Output-Sensitive Avatar Representations for Immersive Telepresence

Adrian Kreskowski, Stephan Beck, and Bernd Froehlich

Abstract—In this paper, we propose a system design and implementation for output-sensitive reconstruction, transmission and rendering of 3D video avatars in distributed virtual environments. In our immersive telepresence system, users are captured by multiple RGBD sensors connected to a server that performs geometry reconstruction based on viewing feedback from remote telepresence parties. This feedback and reconstruction loop enables visibility-aware level-of-detail reconstruction of video avatars regarding geometry and texture data, and considers individual and groups of collocated users. Our evaluation reveals that our approach leads to a significant reduction of reconstruction times, network bandwidth requirements and round-trip times as well as rendering costs in many situations.

Index Terms—Immersive telepresence, avatars, output-sensitive rendering, distributed virtual environments.

1 Introduction

Immersive 3D telepresence systems embody participants through realistic three-dimensional avatars, enabling them to meet in virtual worlds. The most realistic avatars are created by capturing participants using multiple cameras surrounding the workspace [1], [2], [3], [4], [5]. These so-called 3D video avatars are simultaneously distributed over the network between remote locations, integrated into a shared virtual scene and displayed in stereoscopic 3D for each participant. Recent approaches to real-time 3D capturing and reconstruction provide high-quality 3D video avatars using volumetric fusion of multiple color and depth (RGBD) sensors [4], [5], [6]. To reduce the bandwidth requirements for streaming 3D video avatars, several methods for real-time compression of geometry [7], [8], [9] and image streams [10], [11] were proposed. However, these methods do not include information about the visibility and size of avatars as seen by remote participants. As a result, they cannot make optimal use of the available bandwidth, burdening telepresence clients with handling and rendering avatars with much more detail than can be perceived from the perspective of a participant.

To exploit this potential, we present a novel 3D telepresence architecture that creates output-sensitive 3D video avatars at an appropriate level-of-detail by establishing individual closed-loop feedback between pairs of telepresence parties. The central idea of our approach is to reconstruct, compress and transfer only parts of the avatars' geometry and texture that are potentially visible to remote telepresence participants (Figure 1). To achieve this, a volumetric brick-based structure keeps track of visibility, with respect to the participants' perspectives and scene occlusions. The resolutions of the truncated signed distance field (TSDF) for the avatars' geometry creation, as well as the size of the corresponding texture arrays, are adjusted on a per-frame basis, with respect to the viewing feedback information from remote telepresence parties, which leads to quasi-continuous level-of-detail avatar representations. As a result, the creation, transmission and rendering of 3D avatars is significantly accelerated in many situations since invisible surface parts are not reconstructed and consequentially not processed at all.

We especially account for multiple collocated users in a telepresence group by enabling them to request a single merged avatar representation from a remote site. This effectively reduces bandwidth requirements by avoiding the transmission of redundant geometry seen by multiple collocated users at the expense of only slightly increased rendering costs for their respective rendering processes.

Our work is inspired by output-sensitive rendering approaches for large model visualization [12], [13], [14] which generally precompute level-of-detail representations and visibility information. We approach the challenge of highly dynamic scenes captured in real-time, representing the postures and gestures of humans and parts of their surrounding environment.

Our design, implementation and evaluation of a 3D telepresence architecture that creates, transmits and renders output-sensitive 3D video avatars for multi-party communication provides the following contributions:

- A continuous visibility-aware level-of-detail surface extraction and texturing method that significantly reduces avatar reconstruction, transmission and rendering times as well as network bandwidth requirements.
- A novel method for the online creation of pre-blended textures at continuous levels-of-detail which represent each visible point on the surface of an avatar only once.
- An efficient implicit texture mapping approach for avatars based on precomputed calibration volumes.
- A flexible feedback loop that allows to request combined avatar representations for an entire group of collocated users sharing similar perspectives, enabling a trade-off between required network bandwidth and rendering performance.

• *The authors are with the Virtual Reality and Visualization Research Group, Bauhaus-Universität Weimar.*

E-mail:
{adrian.kreskowski, stephan.beck, bernd.froehlich}@uni-weimar.de

Manuscript received July 25, 2019; revised October 22, 2020.

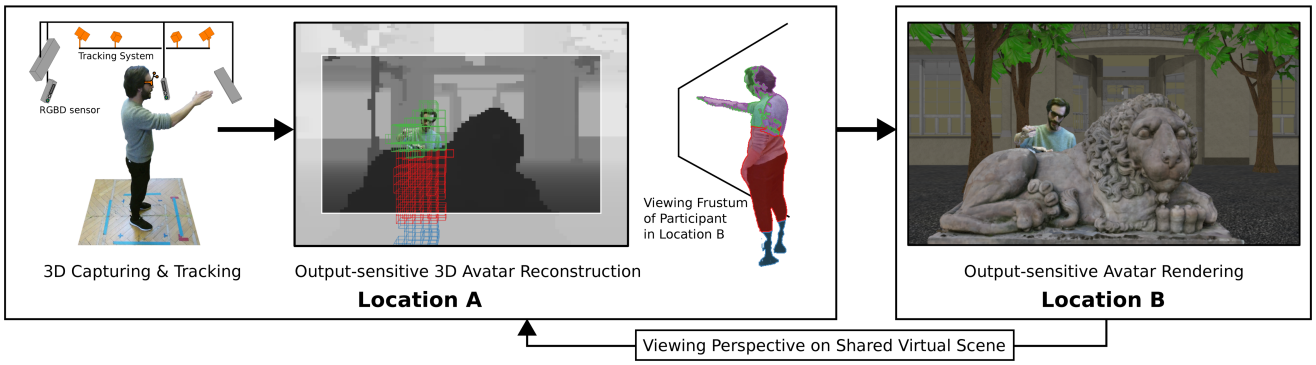


Fig. 1: Output-sensitive avatar reconstruction and rendering in an immersive telepresence application: A user at location **A** (avatar) and users at remote location **B** meet face-to-face in a shared virtual environment and explore a 3D model of a lion statue standing. Our system captures the participant at location **A** using multiple color and depth sensors, and creates output-sensitive 3D video avatar geometry and textures by considering all tracked viewing perspectives of the remote users at **B**. Parts of the avatar that are either occluded (red) or out of frustum (blue) for all all remote users are not reconstructed at all. Furthermore, back-facing parts (purple) are efficiently identified during isosurface extraction and excluded from geometry creation. In this way, our system creates only the potentially visible fraction of the 3D video avatar (green) at an appropriate level-of-detail. At location **B**, the participants' avatars are reconstructed by a second server (not illustrated) based on the viewing feedback obtained from the participant at location **A**.

Our quantitative evaluation reveals that our 3D reconstruction pipeline can create output-sensitive avatars from four RGBD-image streams at more than 200 Hz on current graphics cards throughout different telepresence scenarios.

The implementation of our remote avatar reconstruction approach in a distributed virtual reality framework [15] allows us to demonstrate a reduction in avatar reconstruction time, rendering time and in particular network bandwidth requirements, at low end-to-end latency. In situations where local and remote users are further apart or largely occluded, data rates are quite low and approach zero for invisible avatars. In our experience, users generally did not notice any differences between avatars at original resolution and their level-of-detail counterparts, which was also confirmed by very high SSIM scores.

2 Related Work

In collaborative virtual environments (CVE) users are generally represented as either computer-generated (CG) avatars or real-time reconstructed 3D video avatars. CG avatars are often used in combination with motion-tracking of the participants, e.g. [16], or applied to simulate hundreds [17] or even thousands [18] of agents in CVEs. Obvious advantages of CG avatars are light-weight rendering and transmission between remote sites since only skeleton joint poses are involved for animation, however, at the cost of a-priori generation and rigging. In contrast, immersive telepresence systems use real-time captured user representations called 3D video avatars which enable participants to instantly join CVEs. Moreover, user studies confirmed that 3D video avatars appear natural [2], [3] and can convey mimics better and with less effort than CG avatars [19]. However, efficient real-time 3D reconstruction, transmission and rendering remain challenges when using 3D video avatars [2], [20].

Existing research in real-time 3D capturing and reconstruction provides an essential foundation for our work. Recently proposed techniques build on RGBD sensors for 3D capturing and can be divided into three categories: methods using *explicit* [1], [2], [21], *implicit* [6], [22] or *spatio-temporal* [4], [5] fusion of multiple RGBD-sensor contributions. Alexiadis et al. [21] propose to explicitly stitch mesh segments derived from a step discontinuity constraint triangulation, yielding a watertight 3D mesh. Implicit 3D reconstruction approaches fuse multiple sensor contributions volumetrically using a truncated signed distance field (TSDF) [23]. A major advantage of volumetric fusion is the ability to filter depth measurements in object space and to reconstruct implicit surfaces between unseen or empty space [6]. Moreover, artifacts such as holes or inconsistent appearance for changing viewing perspectives inherent in explicit approaches are mitigated [1], [2]. Current state-of-the-art 3D reconstruction approaches [4], [5] exploit frame-to-frame coherence through spatio-temporal fusion to further improve TSDF volume quality. For 3D reconstruction, most approaches use the Marching Cubes Algorithm [24] to extract a consistent surface at zero crossings within the volume. The 3D video avatar's geometry can then be rendered and textured by blending the color images based on projective texturing [4], [6] or using a texture atlas [5].

Our 3D reconstruction approach builds on volumetric fusion using TSDF integration. In contrast to previous methods [4], [6], our novel brick-based acceleration structure enables efficient culling of subsets of the 3D capturing volume, significantly reducing processing costs during integration and surface extraction. Moreover, we texture the 3D video avatar's geometry using pre-blended textures created at an appropriate level-of-detail with respect to its visibility and projected size. Unlike 3D reconstruction methods that use a texture atlas [5], [25], the color information in our textures is not rearranged. Maintaining the neighborhood of triangles in texture space has the advantage of artifact-free hardware-assisted bilinear interpolation and avoids explicit padding or special triangle packing [26]. In addition, we avoid encoding of per-vertex texture coordinates by extending our existing calibration approach [27] to facilitate mapping vertex positions to texture space during run time using pre-computed calibration volumes. This allows a further

reduction of data rates during both 3D reconstruction and rendering.

A major challenge of immersive 3D telepresence systems is the transmission of data streams between distributed locations. In our application, 3D capturing and reconstruction is performed at a different location per participating party. The reconstructed avatars' 3D geometry and texture information must therefore be transferred from each party's location to all other parties. Existing real-time reconstruction systems typically create data rates ranging from several hundred MBit/s [2] to several GBit/s [3] depending on the resolution and frequency of the 3D capturing and reconstruction system. Transmission over low bandwidth network connections therefore requires the use of compression techniques. Standardized geometry compression techniques (e.g. [28], [29]) have good rate-distortion characteristics, but are not well applicable in 3D telepresence applications due to time consuming mesh analysis required for efficient encoding. Recently, compression approaches have been proposed for 3D telepresence applications, e.g. for meshes derived from RGBD sensor-based 3D reconstruction pipelines [7], [8] or for time-varying sequences of point clouds [9]. In addition, real-time compression and transmission of live-captured 4D performances through mid-to-low bandwidth networks was recently achieved based on compressing TSDF volumes [30], [31]. Although these approaches require pre-training and introduce additional decompression and extraction overhead before rendering, they show that streaming 3D video avatars can be very well achieved, trading some of the geometric fidelity by increased compression performance. However, existing compression schemes for 3D telepresence applications are not designed to provide output-sensitive geometry transmission, because they do not incorporate feedback information from the participants or occlusion information from the shared scene state. In contrast, our approach makes efficient use of this information throughout 3D reconstruction, and prior to data transmission, enabling output-sensitive data rates independent of the compression method used. Although we do not focus on the compression of output-sensitive avatar data in this paper, we demonstrate the impact of our level-of-detail based reconstruction on the required bandwidth by example of intra-frame encoded avatar geometry and texture.

Several research groups propose data reduction methods for 3D telepresence that are orthogonal to the explicit compression of data streams. For example, the distributed 3D reconstruction system *blue-c* [32] transmits point primitives [33] using a differential updating scheme that exploits spatio-temporal coherence to reduce the data rate. Lamboray et al. [34] improves on this by applying predictive coding. The idea of only processing data that is actually visible for a user was investigated for camera arrays [35]. Wang et al. [36] suggest to reduce redundancy in color images from overlapping RGBD sensors [36] prior to transmission. Kuster et al. [37] apply 3D reconstruction and local rendering in a bidirectional telepresence system. In their system, stereo image pairs are transferred to the client site, which enables the use of standardized, and therefore efficient, compression schemes. Such image-based avatar representation comes at the expense of incorrect stereoscopic perception, as the display cannot adapt to the rapidly changing viewing perspectives of the users. In contrast to these systems, our 3D telepresence architecture creates and sends output-sensitive geometric 3D avatar representations, efficiently reduces data rate and rendering load, and enables correct stereoscopic display for the participants.

3 3D Telepresence Architecture

Our proposed 3D telepresence architecture targets scenarios where multiple groups of one or more participants meet in a shared virtual environment. We use the term *group* to refer to members of a *telepresence party*, who request and receive a merged avatar representation of another *party*, in contrast to receiving an individual avatar for each member. The users of a group are typically *collocated*, i.e. they are in the same physical workspace and potentially share a display, such as our projection-based multi-user 3D display [38]. The term *client* refers to the technical view on a group. The client renders the received merged avatars for the group members and sends the group's feedback to the reconstruction processes of the remote parties. Output-sensitivity in this context refers to the generation of only potentially visible avatar geometry and texture information, at an appropriate level of detail for remote telepresence participants. In the following sections, we provide an overview of a generalized telepresence architecture, show how it can be extended with *closed-loop feedback*, and describe our implementation.

3.1 Telepresence Architecture Overview

Figure 2 illustrates software architecture components typically used by immersive telepresence systems. The diagram represents a generic architecture where a 3D reconstruction component receives an (RGBD) image stream, performs 3D reconstruction and forwards the avatar data into a rendering process. For 3D capturing, either custom-built stereo-camera rigs [4], [5] or off-the-shelf RGBD sensors such as the *Microsoft Kinect* [1], [2], [21] can be applied. Besides the 3D capturing stage, the diagram further includes additional components and communication channels for implementing our proposed closed-loop feedback. Note, that the diagram does not enforce a specific distribution of the architecture components to remote and local sites. It therefore includes systems that do not make use of a remote reconstruction approach [21], [22] as well as systems that do [4], [5]. For the latter, an avatar's geometry is created on a remote server and sent to a telepresence client which can render the avatar in a straightforward manner for a group and send their feedback to the server. Compression and decompression are optional and therefore not illustrated.

In general, immersive telepresence systems [2] build on four fundamental components: First, a *3D capturing* system captures the participants geometrically and radiometrically at sufficiently high frame rates ($\geq$ 30 Hz) using multiple surrounding cameras. Second, a *3D reconstruction* process creates 3D video avatars from the captured RGBD image streams. Third, a *client rendering* process receives the avatar data from the 3D reconstruction process and renders the shared virtual scene including the avatars. As a fourth component, a *global scene server* receives local scene changes from individual clients and delivers consistent global scene updates to all of them.

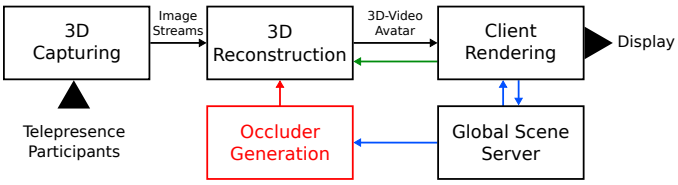


Fig. 2: Schematic illustration of components in 3D telepresence systems: RGBD image streams are transmitted from a 3D capturing to a 3D reconstruction component, which generates an avatar representation and transmits it to the client rendering process (black arrow). Scene updates are streamed from a global scene server to clients and vice versa (blue arrows). We propose to send viewing feedback information from the client to the reconstruction process (green arrow). In addition, we propose to add a new component that creates occluder representations of the virtual scene received from the scene server, and forwards them to the reconstruction process (red arrow).

3.2 Closed-Loop Feedback

We extended the four fundamental components of a 3D telepresence architecture by introducing *closed-loop feedback* mechanism to enable a truly output-sensitive version of the 3D reconstruction process, with regard to the participants' perspectives on the shared virtual environment. Using closed-loop feedback, the reconstruction process generates individual 3D video avatar representations for each user or for an entire group of collocated participants depending on the type of *feedback* information provided. In general, feedback or *viewing feedback* refers to the camera parameters of each virtual eye for stereoscopic display, as well as information about the rendering resolution on the client. In addition to the viewing feedback, the 3D reconstruction server uses lightweight occluder information provided by a *occluder generation* process. This information can be used to identify parts of 3D video avatars that are occluded by virtual scene content. As a result, the 3D reconstruction process only reconstructs parts that are potentially visible from the current viewing perspectives of the participants.

Figure 3 illustrates our approach of sending feedback information from telepresence participants to 3D reconstruction processes for creating, and requesting output-sensitive avatars. Feedback can be either sent individually (*individual feedback*) or as a group (*group feedback*): Individual feedback is used for single user telepresence clients, where each client requests an individual version of an avatar. For many-to-one and many-to-many communication, we suggest to create a combined avatar representation for collocated users, instead of multiple avatars optimized for each user's individual view. In this case, the feedback information is aggregated for all collocated participants and referred to as group feedback.

The motivation for sending group feedback is that collocated telepresence participants, in particular when using projection-based display systems [38], assume similar perspectives and thus see similar parts of a 3D video avatar. As a result, group feedback allows to perform certain steps of the reconstruction process only once for the entire group instead of once per user. Consequently, a combined avatar representation has to be created and transferred only once. Moreover, in comparison to multiple individual feedback requests, group feedback reduces the total amount of geometry created by the reconstruction process and yields

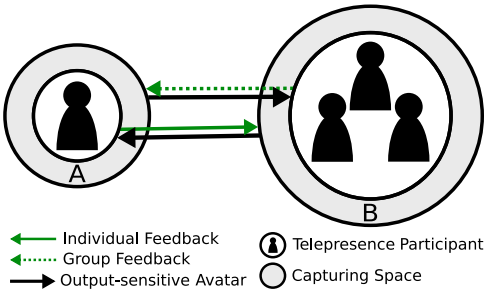


Fig. 3: Output-sensitive avatar reconstruction using our feedback mechanism in a telepresence scenario between locations *A* and *B*. The participant at *A* observes the scene through an own individual display. The collocated participants at *B* share a multi-user 3D display and navigate together through the virtual environment, adopting similar perspectives into the virtual world. At each location, the participants are captured in 3D and a reconstruction process creates output-sensitive avatars based on *individual* or *group feedback* information. The client of the participant in *A*, requests a group avatar representation from the reconstruction process at *B* by sending *individual feedback*. The rendering client at *B* requests a *merged avatar* that is output-sensitive w.r.t. the views of all participants in *B* from the reconstruction process at *A* based on *group feedback*. This simplified example can be extended beyond two parties by adding the corresponding communication channels.

significantly lower bandwidth requirements at only slightly increased rendering costs on the client side (cmp. Subsection 5.4).

3.3 Telepresence Architecture Implementation

Our implementation is based on a server-client model. The *3D capturing*, *3D reconstruction* and *occluder generation* components work together (cmp. Figure 2) as a processing group in a dedicated server and provide the avatar geometry for all *client rendering* components at any remote location. In the following, we provide some implementation details and reasoning about the two components which are fundamental additions for 3D reconstruction pipelines using viewing feedback information as proposed, the *Occluder Generation* process and the *Feedback Channel*.

3.3.1 Occluder Generation

The implementation of the occluder generation process depends on the available hardware (graphics cards) and the desired behavior (i.e. occlusion handling). In our implementation, we opted for the following features:

- Each object in the scene excluding avatars is considered to be an occluder.
- The occluder generation process is executed on a GPU on the reconstruction server machine to avoid latency and reduce CPU load.
- The occluder generation process is executed asynchronously to the 3D reconstruction process. This allows us to provide the reconstruction process immediately with the latest occluder representation generated.
- The occlusion buffer is downsampled to accelerate the 3D reconstruction process, creating a trade-off between occlusion culling performance and effect of the culling.

3.3.2 Feedback Channel

Since pipelined processing as well as communication over network introduces latency, we aimed to keep the main feedback loop between the rendering client and the reconstruction server as short as possible (Figure 4).

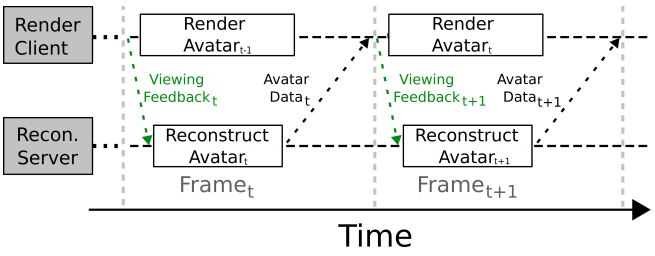


Fig. 4: Processes involved in the main feedback loop. To minimize the round-trip time, the clients' feedback (green arrows) is collected and sent over network to the reconstruction server before the actual rendering of previously obtained avatar data (black arrows) begins.

The viewing feedback from the rendering clients to the reconstruction servers should be sent with a frame rate equal to the rendering frequency. The feedback is therefore sent before the rendering process of the current frame is started. In this way, the time between sending feedback and receiving the reconstructed avatar does not depend on the actual achievable frame rate of the client, but only on the network latency, reconstruction time of the remote server as well as optional compression and decompression times. However, since the received avatar must be visualized on the rendering client side, the time for transferring from CPU to GPU memory and rendering must be added to the round-trip time.

4 3D Avatar Reconstruction

For creating output-sensitive 3D video avatars, we capture collocated users with a cluster of multiple RGBD sensors [2], [4], [5], [6]. The central data structure of our 3D reconstruction pipeline is a truncated signed distance field (TSDF), which is used to integrate and fuse multiple depth-sensor contributions [6], [23]. However, in comparison to previous methods, we apply a brick structure that partitions and tracks the occupied space within the capturing volume, which is a key for output-sensitive reconstruction. In particular, the brick structure enables efficient visibility culling and accelerates the 3D avatar reconstruction significantly, since it allows us to integrate only sensor contributions that are potentially visible for remote participants.

In our lab, a typical 3D capturing space covers a working volume of $3 \times 3 \times 2.5m^3$. We align the TSDF volume with the real world such that the boundaries of the TSDF volume match those of the 3D capturing volume. To align the different spaces, we calibrate the RGBD sensors using a volumetric method [27] which allows us to map from coordinates in depth-sensor space to 3D positions in TSDF volume space ($C_{d \mapsto tsdf}$) and to texture coordinates of a depth sensor's corresponding color sensor ($C_{d \mapsto c}$). To perform precise TSDF volume-to-sensor projections required during TSDF-integration (Section 4.4) and texturing (Section 4.6), we introduce two additional types of calibration volumes. The first type ($C_{tsdf \mapsto d}$) maps coordinates in TSDF volume space to a depth-sensor's coordinate system by inverting the mapping $C_{d \mapsto tsdf}$ provided by our original method. The second type ($C_{tsdf \mapsto c}$) directly maps coordinates in TSDF volume space to a color sensor's coordinate system by merging $C_{tsdf \mapsto d}$ and $C_{d \mapsto c}$. Our calibration scheme enables projections from 3D capturing volume space to individual RGBD-sensor spaces and vice versa through efficient hardware-accelerated volume texture lookups during runtime (cmp. Figure 5).

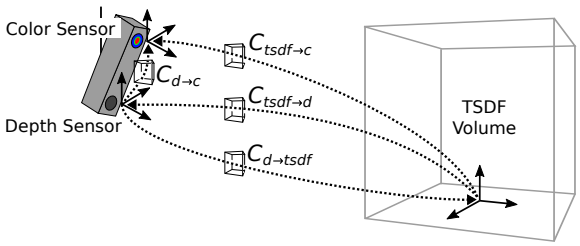


Fig. 5: Schematic illustration of four types of calibration volumes used during integration and texturing. $C_{d \mapsto tsdf}$ maps from depth sensor to TSDF volume space whereas $C_{d \mapsto c}$ maps from depth sensor space to coordinates of the RGBD-sensor's corresponding color image [27]. We use two additional volumes mapping from TSDF volume to depth sensor space ($C_{tsdf \mapsto d}$), and directly to a sensor's color image ($C_{tsdf \mapsto c}$).

As a brief overview, our output-sensitive 3D reconstruction pipeline consists of the following processing steps (Figure 6):

1) Estimation of level-of-detail required during reconstruction, and partition of the capturing volume into bricks of corresponding size (Subsection 4.1).
2) Level-of-detail-aware filtering, downsampling and processing of RGBD image streams, as well as occupancy tracking of TSDF volume bricks. Unoccupied Bricks are not considered in subsequent steps (Subsection 4.2).
3) Frustum and occlusion culling of occupied bricks, using feedback information provided by the rendering clients, as well as scene occluder information provided by the occluder generator. Non-visible bricks are not considered in subsequent steps (Subsection 4.3).
4) TSDF integration of bricks that are both occupied and visible (Subsection 4.4).
5) Brick-based surface extraction of front-facing triangles (Subsection 4.5).
6) Pre-blended texture generation at an appropriate level of detail from color image streams (Subsection 4.6).

After the geometry extraction and pre-blended texture generation, we apply geometry and texture compression to further reduce the required bandwidth for network transmission (Section 4.7).

In the following Subsections, we explain each step of our 3D reconstruction pipeline in detail.

4.1 LOD Estimation and Bounding Box Division

Our reconstruction pipeline is guided by a level-of-detail factor l, determining the granularity at which the individual reconstruction stages operate. To select an appropriate level-of-detail for avatar reconstruction depending on the participants' view, we evaluate their viewing feedback. In

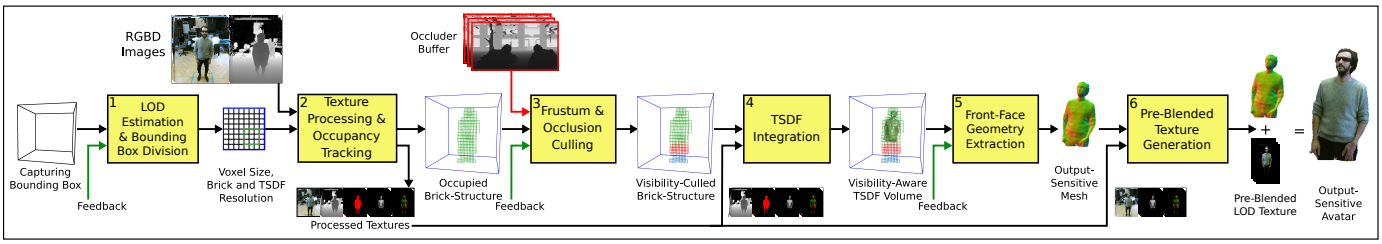


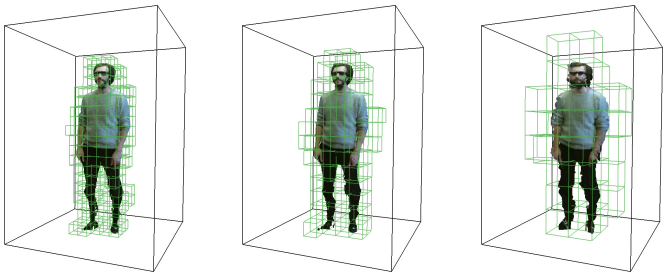
Fig. 6: Processing steps for creating output-sensitive 3D video avatars. Our 3D reconstruction pipeline adapts to the participants' perspectives into the shared virtual environment by incorporating viewing feedback and scene occluder information. In this example, the legs of the video avatar are largely invisible to the participants and are therefore culled prior to TSDF integration. A detailed explanation of the individual processing steps of our 3D reconstruction pipeline can be found in Section 4.

particular, we allow for separate level-of-detail factors for both the geometry reconstruction stage and the texture creation. This is especially important for RGBD sensors with a high discrepancy between depth and color resolution, such as the *Kinect 4 Azure* used in our evaluation. For the sake of simplicity, we use a common level-of-detail factor l for both geometric and texture level-of-detail in our description.

We calculate l by an estimator E_{ssc} projecting the capturing space volume V into screen space and calculating the maximal projected size with respect to all pieces of viewing feedback f_i as follows:

$$l = \min(1.0, \alpha \max_{f_1,\dots f_N} E_{ssc}(V, f_i)) \quad (1)$$

Here, the attenuation α is chosen based on the conservativity of E_{ssc} and may be defined as a function of distance to the viewer. Depending on l, the side length of TSDF voxels is spatially enlarged by a factor of $\frac{1}{l}$. As a result, decreasing l yields coarser TSDF integration and surface extraction.



$l = 1.0$ $l = 0.7$ $l = 0.4$

Fig. 7: Impression of the avatar reconstruction quality with three different geometry and texture level-of-detail factors in a first implementation using four RGBD sensors. From left to right: Varying level-of-detail ratios l affect the avatar reconstruction. The occupied TSDF bricks are outlined in green behind the avatar. Across all l, each brick contains 10^3 voxels. Only voxels contained in the green bricks are integrated and subsequently serve as input for the geometry extraction stage. The brick size increases proportional to $\frac{1}{l}$. The bricks are cubical with a side length of 10 cm, 14.3 cm and 25 cm, respectively.

4.2 Texture Processing and Occupancy Tracking

We filter and downsample the color and depth textures of each sensor with respect to l, such that subsequent texture operations are performed on coarser texture levels-of-detail. Color texture filtering can be performed by a combined low-pass filtering and downsampling operation. In contrast, depth textures have to be downsampled taking the avatar's silhouette into account.

Subsequently, the depth images are smoothed using a bilateral filter in order to reduce sensor noise. We then mark each depth pixel that projects into the bounding box defined by the TSDF volume as a foreground pixel and compute its surface normal using central differences. Subsequently, a quality value is calculated for each foreground pixel which serves as a weighting factor during TSDF integration and texture blending [1], [6]. In addition, the texture processing stage projects each filtered depth pixel to TSDF volume space and tracks the bricks' occupancies using atomic counters. Consequently, only those bricks whose occupancy exceeds a predefined threshold must be processed further, while the other bricks are considered empty and are excluded from this early stage on.

4.3 Frustum and Occlusion Culling

We test the visibility of the occupied bricks against the latest occluder buffers and viewing frustra as computed from the feedback of the remote groups. The visibility of bricks is tracked using a custom *visibility buffer* which stores a visibility flag for each brick.

After initializing the visibility buffer such that all bricks are considered to be invisible (set to zero) in the current frame, we bind the current occluder buffer as a depth attachment to a frame buffer and render the bounding box of each brick from the participant's perspective. If a brick is visible for any participant, its visibility is set to one. As a result, a set of bricks marked as visible is generated, which is then processed in the next stage.

4.4 TSDF Integration

Our brick structure allows us to track regions in the capturing volume that are both visible and potentially contain surfaces. This is a major advantage compared to previous volumetric fusion methods [4], [21], since a significant number of bricks are irrelevant for the avatar's surface and consequently only a fraction of the voxels has to be integrated. Despite depending on the size of the capturing volume, the scene content, and the level-of-detail factor, our bricking technique allows to exclude 50 to 90 percent of the voxels during TSDF integration for our capturing setups (cmp. Figure 7).

The TSDF integration itself is performed per brick and runs entirely on the GPU. For each voxel of a visible, occupied brick, the following steps are executed: first, the voxel is projected into the depth image of each sensor using the calibration volumes. Next, the signed distances between the voxel and the foreground pixels are weighted using the precomputed quality maps and accumulated. Finally, if the absolute value of the signed distance sd is smaller than the TSDF-limit denoted as $limit$, the value is stored at the voxel's position inside the TSDF volume. If sd is smaller than $-limit$ for all sensors, $-limit$ is stored, otherwise, if sd is greater than $limit$ for all sensors, $limit$ is stored.

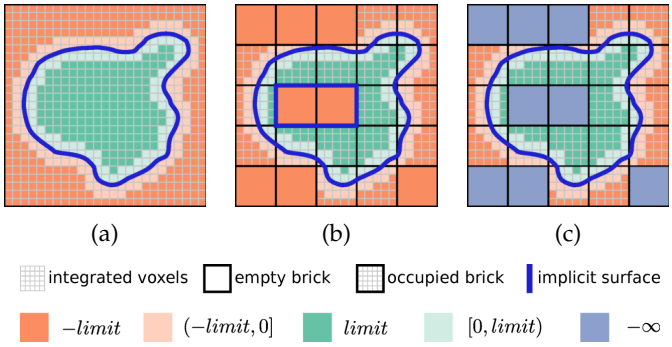


Fig. 8: (a) Conventional integration approaches evaluate the distance between every voxel within a TSDF volume and an implicit surface [21]. (b) Our approach integrates only voxels residing in occupied bricks, empty bricks are omitted. Between occupied and empty bricks on the inside of the surface, a naïve implementation creates artificial zero crossings and therefore erroneous isosurfaces. (c) Clearing the TSDF volume with $-\infty$ instead of $-limit$, interpolation between integrated voxels (green or orange) and unintegrated bricks (violet), cause artificially high negative gradients. These gradients indicate artificial isosurfaces that should not be extracted.

Depending on the brick size and the actual coverage of the avatar within the TSDF volume, bricks inside of the real implicit surface of the avatar may be unoccupied, resulting in unintegrated bricks containing voxels with distance $-limit$. If such bricks reside next to bricks containing positive distance values, they need special consideration such that no artificial surface geometry is created in the subsequent isosurface extraction stage. We clear the volume at the beginning of each integration frame with $-\infty$ instead of $-limit$ to avoid the creation of artificial geometry in the subsequent reconstruction stage (see Figure 8).

4.5 Front-Face Geometry Extraction

To extract the avatars' surfaces we apply the Marching Cubes [24] algorithm for voxels contained in visible and occupied bricks. Note that we are able to reuse the allocated memory for the highest TSDF volume resolution across all levels which allows for a memory efficient implementation of continuous level-of-detail geometry extraction.

As described before, naïve brick-based extraction creates artificial surfaces, which remain hidden but create an overhead of approximately 12%. We identify artificial surfaces based on the high gradients during interpolation between integrated and unintegrated voxels (see Figure 8) and skip the extraction in such cases. Triangles belonging to valid surface data are finally backface-tested against all eye positions obtained from the viewing feedback and stored if any participant can see the triangle.

4.6 Pre-Blended Texture Generation

The gist of our texturing approach is to implicitly assign triangles to the sensor that has the best view of the triangles, using the calibration volume of the sensor. At reconstruction time, a triangle is implicitly assigned to the texture T_B corresponding to the sensor that has the best view of that triangle. If a triangle is visible to multiple sensors, color contributions from all contributing input textures (cmp. Section 4.2) are blended and added to T_B. Regions in other textures where the triangle would have been visible remain empty. The weighting of the texture contributions is done using the normalized weights of the individual color sensor contributions according to the centroid of the triangle. Algorithm 1 outlines the process of pre-blending texture information per triangle. In the algorithm, the triangle footprint refers to the set of pixels that comprises the projected and rasterized triangle, in texture space of the sensor for which it had highest weight.

Algorithm 1 Pre-Blended Texture Creation

Input Sensor Positions $\mathbf{P}$
Input RGBD Textures & Quality Maps per sensor $\mathbf{M}$
Output Blended & Masked RGB Textures per sensor $\mathbf{N}$

procedure BLEND_TEXTURES_PER_TRIANGLE
  for t in 0..num_triangles - 1 **do**
    sensor_weights$_{0..num_sensors-1}$ ← {0,..,0}
    for s in 0..num_sensors - 1 **do**
      sensor_weights$_s$ ← get_weight(t, $\mathbf{P_s}$, $\mathbf{M_s}$)
    h ← get_highest_weight_index(sensor_weights)
    rasterize_triangles(t, $\mathbf{M}$, $\mathbf{N_h}$, sensor_weights)
procedure GET_WEIGHT(t, $\mathbf{P_s}$, $\mathbf{M_s}$)
  cen ← get_triangle_centroid(t)
  depth ← sample_depth_map(cen, $\mathbf{M_s}$)
  if depth < TSDF_limit **then**
    quality ← sample_quality_map(cen, $\mathbf{M_s}$)
    sensor_distance ← length(cen - $\mathbf{P_s}$))
return quality / (sensor_distance + ϵ)
procedure RASTERIZE_TRIANGLES(t, $\mathbf{M}$, $\mathbf{N_h}$, $\mathbf{W}$)
  triangle_footprint ← get_best_tri_footprint(t, $\mathbf{N_h}$)
  for texel_pos in triangle_footprint **do**
    col ← blend_texels(t, $\mathbf{W}$, $\mathbf{M}$)
    write_color(texel_pos, blended_color, $\mathbf{N_h}$)

By collecting and pre-blending texture contributions from multiple input textures into a single texture and sampling it at implicitly determined positions using the calibration volumes, we do not only remove redundancy between the original color images, but also avoid the necessity of storing texture coordinates for the triangles. Texture regions which were not used for storing pre-blended color information for any triangle remain empty. In fact, we only transmit regions inside of a tight texture space bounding box for each original RGBD sensor, which leads to significantly reduced bandwidth requirements for the color texture information. Figure 9 provides an example of the pre-blended texture creation for an avatar that is only partially visible to a participant. Our texture creation approach scales well with

an increasing number of RGBD sensors, because the number of texels which actually store color information in the output texture is mainly determined by the resolution of the sensors and the maximal texture footprint of the set of extracted triangles, but it is largely independent of the number of sensors. Further sensors add information mainly in regions which are not seen by other sensors.

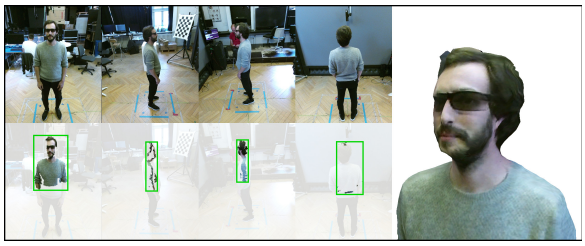


Fig. 9: Example of our pre-blended texture creation. The avatar (right) is seen by a local participant and its lower body is out of frustum. Areas in the input color textures (top row) contributing to visible triangles are blended and stored in the output textures (bottom row). Only texels contained in the tight texture bounding boxes (green) are transmitted to the clients.

4.7 Compression of Reconstructed Avatar Data

Although our output-sensitive reconstruction pipeline greatly reduces the amount of geometry and texture data generated for video avatars, data should be compressed prior to transmission to further reduce bit rates.

Systems similar to ours [9], [39] run both the remote reconstruction server and local rendering client in multi-threaded environments to compress geometry and texture data asynchronously with respect to the 3D reconstruction stage. Although our work does not focus on compression, we want to show realistic data rates achieved through standard compression stages that are easy to incorporate into the system, and are orthogonal to our reconstruction approach. For this reason, we apply standard intra-frame compression methods. Specifically, we compress the extracted geometry data using lossless compression on the CPU by means of deflation [40] and near lossless compression of the texture data using high-quality JPEG compression.

Prior to compression with these standard approaches, we extract the tight fitting texture space bounding boxes and encode the vertex position coordinates using 16 bit uniform quantization [9]. The texture coordinates of the triangles are implicitly defined by our pre-blending heuristic and the calibration volumes, which are sent to the clients at the beginning of the telepresence communication. Hence, we do not store per primitive texture coordinates.

5 Evaluation and Discussion

We evaluated our system in two telepresence scenarios in order to investigate the impact of output-sensitive avatars created by individual and group feedback. Specifically, we investigate and discuss the influence of our techniques on reconstruction, round-trip and rendering times as well as required bandwidth for the transmission of the avatars. We also discuss the scalability of our approach regarding the number of telepresence groups, participants per group, and RGBD sensors.

5.1 Test System Specifications

We set up a one-way telepresence system consisting of the following components (cmp. Section 3.3): a 3D capturing system, a global scene server, a 3D reconstruction server and a telepresence client. All systems are connected via a 10 GBit/s local area network. The 3D capturing system runs at 30 Hz using four synchronized *Kinect 4 Azure* each providing a color resolution of 2560 $\times$ 1440 pixels and a depth resolution of 640 $\times$ 576 pixels. The scene server runs on a machine equipped with an *Intel Xeon CPU E5-1650 v2* running at 3.5 GHz using less than 2 GB of RAM. The 3D reconstruction and occluder generation processes run on the same machine equipped with an *Intel Xeon E5-2687W v3* running at 3.1 GHz using two dedicated graphics cards (*NVIDIA Quadro RTX 6000*) and approximately 10 GB of RAM. The client machine is equipped with three graphics cards (*NVIDIA Quadro RTX 6000*) and an *Intel Xeon E5-2687W v4* running at 3.0 GHz using less than 4 GB of RAM and performs rendering for up to three collocated participants in parallel.

The evaluation was performed with client rendering resolutions of stereoscopic 4096 $\times$ 2160 pixels and a projection-based multi-user 3D display [38] of 3 $\times$ 2 meters size. Head and camera tracking were performed with an optical tracking system from ART (ar-tracking.com). The occluder buffers were rendered with a resolution of 1366 $\times$ 720 pixels per eye and conservatively downsampled [41] to a resolution of 256 $\times$ 144 pixels. The global scene server, the rendering client and the occluder generator are implemented as Avango/Guacamole [15] applications, whereas the 3D reconstruction server is implemented as a standalone OpenGL application.

5.2 Test Scenarios

We evaluated our output-sensitive telepresence architecture in two scenarios using virtual participants. The first scenario (*One-to-Two*) simulates a telepresence meeting between one local participant and a group consisting of two remote participants. The second scenario *(Three-to-Two)* simulates three local and two remote participants.

In the One-To-Two scenario, we positioned the local participant at 1.60 m in front of the screen with an eye

Fig. 10: Our Three-to-Two Scenario: Three collocated telepresence participants meet two remote participants in a shared virtual scene. Due to the size of the multi-user 3D display, the local participants see similar perspectives of the remote participants' 3D video avatars and the scene. Note that the photo camera assumes the role of a third local participant.

(a) Sequence ①, avg. values (184 | 93 | 26 | 7) k Triangles (1,768 | 980 | 317 | 62) MBit/s

(b) Sequence ②, avg. values (177 | 90 | 90 | 44) k Triangles (1,701 | 948 | 948 | 373) MBit/s

(c) Sequence ③, avg. values (183 | 39 | 39 | 39) k Triangles (1,739 | 457 | 457 | 457) MBit/s

(d) Sequence ④, avg. values (186 | 93 | 32 | 32) k Triangles (1,733 | 960 | 359 | 359) MBit/s

Fig. 11: Snapshots of the four telepresence situations we used in our evaluation rendered from the perspective of a local participant. From left to right: The remote avatars are extremely minified and partially occluded, minified and increasingly becoming larger, partially out of frustum and partially occluded. Please also refer to the video figure. The numbers below the images indicate the amount of extracted triangles and bit rates of the combined avatar geometry and texture streams at 60 Hz for our evaluated reconstruction modes (I | II | III | IV).

level at 1.75 m height and an eye distance of 6.5 cm. In the Three-to-Two scenario, the two additional participants were positioned one meter to the left and right of the first scenario's participant (Figure 10).

To increase reproducibility and consistency of our evaluation, we pre-recorded four sequences of two volunteers acting as remote participants using our 3D capturing system. We streamed the pre-recorded RGBD image sequences from hard disk over network to our 3D reconstruction server at a frame rate of 30 Hz, thus, simulating live capturing of remote participants. Note that, although the RGBD streams are captured at 30 Hz, our 3D reconstruction server creates and streams avatars at a frequency of 60 Hz to the rendering client to account for 3D display at 60 Hz as well as changing viewing perspectives.

As basis for our virtual environment we used the publicly available *Sponza* model and placed a lion statue as an additional large occluder in the middle of the hall.

Our emphasis is on different situations where avatars are minified, partially occluded, partially outside of the viewing frustum, or combinations of these (Figure 11). We used keyframe animation for navigating the remote avatars through the virtual scene and evaluated both telepresence scenarios for four different situations, each taking 10 seconds. Please also refer to the video figure.

5.3 Influence of Reconstruction Modes

We created four different reconstruction modes as combinations of *Back-face Culling (BFC)*, *Frustum Culling (FC)*, *Occlusion Culling (OC)* and *Level-of-Detail Geometry Extraction (LOD)* and measured their influence on the performance of our system for the *One-to-Two* scenario in comparison to the baseline, where avatars are created without our feedback mechanism. The modes evaluated are:

I Baseline, nothing enabled
II BFC + FC
III BFC + FC + OC
IV BFC + FC + OC + LOD

In the following, we present results for the four different evaluation sequences (Figure 11) with a focus on the *3D reconstruction time* (Figures 12a), the *number of generated triangles* (Figure 12b) and the *round-trip time* (Figure 12c).

5.3.1 3D Reconstruction Time

The performance of the 3D reconstruction process is illustrated in Figure 12a. The baseline group avatar causes 3D reconstruction times between 6 and 7 ms in all four sequences. Enabling backface and frustum culling (mode II) already discards approximately 50 percent of the geometry and texture information, yielding a performance gain of 1 to 2 ms in comparison to the baseline.

In sequence ① the lower part of the avatars is occluded by the balcony which saves 1 to 2 milliseconds of reconstruction time when occlusion culling (mode III) is enabled. With level-of-detail for geometry and texturing enabled (mode IV) 3D reconstruction takes only 2 to 2.5 ms for sequence ①.In sequence ② the avatars approach the local participant, hence, the geometric level-of-detail reaches full resolution and 3D reconstruction times increases from 2.5 to 4.5 ms, which is similar to using frustum and backface culling (mode II) alone.

In sequence ③ the three reconstruction modes benefit from our proposed feedback mechanism (II-IV) and perform similarly, since frustum culling can be applied effectively. In this situation, reconstruction times can be reduced by 30 to 39 percent in comparison to the baseline.

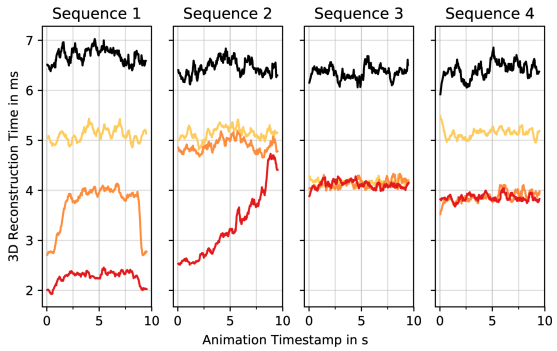
Finally, in sequence ④ the remote avatars meet with the local participants around a lion statue which acts as a large occluder. Consequently, the group avatar is largely occluded (Figure 11d) and the reconstruction server achieves similar performance gains for reconstruction modes III and IV as in sequence ③.

5.3.2 Number of Generated Triangles

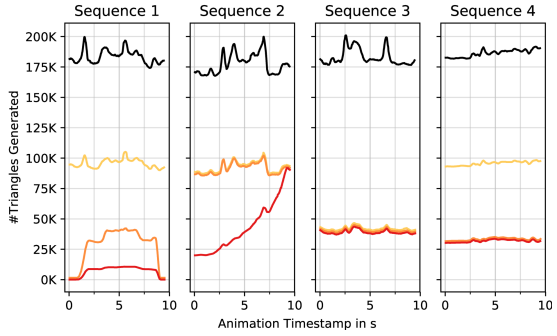
Subfigure 12b exposes similar patterns as reported previously, since the number of generated triangles and the reconstruction time are correlated. Note, that the orange and yellow graphs are slightly offset to avoid overplotting.

It is worth noticing that at the beginning and at the end of sequence ① the avatars are entirely occluded by either the balcony or the wall. For these extreme cases no avatar data is created, whereas in the middle of the sequence the reconstruction process creates a very low-resolution avatar resulting in only a few thousands triangles in mode IV.

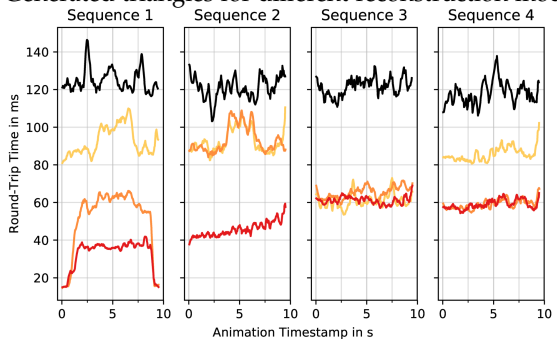
The reduction of geometry and texture resolution due to our level-of-detail approach might slightly change the appearance of the avatars. For this reason, we performed quan-



(a) Reconstruction times for different reconstruction modes.



(b) Generated triangles for different reconstruction modes.



(c) Round-trip times for different reconstruction modes.

Fig. 12: Comparison of the influence of our different reconstruction modes for the *One-to-Two* scenario. Compared to the baseline (black) we benefit from backface culling (yellow to red) in all four sequences. In addition, in sequences ① and ② level-of-detail extraction is effectively applied (red), because the group avatar is minified. In sequence ③ frustum culling is effective (yellow to red), because the avatars are partially outside of the viewing frustum in a face-to-face scenario. In sequence ④ occlusion culling (orange & red) becomes effective because the avatars are partially hidden behind the lion statue.

titative measurements of the perceptual difference between keyframes throughout our evaluation sequences using the structural similarity (SSIM) [42] measure. The observed SSIM values between 0.997 and 1.0 confirm that differences between the output-sensitive avatars and their full resolution counterpart would be difficult to notice which is in conformity with our observation.

5.3.3 Round-Trip Time

Figure 12c illustrates the round-trip times measured for the reconstruction modes using our feedback mechanism

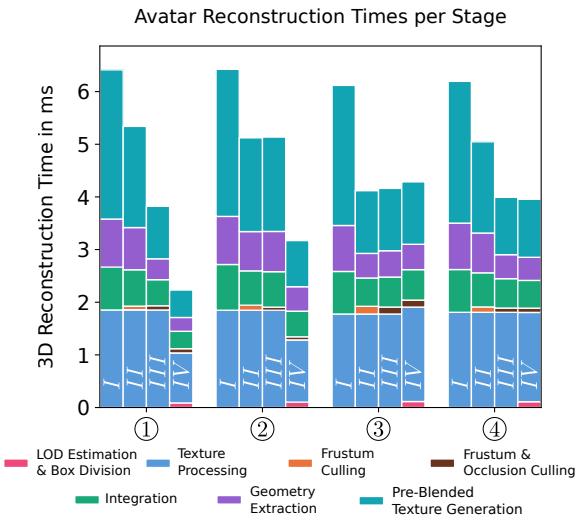


Fig. 13: Reconstruction times for different reconstruction modes (*I-IV*) averaged over evaluation sequences ①–④. The small avatar size in ① leads to the integration of a smaller sub volume of the TSDF volume and thus to faster integration as well as strongly reduced texture atlas generation times. The visibility culling methods notably decrease the integration and geometry extraction time due to exclusion of occupied but invisible bricks for the group in sequences ③ and ④.

in comparison to the baseline. The round-trip time includes sending feedback from the remote client, receiving feedback at the local reconstruction server, executing the 3D reconstruction pipeline, upload of avatar data from GPU to CPU, compression, sending avatar data over network, receiving data on the client side and decompressing the data.

We are aware that round-trip times are typically between 5 and 30 ms between cities in Europe[1]. Since our tests were performed in a local area network, the basic network round-trip time has to be added to our measurements for an estimate of the total round-trip time.

Our measurements reveal round-trip times between 103 to 146 ms for the baseline condition in all tested sequences. Using our approach for 3D reconstruction (mode IV) round-trip times are reduced to 15 to 68 ms depending on the amount of extracted geometry.

Although the 3D reconstruction requires less than 5 ms, the round-trip times are higher in our system since several stages are pipelined and asynchronous to the reconstruction itself, i.e. GPU to CPU upload, compression on the server side and decompression on the client side. However, it is important to emphasize that all pipelined stages benefit from our proposed feedback mechanism, because we generate significantly less data compared to the baseline.

5.3.4 3D Reconstruction Components Performance

Reconstruction times are greatly reduced in any of the four evaluated sequences using our proposed feedback mechanisms. For a more detailed analysis, Figure 13 summarizes the average processing times of the different components used in our reconstruction module.

In the baseline condition (mode I), most time is spent on texture processing. Working on four 2560 × 1440 pixel

1. https://wondernetwork.com/pings/Frankfurt

color streams causes both texture stages together to account for 68 to 69 percent of the total 3D reconstruction time. Therefore, it is crucial to reduce the amount of potentially visible texture information during processing.

The texture preprocessing stage benefits from our level-of-detail mechanism. Early filtering and downsampling allows for texture processing time reduction by 49 percent in case of strong minification (Sequence ①) and on average by 36 percent in case of moderate minification (Sequence ②). Although texture filtering is not fully optimized in our GPU implementation, an ideal system could apply texture downsampling directly within the RGBD sensor.

Moreover, all subsequent processing stages sufficiently benefit from our visibility culling techniques and achieve a significant speedup. As an example, processing time for TSDF integration can be reduced by 44 to 59 percent in situations where geometric level-of-detail extraction is effective and by 28 to 35 percent in situations where avatars have to be extracted at full level-of-detail. For the integration and geometry extraction stage we expect a higher gain from identifying invisible bricks for increasing TSDF volume and depth stream resolutions. Nevertheless, we already observe a notable performance gain of 45 to 71 percent in our evaluation sequences for the geometry extraction stage.

Finally, the amount of triangles retrieved from the geometry extraction stage yields proportional work load in the pre-blended texture atlas generation stage. Depending on the sequence, the time for texture blending and atlas generation can be reduced by 55 to 82 percent.

5.4 Group Feedback versus Individual Feedback

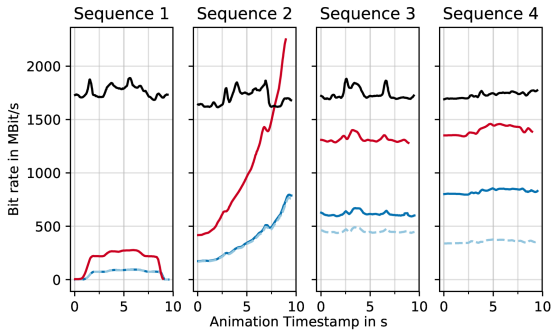
We evaluate the efficiency of our proposed *group feedback* (cmp. Figure 3) in comparison to the *individual feedback* mode in the *Three-to-Two* scenario.

We compared three different feedback configurations: three telepresence clients sending *individual feedback* for the participants (dashed light-blue graph), one client sending *group feedback* (solid dark-blue graph) and a client sending no feedback at all (black graphs) as our baseline. During our evaluation (see Figure 14), we measure notable reduction of bit rates between 52 to 96 percent and reduction of pure avatar draw time between 43 to 90 percent depending on the sequence with no or little overhead to a theoretical optimum a single user providing feedback could achieve. The only scenario, in which the overhead of the group feedback is notable with respect to a theoretical optimum of an individual user is in ④. Here, the occluder buffers are highly different for the three participants in the group, because of the close proximity of the lion statue. Nevertheless, the overhead of the group feedback is still approximately 160 percent lower than the overhead created by sending individual feedback and therefore receiving three individual avatars (red graph).

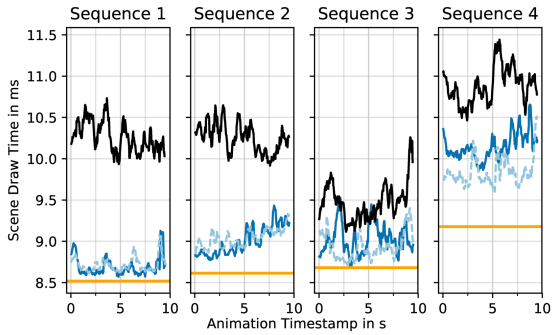
In summary, our evaluation reveals that providing group feedback is clearly more efficient than providing individual feedback or no feedback at all for a group of collocated participants.

5.5 Scalability

We evaluated our architecture in a *One-to-Two* and a *Three-to-Two* scenario using a 3D capturing system consisting of four RGBD sensors. In this section we briefly discuss how our architecture scales with respect to the number of RGBD sensors, telepresence parties, and participants per party.



(a) Bit rates for different feedback modes for 60 Hz avatar transmission.



(b) Draw times for different feedback modes. Scene-only draw time in ms: ①: 8.5, ② 8.6, ③ 8.7, ④ 9.2 (orange lines)

Fig. 14: Overhead of *group* versus *individual feedback* for bit rates and scene draw times. The graphs show a slight overhead of the group feedback compared to the individual feedback of a central user. However, compared to the baseline condition and the sum of individual feedback, our proposed group feedback proves to be effective in most cases.

In our 3D reconstruction pipeline only the texture processing, occupancy tracking and TSDF integration stages are influenced directly by the number of RGBD sensors. Consequently, the reconstruction time increases linearly for these stages if additional sensors are added to a 3D capturing system. However, texture processing could be performed in parallel for all sensors using additional GPUs.

Both the costs for the occluder generation process as well as the combined frustum and occlusion culling stage scale linearly with the number of telepresence participants in a remote party for which the avatar representations of a local party are generated. The number of participants in a telepresence party typically increases the reconstructed surface area of the avatar. If each participant is fully visible to the capturing system, the TSDF processing costs increase linearly with the number of participants.

Besides reconstruction performance, network capacity remains a limiting factor for an increasing number of telepresence parties. In situations where multiple parties meet, our system already reduces network transfer rates for many meeting configurations, since individual user representations are often reconstructed at lower detail or are only partially visible Furthermore, we are also able to reduce the

level-of-detail for the avatar geometry and texture to trade network bandwidth for avatar quality.

6 Conclusion and Future Work

We present the idea of creating, transferring and rendering output-sensitive 3D video avatars in multi-party 3D telepresence systems. Our 3D reconstruction process considers projected screen area, scene occlusion and viewing frustum parameters of groups of remote participants to avoid the creation of avatar geometry and textures that are invisible. This is made possible by adjusting the resolution of the TSDF volume and the color textures in real-time, and using a brick-based data structure that accumulates the visibility information and guides the TSDF extraction process accordingly. The color textures are pre-blended and masked to represent and transmit each visible texel only once. As a result, the output-sensitive avatar is represented by significantly less geometry and texture information, usually by at least 50 percent. In cases where the avatar is largely occluded, outside of the participants' viewing frusta, or extremely minified, our approach creates and transmits barely any data. In our evaluation sequences, we achieve 3D reconstruction time reductions of up to two thirds and by 50 percent on average, as well as avatar rendering time reductions between 43 and 90 percent throughout. Most importantly for telepresence applications in an environment with low network bandwidth, we achieve a bit rate reduction of 77 percent on average, and at least 51 percent at any point in our evaluation scenarios. Our output-sensitive avatar creation can be applied to any reconstruction approach that extracts the final avatar from a TSDF volume [4], [5], and the general idea of output-sensitive avatar reconstruction, compression and transmission is compatible with most immersive telepresence systems.

The scalability of our approach is limited since the workload of a reconstruction server depends on the number of remote telepresence parties taking part in the communication. While typical telepresence scenarios involve only a small number of parties, a larger number could be supported by using personalized model-based representations (e.g. [43]) for avatars that are only seen from a larger distance. However, imperceptible switching between a live reconstructed 3D video avatar and the model-based representation might be a challenge. A promising extension of our proposed group feedback would be dynamic clustering of remote telepresence parties into clusters that share similar viewing perspectives into the virtual world. As a result, the reconstruction server would create avatar representations based on the merged feedback of several groups, and therefore execute the reconstruction pipeline only once per cluster, before broadcasting the result to the cluster.

It is our vision that 3D telepresence will become a standard communication modality. However, bandwidth is and will remain a limited resource in the foreseeable future. Furthermore, considering the increasing resolution of 3D sensors and 3D displays, output-sensitive video avatars are a sustainable contribution to making high quality immersive 3D telepresence communication a reality, since they avoid reconstructing, transferring and rendering information not contributing to the final image of any participant.

Acknowledgments

Our research received funding from the Thuringian Ministry for Economic Affairs, Science and Digital Society under grant 5575/10-5 (*MetaReal*). We thank the members of the Virtual Reality and Visualization Research Group at Bauhaus-Universität Weimar (http://www.uni-weimar.de/vr) for their support and the reviewers of this paper for their constructive feedback that helped to improve the paper significantly.

References

[1] A. Maimone and H. Fuchs, "Encumbrance-free telepresence system with real-time 3d capture and display using commodity depth cameras," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 137–146, 2011.

[2] S. Beck, A. Kunert, A. Kulik, and B. Froehlich, "Immersive group-to-group telepresence," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 4, pp. 616–625, 2013.

[3] S. Orts-Escolano, C. Rhemann, S. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, D. Kim, P. L. Davidson, S. Khamis, M. Dou, V. Tankovich, C. Loop, Q. Cai, P. A. Chou, S. Mennicken, J. Valentin, V. Pradeep, S. Wang, S. B. Kang, P. Kohli, Y. Lutchyn, C. Keskin, and S. Izadi, "Holoportation: Virtual 3d teleportation in real-time," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pp. 741–754, 2016.

[4] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. O. Escolano, C. Rhemann, D. Kim, J. Taylor, P. Kohli, V. Tankovich, and S. Izadi, "Fusion4d: Real-time performance capture of challenging scenes," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 1–13, 2016.

[5] M. Dou, P. Davidson, S. R. Fanello, S. Khamis, A. Kowdle, C. Rhemann, V. Tankovich, and S. Izadi, "Motion2fusion: Real-time volumetric performance capture," *ACM Trans. Graph.*, vol. 36, no. 6, pp. 1–16, 2017.

[6] D. S. Alexiadis, D. Zarpalas, and P. Daras, "Real-time, realistic full-body 3d reconstruction and texture mapping from multiple kinects," in *IVMSP 2013*, pp. 1–4, 2013.

[7] R. Mekuria, D. Alexiadis, P. Daras, and P. Cesar, "Real-time encoding of live reconstructed mesh sequences for 3d tele-immersion," in *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pp. 1–6, 2013.

[8] R. Mekuria, M. Sanna, E. Izquierdo, D. C. A. Bulterman, and P. Cesar, "Enabling geometry-based 3-d tele-immersion with fast mesh compression and linear rateless coding," *IEEE Transactions on Multimedia*, vol. 16, no. 7, pp. 1809–1820, 2014.

[9] R. Mekuria, K. Blom, and P. Cesar, "Design, implementation, and evaluation of a point cloud codec for tele-immersive video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 4, pp. 828–842, 2017.

[10] F. Pece, J. Kautz, and T. Weyrich, "Adapting standard video codecs for depth streaming," in *JVRC11: Joint Virtual Reality Conference of EGVE - EuroVR, Nottingham, UK, 2011. Proceedings*, S. Coquillart, A. Steed, and G. Welch, Eds., pp. 59–66, 2011.

[11] Y. Liu, S. Beck, R. Wang, J. Li, H. Xu, S. Yao, X. Tong, and B. Froehlich, "Hybrid lossless-lossy compression for real-time depth-sensor streams in 3d telepresence applications," in *Advances in Multimedia Information Processing – PCM 2015*, Y.-S. Ho, J. Sang, Y. M. Ro, J. Kim, and F. Wu, Eds., pp. 442–452, 2015.

[12] Sung-Eui Yoon, B. Salomon, R. Gayle, and D. Manocha, "Quick-vdr: interactive view-dependent rendering of massive models," in *IEEE Visualization 2004*, pp. 131–138, 2004.

[13] K. Niski, B. Purnomo, and J. Cohen, "Multi-grained level of detail using a hierarchical seamless texture atlas," in *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, pp. 153–160, 2007.

[14] P. Goswami, F. Erol, R. Mukhi, R. Pajarola, and E. Gobbetti, "An efficient multi-resolution framework for high quality interactive rendering of massive point clouds using multi-way kd-trees," *The Visual Computer*, vol. 29, no. 1, pp. 69–83, 2013.

[15] S. Schneegans, F. Lauer, A. Bernstein, A. Schollmeyer, and B. Froehlich, "guacamole - an extensible scene graph and rendering framework based on deferred shading," in *2014 IEEE 7th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, pp. 35–42, 2014.

[16] P. Bourdin, J. M. T. Sanahuja, C. C. Moya, P. Haggard, and M. Slater, "Persuading people in a remote destination to sing by beaming there," in *Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology*, pp. 123–132, 2013.
[17] M. E. Latoschik, F. Kern, J. Stauffert, A. Bartl, M. Botsch, and J. Lugrin, "Not alone here?! scalability and user experience of embodied ambient crowds in distributed social virtual reality," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 5, pp. 2134–2144, 2019.
[18] A. Beacco, B. Spanlang, C. Andújar, and N. Pelechano, "Output-sensitive rendering of detailed animated characters for crowd simulation," in *CEIG Spanish Conference on Computer Graphic*, 2010.
[19] D. J. Roberts, A. J. Fairchild, S. P. Campion, J. O'Hare, C. M. Moore, R. Aspin, T. Duckworth, P. Gasparello, and F. Tecchia, "withyouan experimental end-to-end telepresence system using video-based reconstruction," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 3, pp. 562–574, 2015.
[20] H. Fuchs, A. State, and J. Bazin, "Immersive 3d telepresence," *Computer*, vol. 47, no. 7, pp. 46–52, 2014.
[21] D. S. Alexiadis, D. Zarpalas, and P. Daras, "Real-time, full 3-d reconstruction of moving foreground objects from multiple consumer depth cameras," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 339–358, 2013.
[22] D. Alexiadis, A. Doumanoglou, D. Zarpalas, and P. Daras, "A case study for tele-immersion communication applications: From 3d capturing to rendering," in *2014 IEEE Visual Communications and Image Processing Conference*, pp. 278–281, 2014.
[23] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pp. 303–312, 1996.
[24] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, 1987.
[25] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan, "High-quality streamable free-viewpoint video," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 1–13, 2015.
[26] N. A. Carr and J. C. Hart, "Meshed atlases for real-time procedural solid texturing," *ACM Trans. Graph.*, vol. 21, no. 2, pp. 106–131, 2002.
[27] S. Beck and B. Froehlich, "Sweeping-based volumetric calibration and registration of multiple rgbd-sensors for 3d capturing systems," in *2017 IEEE Virtual Reality (VR)*, pp. 167–176, 2017.
[28] K. Mamou, T. Zaharia, and F. Prêteux, "Tfan: A low complexity 3d mesh compression algorithm," *Computer Animation and Virtual Worlds*, vol. 20, no. 2, pp. 343–354, 2009.
[29] S. Lee, B. Koo, H. Kim, C. Chu, B. Kim, D. Kim, K. Son, K. Choi, E.-Y. Chang, and E. S. Jang, "Quantization-based compact representation 3d mesh," *ISO/IEC, Technical Report*, 2008.
[30] D. Tang, M. Dou, P. Lincoln, P. Davidson, K. Guo, J. Taylor, S. Fanello, C. Keskin, A. Kowdle, S. Bouaziz, S. Izadi, and A. Tagliasacchi, "Real-time compression and streaming of 4d performances," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–11, 2018.
[31] D. Tang, S. Singh, P. A. Chou, C. Häne, M. Dou, S. Fanello, J. Taylor, P. Davidson, O. G. Guleryuz, Y. Zhang, S. Izadi, A. Tagliasacchi, S. Bouaziz, and C. Keskin, "Deep implicit volume compression," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1290–1300, 2020.
[32] M. Gross, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Gool, S. Lang, K. Strehlke, A. V. Moere, and O. Staadt, "blue-c: a spatially immersive display and 3d video portal for telepresence," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 819–827, 2003.
[33] S. Würmlin, E. Lamboray, and M. Gross, "3d video fragments: Dynamic point samples for real-time free-viewpoint video," *Computers & Graphics*, vol. 28, no. 1, pp. 3 – 14, 2004.
[34] E. Lamboray, S. Wurmlin, and M. Gross, "Real-time streaming of point-based 3d video," in *IEEE Virtual Reality 2004*, pp. 91–281, 2004.
[35] M. Willert, S. Ohl, and O. Staadt, "Reducing bandwidth consumption in parallel networked telepresence environments," in *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, pp. 247–254, 2012.
[36] X. Wang, Y. A. Şekercioğlu, T. Drummond, E. Natalizio, I. Fantoni, and V. Frémont, "Collaborative multi-sensor image transmission and data fusion in mobile visual sensor networks equipped with rgb-d cameras," in *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 1–8, 2016.
[37] C. Plüss, N. Ranieri, J.-C. Bazin, T. Martin, P.-Y. Laffont, T. Popa, and M. Gross, "An immersive bidirectional system for life-size 3d communication," in *Proceedings of the 29th International Conference on Computer Animation and Social Agents*, pp. 89–96, 2016.
[38] A. Kulik, A. Kunert, S. Beck, R. Reichel, R. Blach, A. Zink, and B. Froehlich, "C1x6: A stereoscopic six-user display for co-located collaboration in shared virtual environments," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 1–12, 2011.
[39] A. J. Fairchild, S. P. Campion, A. S. García, R. Wolff, T. Fernando, and D. J. Roberts, "A mixed reality telepresence system for collaborative space operation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 4, pp. 814–827, 2017.
[40] Y. Collet, "Lz4 extremely fast compression algorithm," 2015. [Online]. Available: https://github.com/lz4/lz4
[41] N. Greene, M. Kass, and G. Miller, "Hierarchical z-buffer visibility," in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 231–238, 1993.
[42] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.
[43] T. Waltemate, D. Gall, D. Roth, M. Botsch, and M. E. Latoschik, "The impact of avatar personalization and immersion on virtual body ownership, presence, and emotional response," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 4, pp. 1643–1652, 2018.

Adrian Kreskowski is a PhD candidate with the Virtual Reality and Visualization Research Group at Bauhaus-Universität Weimar. His research interests include output-sensitive real-time rendering, time-varying model visualization and general purpose GPU programming.

Stephan Beck is a postdoctoral researcher with the Virtual Reality and Visualization Research Group at Bauhaus-Universität Weimar. He is enthusiastic about designing and implementing frameworks that enable collaborative virtual reality applications. His current research focuses on 3D telepresence as well as on real-time rendering of large scanned data.

Bernd Froehlich is professor with the Computer Science Department and head of the Virtual Reality and Visualization Research Group at Bauhaus-Universität Weimar. His research interests include real-time rendering, visualization, input devices, 3D interaction techniques, display technology, and support for collaboration in co-located and distributed virtual environments.