Evaluation of Point Cloud Streaming and Rendering for VR-based Telepresence in the OR

Roland Fischer[1], Andre Mühlenbrock[1], Farin Kulapichitr[1], Verena Nicole Uslar[2], Dirk Weyhe[2], and Gabriel Zachmann[1]

[1] University of Bremen, Bremen, Germany
{r.fischer, muehlenb, farin1, zachmann}@uni-bremen.de
https://cgvr.informatik.uni-bremen.de

[2] University Hospital for Visceral Surgery, University of Oldenburg, PIUS-Hospital, Oldenburg, Germany
verena.uslar@uni-oldenburg.de, Dirk.Weyhe@Pius-Hospital.de

Abstract. Immersive and high-quality VR-based telepresence systems could be of great benefit in the medical field and the operating room (OR) specifically, as they allow distant experts to interact with each other and to assist local doctors as if they were physically present. Despite recent advances in VR technology, and more telepresence systems making use of it, most of the current solutions in use in health care (if any), are just video-based and don't provide the feeling of presence or spatial awareness, which are highly important for tasks such as remote consultation, -supervision, and -teaching. Reasons still holding back VR telepresence systems are high demands regarding bandwidth and computational power, subpar visualization quality, and complicated setups. We propose an easy-to-set-up telepresence system that enables remote experts to meet in a multi-user virtual operating room, view live-streamed and 3D-visualized operations, interact with each other, and collaboratively explore medical data. Our system is based on Azure Kinect RGB-D cameras, a point cloud streaming pipeline, and fast point cloud rendering methods integrated into a state-of-the-art 3D game engine. Remote experts are visualized via personalized real-time 3D point cloud avatars. For this, we have developed a high-speed/low-latency multi-camera point cloud streaming pipeline including efficient filtering and compression. Furthermore, we have developed splatting-based and mesh-based point cloud rendering solutions and integrated them into the Unreal Engine 4. We conducted two user studies with doctors and medical students to evaluate our proposed system, compare the rendering solutions, and highlight our system's capabilities.

Keywords: Telepresence · Virtual Reality · Collaborative VR · Point Cloud · Avatar · Point Cloud Rendering · Unreal Engine · Azure Kinect · Mesh Reconstruction.

1 Introduction

Telemedicine plays a major role in medicine and health care and, although it is hardly a novel concept, it has received increased attention lately. The ability to provide assistance from a distance, and collaborate without the need for being physically present, exhibits a huge potential to provide patients with better care, increase efficiency, and save costs [11,19]. There is a wide range of applications for telemedicine and the more advanced telepresence systems: from telementoring and training, remote consultation and collaboration, to remote diagnosis, surgery, and rehabilitation [30,2]. One example scenario is emergency situations with traumatic injuries where fast interventions are critical. Regularly, the dilemma is to either spend valuable time transporting the patient to specialized health care facilities, or to go to the nearest hospital although the local surgeons, especially in rural areas, might be less experienced [10]. These local surgeons could benefit from consultation or even mentoring from remote experts via telepresence systems that could preemptively be integrated into the surgery rooms [20]. Another example would be to reduce health risks by limiting the physical contact with possibly contagious patients and medical staff to a minimum; novice surgeons could consider attending surgeries via telepresence instead of being physically present in the operating room [8]. Other applications could be patient visits/ward rounds in intensive care units, or tumor conferences where normally many experts from different medical areas come together to discuss the situation and the further procedure [31].

Telemedicine in the past, and to a significant degree today too, relied mostly on classical video conferencing systems and other video-based solutions [3,16]. These systems are inherently limited by the fixed point of view, lack of depth perception, and 2-dimensional screens, preventing a distinct feeling of (tele-)presence [34,4]. Continuous technological advances and the emergence of improved, affordable VR/AR devices lead researchers to focus on 3D VR/AR-based telepresence solutions. These systems are intended to deliver a more immersive experience compared to older video-based solutions, enable more natural interactions, and provide a better spatial understanding of the objects and their surroundings [29]. Many studies showed that in such systems the users' representation through personalized high-quality avatars is fundamental [13,7,39]. To create virtual 3D representations of the scene, these systems usually employ RGB-D cameras or other depth sensors, whose data then has to be streamed to the remote location to be viewed in VR. However, VR-based telepresence systems still face several challenges: high bandwidth requirements for transmission of the data, inadequate real-time 3D reconstruction and rendering quality, or hard-to-set-up systems.

Our proposed telepresence system is designed to tackle all the aforementioned challenges with a combination of an immersive multi-user VR system with a real-time RGB-D streaming pipeline and two fast, custom point cloud rendering solutions integrated into a state-of-the-art 3D game engine. Our solution enables remote doctors to meet and interact in a virtual operating room with real-time point cloud avatars as well as assist in operations that are live-streamed and visualized in the virtual room in 3D. Our proposed system is capable of handling multiple cameras per location, as our streaming pipeline includes efficient real-time compression and filtering algorithms. Furthermore, we integrated an easy-to-use registration, i.e. extrinsic calibration, method. To evaluate the benefits of VR-telepresence systems in general and ours specifically, we conducted two user studies with doctors exploring possible use cases, comparing the different rendering solutions, and investigating aspects such as spatial and social presence, realism as well as preference. To summarize, our contributions are:

- A multi-user VR-based telepresence system implemented in the state-of-the-art game engine Unreal Engine 4 (UE4) with a prototype for avatar face reconstruction.
- A modular low-latency multi-camera RGB-D streaming pipeline including filtering, denoising, and compression of RGB-D data, which is easy to extend.
- Custom splat- and mesh-based point cloud rendering solutions and a accompanying user study to compare the two methods.
- An extensive qualitative evaluation of the proposed system as well as a user study exploring clinical benefits and relevant aspects such as spatial and social presence.

2 Related Work

Many VR/AR-based telepresence systems have been proposed in the past, some still rely on video feeds that can be augmented [35,17], others do make use of real-time point clouds and 3D reconstruction [18]. For instance, Boehlen et al. [5] recently presented a real-time telepresence system intended for usage in caregiving that uses multiple RGB-D cameras as well as point cloud visualization and Anton et al. [2] proposed an augmented telemedicine platform for real-time medical consultation in which the patient is captured via an RGB-D camera, a remote expert can assist using a 3D display, and annotated feedback is sent back to the patient-side. The former system doesn't support multiple users or avatars and doesn't consider compression to reduce the required bandwidth. The latter uses only one RGB-D camera and a limited 8-bit YUV-based compression. Thoravi Kumaravel et al. [36] successfully demonstrated the benefits of immersive telepresence for remote teaching of physical tasks via a bi-directional Mixed-Reality system that combines AR and VR. In this system, point cloud hologlyphs visualize the remote scene, although only with 10 Hz on the AR side. Gasques et al. [14] developed a collaborative mixed reality system based on multiple color and depth cameras for live 3D scene scanning and reconstruction, OptiTrack marker-based tracking which is used for registration and annotations, and AR and VR devices. The focus of this system lies more in tracking and interaction, as the depth images are transmitted unprocessed to the remote location. Based on the work by Gasques et al., Roth et al. [31] recently presented another mixed reality teleconsultation system that is intended for telepresence in ICUs. In their system multiple RGB-D cameras are mounted on the ceiling and, as commonly done, connected to dedicated PCs ("capture nodes"). The data is then compressed and transmitted to the remote location where the point cloud is computed but eventually rendered as surface mesh via a custom shader. One drawback they report is the high latency of 300-400 ms.

For telepresence applications, it is of high interest to constrain the required bandwidth to a reasonable level which makes real-time streaming of RGB-D sensor data and point clouds a difficult task. Therefore, one important but often still lacking aspect of such streaming systems is efficient data compression, particularly of depth data. Not only must the amount of data be reduced as much as possible but also as quickly as possible. Often this is only achieved by sacrificing the quality, which is problematic in medical contexts though. The two main approaches in which most of the previous dedicated research on RGB-D compression can be split into are 2D approaches using image- and video compression techniques that compress the individual color- and depth images [38] and 3D approaches which directly compress the point cloud. The latter often rely on hierarchical subdivision using octrees [24] and tend to be slower and less effective compared to 2D approaches if the reconstruction quality should remain high or even be lossless [22]. To achieve convincing results, the image- and video compression techniques need to be adapted to depth images and their specific characteristics though [28]. A comprehensive overview of the recent work in this department was recently provided by Cao et al. [6]. Even with the ongoing work in this field, the problem is far from solved.

Another important task is to produce high-quality 3D visualizations of the RGB-D data. One persistent obstacle is the inherently noisy output of the depth sensors, even the newest ones like the Azure Kinect suffer from temporal noise and effects such as the flying pixel and multipath effects [37]. Since the emergence of low-cost RGB-D cameras, much research was done to enhance the depth images by proposing different denoising, inpainting, and filtering approaches [1,21,12]. Though, the problem is still relevant today, as the proposed solutions often have difficulties with dynamic content, e.g., in form of ghosting artifacts, or take too much time for real-time application.

To eventually visualize point clouds and, specifically, point cloud and 3D reconstructed avatars, different techniques were proposed. While, in principle, very high-quality representations can be achieved via offline scanning and elaborate reconstruction techniques, these methods are not suitable for real-time telepresence applications [23]. Dou et al. [9] presented an online performance capture system with advanced volumetric 3D reconstruction achieving real-time speed, however, multiple high-end PCs were necessary to achieve 30Hz which leaves no room for other necessary tasks in a telepresence system such as compression, rendering of multiple users, and rendering the scene itself in VR. Similarly, Orts et al. [27] were able to stream high-quality 3D reconstructed avatars in real-time. The avatars are produced via temporal-volumetric fusion of the data of multiple RGB-D cameras, however, the proposed system is computationally highly demanding, requires a 10 Gigabit connection, and is complicated to set up. On the other hand, Gamelin et al. [13] showed that even simpler and faster point cloud visualization techniques such as splatting are sufficient to outperform preconstructed animated avatars in collaborative spatial tasks. Yu et al. [39], too, compared point cloud avatars with tracked mesh avatars in their telepresence

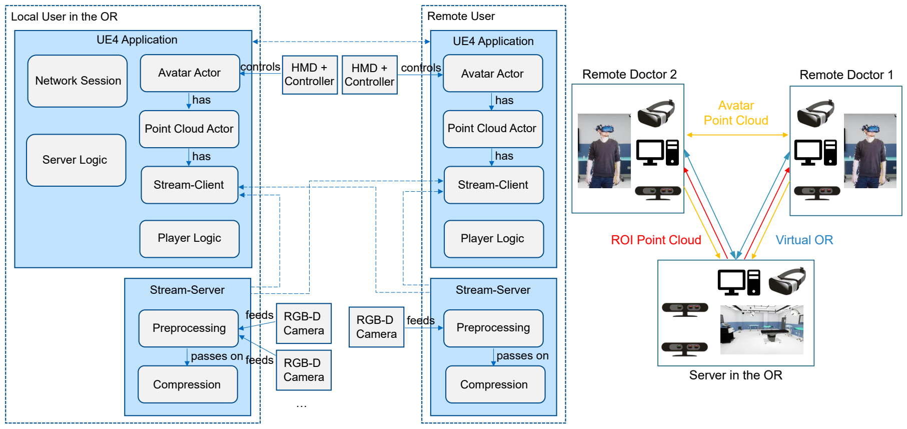

Fig. 1. Left: System architecture of our application. Right: System setup and communication channels between the server in the OR and the remote users.

prototype and found the point cloud avatars to be superior regarding perceived co-presence and social presence.

3 System Overview

In this section we present our telepresence system for remote consultation and collaboration in healthcare. In our system, the doctors and the patient in the physical operating room as well as the remote experts in VR are visualized via accurately registered live-streamed point cloud-based 3D representations. To provide high-quality graphics and robust network components, we decided to use the Unreal Engine 4 as a basis. This also has the benefit that a lot of basic aspects such as collision handling and different Virtual Reality Headsets (HMDs) are supported out-of-the-box. A core pillar in our system architecture is the RGB-D streaming pipeline that we realized not with the Unreal Engine, as the engine and its network components specifically are not suited for low-latency transmission of huge data loads. Instead, the RGB-D data is streamed via custom client-server connections that we implemented using C++ and CUDA; we integrated these into our Unreal Engine application. This is illustrated by the system diagram on the left side of Fig. 1 which shows our system's components in more detail.

The image on the right side of Fig. 1 illustrates the general setup and the communication channels between the participants: The server, which acts as a client too, is located in the operating room, has multiple RGB-D cameras connected to it, and hosts the virtual operating room scene. Remote doctors connect to the server and receive the point cloud visualization of the physical scene – the region of interest (ROI) of the OR – from the server. If the users have an RGB-D camera for a personalized point cloud avatar themselves, they directly broadcast the corresponding data to all participants. The application's

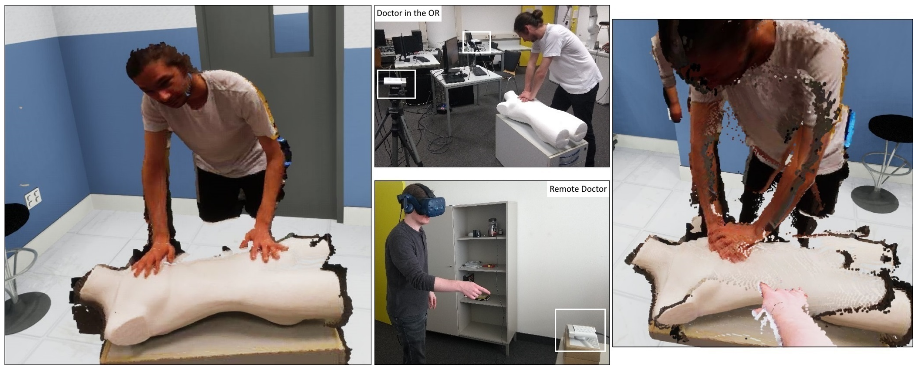

Fig. 2. Example of our system. Left: Live-streamed point cloud visualization of the local doctor in the real OR, seen from the perspective of a remote doctor. Middle: Physical environments of the local (top) and remote (bottom) doctors with RGB-D cameras (see white rectangles). Right: Point cloud visualization of the local doctor using two cameras to minimize occlusions, and part of the remote doctor's self-avatar.

network traffic, apart from the RGB-D data, however, is always routed via the server.

An example of our telepresence system in action can be seen in Fig 2. The left image shows the doctor in the real OR as a live-streamed point cloud visualization in the virtual OR, seen from the viewpoint of a remotely connected doctor. The right image shows the same doctor rendered with the point clouds from two cameras which helps to minimize occlusions. The middle images illustrate the physical environments of the local and remote doctors with their respective RGB-D cameras.

3.1 Multi-User VR Environment

The central part of our system is a virtual operating room in which all the RGB-D data gets streamed and rendered, and in which the remote doctors can meet using HMDs, see Fig. 3 (left). The network architecture is based on Unreal's session system. After starting the application, the users arrive in a lobby where they can start a session or join an existing one. The focus of our system lies in users making use of HMDs to have an immersive VR experience and having real-time personalized point cloud avatars. However, for the case that the required technology is not available, we made sure the system is usable with a mouse and keyboard, too, and integrated flying mesh avatars as a fallback. For VR locomotion the room scale system is used in which the users can physically walk to move. We also provide a teleport functionality for cases where the physical space runs out. Research showed that this locomotion metaphor exhibits the least risk of inducing cybersickness. Also, to not confuse other present users, a simple particle effect is shown to indicate the deliberate nature of the teleport. Other features of our virtual operating room are a virtual monitor to show

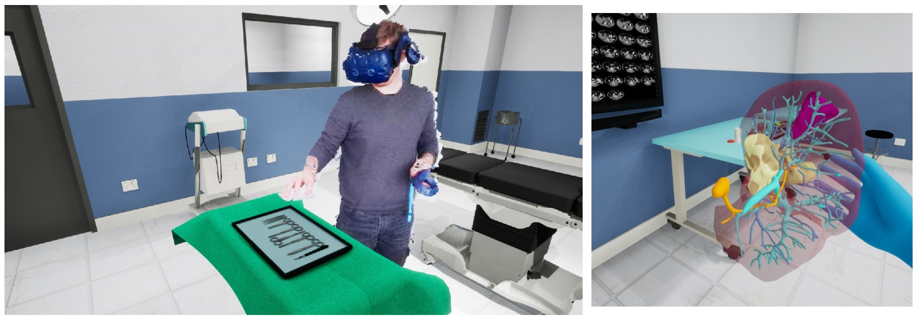

Fig. 3. Our virtual operating room with a point cloud avatar of a remote user on the left and an interactive virtual liver on the right.

medical image data and 3D organ meshes modeled based on real patient data, in this case, livers. The fully synchronized organ models consist of separate parts for arteries, tumors, and a half-transparent outer shell and can be grabbed and inspected by the users, as can be seen in Fig. 3 (right). Also, the organ data is quickly replaceable to represent new cases.

3.2 Point Cloud Streaming

In this section, we describe the point cloud streaming pipeline of our telepresence system, which is one of its core parts. Instead of implementing everything from scratch, which would be tedious and time-consuming, we opted to use "Dyn-Cam: A Reactive Multithreaded Pipeline Library for 3D Telepresence in VR" by Schröder et al. [32] as a base and extended it for our needs. Generally, the Dyncam library provides a good foundation, as it has low latency streaming capabilities, which are crucial for telepresence in VR, and is easily extendable. However, we found some aspects such as compression and rendering to be lacking and decided to extend or replace them. We also adapted the architecture so that a single server-client connection can handle multiple cameras to reduce overhead and integrated functionality to record point cloud sequences and replay them later without the need for cameras to be connected. In Fig. 4 you can see an illustration of our final streaming pipeline. One or multiple RGB-D cameras are connected to the streaming server and will be processed individually. We use the new Azure Kinect RGB-D camera from Microsoft, as it has a high resolution and precision, hardware synchronization, and uses the TOF principle which is very suited for indoor use. The first step of our pipeline is the preprocessing of the color and depth images, which consists of lens correction, cropping, and filtering algorithms including background subtraction, and morphological filters for hole-filling and denoising. Then the images get compressed and transmitted to the client where they will be decompressed. We decided to compress and transmit the color- and depth images instead of point clouds, as this enables us to use more efficient image- and video-based compression algorithms. This means that

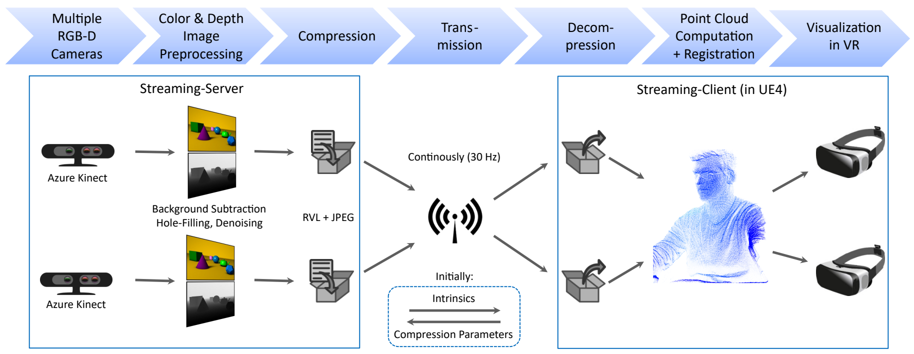

Fig. 4. Our point cloud streaming pipeline. Color and depth images of multiple cameras get individually preprocessed and compressed server-side and then transmitted to the clients where the point clouds get computed and registered before rendering.

the point cloud will only be computed client-side. For this, the camera's intrinsic data will be transmitted once too. Lastly, the point clouds of multiple cameras will be registered to each other and to the virtual scene and then rendered.

We have implemented the filtering algorithms using CUDA to minimize latency. For background subtraction, there are two options. The first one is to set a simple depth threshold per camera to exclude points from rendering (depth set to zero). As a second option, the user can do a one-time recording of the scene, e.g., the empty operating room. In this case, the pixel-wise minimum valid value will be stored and acts as the threshold to distinguish between foreground and background. For hole-filling and depth denoising, we did extensive experiments with various filter algorithms, such as optical flow, Navier-Stokes-based inpainting, local regression, etc, but found them to be too slow, or the provided improvements were not significant enough to warrant the additional performance cost. Thus, we settled for faster solutions that achieve a good trade-off in this regard. Hole filling is done via multiple iterations of median-based morphological filtering and denoising using an adapted Kalman filter. As the Azure Kinect camera masks off the corners of the depth images with non-usable data, we added the option to do a cropped transmission and save bandwidth. Regarding the compression of the depth images, we extend the solution present in Dyncam (quantization plus LZ4) by integrating the H.264 video codec, and multiple efficient lossless algorithms, i.e., an ANS coder, RVL, and Zstandard. We found that even after compression the depth images are responsible for most network traffic while the color images are sufficiently small by just using jpeg compression. Therefore, we adapted the integrated lossless depth compression algorithms to achieve higher compression ratios by adding temporal delta compression. As with all compression and streaming systems, at some point, a trade-off has to be made between the required bandwidth and the computational speed. To allow the users to adapt this to their local capabilities, i.e., hardware power and network bandwidth, our design allows the client-side user to select the compression algorithm and parameters which will be used for their individual connection. For example, one user could choose to use RVL compression, which is fast and efficient, while another user may have a slower internet connection and opt for a different compression technique with a higher compression ratio at the cost of increased computational costs and lower speeds. The modular design of our pipeline also makes it easy to implement other and even more efficient compression algorithms in the future.

3.3 Point Cloud Registration and Rendering

To be able to have individual point cloud avatars for remote users, a registration procedure between the RGB-D camera and the VR coordinate system has to be done. When using multiple cameras, these have to be registered to each other too. For these registration tasks, we use the novel method by Mühlenbrock et al. [25]. It uses a grid-like registration target that is visible in the depth images to register multiple RGB-D cameras with each other and the VR coordinate system. This registration method proved to be very quick and easy to use. In the virtual scene, we have a hierarchy of actors in which each one is responsible for rendering one point cloud/camera. To account for the registration occasionally being slightly off, we allow the user to manually tweak the transformation in-game via sliders.

Of paramount interest is the fast and visually pleasing rendering of the point clouds. The Unreal Engine historically does not natively support point cloud rendering, however, by now, there is a publicly available point cloud rendering plugin “LiDAR Point Cloud”[1]. We experimented with the plugin but found it to be too slow with dynamic point clouds and, therefore, not suitable for our application. Our investigations indicate that the reason for the poor performance is that the plugin was designed to handle huge but static point clouds – it builds a spatial acceleration data structure intended for LOD. With dynamic point clouds, this costly operation would have to be done in each frame. The rendering solution provided in Dyncam was not convincing to us, both visually and from a performance point of view. We also considered implementing more complex volumetric reconstruction techniques similar to the ones in [27] and [9] but eventually decided against it, as they are computationally highly demanding, and we prioritized keeping the latency, which is critical in VR, to a minimum. Therefore, we have developed two different and very quick rendering solutions that we both integrated and tested in the Unreal Engine.

The first method is splatting-based but uses Unreal’s new Niagara particle system. To get the point cloud positions and colors from the CPU to the GPU, we adopted Dyncam’s approach of using two dynamic textures. In our case, all cameras from one user share a single texture with the size of 2048^2 which is sufficient for at least 11 cameras. Via Niagara module scripts we then calculate the UV coordinates based on the particle ID and forwarded parameters such as the point count per camera, texture size, etc., exploiting the fact that the point clouds are ordered. For point clouds meant to represent avatars of VR users, we additionally filter out all points that exceed a set distance from the

[1] https://www.unrealengine.com/marketplace/en-US/product/lidar-point-cloud

center of mass which we compute dynamically via the HMD position. Also, points representing the HMD are filtered out for the local avatar's user in a similar fashion to prevent occluding the vision. When using splatting methods, the size of the points is important. To minimize holes and overlaps, we compute the diameter of each point based on the distance recorded from the sensor, as, because of the parallel projection, the density of points decreases with the distance. Another effect to consider is that surfaces perpendicular to the line of sight of the camera get sampled with a significantly lower density which results in bigger holes. To account for this fact, we also dynamically compute the local density of points, which is computationally cheaper than approximating their normals, and adjust the diameter accordingly. As blend mode in the material we use the masked mode instead of the translucent mode to circumvent the costly depth sorting.

Our second rendering method is based on a very fast mesh reconstruction and is intended to prevent visible holes between individual points altogether and instead provide continuous surfaces. Fig. 5 shows a comparison of the two renderers. As can be seen, in some instances the individual points are still clearly visible with the splatting method. With our mesh reconstruction method, we can again exploit the fact that the point cloud is ordered and establish a one-to-one relation between point cloud points and the vertices of a plane mesh. At start-up, we once automatically create a plane mesh with the exact vertex dimensions and structure as the depth image on which the point cloud is based. E.g., a rectangular grid-like pattern of 640×576 vertices. At runtime, we then make the point cloud positions and colors available to the mesh's material, again, via dynamically updated textures. In the material, the vertices get transformed via Unreal's WorldPositionOffset-function according to the corresponding point cloud positions in the RGB-D camera's local space and the world-transform. We wrote custom shader nodes to exclude triangles from being rendered (alpha set to zero) if, based on the original position, the point was invalid, or one of the triangle's edges is too long. This prevents long, stretched triangles between foreground and background objects.

4 Evaluation and Results

4.1 Performance

To quantitatively evaluate our system's performance, we measured the time needed for filtering, compression, and rendering as well as the frame rate with which camera updates can be processed. All performance measurements were done using a PC with Windows 10, an Intel i7 7800x processor, 16 GB of main memory, and an Nvidia GeForce 2070 graphics card. As HMD we used the HTC Vive Pro Eye with a mounted Facial Tracker. Our application was developed using the Unreal Engine 4.26. All measurements were done without background subtraction to maintain the full worst-case workload.

First, we evaluated the speed of the filtering step of our pipeline (see the top left part of Table 1). As a whole, it took 1.34 ms to filter an un-cropped

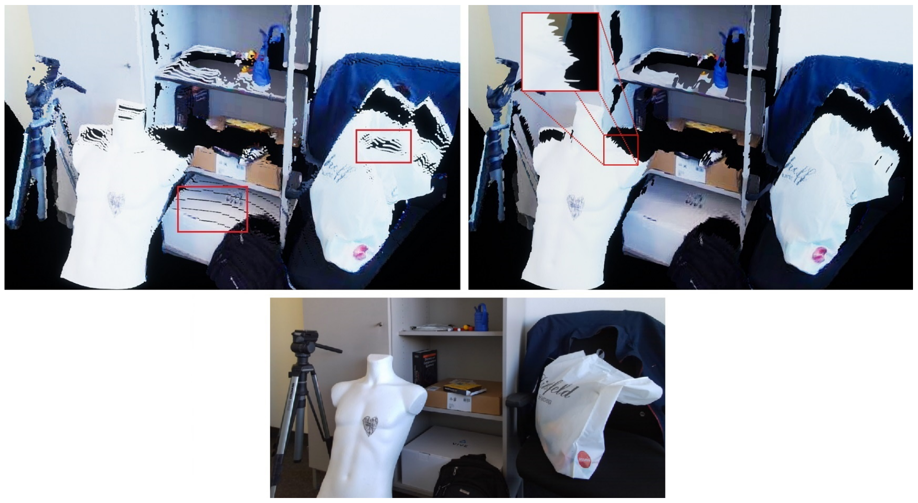

Fig. 5. Comparison of the two point cloud renderers, splatting on the left, fast mesh reconstruction on the right. Both look quite good but the splatting method still has visible holes in some areas, see the red rectangles. With the mesh however, object borders can look jagged (see highlighted region). The bottom image shows the captured scene.

depth image (640×576 pixel), of which 0.59 ms were spent on hole-filling. Next, we measured the time for compression and decompression of the registered color image using JPEG: each took 1.5 ms. Using the cropped transmission (540×476 pixel) to discard the Azure Kinect camera's unused border areas resulted in only 1.08/1.22 ms being needed. Using H.264 (preset: ultrafast, tune: zerolatency), the computation was significantly slower: 7.72 ms for compression and 4.84 ms for decompression of the full-sized color images, and 11.04 ms and 9.4 ms for the depth images, respectively. Compression and decompression of the un-cropped depth images using the RVL method, which we found to be the most efficient, took 1.76 ms and 1.19 ms, respectively. Cropping reduced the time needed to 1.31 ms and 0.987 ms, respectively. An important detail to note is that the (de-)compression of color- and depth are done in parallel, meaning that the time won't stack on top of each other. The results show that in our pipeline, preprocessing and compression are very fast and can be accelerated even further by cropping the unused borders of the Azure Kinect's depth images, although the speed-up doesn't reach its theoretical potential (19-28 % less time for 30 % fewer pixels).

After these individual measurements, we evaluated the overall performance by measuring the more comprehensive metrics of the point cloud (PC) update rate that was maintainable as well as the eventual fps/frame times in VR. We measured the point cloud update rate by calculating the delta time both in our streaming server application as well as in the UE4 telepresence application that received and rendered the data. Throughout all cases, even using two cameras

Table 1. Performance measurements of our application.

Task Duration	Time (ms)	
	Full	Cropped
Filtering	1.34	-
Color Comp. (jpeg)	1.5	1.08
Color Decomp. (jpeg)	1.5	1.22
Color Comp. (H.264)	7.72	-
Color Decomp. (H.264)	4.84	-
Depth Comp. (tRVL)	1.76	1.31
Depth Decomp. (tRVL)	1.19	0.987
Depth Comp. (H.264)	11.04	-
Depth Decomp. (H.264)	9.4	-
Latency		
VR Round-Trip Time	22-29	-
PC MotionToPhoton	120-150	-

Compression	Image Size (kB)	
	Full	Cropped
Color (JPEG)	21.9	18
Color (H.264)	15.85	-
Depth (RVL)	208	-
Depth (tRVL)	96.6	81.78
Depth (H.264)	59.69	-

Rendering Perf. (ms)	PC	Mesh
VR Frametime (1 Cam)	10.9	8.5
VR Frametime (2 Cam)	15	13

sending in full resolution and being in VR at the same time, our system was able to maintain a delta time of 33 ms which corresponds to the 30 fps capturing capability of the Azure Kinect cameras. The final performance in the packaged VR application was not only dependent on the number of cameras and the rendering technique but also heavily dependent on the general graphics settings the scene was rendered with ("scalability settings" in Unreal). Using the mesh rendering technique, Unreal Engine's "high" graphics preset, and the full depth resolution, we achieved a frame time of 12.5 ms with one camera and 15.5 ms with two. Setting the graphics preset to "low", we were able to reach 8.5 ms and 13 ms, see the bottom right part of Table 1. We found the splatting technique to be slower than the mesh variant, achieving only 10.9 ms and 15 ms under the "low" graphics preset. Important to note here is that the rendering resolution was held constant throughout the presets, and the graphics preset had no effect on the visualization of the point cloud rendering but only on the surrounding scene, most noticeably on reflective materials, anti-aliasing, and ambient occlusion. As can be seen from the performance measurements, our pipeline is very efficient throughout all stages and enables VR performance even with just a single PC per location. Both of the rendering techniques are very quick to compute, but especially the mesh version is highly efficient. As the performance scales with the number of cameras, at some point (e.g., 4+ cameras), more powerful hardware or a second PC would be needed to maintain the real-time VR performance, though.

4.2 Network

Regarding network performance metrics, we measured the round-trip time for interactions in VR, and the motion to photon latency of the whole pipeline, see the bottom left part of Table 1. The round-trip time – the time it takes for a client-side action to be transmitted to the server and back to a client – was between 22 ms and 29 ms., depending on the tick rate the server and client could achieve. The tests were conducted with 2 PCs in a local network. With greater distances, e.g., different cities, the time will likely be higher. The time between the camera capturing the scene and it being rendered on the display – the motion to photon latency – was measured by pointing a camera in such a way that both the physical scene and the display were recorded by an external camera. By analyzing the videos frame-by-frame, we found a latency of 4 to 5 frames which corespondents to a 120-150 ms delay. However, roughly half of this is caused by the camera itself, as it is reported that delivery of the raw images by the Azure Kinect SDK needs $\sim$ 75 ms, depending on various parameters. We couldn't find any significant differences in delay for a varying number of cameras, between the rendering methods, or different graphics presets, which may also be because the external camera that we used for the measurement itself only captured with 30 Hz. Although the measurements were not highly precise due to the limited temporal resolution of the external camera, the results show a rather low delay for such a system.

Lastly, we measured the compression efficacy and bandwidth required to transmit the RGB-D Data. The color images with an original size of 1440 kB were compressed with JPEG to 21.9 kB (on average) with a compression ratio of 66. Cropping further reduced the size to 18 kB, as can be seen in the top right part of Table 1. With H.264, the color images were compressed to 15.85 kB, and the the depth images to 59.69 kB. However, the size and image accuracy are heavily dependent on the parameters; we used CRF values of 20 and 10 for the color and depth images, respectively. For depth images, we found the RVL algorithm to be a good trade-off between speed and compression ratio. Using it, depth images were losslessly compressed from the original 720 kB down to 208 kB with a compression rate of 3.46. With our temporally extended RVL, the average compressed size shrunk to 96.6 kB with a compression rate of 7.45, though the achievable compression here strongly depends on the amount of motion in the scene. With the cropped transmission, the size was further reduced to 81.78 kB. With the 30 images per second that the cameras provide, our system requires per camera 3,555 kB/s in full and 2,993 kB/s in cropped mode, which corresponds to 23.4 and 27.8 Mbps. This is a very good result considering that the depth images are transmitted losslessly and the high computational speed the compression runs on. Naturally, using lossy compression algorithms such as H.264, the bandwidth could easily be reduced further at cost of higher latency, if need be. However, we noticed visible artifacts on H.264-encoded depth images.

5 User Studies

To demonstrate the capabilities of our telepresence system and evaluate it regarding crucial aspects such as visualization quality, realism, and spatial- and social presence, we conducted two user studies with doctors and medical students in a hospital.

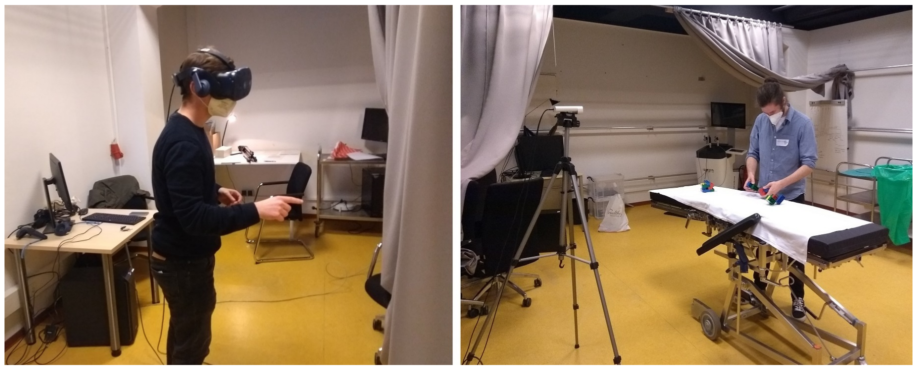

Fig. 6. Our telepresence system in action during the studies: on the left the remote doctor and on the right the live-captured operating room scene.

5.1 Study 1: Qualitative Feedback, Presence, and Preference

The goal of the first user study was to get general feedback from the doctors, evaluate it regarding relevant aspects like presence, and see how beneficial the telepresence system could be in clinical practice. The study was conducted with $N_1 = 12$ doctors and medical students with varying amounts of experience in the operating room and with AR/VR. The experimental setup for this study was as follows: In a room similar to an operating room, we divided the space in half. In the one half, a PC was set up that acted as the server for the streaming pipeline and host for the VR session. Connected to the PC was an RGB-D camera facing an operating table and a staff member acting as a surgeon, see Fig. 6 (right). In the second half of the room was a second PC with an HMD which the participants had to put on and join the session in the virtual operating room where they could see the operating table and staff member as a live-streamed point cloud, see Fig. 6 (left). The PCs were connected via a local network.

The task for the participants was to observe the staff member and help him with specific spatial tasks he had to perform with interlocking bricks (Lego). We did use interlocking bricks for this study, as their shape, size, and inherent ability to be combined into various more complex structures makes them suitable to recreate spatial tasks done by surgeons. First of all, the staff member did work alone so the participants could familiarize themselves with the VR experience and the scene. After roughly one minute, the staff member started asking questions and presenting problems to facilitate interaction with the participant and steer his attention to individual bricks. Questions were, for example, how many bricks of one color were used in a construction, how many studs of one color were visible, or how a specific construction could be built with a set amount of available bricks. After the VR experience, which lasted roughly 8 minutes, the participants had to fill out a questionnaire. The questionnaire consisted of a demographical part (age group, sex, experience in the operating room, experience with AR/VR), the Igroup Presence Questionnaire (IPQ) [33], which is

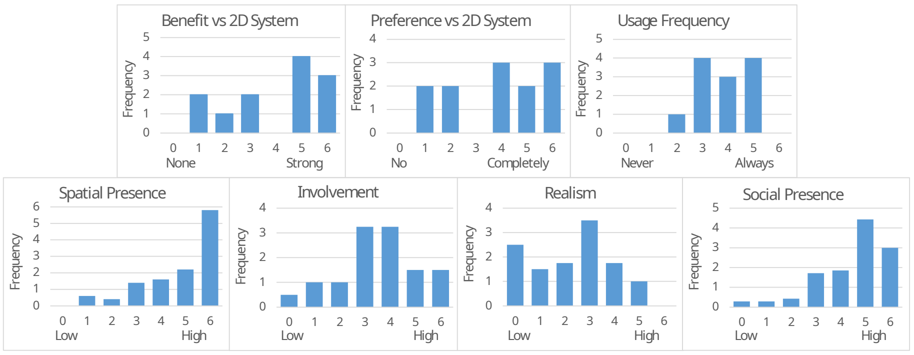

Fig. 7. Results of our first user study (in Likert scores, higher scores are better). Especially the spatial- and social presence scored very well, involvement and realism mediocre. The users see moderate to high benefits of the system and most of them would like to use it.

split into the three subscales spatial presence, involvement, and realism, and a social-presence part taken from Nowak et al. [26]. Additionally, we added various specific questions: about cybersickness; if the participants see benefits of our system compared to more traditional video-based systems; if they would prefer it to those systems; how often they would want to use our system. For all questions, we used a 7-point Likert scale (1-7, but shifted to 0-6 for the evaluation; higher scores are better).

The results of this study can be seen in Fig. 7. Note that the scores can be fractional in some cases, as the IPQ subscales are aggregations of multiple questions. Our system scores especially high regarding both spatial- and social presence. 58 % of the participants stated that they had a strong feeling of being present in the virtual world (Likert scale >= 5), the most often given score even being the maximal one. For the rest, the feeling was still moderately pronounced. Similarly, 62 % stated that they had a strong feeling of actually being in the same room with the other person/having a personal meeting with another real person. Only 5 % definitely had not the feeling. We also found that our system fares very well with cybersickness, as no participant had a significant occurrence of it, and 75 % had nearly none to none. The results for the involvement and realism subscales of the IPQ are moderately good, most participants gave scores relating to “somewhat captivated by the virtual world” or “moderately real world”, although, especially on the question “The virtual world seemed more realistic than the real world” of the IPQ, 75 % gave the minimal score dragging down the subscale. One possibility for this particularly low result could be that the participants took the question too literally, and the question may be not that appropriate in our context. The other questions of the realism subscale[1] scored significantly better. In the end, 25 % of the participants would attest moderate advantages and 58 % even strong advantages to our system compared to

[1] http://www.igroup.org/pq/ipq/download.php

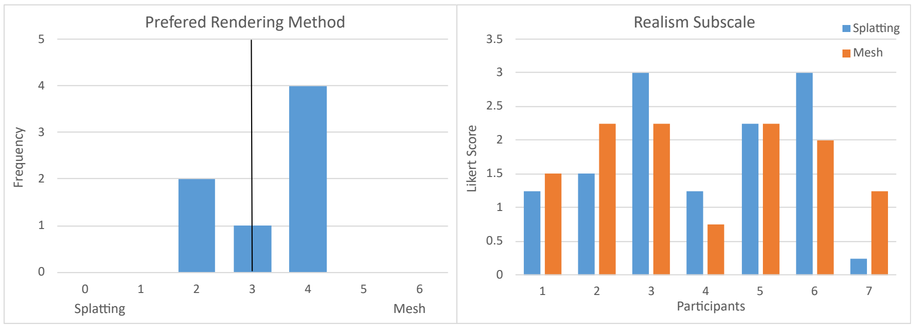

Fig. 8. Results of our second study in which we compare the two rendering methods. The left diagram illustrates that the users have no definitive preference, some slightly prefer the splatting (left side of the x-axis), others the mesh rendering (right side of the x-axis). The right diagram shows that also the realism scores fairly similar between the two rendering methods (higher scores are better).

more traditional videoconferencing systems. Also, 83 % would prefer this system at least somewhat over teleconferencing systems, and 33 % would use it on all possible occasions, while the other 77 % at least sometimes. The results for the realism subscale as a whole are in-line with the comments made by some participants during and after the study – that the point cloud visualization is still somewhat grainy and low precision.

5.2 Study 2: Comparison of Point Cloud Rendering Solutions

In a second study, we compared our two point cloud rendering methods and further investigated the specific clinical use cases in which our system could provide benefits. The experimental setup was similar to the one in the first study, but this time there were two VR phases for the participants. First, they saw the live-streamed staff member with the interlocking bricks using one rendering method, filled out a questionnaire, and then repeated the procedure with the second rendering method and a second part of the questionnaire. Which rendering method was seen first, point cloud or mesh, was randomized. The task in both phases was the same as in the first study, helping the staff member with the interlocking bricks. For this study, we discarded the social presence part from the first study and the spatial presence and involvement subscales of the IPQ, retaining only the realism one. Instead, we focused on getting more precise feedback regarding the potential benefits and use cases of the system. We also directly asked which rendering method would be preferred. The study was conducted with $N_2 = 7$ doctors and medical students in a hospital.

The results can be seen in Fig. 8. The left diagram shows the answers to the question of which rendering technique the doctors would prefer, the value of 3 meaning they found them similar, lower values meaning the splatting technique was preferred, and higher values correspond to the mesh visualization. The right diagram shows the scores of the realism subscale for the two rendering techniques. Again, scores are fractional, as the IPQ subscale is an aggregation of multiple questions. The results show that there was no absolute preference for one rendering method or the other. According to the data, the participants found them to be rather similar, some slightly preferring the splatting and others the mesh method. From the direct question and its result in the left diagram, we can see a small tendency for the mesh technique, which lines up with verbally given statements from two doctors after the study that they found the mesh rendering a bit better. The results for the realism subscale also indicate that there is no clear advantage for one or the other method. With a sample size of seven, it is hard to make statements with any significance, though. We did perform a Wilcoxon rank-sum test to test the null hypothesis of both distributions being equal and had not enough evidence to reject this hypothesis. Similar to the first study, the question about if the virtual world was more realistic than the real world scored particularly bad. Asked about it, one doctor argued that a positive answer to this would be impossible, as he obviously knew that he was in a virtual world during the experiment. The rest of the questionnaire's answers confirmed the results of the first study: most doctors would like to use such a telepresence system from time to time (57.1 %), and some would even be eager to use it very often (28.5 %). The more in-depth questions about the benefits and use-cases showed that most doctors see moderate benefits over video-based solutions for our proposed system in its current state and very strong benefits if, in the near future, the point cloud visualization quality and precision could be improved. The doctors stated that the system could be advantageous and helpful in emergency operations, or if an inexperienced doctor is on duty. Additional use-cases given by the doctors were educational operations, learning of the anatomy, patient anamnesis, and learning and teaching of practical skills in general.

6 Limitations

From the results of the first and particularly, the second study, we conclude that our telepresence system is a very good basis and has very high potential, but the RGB-D cameras' sensor resolution is a main limiting factor at the moment, at least concerning tasks involving high precision or small details. Naturally, sensors with higher resolutions being released would bring the biggest relief, however, we also think about combining 3D cameras with different sensing techniques to complement each other. E.g., stereo cameras, which typically produce higher-resolution color and depth images, could be added and used to enhance the fidelity of our system. Another limitation of our current system is the lack of a dedicated point cloud/mesh fusion process. Individually rendering multiple point clouds or meshes that depict the same object leads to visible seams or artifacts, even though they are registered precisely. The flying pixel effect and internal interpolation routines in the cameras may be one of the causes. Efficient and precise real-time fusion is a challenging topic in itself though, for example, Dou.

et al. [9] proposed a sophisticated but computationally demanding approach, and, thus, was not the focus of this work.

6.1 Face Reconstruction

A glaring problem with real-time reconstructed avatars in VR/AR telepresence applications is the HMD blocking the face. To be able to see the people's faces is highly important in collaborative virtual environments, though. To solve this issue, we developed a prototype combining the HMD's eye- and mouth-trackers with 3D Morphable Face Models (3DMM). A straightforward solution would be to use the deformable face model delivered with the HTC eye-tracking SDK and let the trackers drive the deformation. However, this generic model wouldn't be too realistic, as it can't be personalized. Also, we found the resulting facial expressions often do not match the actual mimic that well, sometimes even being weird-looking. Therefore, we acquired a more advanced morphable face model from "eos: A lightweight header-only 3D Morphable Face Model fitting library in modern C++11/14" [15] which can be automatically adapted to the person's facial geometry via a photo taken beforehand. Instead of directly applying the trackers' output to the morphable face, we pre-animated six facial expressions relating to basic emotions that are dynamically selected, interpolated, and applied to the face. The selection is based on a custom neural network that we pre-trained to map the trackers' output to the emotions. The six basic emotions we decided on are sadness, disgust, happiness, surprise, anger, and neutrality. To train the neural network, we created a small data set of facial expressions by asking multiple people to mimic the six emotions while we record the tracker output. To further customize the face model, we initially take six photos of the user's face, one for each of the facial expressions, and apply the most appropriate as a texture on the face at run-time. Again, the selection and interpolation is being driven by the neural network's output. The morphable face model is eventually rendered at the HMDs' 3D position while point cloud points or mesh triangles corresponding to the face are hidden. An example is illustrated in Fig. 9. To get a homogeneous avatar appearance for the point cloud rendering method, we wrote a custom shader transforming the mesh of the morphable face to look like a point cloud.

7 Conclusion and Future Work

In this paper, we have presented an immersive VR telepresence system for telementoring and remote collaboration in the operating room. Multiple Azure Kinect RGB-D cameras and point cloud avatars enable doctors to interact with each other and to view and assist in operations from a distance as if they were there. Thanks to our modular, low latency RGB-D streaming pipeline and efficient point cloud rendering and reconstruction techniques implemented in the Unreal Engine 4, we achieve very low latencies: our evaluation shows motion-to-photon latencies of only 120-150 ms, of which half of the latency is caused

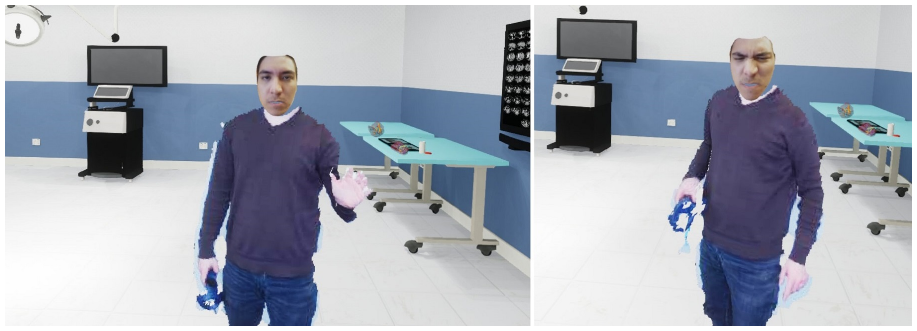

Fig. 9. Our mesh-rendered point cloud avatar with the reconstructed face, on the left with a neutral look and on the right showing disgust.

by the camera itself. At the same time, our pipeline handles all relevant tasks from filtering, denoising, and compression of the RGB-D data, to registration and computation of the point clouds. Using lossless compression, a bandwidth of only 23.4 Mbit/s per camera is required, although this can be further reduced by lossy compression when appropriate. In contrast to many other telepresence systems, our proposed solution is easy to set up and doesn't require multiple high-end PCs to run. We also presented a prototype to tackle the issue of occluded faces via personalized semi-real-time face reconstruction. A user study we conducted with doctors indicated that our system is capable of inducing very strong feelings of spatial and social presence. 83 % of the participating doctors attested moderate to high benefits to our system compared to video-based solutions, and one-third would like to use it at every opportunity. Lastly, we compared two point cloud rendering solutions, splatting and fast mesh reconstruction, via another study. The results showed that they scored quite similar regarding the IPQ's realism subscale, and the doctors had no clear preference for one or the other method. If directly asked for a comparison, there was a slight tendency in favor of the mesh method. In general, though, the RGB-D sensors' lacking resolution seems to be a limiting factor. Accordingly, most doctors would attest medium-high benefits to the system in the current state, reaching from educational scenarios to consultation in emergency situations, and very strong benefits if the fidelity could be improved upon in the near future.

In the future, we plan to enhance the rendering quality by improving both the real-time preprocessing of the RGB-D data as well as the rendering solutions themselves. For instance, neural networks became popular in many related fields lately and may lead to improvements in hole filling, denoising, or merging of RGB-D data and point clouds, too. In order to further increase the fidelity, we plan to add dedicated color or stereo cameras. Deep-learning-based up-sampling of the depth images could also be promising. Another aspect that could be further improved upon is the compression of the depth data. State-of-the-art video compression algorithms may be adapted to better suit compression of depth images, however, they are computationally expensive and not necessarily lossless which is crucial to not loose information from medically important areas/the situs. Using different compression algorithms and a combination of lossy and lossless techniques depending on the area of the scene and the visualized object may be a valid solution to circumvent this problem. To tackle the issue of the obstructed faces, we plan to finalize the prototype of our proposed reconstruction pipeline. Point cloud/mesh fusion is also an important topic we want to explore. Lastly, integrating the option to also use AR HMDs would be a great addition, especially for the doctors in the operating room.

Acknowledgements This work was partially funded by the German Federal Ministry of Education and Research (BMBF) under the grant 16SV8077.

References

1. Amamra, A.: Gpu-based real-time rgbd data filtering. Journal of Real-Time Image Processing **14** (09 2014)
2. Antón, D., Kurillo, G., Yang, A., Bajcsy, R.: Augmented telemedicine platform for real-time remote medical consultation. pp. 77–89 (01 2017)
3. Augestad, K., Lindsetmo, R.O.: Overcoming distance: Video-conferencing as a clinical and educational tool among surgeons. World Journal of Surgery **33**, 1356–1365 (07 2009)
4. Baños, R., Botella, C., Alcañiz Raya, M., Liaño, V., Guerrero, B., Rey, B.: Immersion and emotion: Their impact on the sense of presence. Cyberpsychology & behavior : the impact of the Internet, multimedia and virtual reality on behavior and society **7**, 734–41 (01 2005)
5. Böhlen, C.F.v., Brinkmann, A., Mävers, S., Hellmers, S., Hein, A.: Virtual reality integrated multi-depth-camera-system for real-time telepresence and telemanipulation in caregiving. In: 2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR). pp. 294–297 (2020)
6. Cao, C., Preda, M., Zaharia, T.: 3d point cloud compression: A survey. pp. 1–9 (07 2019)
7. Cho, S., Kim, S.w., Lee, J., Ahn, J., Han, J.: Effects of volumetric capture avatars on social presence in immersive virtual environments. In: 2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). pp. 26–34 (2020)
8. Dedeilia, A., Sotiropoulos, M., Hanrahan, J., Janga, D., Dedeilias, P., Sideris, M.: Medical and surgical education challenges and innovations in the covid-19 era: A systematic review. In Vivo **34**, 1603–1611 (06 2020)
9. Dou, M., Khamis, S., Degtyarev, Y., Davidson, P., Fanello, S., Kowdle, A., Orts, S., Rhemann, C., Kim, D., Taylor, J., Kohli, P., Tankovich, V., Izadi, S.: Fusion4d: Real-time performance capture of challenging scenes. ACM Transactions on Graphics **35** (07 2016)
10. Dussault, G., Franceschini, M.: Not enough there, too many here: Understanding geographical imbalances in the distribution of the health workforce. Human resources for health **4**, 12 (02 2006)
11. Flodgren, G., Rachas, A., Farmer, A., Inzitari, M., Shepperd, S.: Interactive telemedicine: effects on professional practice and health care outcomes. The Cochrane database of systematic reviews **9**, CD002098 (09 2015)

12. Gallo, L., Essmaeel, K., Damiani, E., De Pietro, G., Dipanda, A.: Comparative evaluation of methods for filtering kinect depth data. Multimedia Tools and Applications **74** (05 2014)
13. Gamelin, G., Chellali, A., Cheikh, S., Ricca, A., Dumas, C., Otmane, S.: Point-cloud avatars to improve spatial communication in immersive collaborative virtual environments. Personal and Ubiquitous Computing **25** (06 2021)
14. Gasques, D., Johnson, J.G., Sharkey, T., Feng, Y., Wang, R., Xu, Z.R., Zavala, E., Zhang, Y., Xie, W., Zhang, X., Davis, K., Yip, M., Weibel, N.: Artemis: A collaborative mixed-reality system for immersive surgical telementoring. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. CHI '21, Association for Computing Machinery, New York, NY, USA (2021)
15. Huber, P., Hu, G., Tena, R., Mortazavian, P., Koppen, P., Christmas, W.J., Ratsch, M., Kittler, J.: A multiresolution 3d morphable face model and fitting framework. In: Proceedings of the 11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. University of Surrey (2016)
16. Jang-Jaccard, J., Nepal, S., Celler, B., Yan, B.: Webrtc-based video conferencing service for telehealth. Computing **98**, 169–193 (01 2016)
17. Kamimura, K., Fujita, Y., Miura, T., Matsumoto, Y., Maeda, Y., Zempo, K.: Teleclinical support system via mr-hmd displaying doctor's instructions and patient information. pp. 477–479 (03 2021)
18. Kolkmeier, J., Harmsen, E., Giesselink, S., Reidsma, D., Theune, M., Heylen, D.: With a little help from a holographic friend: the openimpress mixed reality telepresence toolkit for remote collaboration systems. pp. 1–11 (11 2018)
19. Kvedar, J., Coye, M., Everett, W.: Connected health: A review of technologies and strategies to improve patient care with telemedicine and telehealth. Health affairs (Project Hope) **33**, 194–9 (02 2014)
20. Latifi, R., Weinstein, R., Porter, J., Ziemba, M., Judkins, D., Ridings, D., Nassi, R., Valenzuela, T., Holcomb, M., Leyva, F.: Telemedicine and telepresence for trauma and emergency management. Scandinavian journal of surgery : SJS : official organ for the Finnish Surgical Society and the Scandinavian Surgical Society **96**, 281–9 (02 2007)
21. Lin, B.S., Su, M.J., Cheng, P.H., Tseng, P.J., Chen, S.J.: Temporal and spatial denoising of depth maps. Sensors (Basel, Switzerland) **15**, 18506–25 (08 2015)
22. Liu, Y., Beck, S., Wang, R., Li, J., Xu, H., Yao, S., Tong, X., Froehlich, B.: Hybrid lossless-lossy compression for real-time depth-sensor streams in 3d telepresence applications. pp. 442–452 (09 2015)
23. Mao, A., Zhang, H., Liu, Y., Zheng, Y., Li, G., Han, G.: Easy and fast reconstruction of a 3d avatar with an rgb-d sensor. Sensors (Switzerland) **17** (05 2017)
24. Mekuria, R., Blom, K., César, P.: Design, implementation and evaluation of a point cloud codec for tele-immersive video. IEEE Transactions on Circuits and Systems for Video Technology **27**, 1–1 (01 2016)
25. Mühlenbrock, A., Fischer, R., Weller, R., Zachmann, G.: Fast and robust registration of multiple depth-sensors and virtual worlds. In: 2021 International Conference on Cyberworlds (CW). pp. 41–48 (2021)
26. Nowak, K., Biocca, F.: The effect of the agency and anthropomorphism on users' sense of telepresence, copresence, and social presence in virtual environments. Presence Teleoperators and Virtual Environments **12**, 481–494 (10 2003)
27. Orts, S., Rhemann, C., Fanello, S., Kim, D., Kowdle, A., Chang, W., Degtyarev, Y., Davidson, P., Khamis, S., Dou, M., Tankovich, V., Loop, C., Cai, Q., Chou,

P., Mennicken, S., Valentin, J., Kohli, P., Pradeep, V., Wang, S., Izadi, S.: Holoportation: Virtual 3d teleportation in real-time (10 2016)
28. Pece, F., Kautz, J., Weyrich, T.: Adapting standard video codecs for depth streaming. pp. 59–66 (01 2011)
29. Ragan, E., Kopper, R., Schuchardt, P., Bowman, D.: Studying the effects of stereo, head tracking, and field of regard on a small-scale spatial judgment task. IEEE transactions on visualization and computer graphics **19** (08 2012)
30. Rojas, E., Cabrera, M., Lin, C., Sánchez-Tamayo, N., Andersen, D., Popescu, V., Anderson, K., Zarzaur, B., Mullis, B., Wachs, J.: Telementoring in leg fasciotomies via mixed-reality: Clinical evaluation of the star platform. Military Medicine **185**, 513–520 (01 2020)
31. Roth, D., Yu, K., Pankratz, F., Gorbachev, G., Keller, A., Lazarovici, M., Wilhelm, D., Weidert, S., Navab, N., Eck, U.: Real-time mixed reality teleconsultation for intensive care units in pandemic situations. pp. 693–694 (03 2021)
32. Schröder, C., Sharma, M., Teuber, J., Weller, R., Zachmann, G.: Dyncam: A reactive multithreaded pipeline library for 3d telepresence in vr. In: Proc. of the 20th ACM Virtual Reality International Conference (VRIC 2018). ACM (2018)
33. Schubert, T., Friedmann, F., Regenbrecht, H.: The Experience of Presence: Factor Analytic Insights. Presence: Teleoperators and Virtual Environments **10**(3), 266–281 (06 2001)
34. Söderholm, H., Sonnenwald, D., Cairns, B., Manning, J., Welch, G., Fuchs, H.: The potential impact of 3d telepresence technology on task performance in emergency trauma care. pp. 79–88 (11 2007)
35. Teng, C.C., Jensen, N., Smith, T., Forbush, T., Fletcher, K., Hoover, M.: Interactive augmented live virtual reality streaming: A health care application. pp. 143–147 (06 2018)
36. Thoravi Kumaravel, B., Anderson, F., Fitzmaurice, G., Hartmann, B., Grossman, T.: Loki: Facilitating remote instruction of physical tasks using bi-directional mixed-reality telepresence. pp. 161–174 (10 2019)
37. Tölgyessy, M., Dekan, M., Chovanec, L., Hubinský, P.: Evaluation of the azure kinect and its comparison to kinect v1 and kinect v2. Sensors **21**, 413 (01 2021)
38. Wilson, A.: Fast lossless depth image compression. pp. 100–105 (10 2017)
39. Yu, K., Gorbachev, G., Eck, U., Pankratz, F., Navab, N., Roth, D.: Avatars for teleconsultation: Effects of avatar embodiment techniques on user perception in 3d asymmetric telepresence. IEEE Transactions on Visualization and Computer Graphics **PP**, 1–1 (08 2021)