebGL shader code is given in the following (please note that in the actual implementation the thresholds are not hard-coded but fully and dynamically configurable):

$$float\ depthDiff = distance(z,\ pixel\_neighbor.z); \tag{6}$$

$$if\ (depthDiff >= 0.1)\{ \tag{7}$$

$$gl\_PointSize = 0.0; \tag{8}$$

$$gl\_Position = vec4(0.0,\ 0.0,\ -1.0,\ 0.0); \tag{9}$$

$$return;\} \tag{10}$$

$$if\ (depthDiff <= 0.03)\{ \tag{11}$$

$$z = (z + pixel\_neighbor.z)/2.;\} \tag{12}$$

3.4.3 Point size

Ultimately, when rendering a user representation as a point cloud, each point will be rendered as pixels on the display. The final optimization step of the 3D rendering is to adjust the pixel size of each point based on the distance in space. The reason for this is that by default a capture sensor will capture more dense points if they are closer to the capture device rather than points that are far away. This results in that points further away (even though they are physically close) may be rendered with space in between them, and thus diminish the visual representation (note that this is only relevant when rendering the 3D representation as points, in a Mesh representation points are already connected by default). Thus, ideally, we would like to render points in the same size as the minimal distance to its direct neighbors (neighbors at the same distance in space) according to the focal properties of the capture device. This results in the following formula as WebGL shader code:

$$gl\_PointSize = z\ /\ camera\_focal\_length.x * 1000.; \tag{13}$$

Note that in WebGL we cannot create non-uniform pixel sizes on the x and y-axes, and thus create the pixel size only based on the x-axis. In Unity (e.g., VFX graph) more complex rendering techniques are possible. Further, in WebGL the z value is in meters, while the camera focal length is in mm, thus requiring a multiplication by 1,000.

The combination of the "Point Size" and "Edge Smoothing" (Section 3.4.2) are shown as example in Figure 3 3D Points Optimized).

4 Results and evaluation

We evaluate our modular capture application in two ways, a technical evaluation of the operation of the different modules (Section 4.1) and an integrated application evaluation (Section 4.2) of two use cases with a small-scale user study each. The

14 https://threejs.org/docs/\#api/en/core/BufferGeometry.

15 https://threejs.org/docs/\#api/en/geometries/PlaneGeometry.

technical evaluation measures the processing delay, CPU, GPU and memory resource usage of each individual module of the capture application, as well as the CPU, GPU and memory resource of the different rendering methods proposed. The application evaluation presents two XR communication solutions integrated with a dedicated capture instantiations.

4.1 Technical evaluation

In this section we present different performance measures for the capture and rendering as proposed in Section 3. For the different (processing and rendering) performance measures, we use a customized version of the Resources Consumption Metrics (RCM) (Montagud et al., 2020)[16]. The RCM utility is a native Windows application that enables 1-s interval system statistics capture for the CPU, GPU, and memory utilization per process. In all measurements, our capture application was deployed on a XMG NEO 15 [E21] laptop PC (with AMD Ryzen 9 5900HX, GeForce® RTX 3080 and 32 GB RAM). With a capture framerate of 15fps.

For (capture-to-display) processing delay measurements, we use VideoLat (Jansen, 2014)[17]. VideoLat is a tool for dedicated one-way delay measurements. A PC (Mac) with a webcam serves as a measuring device, constantly depicting QR-codes and measuring the time from display to webcam capture. VideoLat is calibrated in an initial step by pointing its on camera to the own screen, it measures the default (hardware) delay of the system (which will be subtracted from subsequent measures). For each measure condition, our capture application records the images of VideoLat and displayed them locally (while deploying different module configurations). In our setup VideoLat was used on a MacBook Retina (15-inch, early 2013 model with 2.4 GHz Intel Core i7, 8 GB RAM and MacOS 10.13.6) with a Logitech Brio[18] as capture camera.

4.1.1 Capture performance

We measure the delay of each capture module by measuring the displayed output of each module against the delay from frame grabbing with VideoLat. Different module configurations were deployed and tested as shown in Figure 1, with an additional step of a video conferencing application displaying the results of the virtual webcam. This is to show the delay in a realistic deployment, we used Microsoft Teams (MS Teams)[19] to show a self-view of the virtual webcam output.

For all ZED sensor conditions this means the following module deployment:

- Capture = ZED Frame Grabber
- FGBG = Mediapipe ML FGBG
- Enhance = ZED Hole Filling
- Depth = GrayAVG
- Output = OBS Virtual Webcam
- Application = Microsoft Teams (camera self view)

For all ZED sensor conditions this means the following module deployment:

- Capture = Kinect Frame Grabber
- FGBG = Simple FGBG
- Enhance = Kinect Edge Smoothing
- Depth = Intel HSV
- Output = Unity Virtual Webcam
- Application = Microsoft Teams (camera self view)

Table 1 shows the different delays measured after each module. Overall, to read the RGBD information in an application (i.e., MS Teams), the delay is approximately doubled to the input (fame grabber capture) delay. Further, the capture delay of the Kinect Azure can be seen as comparable to a "normal" webcam (e.g., the Logitech Brio calibration delay in VideoLat was measured with approx. 190 ms) while the ZED capture delay is slightly higher, which is expected as to the expense of stereo matching depth image creation. Otherwise, all processing delays were in a somewhat expected range. However, it is also important to recognize that higher resolution capture significantly increases processing delay. With the highest delay being over 500 ms (K4A sensor, high resolution in MS Teams), this configuration might not be suitable for every XR communication use case.

To measure the resource utilization of each individual module, we added a "pass-through" debug option into each module. In pass-through mode each module does not execute any processing but simply forwards pre-defined example data (i.e., data that is representative for the normal operation of this module). With this feature, we tested each individual module by setting all modules, besides the one to test, into pass trough and measure the CPU, GPU and memory usage via RCM. The module deployment is otherwise identical to the delay measures, with two additional conditions:

- None = No module active, only pass through
- All = All modules active, "normal" operation of the capture application

Table 2 shows the measurements of the different individual modules. As mentioned in all conditions (besides "All" and "None"), only one module is fully active in each condition. Overall, the results show a stable and acceptable performance overhead, even for the higher resolution RGBD streams. This might offer some possibility to utilize further resources in order to add more spatial image recognition and improvement tasks (like HMD replacement) or to further split processing into more parallel tasks to further decrease the processing delay.

4.1.2 Rendering performance

In order to evaluate the different rendering options (see Figure 3) we deployed a dummy capture module and used the VRComm system (Gunkel et al., 2021) with different devices and browsers as clients. The dummy capture module facilitates a playback of representative pre-recorded RGBD content based on the following modules: Azure Kinect Frame Grabber (with a

16 https://github.com/ETSE-UV/RCM-UV.

17 https://videolat.org/.

18 https://www.logitech.com/nl-nl/products/webcams/brio-4k-hdr-webcam.html.

19 https://www.microsoft.com/en-us/microsoft-teams/group-chat-software.

TABLE 1 Capture Module Processing Delays (VideoLan; 500+ samples each; mean values in ms; maximum confidence interval {alpha = 0.05} of 5 ms).

Sensor	Res	Capture	FGBG	Enhance	Depth	Output	Application
K4A	512	156 (SD 32)	159 (SD 37)	194 (SD 42)	241 (SD 41)	319 (SD 40)	297 (SD 40)
K4A	1,024	189 (SD 33)	225 (SD 45)	333 (SD 46)	480 (SD 57)	572 (SD 58)	573 (SD 59)
ZED	512	209 (SD 19)	240 (SD 36)	252 (SD 34)	287 (SD 30)	285 (SD 18)	390 (SD 37)
ZED	1,024	245 (SD 38)	324 (SD 39)	380 (SD 42)	429 (SD 52)	416 (SD 45)	475 (SD 48)

TABLE 2 Capture Performance of different modules (RCM; 5 min samples each; average and SD values; maximum confidence interval {alpha = 0.05} is CPU = 1.98%, GPU = 4.3%, and Memory = 48.86 MB).

Sensor	Res	Measure	None	Capture	FGBG	Enhance	Depth	Output	All
K4A	512	CPU in %	1.22	1.62	0.94	3.2	1.5	1.04	4.13
		SD in %	0.85	1.16	0.54	2.46	0.89	0.71	1.9
K4A	512	GPU in %	7.65	10.02	6.73	6.35	6.72	6.56	10.62
		SD in %	4.39	2.18	1.6	1.85	1.68	1.68	2.27
K4A	512	Memory in MBs	253.63	219.87	254.12	268.49	294.51	262.33	277.74
		SD in %	40.67	33.49	38.77	2.32	2.82	1.92	6.09
K4A	1,024	CPU in %	3.25	5.01	3.3	7.06	7.33	2.88	15.58
		SD in %	3.13	2.63	4.06	5.04	2.76	2.38	6.1
K4A	1,024	GPU in %	6.49	13.52	6.63	6.68	6.71	6.74	13.36
		SD in %	1.88	2.62	1.68	1.6	1.63	1.64	2.55
K4A	1,024	Memory in MBs	388.71	321.06	387.27	406.81	420.41	406.21	414.34
		SD in %	60.31	3.05	60.05	4.13	66.81	3.79	68.59
ZED	512	CPU in %	0.27	2.28	0.0	0.45	0.04	0.37	6.71
		SD in %	0.38	1.02	0.0	0.56	0.1	0.55	1.65
ZED	512	GPU in %	0.0	27.45	0.0	0.0	0.0	0.0	15.34
		SD in %	0.0	12.88	0.0	0.0	0.0	0.0	13.09
ZED	512	Memory in MBs	406.19	393.5	231.35	421.38	406.63	417.79	480.18
		SD in %	62.13	60.24	34.81	0.69	0.18	0.35	20.47
ZED	1,024	CPU in %	1.17	4.81	6.8	2.77	0.08	1.06	16.03
		SD in %	1.23	2.36	4.21	1.3	0.14	0.68	4.2
ZED	1,024	GPU in %	0.0	22.78	0.0	0.0	0.0	7.38	16.08
		SD in %	0.0	4.0	0.0	0.0	0.0	3.79	9.08
ZED	1,024	Memory in MBs	587.61	538.67	780.41	609.11	621.25	255.36	758.06
		SD in %	93.53	85.77	71.5	4.52	1.01	31.97	144.41

resolution of 1,024 and 512 pixel), Simple FGBG, Kinect Edge Smoothing, GrayAVG, and OBS Virtual Webcam. Besides the three 3D rendering conditions (3D Points, 3D Points Optimized and 3D Mesh), as presented in Figure 3 (see Section 3.4), we added two baseline conditions: 1) "None" not rendering any user and 2) "2D" rendering the user as a 2D sprite. This is the case where for each condition (besides "None") an upload client is uploading a RGBD video stream into VRComm (WebRTC/peer-to-peer) and the receiving client (marked as "Device/Browser" in the table) renders the RGBD image texture according to the condition. The performance of the "PC" condition was measured with RCM and the performance of the "HoloLens 2" condition was measured with the HoloLens Windows Device Portal[20]. For the HoloLens 2 we

20 https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/using-the-windows-device-portal.

TABLE 3 Render Performance of different Browsers (RCM; 5 min samples each; average and SD values; maximum confidence interval {alpha = 0.05} is CPU = 1.35%, GPU = 1.17%, and Memory = 8.81 MB).

Device/Browser	Res	Measure	None	2D	3D	3D optimized	3D Mesh
PC/Chrome	512	CPU in %	7.04	8.7	8.57	8.69	8.84
PC/Chrome	512	CPU-sd in %	1.07	1.25	1.35	1.18	1.22
PC/Chrome	512	GPU in %	34.09	39.17	49.12	48.83	49.43
PC/Chrome	512	GPU-sd in %	6.81	3.74	4.1	4.89	6.52
PC/Chrome	512	Memory in MBs	600.99	501.38	531.09	540.92	513.67
PC/Chrome	512	MEM-sd in %	50.07	31.45	18.4	31.7	48.75
PC/Chrome	1,024	CPU in %	7.71	9.09	9.81	9.66	9.76
PC/Chrome	1,024	CPU-sd in %	1.34	1.19	1.64	1.53	1.42
PC/Chrome	1,024	GPU in %	20.26	31.69	68.09	69.71	62.21
PC/Chrome	1,024	GPU-sd in %	6.52	2.63	6.62	8.38	8.31
PC/Chrome	1,024	Memory in MBs	446.15	522.26	602.88	604.21	637.08
PC/Chrome	1,024	MEM-sd in %	26.14	17.57	34.95	38.52	34.89
PC/Edge	512	CPU in %	4.85	37.24	18.06	12.79	17.75
PC/Edge	512	CPU-sd in %	0.99	2.04	2.24	2.36	2.35
PC/Edge	512	GPU in %	23.79	58.64	67.21	65.35	63.1
PC/Edge	512	GPU-sd in %	7.49	7.59	3.54	3.23	4.47
PC/Edge	512	Memory in MBs	450.66	540.18	614.07	565.66	625.57
PC/Edge	512	MEM-sd in %	21.65	78.1	81.07	84.3	72.9
PC/Edge	1,024	CPU in %	3.58	34.19	13.63	13.86	8.05
PC/Edge	1,024	CPU-sd in %	1.1	1.96	2.54	2.7	1.85
PC/Edge	1,024	GPU in %	21.74	29.55	48.53	48.82	76.47
PC/Edge	1,024	GPU-sd in %	5.0	4.55	6.68	6.38	2.33
PC/Edge	1,024	Memory in MBs	388.28	584.98	675.19	688.83	750.92
PC/Edge	1,024	MEM-sd in %	22.57	17.04	33.68	52.53	15.4
PC/Firefox	512	CPU in %	2.69	8.9	8.79	8.63	8.54
PC/Firefox	512	CPU-sd in %	0.98	1.37	1.4	1.42	1.54
PC/Firefox	512	GPU in %	19.78	49.63	55.65	55.33	56.26
PC/Firefox	512	GPU-sd in %	2.96	6.08	6.58	6.04	6.16
PC/Firefox	512	Memory in MBs	398.55	501.1	499.5	525.81	534.39
PC/Firefox	512	MEM-sd in %	10.99	52.0	52.95	14.86	53.35
PC/Firefox	1,024	CPU in %	3.35	9.76	9.97	9.95	9.96
PC/Firefox	1,024	CPU-sd in %	1.17	1.34	1.36	1.38	1.41
PC/Firefox	1,024	GPU in %	32.34	30.1	54.02	55.29	49.1
PC/Firefox	1,024	GPU-sd in %	8.77	2.62	6.1	5.58	5.26
PC/Firefox	1,024	Memory in MBs	446.71	558.87	690.56	699.74	701.23
PC/Firefox	1,024	MEM-sd in %	13.41	8.5	47.97	25.97	12.71
HoloLens 2/Edge	512	CPU in %	30.15	35.54	32.36	30.02	35.66
HoloLens 2/Edge	512	CPU-sd in %	3.79	2.6	3.41	3.84	2.12

(Continued on following page)

TABLE 3 (*Continued*) Render Performance of different Browsers (RCM; 5 min samples each; average and SD values; maximum confidence interval {alpha = 0.05} is CPU = 1.35%, GPU = 1.17%, and Memory = 8.81 MB).

Device/Browser	Res	Measure	None	2D	3D	3D optimized	3D Mesh
HoloLens 2/Edge	512	GPU in %	33.0	40.44	80.88	78.64	85.26
HoloLens 2/Edge	512	GPU-sd in %	1.96	3.93	11.28	13.44	2.83
HoloLens 2/Edge	1,024	CPU in %	30.37	33.41	29.06	27.5	28.63
HoloLens 2/Edge	1,024	CPU-sd in %	7.8	12.03	3.89	4.52	4.23
HoloLens 2/Edge	1,024	GPU in %	32.75	38.62	88.53	91.7	80.77
HoloLens 2/Edge	1,024	GPU-sd in %	3.67	10.42	4.58	4.9	4.76

measured the complete device performance (only running the browser client). Under all conditions, the memory usage of the HoloLens 2 device was measured below 3 GB.

Table 3 shows the rendering performance under the different device/browser and rendering conditions. Overall, the different browsers all perform very similar. Generally, all three browsers are very capable for WebXR processing and rendering. However, it is important to note that even though Chrome&Edge indicate a significantly higher GPU usage than Firefox, this is mainly due to internal rendering fame rate management. While Chrome&Edge targeted a framerate of approximately 90 fps, Firefox fluctuated around approximately 60 fps. On the Hololens 2 however we could observe a drop in framerate due to reaching the maximum resource utilization. While most conditions (None and 2D) run with the expected 60fps, including the 512 3D Mesh condition, the higher resolution and point cloud conditions decreased in framerate (512 3D and 3D Optimised had a framerate of ~30fps, 1,024 3D and 3D Optimised had a framerate of ~15fps, and 1,024 3D Mesh had a framerate of ~30fps). We can observe that the different configurations are technically suitable for photorealistic XR communication as previously observed in [(Gunkel et al., 2018; Dijkstra-Soudarissanane et al., 2019; Gunkel et al., 2021)]. Adding multiple users with high resolution might still be challenging according to the high GPU usage in our measures (this is also because for the high resolution more than 1M pixels point needs to be rendered and constantly updated). However, the different 3D render options (and whether optimizations were applied) had little influence on the resource usage, and thus can be chosen according to the use case and users needs. This is for the Hololens 2 the 3D Mesh rendering seems to result into a better technical performance.

4.2 XR application and use cases evaluation

4.2.1 Virtual reality for business meetings

Face-to-face (f2f) meetings are widely acknowledged to be the most productive and interesting way to conduct business meetings (Denstadli et al., 2012). One of the causes is that meetings using existing 2D videoconferencing solutions often lack engagement and effectiveness. Participants in meetings often struggle with background noise, lack of social presence, and not recognizing who is speaking. Most of the time, these problems arise in large group meetings (op den Akker et al., 2009). Thus, we assume that business meetings are a good testing use case for immersive communication in VR. To test our modular capture system in a business meeting use case, we developed an immersive VR communication tool that integrates 3D photorealistic capture and rendering (RGB + depth) into the Connec2[21] commercial VR application. Our approach enables movement in a 3D environment with auditory and visual spatial awareness, in the ideal view frustum. This is due to deploying only a single RGBD camera solution, individuals are advised to stay in a small area in order to prevent 3D point cloud distortions (i.e., simply because sections that are not captured cannot be displayed). However, staying in a small, confined space can be considered normal for most business meetings.

4.2.1.1 VR user study setup

To evaluate the technical feasibility of our modular capture application, we created a unique communication tool that integrates the presented capture application (for 3D photorealistic user representations) into the commercial VR application and communication platform Connec2. The capture application consisted of the following modules: ZED Frame Grabber (with a resolution of 512 × 512 pixel), Mediapipe ML FGBG, ZED Hole Filling, GrayAVG, and OBS Virtual Webcam. Deployed with a single RGBD (ZED) sensor and an Oculus Quest for business (same hardware as Oculus Quest 2). Figure 4 shows an example of the rendering of a remote user within the Connec2 VR communication application.

The complete application pipeline can be seen in Figure 5, and includes the following components:

- Modular capture as presented in Section 3
- Ingest client receiving RGBD images from the OBS virtual webcam driver and uploading them into the Connec2 system
- Connec2 utilizes a motion JPG compression sent via structured networked message queues
- The render client runs on an Oculus Quest device sending/receiving audio and rendering the audio as well as the 3D point cloud (via VFX-graph)

Part of this study was previously presented in (Singh et al., 2022). In total, 12 participants (6 meeting pairs) used the Social VR system

21 https://connec2.nl/.

FIGURE 4
Side view of example user inside the immersive VR communication application (Connec2) (Singh, 2022).

to have 1 or 2 meetings with each other from various offices in Netherlands. There are 11 male subjects and 1 female subject, and the participants' ages range from 20 to 60+. All pairs already knew each other through their work on previous or ongoing projects. Subsequent encounters were held with each other using the VR system over a 12 week period. All participants were required to complete a questionnaire after each meeting that contained questions based on several established surveys on social presence, immersion, usability, embodiment, quality of experience (QoE), quality of interaction (QoI), and quality of communication (QoC) (Brooke et al., 1996; Witmer and Singer, 1998; Biocca et al., 2001; Garau et al., 2001; Nowak and Biocca, 2003; Jennett et al., 2008; Wiebe et al., 2014; Toet et al., 2021).

4.2.1.2 Evaluation

To analyze the scores of the different measured aspects of the communication and the system, the answers to the questions corresponding to either embodiment, Quality of Service (QoS), usability, Quality of Interaction (QoI), immersion, social presence, and Quality of Communication (QoC) have been combined for their respective elements. The average rating for each of the 22 meetings for each of the elements for which the data was gathered is shown in Table 4. The average score for the embodiment was the lowest, coming in at −1.44 on a scale from −3 to 3, with 0 being neutrality. Positive numbers show a satisfactory or excellent response to the element, whilst negative values show an inadequate or bad response from the participants. With a score of 0.9, QoC received the best rating of all elements. Thus while technically users were able to operate and use the system with an acceptable level of social presence and quality of communication, there is still room to improve many of the technical aspects. Some of the limitations in this study are the low image resolution of 512 pixels, the VFX graph used in Unity did not apply edge smoothing, and the users face being occluded by the VR HMD. Currently, (whiteout applying any HMD replacement strategy) using HMDs is counterintuitive for photorealistic communication as it obfuscates the face and prevents eye gaze.

4.2.2 Augmented reality for personal communication

A demographic that is often neglected in daily personal communication is the elderly. There is a severe absence of contact between elderly residents of nursing facilities and their families and loved ones. Elderly people's health, cognitive decline, and quality of life are negatively impacted by social isolation and loneliness. Furthermore, the COVID-19 measures applied in care institutions in the Netherlands further "increased loneliness and restricted decision-making for" care home residents (Rutten et al., 2022). To address this problem, an AR tool with 3D user capture and rendering on an iPad and a large TV screen was previously presented in (Dijkstra-Soudarissanane et al., 2021; Dijkstra-Soudarissanane et al., 2022) and enabled high-quality mediated social communication (effectively rendering remote users is life size). We identified this use case to integrate and test our capture application in AR communication (i.e., with AR glasses).

4.2.2.1 AR user study setup

To test our capture application within an AR communication use case, we integrated our capture application into a web-based immersive communication platform called VRComm (Gunkel et al., 2021). VRComm is a web-enabled (WebXR) XR system that allows both VR and AR rendering on OpenXR devices. In order to enable our AR usecase in VRComm we made minor improvements to the system, like a new room configuration and the placement of users in AR. Users are placed opposite each other in AR, thus a remote user appearing in your own environment, with the same geometric and size properties as captured. For this integration, we deployed the

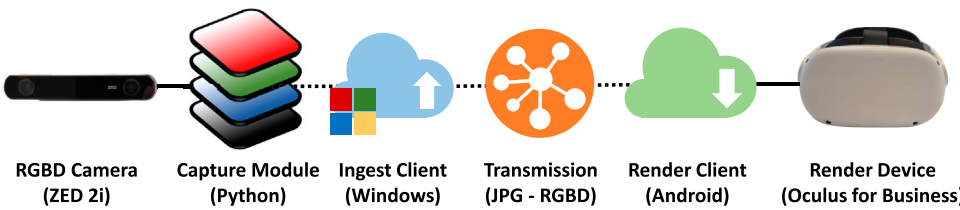

FIGURE 5
Application pipeline VR (Singh et al., 2022).

TABLE 4 Overview of variables of VR communication, scores are presented as mean average with standard deviation (SD) and confidence interval (CI; alpha = 0.05).

Variable	Embodiment	QoS	Usability	QoE	QoI	Immersion	Meeting Engagement	Social Presence	QoC
Mean	−1.44	−0.81	−0.30	−0.25	−0.10	0.61	0.68	0.75	0.90
SD	1.24	1.26	1.17	0.78	0.87	1.04	0.79	0.65	0.96
CI	0.52	0.54	0.49	0.33	0.37	0.43	0.33	0.27	0.40

FIGURE 6
Example user experiencing immersive communication in AR (with RGBD capture shown on the screen).

following capture modules: ZED Frame Grabber (with a resolution of 1024 × 1024 pixel), Mediapipe ML FGBG, ZED Hole Filling, GrayAVG, and OBS Virtual Webcam. The final user setup consists of a single RGBD (ZED) sensor, a capture ingest PC and a HoloLens 2 running a Social XR web client. An example of a user in an immersive communication, while wearing a Hololens 2 AR device, is shown in Figure 6. Note that the Kinect Azure could not be used, as it resulted in interference with the HoloLens 2.

The complete application pipeline can be seen in Figure 7, and includes the following components:

- Modular capture as presented in Section 3
- Ingest web client receiving RGBD images from the OBS virtual webcam driver and uploading them into the VRComm system
- VRComm utilizes per-to-peer WebRTC transmission for the video (and audio)
- The render web client runs on a HoloLens2 device sending/receiving audio and rendering the audio, as well as the 3D point cloud

We replicated the same testing environment as described in (Alvarez et al., 2022), to measure social presence in a personal communication environment of AR. We applied two different conditions than (Alvarez et al., 2022) for our small-scale user study: 1) a condition using MS teams on a 46 inch (life-size) display and 2) using VRComm and the HoloLens 2 as described above. To make both conditions comparable, users had to wear the HoloLens 2 in both conditions (i.e., the HoloLens 2 was turned off in the MS Teams condition). We conducted our study in a setup between subjects with 18 pairs (N = 36; 9 pairs per condition) of people who know each other (friends, couples, family, and close colleagues). This means in 7 male and 11 female (with an average age of 29.5) in the MS Teams condition, and 9 male and 9 female (with an average age of 34.39) in the HoloLens 2 condition. In each condition, two participants have a conversation in two different rooms followed by questionnaires. We included the following questionnaires in our study:

- Holistic Framework for Quality Assessment of Mediated Social Communication (H-MSC-Q) (Toet et al., 2022): to assess both the spatial and social presence.
- Networked Minds questionnaire (NMQ) (Biocca et al., 2001): assessing co-presence, psychological involvement, and behavioral engagement.
- Inclusion of Other in the Self (IOS) (Aron et al., 1992): to assess interpersonal closeness via “a single-item, pictorial measure of closeness”.
- User Experience Questionnaire (UEQ) (Schrepp, 2015): evaluating the user experience on six scales attractiveness, perspicuity, efficiency, dependability, stimulation, and novelty

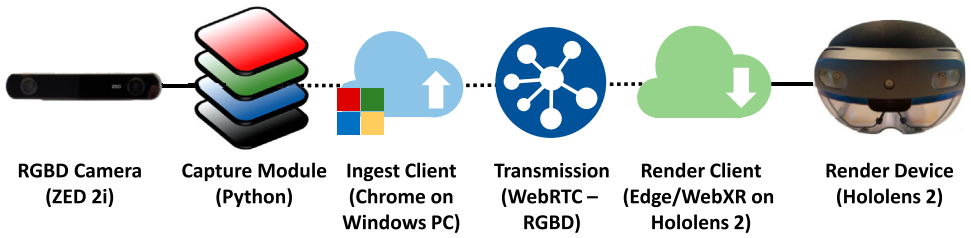

FIGURE 7
Application pipeline AR.

TABLE 5 Overview of presence and quality scores for AR communication (HoloLens 2), scores are presented as mean average with standard deviation (SD) and confidence interval (CI; alpha = 0.05).

Condition	IOS	H-MSC-Q	NMQ	Overall quality	Image quality
MS Teams 46" and HL2	3.78	6.1	5.27	4.11	4.16
	(SD 0.82)	(SD 0.86)	(SD 0.77)	(SD 0.66)	(SD 0.5)
	(CI 0.66)	(CI 0.40)	(CI 0.36)	(CI 0.30)	(CI 0.23)
HoloLens 2	3.44	5.73	5.6	3.78	2.72
	(SD 1.42)	(SD 0.81)	(SD 0.58)	(SD 0.71)	(SD 0.80)
	(CI 0.66)	(CI 0.39)	(SD 0.27)	(CI 0.33)	(CI 0.37)

We also asked users to rate the overall perceived video quality and performed post-experiment interviews for qualitative feedback. We ensured good quality audio, given its importance in video conferencing (Beerends and Neumann, 2020).

4.2.2.2 Evaluation

The results of our user study can be seen in Table 5, the perceived quality of our 3D video pipeline was lower compared to the conditions of the MS Teams. Interestingly, the feeling of social presence did not decrease significantly under these conditions, despite the lower perceived video quality. We performed a Principal Component Analysis of the various conditions to gain further insight into all the factors that influence the results. This analysis basically shows two main factors. The first is the novelty effect of the use of the HoloLens 2, for which the scores on factors such as attractiveness, novelty, and stimulation had a great influence on the social presence scores. However, perceived quality has a large positive influence on social presence scores.

This explorative study strengthens our belief that Augmented Reality can be used to increase the social presence when communicating at a distance. By further improving 3D video quality in the future, immersive communication will ultimately outperform current video conferencing in delivering social presence. This is further supported by remarks during the post-experiment interviews. Users indicated that the HoloLens 2 brought a feeling of closeness, of experience of the other person as being actually there with them in the room. Multiple users specifically mentioned the lack of a screen as a big plus. Even with these remarks, we still feel that AR headsets are not good enough yet for AR conferencing. Multiple users also remarked that they had to sit still, because with (head) movement, the other user would disappear, pointing to the limited field of view of the HoloLens 2. Additionally, with our limited single capture solution, the eyes were not visible, which was a serious detriment. In the Teams + HoloLens condition, the eyes were somewhat visible, but ideally more transparent glasses are available for eye contact. Given current announcements on new AR headsets, we feel they may be suitable in 2–4 years from now, both offering a bigger field of view and more transparent glasses.

Although the user study confirms the technical feasibility of integrating the capture modules into an immersive communication application that allows people to communicate in a way comparable to Microsoft Teams, it still raises multiple questions about the technical readiness of multiple components (such as the AR glasses itself) as well as multiple questions relating to the user experience.

5 Discussion

To discuss our modular capture application approach and evaluation results, we follow the structure of our research questions:

RQ1. What are the advantages and disadvantages of different RGBD capture modalities, individual processing steps, and their resource needs?

In this paper, we present an extensive performance analysis of the different capture and processing modalities. Each processing step was included in an individual module and independently measured, both in terms of processing delay and resource usage. Our results show low performance overhead with a good balance between CPU and GPU utilization (and a maximum of 16% for GPU and CPU for high

resolution streams). This means that the performance requirements are significantly higher than for traditional video conferencing capture but still acceptable for any modern PC (H1.1). Furthermore, we have an acceptable delay overhead for low-resolution streams. This is because with a complete delay of ~300 ms (K4A) and ~400 ms (ZED) we can assume a complete capture-to-glass delay for a communication application of less than 500 ms (depending on the encoding and network conditions), which is acceptable for most communication scenarios (H1.2). This may not be true for the high-quality stream setting with a complete delay of < 500 ms (ZED) and < 600 ms (K4A). One of the main reasons for the 100 ms higher delay in the K4A condition is that the Intel_HSV depth conversion seems to not perform optimal for the higher-resolution streams (needing ~150 ms execution time vs. ~50 ms execution time of GrayAVG), as well as the Unity Virtual Webcam driver (adding a delay of ~100 ms vs. ~50 ms for the OBS virtual webcam driver). We assume in its current state our proposed capture application, reconstruction and rendering might result into a delay double to a 2D webcam video approach. In an operational deployment, however, this delay could further be optimized, i.e., by processing within the integrated circuit (IC) of the capture device itself.

RQ2. How can RGBD data be utilized for 3D photorealistic user representations and what is its resource overhead?

In our rendering performance measures, all rendering conditions performed in an acceptable range of CPU/GPU usage within the laptop/browser combination (see Table 3). Furthermore, all browsers performed similar, which is to be expected since we choose main browsers that are in long-term development and well equipped for 3D WebGL processing. One surprising aspect is that the Chromium-based Edge browser seemed to perform slightly less than Chrome, even though both rely on the same rendering engine. However, we can conclude that the RGBD-based photorealistic user rendering is easily possible on a modern PC with any of the tested browsers (H2.1). With the internal structure of the WebGL rendering engine in mind, we expect that any native application (e.g., Unity or Unreal XR application) would achieve similar or better performance results.

Concerning the different 3D rendering techniques, we were unable to observe any significant differences between them. This means that our introduced shader optimizations are able to increase the render quality while not introducing any further processing overhead (H2.2). Furthermore, we could not observe much difference between 3D Mesh and 3D point-based rendering, allowing us to choose the optimal rendering technique based on the user case and user aesthetic preferences. When comparing the 3D rendering with the 2D baseline, we do see a significant increase in GPU and memory usage. This is expected as, for example, in the high-resolution case, the GPU needs to handle 1M individual pixel data points and its geometric properties. Furthermore, the stable CPU utilization between the 2D and 3D conditions indicates that the rendering pipeline works optimally and reliably.

Our tests also show that rendering on the HoloLens 2, even though possible, is still challenging, as it practically maxes out the CPU/GPU usage of the device (H2.1). This effectively only allows to add one high resolution user. Adding more users would only be possible in a lower resolution, and thus accepting a low-quality image for the user representations. However, with the next-generation of hardware and further optimizations such as remote rendering, this could be mitigated in the future.

RQ3. Do RGBD-based user representations result in a high social presence for remote communication?

We integrated our modular capture application into two XR communication applications, covering both AR and VR modalities, including one commercial application. Our examples show that a modular integration is feasible and can add 3D photorealistic user representations to existing video communication transmission systems. Even though motion JPEG compression (as in the case of Connec2) can be considered suboptimal to transmit RGBD data (and thus is currently restricted to 512 × 512 pixel resolution for the color and depth stream), it further shows the usability and technical feasibility of our approach. However, the question of an increased level of social presence can only be partially answered due to remaining technical limitations and the complexity of user studies itself. Although the AR personal communication use case study showed an indicative result of its benefits, technically the AR glasses do not appear to be fully ready for our use case. Multiple technical improvements are required in the pipeline to reach the same sophisticated state of a commercial application like Microsoft Teams. Further, more user research is needed to investigate XR communication use cases in more detail. This includes a need for a better understanding of technical and functional limits as well as the benefits of immersive photorealisitc communication. We believe that our current modular capture approach offers the ideal testbed for dedicated user studies under different constraints.

Finally, we like to stress that the modular capture application and rendering, presented in this paper, can be reused in any established existing 2D video conferencing system (incl. its 2D video transmission). However, building such immersive communication systems in a reliable, scalable way with ever-growing higher resolution of streams and simultaneous users remains a challenge.

6 Conclusion and future work

In this paper, we present a modular capture application as well as two examples of integration with two XR communication applications, one of which is a commercial product. We show the advantages and disadvantages of the different modules with dedicated in-depth evaluations of each module. This includes two types of foreground background extraction and two RGBD image quality enhancement strategies. In addition, we propose and evaluate different techniques to render the 3D user representation. Our small-scale user evaluation of the two XR communication applications shows the technical suitability of our approach and shows that an easy integration of 3D photorealistic user representations can be achieved to allow immersive communication.

6.1 Future work

We are currently planning multiple extensions to the modular capture application. Most importantly, will be the extension to include HMD replacement, as a new spatial computing functional block. The HMD replacement will allow users to gaze into each other's eyes while wearing a VR HMD. We see that the occlusion of the face by the XR HMD is currently one of the main obstacles from a user experience point of view (within VR

experiences). Further, as many of the processing blocks are still resource intensive (i.e., the rendering), we plan to add a network layer into the processing chain, and thus allow processing and rendering tasks to move freely between central processing units (in the edge and cloud) and the user's device. We consider remote rendering and remote processing particularly important for the next-generation of AR glasses (which will have a small form factor and thus limited battery and resource power). Finally, the modular capture application offers a basis for more user experience testing. Therefore, integration into more XR communication use cases is important to gain a broad understanding of technical and non-technical limitations and ultimately to have a reliable QoS-to-QoE mapping of 3D user capture, transmission, and rendering.

Data availability statement

The datasets presented in this article are not readily available because all data was handled conform to GDPR and is only presented as results of the analysis in this paper. Requests to access the datasets should be directed to simon.gunkel@tno.nl.

Ethics statement

The studies involving human participants were reviewed and approved by TNO internal ethics board. The patients/participants provided their written informed consent to participate in this study. Written informed consent was obtained from the individual(s) for the publication of any potentially identifiable images or data included in this article.

Author contributions

SG contributed to the conception and design of the technical aspects, technical evaluation, analysis, leading the research and paper writing. SD-S contributed to the overall execution of the research. HS contributed to the conception and design of the user studies, as well as the statistical analysis. ON contributed to the shaping of the research questions. All authors contributed to manuscript writing and revision.

Acknowledgments

In the context of this article, we would like to acknowledge the technical contributions made by our project partner Connec2 B.V. and the utilization of their communication platform. The authors especially thank Tim Moelard and Stefan Leushuis for their technical help and contributions to the system's operation. Furthermore, we thank Simardep Singh and Deborah van Sinttruije for setting up the AR and VR user studies, as well as Tessa Klunder, Marina Alvarez, and Veronne Reinders for their participation in this research. Also, we thank Dr. Evangelos Alexiou for his feedback and spellchecking shaping the final version of the paper. Finally, we thank all the participants in our user study for their time and valuable feedback.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Alexiadis, D. S., Chatzitofis, A., Zioulis, N., Zoidi, O., Louizis, G., Zarpalas, D., et al. (2017). An integrated platform for live 3d human reconstruction and motion capturing. *IEEE Trans. Circuits Syst. Video Technol.* 27, 798–813. doi:10.1109/tcsvt.2016.2576922

Alvarez, M., Toet, A., and Dijkstra-Soudarissanane, S. (2022). "Virtual visits: Ux evaluation of a photorealistic ar-based video communication tool," in Proceedings of the 1st Workshop on Interactive eXtended Reality, 69–75.

Aron, A., Aron, E. N., and Smollan, D. (1992). Inclusion of other in the self scale and the structure of interpersonal closeness. *J. personality Soc. Psychol.* 63, 596–612. doi:10.1037/0022-3514.63.4.596

Aubert, B. A., and Kelsey, B. L. (2003). Further understanding of trust and performance in virtual teams. *Small group Res.* 34, 575–618. doi:10.1177/1046496403256011

Beerends, J., and Neumann, N. (2020). "Conversational quality assessment of advanced video conferencing systems," in Presented at the 4th International Conference of the Acoustical Society of Nigeria.

Biocca, F., Harms, C., and Gregg, J. (2001). "The networked minds measure of social presence: Pilot test of the factor structure and concurrent validity," in 4th annual international workshop on presence, Philadelphia, PA, 1–9.

Boyle, E. A., Anderson, A. H., and Newlands, A. (1994). The effects of visibility on dialogue and performance in a cooperative problem solving task. *Lang. speech* 37, 1–20. doi:10.1177/002383099403700101

Brooke, J. (1996). "Sus-a quick and dirty usability scale," in *Usability Eval. industry*. Editor P. W. Jordan, B. Thomas, B. A. Weerdmeester, and I. L. McClelland, 189–194. CRC Press: London. doi:10.1201/9781498710411

Carballeira, P., Carmona, C., Díaz, C., Berjon, D., Corregidor, D., Cabrera, J., et al. (2021). Fvv live: A real-time free-viewpoint video system with consumer electronics hardware. *IEEE Trans. Multimedia* 24, 2378–2391. doi:10.1109/tmm.2021.3079711

Cascio, W. F. (2000). Managing a virtual workplace. *Acad. Manag. Perspect.* 14, 81–90. doi:10.5465/ame.2000.4468068

Cho, S., Kim, S., Lee, J., Ahn, J., and Han, J. (2020). "Effects of volumetric capture avatars on social presence in immersive virtual environments," in 2020 IEEE Conference on Virtual Reality and 3D User Interfaces (New York, NY: VR), 26–34.

Coelho, C., Tichon, J., Hine, T. J., Wallis, G., and Riva, G. (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* (Amsterdam: IOS Press), 25–45.

Collet, A., Chuang, M., Sweeney, P., Gillett, D., Evseev, D., Calabrese, D., et al. (2015). High-quality streamable free-viewpoint video. *ACM Trans. Graph.* 34, 1–13. doi:10.1145/2766945

De Simone, F., Li, J., Debarba, H. G., El Ali, A., Gunkel, S. N., and Cesar, P. (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 (VRIEEE), 890–891.

Denstadli, J. M., Julsrud, T. E., and Hjorthol, R. J. (2012). Videoconferencing as a mode of communication: A comparative study of the use of videoconferencing and face-to-face meetings. *J. Bus. Tech. Commun.* 26, 65–91. doi:10.1177/1050651911421125

Dijkstra-Soudarissanane, S., Assal, K. E., Gunkel, S., Haar, F. t., Hindriks, R., Kleinrouweler, J. W., et al. (2019). "Multi-sensor capture and network processing for virtual reality conferencing," in Proceedings of the 10th ACM Multimedia Systems Conference, 316–319.

Dijkstra-Soudarissanane, S., Gunkel, S. N., and Reinders, V. (2022). "Virtual visits: Life-size immersive communication," in Proceedings of the 13th ACM Multimedia Systems Conference, 310–314.

Dijkstra-Soudarissanane, S., Klunder, T., Brandt, A., and Niamut, O. (2021). "Towards xr communication for visiting elderly at nursing homes," in ACM International Conference on Interactive Media Experiences, 319–321.

Ekong, S., Borst, C. W., Woodworth, J., and Chambers, T. L. (2016). "Teacher-student vr telepresence with networked depth camera mesh and heterogeneous displays," in Advances in Visual Computing: 12th International Symposium, ISVC 2016, Las Vegas, NV, USA, December 12-14, 2016 (Springer), 246–258.12

Elvezio, C., Sukan, M., Oda, O., Feiner, S., and Tversky, B. (2017). "Remote collaboration in ar and vr using virtual replicas," in *ACM SIGGRAPH 2017 VR village* (New York, NY, USA: Association for Computing Machinery). SIGGRAPH '17. doi:10.1145/3089269.3089281

Fehn, C. (2004). Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3d-tv. Stereosc. displays virtual Real. Syst. XI *(SPIE)* 5291, 93–104.

Garau, M., Slater, M., Bee, S., and Sasse, M. A. (2001). "The impact of eye gaze on communication using humanoid avatars," in Proceedings of the SIGCHI conference on Human factors in computing systems, 309–316.

Grier, R., Thiruvengada, H., Ellis, S., Havig, P., Hale, K., and Hollands, J. (2012). "Augmented reality–implications toward virtual reality, human perception and performance," in Proceedings of the Human Factors and Ergonomics Society Annual Meeting (Los Angeles, CA): SAGE Publications Sage CA), 1351–1355.

Gunkel, S. N., Dohmen, M. D., Stokkinq, H., and Niamut, O. (2019). 360-degree photo-realistic vr conferencing. In 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VRIEEE), 946–947.

Gunkel, S. N., Hindriks, R., Assal, K. M. E., Stokking, H. M., Dijkstra-Soudarissanane, S., Haar, F. t., et al. (2021). "Vrcomm: An end-to-end web system for real-time photorealistic social vr communication," in Proceedings of the 12th ACM Multimedia Systems Conference, 65–79.

Gunkel, S. N., Stokking, H., De Koninck, T., and Niamut, O. (2019). "Everyday photo-realistic social vr: Communicate and collaborate with an enhanced co-presence and immersion," in Technical Papers International Broadcasting Convention (IBC).

Gunkel, S. N., Stokking, H. M., Prins, M. J., van der Stap, N., Haar, F. B. t., and Niamut, O. A. (2018). "Virtual reality conferencing: Multi-user immersive vr experiences on the web," in Proceedings of the 9th ACM Multimedia Systems Conference, 498–501.

Jansen, J. (2014). "Videolat," in Proceedings of the ACM International Conference on Multimedia - MM. doi:10.1145/2647868.2654891

Jennett, C., Cox, A. L., Cairns, P., Dhoparee, S., Epps, A., Tijs, T., et al. (2008). Measuring and defining the experience of immersion in games. *Int. J. human-computer Stud.* 66, 641–661. doi:10.1016/j.ijhcs.2008.04.004

Kurillo, G., Hemingway, E., Cheng, M.-L., and Cheng, L. (2022). Evaluating the accuracy of the azure kinect and kinect v2. *Sensors* 22, 2469. doi:10.3390/s22072469

Langa, S. F., Climent, M. M., Cernigliaro, G., and Rivera, D. R. (2021). Toward hyper-realistic and interactive social vr experiences in live tv scenarios. *IEEE Trans. Broadcast.* 68, 13–32. doi:10.1109/tbc.2021.3123499

Liu, Y., Beck, S., Wang, R., Li, J., Xu, H., Yao, S., et al. (2015). "Hybrid lossless-lossy compression for real-time depth-sensor streams in 3d telepresence applications," in Pacific Rim Conference on Multimedia, 442–452. doi:10.1007/978-3-319-24075-6_43

Lombard, M., and Ditton, T. (1997). At the heart of it all: The concept of presence. *J. computer-mediated Commun.* 3, 1. doi:10.1111/j.1083-6101.1997.tb00072.x

Mekuria, R., Blom, K., and Cesar, P. (2017). "Design, implementation, and evaluation of a point cloud codec for tele-immersive video," in IEEE Transactions on Circuits and Systems for Video Technology, 828–842. doi:10.1109/tcsvt.2016.254303927

Montagud, M., De Rus, J. A., Fayos-Jordan, R., Garcia-Pineda, M., and Segura-Garcia, J. (2020). "Open-source software tools for measuring resources consumption and dash metrics," in Proceedings of the 11th ACM Multimedia Systems Conference (MMSys '20). New York, NY: Association for Computing Machinery, 261–266. doi:10.1145/3339825.3394931

Ndjiki-Nya, P., Koppel, M., Doshkov, D., Lakshman, H., Merkle, P., Muller, K., et al. (2011). Depth image-based rendering with advanced texture synthesis for 3-d video. *IEEE Trans. Multimedia* 13, 453–465. doi:10.1109/tmm.2011.2128862

Nesher Shoshan, H., and Wehrt, W. (2022). Understanding "zoom fatigue": A mixed-method approach. *Appl. Psychol.* 71, 827–852. doi:10.1111/apps.12360

Nowak, K. L., and Biocca, F. (2003). The effect of the agency and anthropomorphism on users' sense of telepresence, copresence, and social presence in virtual environments. *Presence Teleoperators Virtual Environ.* 12, 481–494. doi:10.1162/105474603322761289

op den Akker, R., Hofs, D., Hondorp, H., op den Akker, H., Zwiers, J., and Nijholt, A. (2009). "Supporting engagement and floor control in hybrid meetings," in *Cross-modal analysis of speech, gestures, gaze and facial expressions* (Springer), 276–290.

Orts-Escolano, S., Rhemann, C., Fanello, S., Chang, W., Kowdle, A., Degtyarev, Y., 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.

Park, J., Chou, P. A., and Hwang, J. (2019). Rate-utility optimized streaming of volumetric media for augmented reality. *IEEE J. Emerg. Sel. Top. Circuits Syst.* 9, 149–162. doi:10.1109/jetcas.2019.2898622

Pece, F., Kautz, J., and Weyrich, T. (2011). Adapting standard video codecs for depth streaming. *EGVE/EuroVR.*, 59–66.

Pinsonneault, A., and Boisvert, M. (2001). "The impacts of telecommuting on organizations and individuals: A review of the literature," in Telecommuting and virtual offices: Issues and opportunities (IGI Global), 163–185.

Priest, H. A., Stagl, K. C., Klein, C., and Salas, E. (2006). "Virtual teams: Creating context for distributed teamwork," in Creating high-tech teams: Practical guidance on work performance and technology. Editors C. Bowers, E. Salas, and F. Jentsch (American Psychological Association), 185–212. doi:10.1037/11263-009

Prins, M. J., Gunkel, S. N., Stokking, H. M., and Niamut, O. A. (2018). Togethervr: A framework for photorealistic shared media experiences in 360-degree vr. *SMPTE Motion Imaging J.* 127, 39–44. doi:10.5594/jmi.2018.2840618

Rasmuson, S., Sintorn, E., and Assarsson, U. (2021). A low-cost, practical acquisition and rendering pipeline for real-time free-viewpoint video communication. *Vis. Comput.* 37, 553–565. doi:10.1007/s00371-020-01823-7

Reimat, I., Mei, Y., Alexiou, E., Jansen, J., Li, J., Subramanyam, S., et al. (2022). "Mediascape xr: A cultural heritage experience in social vr," in Proceedings of the 30th ACM International Conference on Multimedia, 6955–6957.

Roberts, D., Duckworth, T., Moore, C., Wolff, R., and O'Hare, J. (2009). "Comparing the end to end latency of an immersive collaborative environment and a video conference," in 2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (IEEE), 89–94.

Rutten, J. E., Backhaus, R., Ph Hamers, J., and Verbeek, H. (2022). Working in a Dutch nursing home during the Covid-19 pandemic: Experiences and lessons learned. *Nurs. open* 9, 2710–2719. doi:10.1002/nop2.970

Schmitt, M., Gunkel, S., Cesar, P., and Bulterman, D. (2014). Asymmetric delay in video-mediated group discussions. 2014 sixth international workshop on quality of multimedia experience. QoMEX IEEE, 19–24.

Schreer, O., Feldmann, I., Renault, S., Zepp, M., Worchel, M., Eisert, P., et al. (2019). "Capture and 3d video processing of volumetric video," in 2019 IEEE International Conference on Image Processing (ICIP), 4310–4314.

Schrepp, M. (2015). "User experience questionnaire handbook," in *All you need to know to apply the UEQ successfully in your project*.

Schwarz, S., Preda, M., Baroncini, V., Budagavi, M., Cesar, P., Chou, P. A., et al. (2019). Emerging mpeg standards for point cloud compression. *IEEE J. Emerg. Sel. Top. Circuits Syst.* 9, 133–148. doi:10.1109/JETCAS.2018.2885981

Sellen, A. J. (1995). Remote conversations: The effects of mediating talk with technology. *Human-computer Interact.* 10, 401–444. doi:10.1207/s15327051hci1004_2

Singh, S., Dijkstra-Soudarissanane, S., and Gunkel, S. (2022). "Engagement and quality of experience in remote business meetings: A social vr study," in Proceedings of the 1st Workshop on Interactive eXtended Reality, 77–82.

Singh, S. (2022). Towards guidelines for facilitating engaging social VR business meetings. Master's thesis

Stanney, K. M., Nye, H., Haddad, S., Hale, K. S., Padron, C. K., and Cohn, J. V. (2021). Extended reality (xr) environments. *Handb. Hum. factors ergonomics*, 782–815. doi:10.1002/9781119636113.ch30

Straus, S. G. (1997). Technology, group process, and group outcomes: Testing the connections in computer-mediated and face-to-face groups. *Human–Computer Interact.* 12, 227–266. doi:10.1207/s15327051hci1203_1

Subramanyam, S., Li, J., Viola, I., and Cesar, P. (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 (Atlanta, GA: VR), 127–136. doi:10.1109/VR46266.2020.00031

Tadic, V., Toth, A., Vizvari, Z., Klincsik, M., Sari, Z., Sarcevic, P., et al. (2022). Perspectives of realsense and zed depth sensors for robotic vision applications. *Machines* 10, 183. doi:10.3390/machines10030183

Thompson, L. F., and Coovert, M. D. (2006). "Understanding and developing virtual computer-supported cooperative work teams," in *Creating high-tech teams: Practical*

guidance on work performance and technology. Editors C. Bowers, E. Salas, and F. Jentsch (American Psychological Association), 213–241. doi:10.1037/11263-010

Toet, A., Mioch, T., Gunkel, S. N., Niamut, O., and van Erp, J. B. (2022). Towards a multiscale qoe assessment of mediated social communication. *Qual. User Exp.* 7, 4–22. doi:10.1007/s41233-022-00051-2

Toet, A., Mioch, T., Gunkel, S. N., Niamut, O., and van Erp, J. B. (2021). "Holistic framework for quality assessment of mediated social communication," in Virtual Reality and Augmented Reality. EuroVR 2020, Valencia, Spain. doi:10.1007/978-3-030-62655-6_13

Wiebe, E. N., Lamb, A., Hardy, M., and Sharek, D. (2014). Measuring engagement in video game-based environments: Investigation of the user engagement scale. *Comput. Hum. Behav.* 32, 123–132. doi:10.1016/j.chb.2013.12.001

Wilson, J. M., Straus, S. G., and McEvily, B. (2006). All in due time: The development of trust in computer-mediated and face-to-face teams. *Organ. Behav. Hum. Decis. Process.* 99, 16–33. doi:10.1016/j.obhdp.2005.08.001

Witmer, B. G., and Singer, M. J. (1998). Measuring presence in virtual environments: A presence questionnaire. *Presence* 7, 225–240. doi:10.1162/105474698565686

Yu, C., Xu, Y., Liu, B., and Liu, Y. (2014). "Can you see me now?" a measurement study of mobile video calls," in IEEE INFOCOM 2014-IEEE Conference on Computer Communications (IEEE), 1456–1464.

Zioulis, N., Alexiadis, D., Doumanoglou, A., Louizis, G., Apostolakis, K., Zarpalas, D., et al. (2016). "3d tele-immersion platform for interactive immersive experiences between remote users," in 2016 IEEE International Conference on Image Processing (ICIP), 365–369.