IADIS International Journal on WWW/Internet
Vol. 20, No. 2, pp. 99-116
ISSN: 1645-7641

AR IN VR: OMNISTEREOSCOPIC TELEPRESENCE WITH HOLOGRAMS FOR REMOTE MAINTENANCE AND COLLABORATION

Furkan Kaynar[1], Markus Hofbauer[1], Asa MacWilliams[2], Joseph Newman[2], Andreas Hutter[2] and Eckehard Steinbach[1]
[1]Technical University of Munich, School of Computation, Information and Technology, Department of Computer Engineering, Chair of Media Technology, Munich Institute of Robotics and Machine Intelligence (MIRMI), Germany
[2]Siemens Technology, Munich, Germany

ABSTRACT

With the development of mixed reality technologies, remote maintenance and collaboration applications receive increasing interest in the industry. The combination of 360-degree telepresence and augmented reality (AR) cues have been shown to be an effective way of remote collaboration. However, most existing methods do not provide depth perception, which can improve the remote inspection. Also, many methods that use holograms for live collaboration are limited with only pointing functionality. In this paper, we present a 360-degree telepresence system interacting with an AR backend to create an immersive mixed reality interface for remote collaboration. The proposed camera system does not need to be carried by any on-site person, but can be controlled by the remote operator intuitively. We deploy network delay compensation methods and propose novel projection strategies for a correct and efficient rendering. A comparative analysis shows advantages of the proposed system over previous work, and that it is a promising approach for improving remote collaboration and maintenance. Our experiments indicate that our system has a glass-to-glass delay of roughly 106 ms, and a data rate to the client varying between 10-15 Mbps, which can be further optimized for specific applications. Exploratory tests with VR-experienced users showed that the motion-to-photon latency of the system is in acceptable ranges. The system can be deployed in various industrial applications for a live or asynchronous collaboration between an on-site user (with AR) and an off-site user (with AR in VR).

KEYWORDS

Remote Collaboration, Remote Maintenance, Augmented Reality, Virtual Reality, 360-Degree Telepresence

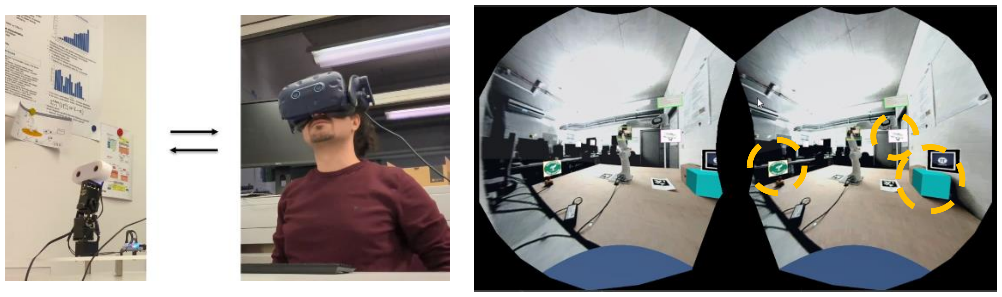


Figure 1. Overview of the proposed telepresence system with Augmented Reality (AR) cues. The camera unit on the left mimics the head motion of the user. The user is provided with stereo views in all directions (360°) augmented with the AR cues (on the right). Some of the holograms are highlighted with orange circles for better visibility

1. INTRODUCTION

With the increasing availability of Augmented Reality (AR) and Virtual Reality (VR) systems, different remote inspection and collaboration systems have been developed based on AR and telepresence (Lee et al., 2018; Speicher et al., 2018).

The design of the telepresence system defines the capabilities and limitations of the remote session and hence determines what the operator perceives. Among different systems, 360° stereoscopic vision systems provide a remote presence experience with a high level of immersion and 3D perception, which improves the task performance for indoor applications (Aykut et al., 2019). On the other hand, capturing a live stereoscopic 360° view of the environment is challenging due to problems like computational complexity, limited stereoscopic budget (the level of flexibility of changing the inter-pupillary distance), and constrained 3D impression due to unequal depth perception in different orientations (Aykut et al., 2019).

AR can be used for augmenting the scene with important technical details or auxiliary information, increasing the effectiveness of a remote inspection or collaboration session. The vast majority of the AR-based live collaboration systems either use a 2D view or a monoscopic panorama of the remote environment. A 2D view often leads to a limited field-of-view (FoV) and situational awareness, while a monoscopic panorama lacks the depth information.

In this paper, we present a remote AR system combined with a 360° stereoscopic telepresence system to improve real-time remote inspection and collaboration applications. The operator can freely observe the remote environment without depending on any other person, and with depth perception, due to omnistereoscopic vision. In addition, we deploy a state-of-the-art network delay compensation method (Aykut et al., 2019), to allow a natural and smooth Mixed Reality (MR) experience.

The proposed system uses an AR backend that provides hologram poses with respect to the anchor points initially defined for indoor localization (Lehrbaum et al., 2022). Any client device can add or modify the AR components, consisting of images, text information, web-content, or the live sensory data of an industrial environment. The holograms reappear at the saved 6-DOF poses once a client device revisits the same environment. This allows for an effective live collaboration as well as the generation of an information database improving the remote inspection in the long-term. We propose a novel system to efficiently map the hologram imagery onto the spherical VR scene created with fisheye lenses.

The rest of the paper is structured as follows. In Section 2, we summarize the related work in the field of mixed-reality telepresence systems. In Section 3, we introduce the architecture of the proposed 360° telepresence system interacting with an AR backend. We present a comparative analysis against similar systems in Section 4. In Section 5, we provide an experimental evaluation with measurements. Section 6 gives an outlook on future work and possible industrial applications. Section 7 concludes the paper.

2. RELATED WORK

In this section, we summarize existing systems proposed for 360° telepresence with AR overlays. Druta et al. (2021) provide a general overview on remote collaboration. Pretlove (1998) presented a mobile robotic platform equipped with an actuated stereo system synchronized with the operator's head-mounted display (HMD). The scene is overlayed with computer graphics to help the operator while navigating the remote robot in low visibility cases. While this system only focuses on navigation by teleoperation, it does not allow live manipulation of holograms for collaboration.

Wang et al. (2012) proposed a pan-tilt stereo camera unit and AR overlays of objects. They estimate the user's head orientation from the camera view attached to the head using visual processing. To avoid the slow response of the actuated system, the authors warp the video shown to the user to mitigate the inconsistency between the user's and the remote camera's orientations. However, this can lead to incomplete visualization of the remote environment and visual discomfort.

Speicher et al. (2018) proposed a system with a static 360° monoscopic camera and a projector for creating AR cues in a room for remote collaboration. The application scenarios are limited to putting markers on a straight wall and visualizing them with a projector.

Lee et al. (2018) proposed a live panorama system where the remote user views the panoramic video stream captured by the head-mounted 360° monoscopic camera carried by the local host. AR cues and hand gestures are used for non-verbal collaboration. The users reported motion sickness triggered in the dependent mode where they observe the viewport of the local host. Furthermore, the users reported arising discomfort due to the jittery motion introduced by the head-mounted device during the independent mode. Since this system only provides a monoscopic view, no 3D perception of the remote environment is available.

Teo et al. (2019) proposed a system with two modes. The first mode is 360° monoscopic view obtained by a head-mounted camera carried by the local host. The second mode provides a 3D model of the room for an independent viewing experience. While the second method addresses the problem of dependency to the host, it requires a 3D reconstruction of the room in advance.

Kasahara et al. (2016) proposed a head-mounted multi-camera arrangement that creates the 360° view of the environment and stabilizes the view to allow independent remote inspection by the second user. This system does not only bring high computational cost for combining multi-camera frames, but also results in severe stitching errors as viewed in the demonstration videos.

Most of the methods discussed in this section create and transmit a full 360° monoscopic video, which typically requires a high bandwidth. This is problematic for cases where the network throughput is limited or time-varying. The bandwidth limitations are an even more severe problem for transmitting a 360° stereoscopic video.

The head-mounted perception systems do not only cause viewing discomfort and motion sickness by the remote viewer, but they also add the burden to the local host to carry the system on the head in addition to the maintenance task. For an efficient video transmission and processing, head-mounted camera systems are typically connected to the workstations via cables, which further limits the freedom of movement. Many works use AR cues for pointing at different objects, but they do not build a long-term information base that can be used in further telemaintenance sessions.

Our proposed system addresses all of the problems mentioned above. Our actuated camera system has a steady basis, eliminating the need for carrying the setup. The fisheye-based delay compensation is deployed to avoid motion sickness. We do not transmit the entire FoV per eye, but only the 180° FoV around the desired viewport, leading to less bandwidth usage. Finally, our AR backend brings a lot of flexibility for short term and also long term development of a remote AR system. This approach has been first described in (Kaynar et al., 2022). In this paper we add more technical information, experiments with glass-to-glass delay and data rate measurements, and a detailed outlook that includes possible improvements, extensions and industrial use cases.

3. 360° TELEPRESENCE SYSTEM WITH AR BACKEND

In this section, we introduce the 360° telepresence system interacting with an AR backend. Figure 2 summarizes the architecture of the proposed framework.

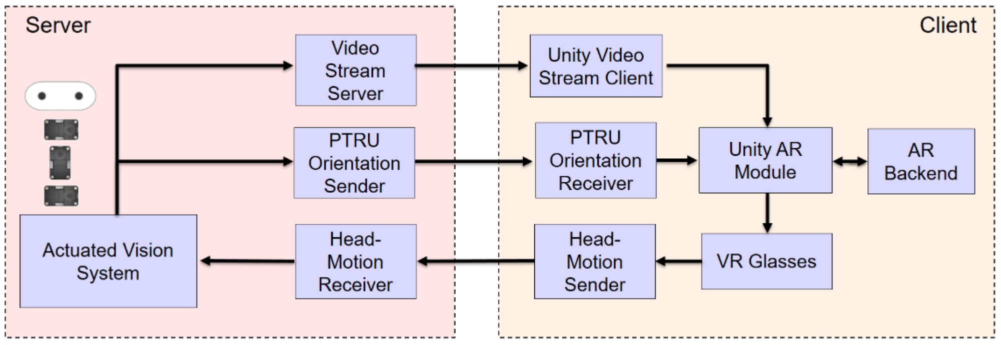


Figure 2. Overview of the proposed stereoscopic 360° AR/VR system

The server side includes an actuated camera unit, video streaming server, and modules for sending and receiving orientation data. To create a modular and scalable system, we designed each module on the server side as a separate ROS node. The client side uses Unity for the interaction with the HMD and rendering of the remote scene. The Unity system includes a video streaming client, an AR module communicating with the AR backend, as well as orientation sender and receiver modules. The user observes the remote scene enriched with holograms intuitively through VR glasses.

In the following sections, we discuss the individual modules of the proposed framework in detail.

3.1 Telepresence Server

The telepresence server provides the functionality of acquiring and streaming the desired viewport of the remote environment. We use a stereo fisheye camera system with network delay compensation, to avoid visual discomfort and motion sickness. We introduce each module in detail in the following sections.

3.1.1 Actuated Vision System

Acquiring live omnistereoscopic views of the environment is challenging. Many vision systems with multiple cameras and mirrors suffer from stitching errors, being not real-time capable, having a limited stereoscopic budget, and constrained depth perception (Aykut et al., 2019). To overcome these challenges, we use an actuated stereo vision system that allows the remote operator to control the stereo camera system with their head movement. Using an actuated system inevitably introduces a network delay. Since humans are very sensitive to delays in visual response of the system, we deploy a state-of-the-art network delay compensation method using fisheye lenses (Aykut et al., 2019).

For the hardware system, we deploy three fast servo motors for pan-tilt-roll rotation of the camera system. The resulting pan-tilt-roll unit (PTRU) is depicted in Fig. 1 and Fig. 2 on the left. The servo motors communicate with the server workstation via a serializer, which allows communication over USB. The motors follow the target pan-tilt-roll angles as received from the head-motion data of the user. Further, the motors provide their current orientation as sensed by their encoders to be used by the client for rendering the FoV of the fisheye view. The stereoscopic camera publishes the stereo images as a single frame that is obtained through a USB interface. Since we use USB communication for both the servo motors and cameras, the system is plug and play.

The fisheye lens-based network delay compensation has been shown to be effective in the literature (Aykut et al., 2019). Our stereoscopic camera has fisheye lenses to acquire a 180° FoV of the remote environment for each eye. We do not acquire and transmit the entire 360° FoV for each eye, but only a 180° FoV around the desired viewport for each eye. This yields a sufficiently larger image than the viewport rendered for the user wearing HMD. We leverage the extra image region around the user's viewport in each eye for network delay compensation. This image buffer region allows for providing an immediate rendering of the environment when the user rotates the head before the PTRU moves to the new orientation and the new image is captured and transmitted back to the viewer. For usual head rotation speeds, this scheme can compensate network delays up to few seconds. This delay compensation mitigates the visual discomfort due to rendering delays during remote collaboration or maintenance tasks.

3.1.2 Video Streaming

The video stream captured by the cameras are encoded by the x264 software video encoder (VideoLAN) and transmitted to the client using the Real Time Streaming Protocol (RTSP). The stereo camera outputs a single frame with the two frames coming from the two cameras combined. We configure the encoder for ultrafast and low-delay encoding. As an alternative streaming server, we integrated the video streaming system of TELECARLA (Hofbauer et al., 2020) to enable a dynamic video stream adaptation based on the available network transmission rate. The TELECARLA streaming system together with a multi-dimensional adaptation scheme such as deployed in the work of Hofbauer et al. (2022) enables an optimized spatio-temporal video stream adaptation.

3.1.3 Orientation Communication

For orientation data transmission, we use the ROS# package of Bischoff (2019), providing a WebSocket communication between ROS and Unity implemented in C#. The server subscribes to the ROS topics showing the HMD orientation published by the client. The PTRU follows the target angles to mimic the HMD motion. We use the HMD's motion tracking system and do not require any additional hardware.

We also transmit the current PTRU orientation back to the client side via ROS#. This information is used to place the remote scene in the correct 3D orientation for rendering in the HMD.

3.2. Telepresence Client with AR

The telepresence client consists of the client workstation and the HMD worn by the user inspecting the remote environment. We do not use any additional hardware other than the HMD's own trackers to capture the orientation. No training for the user is necessary, because of the intuitive usage of HMD. We use Unity as our main rendering framework, due to its compatibility with a wide range of devices. The modules of the proposed client system are explained in the following.

3.2.1 Scene Rendering

In this section, we detail how we created our rendering pipeline for a smooth and efficient visualization. Our aim is to visualize the remote scene in an immersive way and overlaying the holograms, which were created in world coordinates, at the correct image locations.

By designing our pipeline, we must consider the following points: The stereo camera outputs two images with fisheye projection. The viewport of the camera and the user can be at different orientations at a given time. The obtained scene must be placed to the exact orientation of the PTRU to depict the correct part of the remote environment. The hologram poses are saved in cartesian space and they must be transformed properly to the fisheye projected scene.

We start our pipeline with the video acquisition. The client application connects to the RTSP server on the server side, and starts to receive the video stream using UDP. The video stream is then decoded and rendered into the Unity scene. Since the camera system provides a single image that contains both camera views, we cannot directly render the image to each eye of the user. We first render the decoded stereo frame on a Quad object in Unity, as shown in Figure 3 on the left. To render the remote scene correctly, we must project the spherical images properly, and place them at the correct 3D orientation. In addition, we want to benefit from the image buffer-based delay compensation with the fisheye cameras.

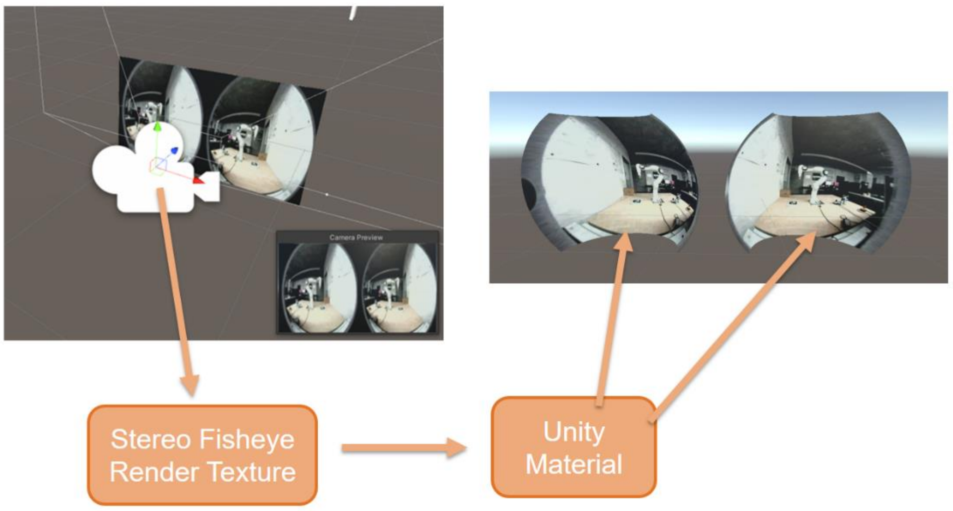


Figure 3. Projection from the stereo frame to hemispheres. Our custom render texture defines the mapping. The render texture is assigned to the Unity material used for rendering the hemispheres

The stereo images contain the fisheye projected view of the remote environment, each eye covering a horizontal FoV of 180°. We project each fisheye image onto a hemispherical shape, covering the half of the entire FoV. Hence, the stereo frame yields two hemispheres with the scene content after the mapping step as shown in Figure 3 in the right.

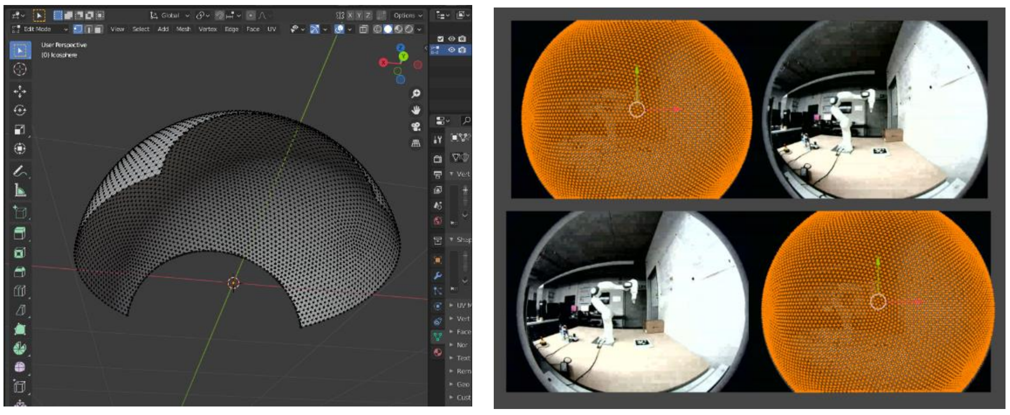


Figure 4. The hemisphere model for a single eye created in the software Blender (on the left). UV mapping from the stereo frame (on the right)

For this mapping, we create 3D hemisphere models with a custom UV mapping from the stereo frame, assuming an equidistant fisheye projection, by which the radial distance from the center on the image plane is proportional to the angle of incidence (Hughes et al., 2010). Figure 4 shows the hemispherical model designed in Blender for a single eye, on the left. The hemisphere model is cut from top and bottom since those parts are not covered by the camera sensor.

The custom UV mapping projects each pixel on the stereo frame onto a point on the hemisphere. We define the UV mapping on the stereo frame differently for the left and right hemispheres, as shown in Fig. 4 on the right, such that each hemisphere receives pixel values from the respective part of the frame. This brings the advantage of being able to decode the frame only once (on a single Quad object), but render it twice on two hemispheres. This decreases the computational load, which is of importance for having a smoothly running Unity application on a computer with decent graphical computation resources.

The hemisphere orientations in the scene must be the same as the fisheye lenses' orientation on the PTRU to show the correct portion of the remote environment. The two hemispheres are placed in the scene to have the same distance between each other as the physical cameras have. Note that we show the hemispheres distinctly in Fig. 3 (on the right) for better visibility.

Each hemisphere is observed by a Unity camera at its center that renders to the left or right display of the HMD. Thereby, we create a spherical surrounding that the user can freely observe by rotating the head. We keep the two hemispheres in separate rendering layers, and each Unity camera observes only one layer. These Unity cameras follow the HMD orientation to let the user look at any desired orientation, while the hemisphere orientations are following the current orientation of the PTRU. Since the FoV of each eye is smaller than the available FoV, we can compensate the delayed response of the PTRU and visual feedback due to network and system delays. When the user rotates the head, we have an available image region to render, until the PTRU receives the new head orientation and transmits the new frame at the new orientation.

3.2.2 AR Backend

The holograms show technical information about the scene, and are kept on our AR backend. The AR backend holds the previously recorded holograms and their 6-DOF poses with respect to the defined anchor points in the corresponding room. The holograms can be modified at any time with a mobile device visiting the same place or via a web-based editor. This makes it possible to communicate between an on-site worker and remote collaborator.

Rendering of holograms. Upon fetching the holograms from the AR backend, they are loaded into the Unity scene at the locations that correspond to the real world locations in the room. The holograms live in a separate Unity layer and are rendered on top of the hemisphere layers. As a result, they are always visible to the Unity cameras rendering the scene on the HMD.

If the holograms are simply rendered on top of the hemispheres as seen by the Unity cameras, the locations of them will be wrong on the spherical scene. This is because of the remote scene being acquired through fisheye projection and mapped on a hemisphere. In contrast, the holograms are just placed in the world coordinates and observed by a perspective camera in Unity. Fig. 5 demonstrates the result of this coordinate mismatch. While the pan angle of the HMD changes, the holograms do not stick to the correct locations but they move further away from their initial locations in the scene. This is especially visible by the hologram in the back, placed above the monitor on the wall, indicated with a red arrow on the scenes. The hologram should stick to the monitor where the red arrow originates.

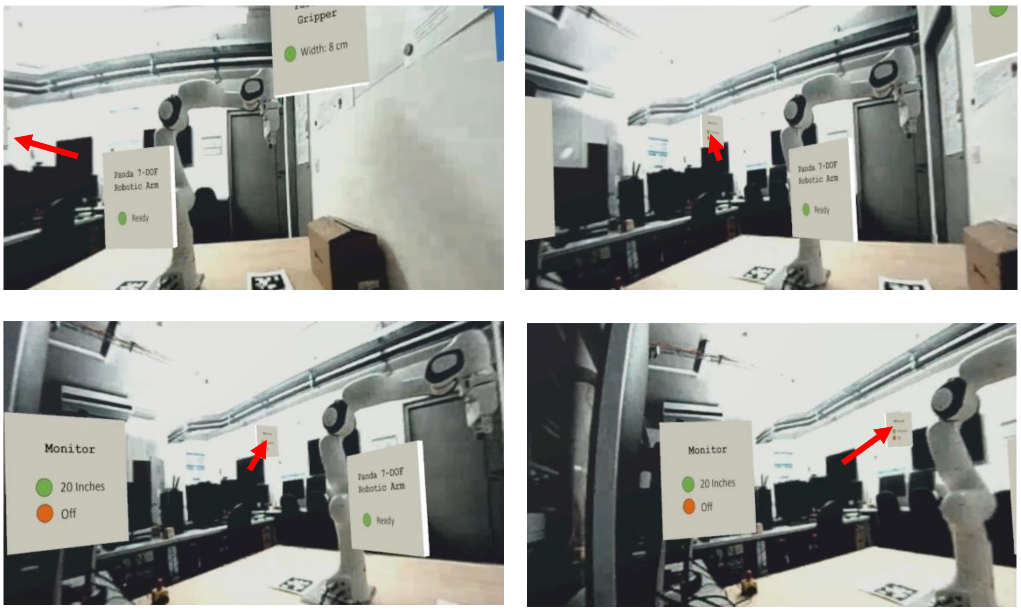


Figure 5. The Hologram coordinate mismatch shown at different pan angles

We solve this problem by adding a fisheye projection step in Unity. We only warp the holograms with this fisheye projection, for them to be in accordance with the remote scene projected on the hemispheres. The fisheye warping of holograms is shown in Fig. 6. We use a shader performing fisheye projection and add a fisheye camera in Unity that only observes the hologram layer. Then we render the view on a hemisphere model with transparent texture, resulting in the warping of holograms as if they were observed by a fisheye camera. We then place the transparent hemisphere in front of the other hemispheres, with its center at the middle of two hemispheres. Thereby, we create a spherical view for both the remote scene, and the holograms. The holograms are static in their positions, but the optical effect of AR-warping changes when the hemispheres move, because of the fisheye projection.

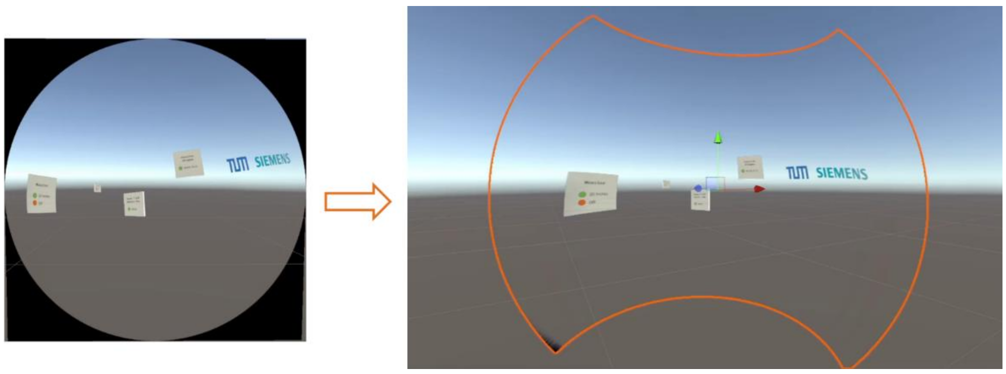


Figure 6. Fisheye warping of holograms

The resulting improvement is shown in Figure 7. It is observed that the holograms stick to their intended locations with a much higher accuracy and do not move away while the HMD orientation changes.

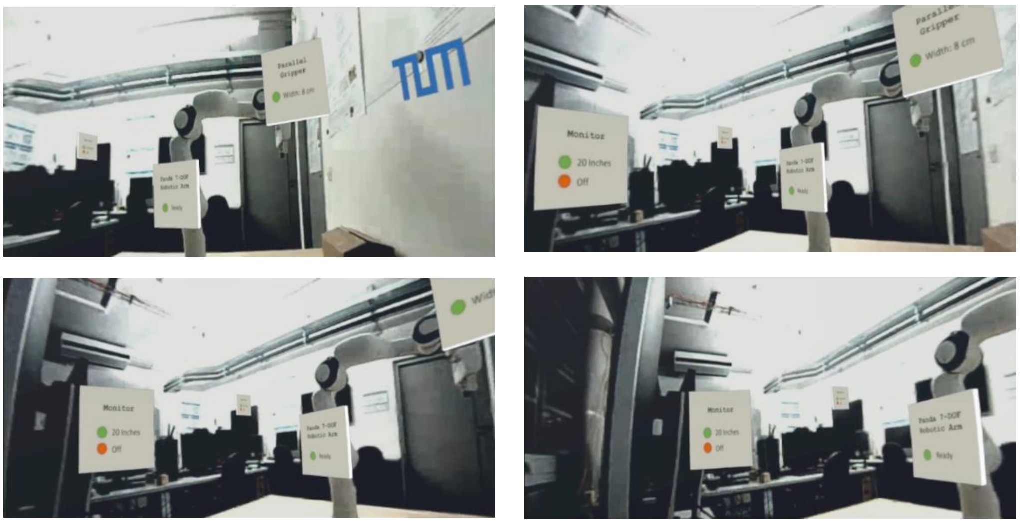


Figure 7. Corrected Hologram rendering after applying fisheye distortion

PTRU Modeling for Correct Pose of Hemispheres. The PTRU has separate joints for pan, tilt, and roll. The two cameras are placed at a specific distance to each other. Ideally, the hemispheres should follow the rotation of the cameras realistically, rather than rotating around a single point using the pan-tilt-roll angles of the motors. For a realistic geometrical representation of the PTRU, we measure the distances between the joints and model the physical PTRU. We create three hinge (revolute) joints in Unity, combine these joints with links of specific sizes, and center the hemispheres at the positions of the real cameras. This mimics the real setup by size and motion and ensures that we are moving the hemispheres in the scene in the same way as the physical cameras move on the PTRU.

Fig. 8 shows an exemplary scene with VR and AR components. The editor scene on the left shows the AR components placed in the world coordinates and visible behind the hemispheres. The projections of the AR components are rendered on the hemispheres with AR-warping. Note that both hemispheres are shown in the scene, however they are hardly distinguished due to the small distance between them. The viewport shown to the user for this scene is shown in Fig.8 on the right. In this example scene, holograms indicate important technical information on the respective objects in the environment.

3.2.3 Orientation Communication

ROS# is used for receiving and transmitting the orientation data. We create a custom joint state publisher in Unity, read the HMD orientation in Unity, and publish it to a ROS topic available to the server. Secondly, the client subscribes to respective ROS topics showing the current orientation of the pan-tilt-roll motors. We mimic this current orientation by the joints of the PTRU model in Unity. These Unity joints move the hemispheres to the same pose as the real cameras have at the moment.

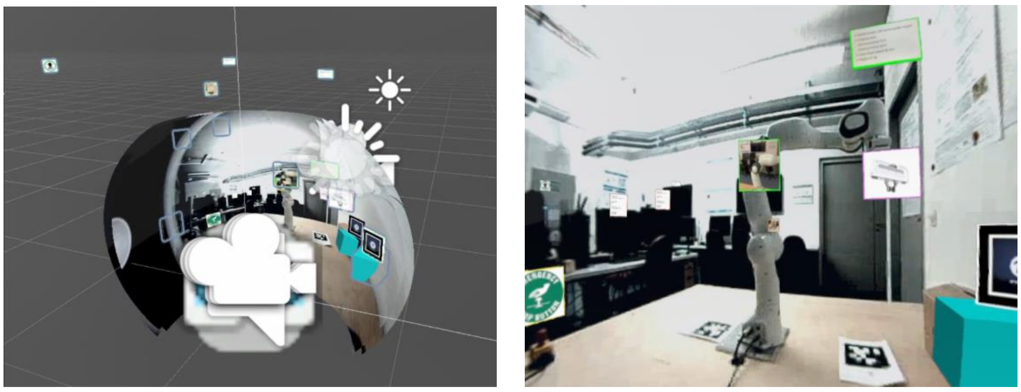


Figure 8. A scene with VR and AR components. Editor view on the left, user's viewport on the right

4. ANALYSIS AND DISCUSSION

In this section, we compare our method to previous works, by contrasting important system properties. Table 1 shows a comparison of our system against alternative systems.

We tested our system at 2560 x 920 resolution for two eyes up to 60 Hz frame rate and observed a smooth video transmission with a hardly noticeable motion-to-photon latency. As opposed to many other remote collaboration works, our camera setup is independent of a person on-site. This brings several advantages. Firstly, the remote user can observe the environment in a completely free way, rather than having to be dependent on another person's viewport on-site. Secondly, the person on-site does not have to carry a sensor setup which can be bulky and limiting the motions, hence the work efficiency is increased.

Furthermore, we do not need to use image stabilization techniques to create an independent view for the remote observer, and thereby decrease the computational complexity and avoid possible visual defects.

As opposed to methods using multi-camera stitching approaches, we use an actuated stereo camera with the vision-on-demand approach and thereby we eliminate the issues of stitching and visual defects that can be uncomfortable or misleading for the remote collaborator.

We know where the camera setup is placed in the room and do not require any 3D reconstruction or digital twin of the remote environment. Since our AR system is based on the AR backend, the holograms can be created, modified and saved at any time from any client device. This brings a lot of flexibility for remote collaboration with non-verbal cues. In addition, holograms which stay in a given position and show important information can improve the remote maintenance not only for the moment, but also in the long term.

The holograms are transmitted from the backend to the client, and the transmission is unaffected by the network quality between the server and the client. Even though the quality of the video content is decreased, the holograms can still be viewed in high quality by the client.

Table 1. Comparison of 360° remote collaboration systems

Method	**AR in VR (Ours)**	Pretlove, 1998	Wang et al, 2018	Speicher et al, 2018	Lee et al, 2018	Teo et al, 2019	Kasahara et al, 2016
Camera type	Stereo fisheye	Stereo	Stereo	360° mono	360° mono	360° mono	6 cameras
Depth perception	✓	✓	✓	✗	✗	(✓) partial	✗
No need for stitching	✓	✓	✓	(✓) (stitching in camera)	(✓) (stitching in camera)	(✓) (stitching in camera)	✗
Network delay compensation	✓	✗	✗	✗	✗	✗	✗
No need to transmit entire FoV	✓	✓	✓	✗	✗	✗	✗
Types of AR Cues	6 DOF static & modifiable	Static	Static	Pointing cues	Pointing cues	Pointing cues	Pointing cues
No need for 3D reconstruction	✓	✓	✓	✓	✓	✗	✓
Camera independent from on-site person	✓	✓	✓	✓	✗	✗	✗
No need for image stabilization	✓	✓	✓	✓	✗	✓	✗
3 DOF orientation capability	✓	✓	✗	✓	✓	✓	✓

5. EXPERIMENTAL MEASAUREMENTS

In this section, we provide the results of the measurement experiments addressing the glass-to-glass delay and the transmitted data rates.

5.1 Measurement of Glass-to-Glass Delay

The glass-to-glass delay describes the time that passes from when the photons of an event pass through the lens of the camera, until they are displayed on the screen of the viewer (Bachhuber and Steinbach, 2016). It is an important measure to evaluate the overall time required for video acquisition, encoding, transmission, decoding, and rendering. Hence, the glass-to-glass delay tells us how quickly a change on-site (e.g. a machine part moving, or an on-site collaborator moving) gets propagated to the VR user. A value near 100 ms is fine for many applications.

There are different methods to measure the glass-to-glass delay. We follow a similar approach as Bachhuber and Steinbach proposed (Bachhuber and Steinbach, 2016) and measure the glass-to-glass delay with a light source visible to the camera. In this approach, the glass-to-glass delay is computed via a light source visible to the camera, and a light sink (phototransistor) that detects the time the light change is displayed in the target display. The time difference between the light turning on and getting displayed after video acquisition, encoding, transmission, decoding and rendering gives the glass-to-glass delay estimation.

Differently from the stated approach, we use a high-speed camera that views both the light source at the server side and the target displays at the client side in the same frame. As a light source at the server side, we use a monitor that turns black and white repeatedly, effectively serving as a blinking light source. This is shown in the Fig.9 on the left, while the light source monitor is displaying white. We choose this approach to cover the most of the FoV of the fisheye cameras with the light source, because this makes it easier to determine the light change happening on the client displays, especially at the HMD’s viewport. An example frame captured by our high-speed camera that views the light source at the server side, the HMD lenses and the Unity editor screen at the client side is given in Fig. 9 on the right.

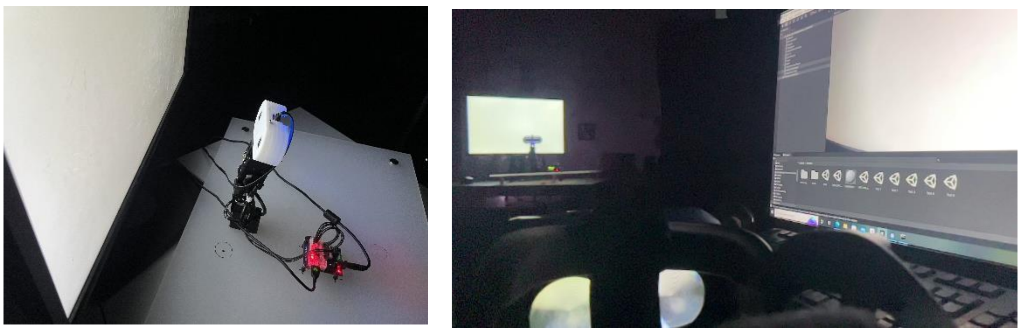
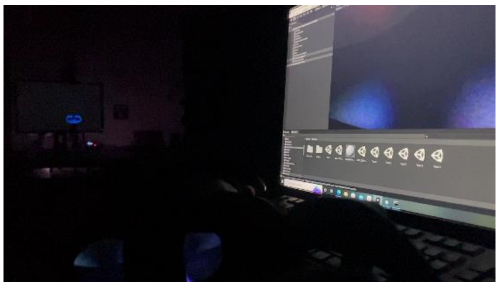
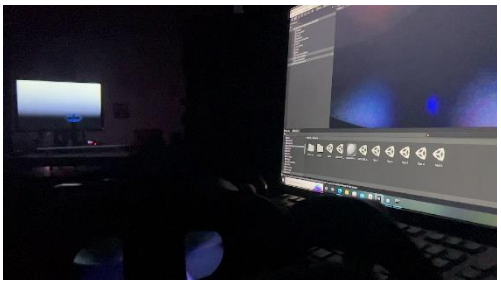


Figure 9. Glass-to-glass delay measurement setup. The light source monitor is placed in front of the PTRU (on the left). The measurement camera’s image (on the right) frames the light source monitor, the HMD lenses and the Unity editor screen to detect the time difference between the changes at each display

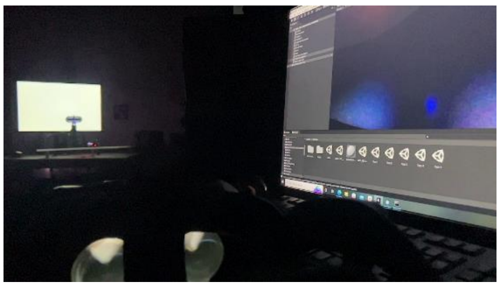
We determine the frames that show light changes, which happen at different times for the light source monitor, the HMD display and the Unity editor display. The time difference between these frames is used to estimate the glass-to-glass delay. The glass-to-glass delay is closely related to the refresh rates of the HMD display (90 Hz) and the display of the Unity editor (60 Hz). An example glass-to-glass delay observation is shown in Figure 10.



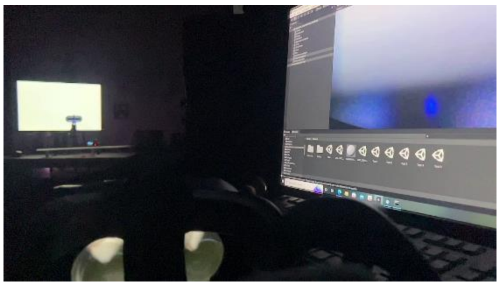
1. Light source displays a black screen.



2. Light source starts turning white. (t = 27.289 s)



3. HMD display starts turning white. (t = 27.394 s)



4. Unity editor screen starts turning white. (t=27.427 s)

Figure 10. An example glass-to-glass delay measurement

Challenges: The measurement accuracy is limited by the frame-rate of the high-speed camera. We observe that the high-speed camera has non-regular intervals, with a time difference of 4 ms or 12 ms between the frames, leading to varying accuracy for different measurements. Another challenge is detecting a single time instance for the light changes, since the displays are updated line by line and the light change is mostly shared among multiple frames.

Results: We do 43 readings and calculate the mean value. We determine the mean glass-to-glass delay from the server camera to the HMD display as 106.326 ms, with a standard deviation of 11.730 ms. The mean glass-to-glass delay from the server camera to the Unity editor screen of the client PC is found as 132.233 ms, with a standard deviation of 9.616 ms. Note that in an immersive telepresence scenario, the HMD is the display device, hence its glass-to-glass delay should be considered primarily. The difference in the delays observed at the screen and the HMD is expected, since the HMD has a higher refresh rate, and will display the changes earlier on average. The glass-to-glass delay values indicate the overall temporal delay including the time required for video acquisition, encoding, decoding, projection onto the hemispherical surfaces, rendering to the display, and the acquisition of the rendered pixels by the measurement camera. The video transmission delay can be neglected in the given results, since the server and client of our setup communicate in the same local network and the packet transmission latency is under 1 ms.

Another relevant measure of delay is the motion-to-photon delay. This is the delay between the time the VR user rotate their head, and the time the HMD displays a frame with the correctly updated perspective. A value above 20 ms will often lead to VR motion sickness. We have not yet experimentally measured the motion-to-photon delay in our system, but first explorative tests by VR-experienced users indicate that it is within an acceptable range. In our system, the motion-to-photon delay is independent of the glass-to-glass delay by design, as long as the user's head rotations are below a certain speed. Therefore, the measured glass-to-glass delay near 106 ms does not induce VR motion sickness.

5.2 Measurement of the Transmitted Data

We measure the transmitted data for the video and orientation communication for an example telepresence case. The incoming data rate to the client is fluctuating between 10Mbps for a steady camera and 15 Mbps at the moment of orientation change. This data includes both the video stream and the orientation data of the camera at the server side. The data volume increases during orientation changes, which is expected due to the differential information coding by the video encoder.

In order to discriminate the data required for the video and orientation data, we deactivate the camera and measure the incoming and outgoing data rate at the client. We observe approximately 2.2 Mbps incoming data rate from the server to the client and 320 kbps outgoing data rate from the client to the server, which are mostly constant during the telepresence session. The difference between the incoming and outgoing orientation data volumes is expected, due to the differences in the sampling rates of the orientation readings. The HMD orientation is refreshed at a rate of 50 Hz, while the actuator orientations are read at a much higher rate, around 600 Hz. Note that the sampling rate of the actuator readings can be decreased further to save bandwidth.

Computing the difference, the data rate for the video stream is found as 7.8-12.8 Mbps, for the resolution of 2560 x 920 and 60 Hz refresh rate. Note that the RTSP server is configurable for a desired target bitrate and video quality, hence the data volume for the video is highly adjustable.

6. FUTURE WORK AND INDUSTRIAL USE CASES

6.1 System Improvements

Improvement of glass-to-glass delay. In our experiments the average glass-to-glass delay is found as approximately 106 ms, which can be improved further by optimizing individual parts of the system. The stereoscopic camera currently used has a frame rate of 60 fps. The refresh rate of the HMD is at 90 Hz. Using a camera and a display with higher frame rates would improve the glass-to-glass delay of the system. We are using a software video encoder for video streaming. Using a hardware encoder would also decrease the respective delay. The rendering pipeline in Unity can also be optimized further with a specific focus on the rendering delay.

Video and orientation data synchronization. In the current design, the video stream and the orientation data are transmitted separately from each other. The client renders the video frame on the hemispherical objects once the video data is received and decoded. The hemispheres are oriented based on the orientation data of the PTRU, once this data is received at the client. Under stable network conditions, the both data arrive almost at the same time and the system runs smoothly. To increase the robustness of the system against network fluctuations, the video frame data and the PTRU orientation data can be multiplexed accordingly.

Motion-to-photon delay synchronization of VR and AR-in-VR content. We see a potential for the improvement of the stability of the AR-in-VR holograms with relation to the stereoscopic video while the user is moving their head. Explorative tests have shown that there is a small, but noticeable "swimming" effect of the holograms upon head motion. This indicates that there are two different motion-to-photon delays in our system: one for the VR content (the 360° video) and one for the AR-in-VR content (the holograms). We believe that this is due to different delays introduced in the rendering pipeline (e.g. the VR and AR contents are rendered separately, the fisheye shader for AR warping introduces additional delay), and to lacking synchronization of the video and motion data over the network. Hence, one additional area of future work is to measure these two different motion-to-photon delays in our system and then optimize both of them, assuming that they are both basically limited only by current VR system constraints. We can also optimize the difference between the two motion-to-photon delays, which should be zero in order to avoid an AR-in-VR "swimming" effect.

6.2 Possible Extensions

In the current design, we provide 360° view on a static location where the PTRU is placed manually which provides the omnidirectional view on the given location. Extending the system with translation capabilities in addition to orientation can be an interesting future work direction. One straightforward way for this extension is adding more actuation units for translating the camera system or placing the system on a mobile platform. Another interesting direction is combining the system with virtual rendering techniques to generate new viewpoints in the vicinity of the initial location where the first viewpoint is obtained. Alternatively, a 3D reconstruction of the environment can be generated from the stereo vision, and can be combined with VR rendering solutions.

6.3 Subjective Analysis

Running user studies with subjective evaluations is important while developing systems for the human usage. For this, the future work should cover user experiments to test the visual comfort and the benefits brought by our system. Some aspects to test could be the effects of stereo vision, network delay compensation, and holograms for a given industrial teleassistance task.

6.4 Industrial Use Cases

We can see industrial use cases for our proposed system in two main areas.

Firstly, situations where there is a live collaboration between an on-site user (with AR) and an off-site user (with VR). These include remote factory acceptance tests, remotely assisted commissioning of equipment, and MRO (maintenance-repair-overhaul) tasks. Here, both users are interested in seeing the real equipment from various angles, and creating and exchanging annotations in AR. This becomes particularly interesting when some of the AR annotations come from live connections to IoT systems, e.g. to show current measurement states in a process plant.

Secondly, situations where there is asynchronous collaboration between on-site users (with AR) and off-site users (with VR). One example of this is regular inspection rounds at an industrial site. For this, we assume the site is equipped with a fixed installation of omnidirectional robot camera setups, as we describe in this paper, in different parts of the site. Then, in a regular inspection round, an on-site quality inspector uses an AR device to note potential quality issues, storing photos, sounds, and personal impressions with the appropriate 3D coordinates. Later, a remote quality inspector, using VR, can review these, together with live video from the site, in order to rate the severity of the problem. Of course, the order of collaboration can be reversed: A remote quality inspector performs a virtual inspection round in VR, highlighting potential problems, and determines which issues warrant sending an on-site quality inspection to find out more details using AR.

We are exploring the use of both AR and telepresence in such industrial scenarios (e.g. see Labisch, 2021) and see a great potential in combining the two.

7. CONCLUSION

In this paper, we presented a system realizing "AR in VR" in real time. The proposed system can be used for remote collaboration and maintenance tasks. An actuated stereoscopic camera unit following the head-motion of the viewer captures the remote view. We use fisheye lenses for network delay compensation. Additionally, we present a novel scene rendering with UV mapping from a stereo frame for efficient video rendering. Holograms are fetched from a backend and warped with fisheye projection to be placed at the correct positions in the frame.

Experimental evaluation showed that the resulting system has a mean glass-to-glass delay of roughly 106 ms. The motion-to-photon delay is independent of the glass-to-glass delay by design. Explorative tests with VR-experienced users showed that the glass-to-glass and motion-to-photon delays are in acceptable ranges.

Our camera captures images with a resolution of 2560 x 920 at 60 frames per second. The low computational complexity of the system allows such high resolution and high frame rates. The data rate from the server to the client fluctuates between 10 Mbps and 15 Mbps, while the data rate from the client to the server mostly stays around 320 kbps.

Our system does not face any stitching problems as some similar methods suffer from. The depth perception quality is unaffected by the viewing orientation, since the stereo acquisition is the same in all directions.

The proposed system can be used in many remote collaboration scenarios. Both the on-site person and the remote collaborator can add or modify the holograms, sharing and saving important technical information. The remote person can discover the scene independently of an on-site person and collaborate with the on-site person efficiently.

Future work can focus on improving and extending the system for better synchronization and less noticeable visual delays. Subjective evaluation is also an important future work for the deployment of the system in real life applications.

REFERENCES

Aykut, T. et al., 2019, Realtime 3D 360-degree telepresence with deep-learning-based head-motion prediction. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, *9*(1), pp. 231-244.

Bachhuber, C. and Steinbach, E., 2016, A system for high precision glass-to-glass delay measurements in video communication. *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 2132-2136.

Bischoff, M., 2019, Ros#.

Druta, R. et al., 2021, A Review on Methods and Systems for Remote Collaboration. Applied Sciences: 10035.

Hofbauer, M. et al., 2022, Traffic-Aware Multi-View Video Stream Adaptation for Teleoperated Driving. *2022 IEEE 95th Vehicular Technology Conference: VTC2022-Spring*.

Hofbauer, M. et al., 2020, Telecarla: An open-source extension of the carla simulator for teleoperated driving research using off-the-shelf components. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 335-340.

Hughes, C. et al., 2010, Accuracy of fish-eye lens models. *Applied optics*, *49*(17), pp. 3338-3347.

Kasahara, S. et al., 2016, Jackin head: Immersive visual telepresence system with omnidirectional wearable camera. *IEEE transactions on visualization and computer graphics*, *23*(3), pp. 1222-1234.

Kaynar, F., et al., 2022, "AR in VR: Augmented Reality Cues in 360-Degree Stereoscopic Telepresence for Remote Collaboration and Maintenance." *16th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing 2022 (CGVCVIP 2022)*.

Labisch D.: Augmented Reality in Process Plants. Presentation at *Achema Pulse*, 2021.

Lee, G. A. et al., 2018, A user study on mr remote collaboration using live 360 video. *In 2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 153-164.

Lehrbaum V., MacWilliams A., Newman J., Sudharsan N., Bien S., Karas K., Eghtebas C., Weber S., Klinker G., 2022, Enabling Customizable Workflows for Industrial AR Applications. To appear in *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR),* October 2022, Singapore.

Pretlove, J., 1998, Augmenting reality for telerobotics: unifying real and virtual worlds. *Industrial Robot: An International Journal*.

Speicher, M. et al., 2018, 360anywhere: Mobile ad-hoc collaboration in any environment using 360 video and augmented reality. *Proceedings of the ACM on Human-Computer Interaction*, *2*(EICS), pp. 1-20.

Teo, T. et al., 2019, Mixed reality remote collaboration combining 360 video and 3d reconstruction. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pp. 1-14.

Wang, Y. et al., 2012, Tele-ar system based on real-time camera tracking. In *2012 International Conference on Virtual Reality and Visualization*, pp. 19-26.