Volumetric Avatar Reconstruction with Spatio-Temporally Offset RGBD Cameras

Gareth Rendle* Adrian Kreskowski† Bernd Froehlich‡

Virtual Reality and Visualization, Bauhaus-Universität Weimar, Germany

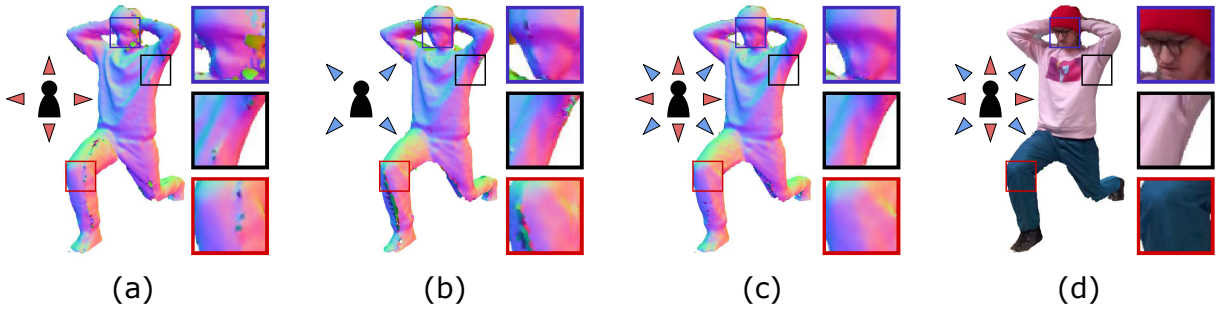

Figure 1: We propose a volumetric avatar capture technique that can increase the effective capture frame rate of a set of RGBD cameras from 30 Hz to 60 Hz. In our configuration, cameras are assigned to one of two spatio-temporally offset capture groups, group A (red triangles) and group B (blue triangles). The surfaces reconstructed from each group exhibit different artefacts, as shown in images (a) and (b). We also propose a reconstruction pipeline that consumes images from alternating capture groups, leveraging temporal volumetric fusion to blend surfaces from previous frames into the current reconstruction frame, mitigating temporal artefacts (c). Colors from previous frames are also blended into the texture applied to the geometry (d).

Abstract

RGBD cameras can capture users and their actions in the real world for reconstruction of photo-realistic volumetric avatars that allow rich interaction between spatially distributed telepresence parties in virtual environments. In this paper, we present and evaluate a system design that enables volumetric avatar reconstruction at increased frame rates. We demonstrate that we can overcome the limited capturing frame rate of commodity RGBD cameras such as the Azure Kinect by dividing a set of cameras into two spatio-temporally offset reconstruction groups and implementing a real-time reconstruction pipeline to fuse the temporally offset RGBD image streams. Comparisons of our proposed system against capture configurations possible with the same number of RGBD cameras indicate that it is beneficial to use a combination of spatially and temporally offset RGBD cameras, allowing increased reconstruction frame rates and scene coverage while producing temporally consistent volumetric avatars.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities

1 Introduction

In social virtual reality applications, the nature of users' visual representations can affect the level of social presence [6, 49, 50] and co-presence [2, 23, 50] experienced by users, as well as the quality of social interactions [40] and the trust that users feel towards communication partners [2, 26]. Steed and Schroeder [42] highlight the difference between *tracked* avatars, like those animated from tracked headset and controller positions commonly used in current social virtual reality platforms[1,2], and *reconstructed* avatars, where an accurate representation of the user is reconstructed in real time. Reconstructed avatars, also referred to as *volumetric avatars*, can facilitate rich communication between collaborators by conveying facial expressions and accurate full-body movements. In application areas where interpersonal communication is paramount, such as in immersive telepresence scenarios, systems are required that can reconstruct and transmit avatars with high-resolution geometry and textures in real-time.

Consequently, real-time reconstruction of volumetric avatars has been an active field of research in recent years, producing methods that reconstruct detailed geometry at high frame rates from images captured by custom depth sensors [10, 11, 38]. A parallel research thread, initiated by the introduction of consumer-level RGBD cameras such as the Microsoft Kinect, focuses on reconstructing avatar streams with commercially available sensors [1, 4, 32–34]. Commodity RGBD cameras typically have a lower frame rate than the cameras used in the state-of-the-art methods. For example, Microsoft's latest RGBD camera, the Azure Kinect[3], has a maximum frame rate of 30 Hz, compared to the frame rate of 200 Hz achieved by the custom depth sensor used in the *Motion2Fusion* system [10]. Lower capture frame rates mean that reconstructed motion appears less smooth than that produced by state-of-the-art methods. Reconstructing volumetric avatar streams with a high frame rate is of particular interest in virtual reality applications, where display refresh rates are typically 60 Hz or higher, and rendering moving visual stimuli at a significantly lower frame rate than the display's refresh rate leads to unpleasant judder effects [8, 41].

The work presented in this paper aims to enable reconstruction of volumetric avatar streams at high frame rates with commodity RGBD cameras. We propose a capture configuration and reconstruc-

*e-mail: gareth.rendle@uni-weimar.de
†e-mail: adrian.kreskowski@uni-weimar.de
‡e-mail: bernd.froehlich@uni-weimar.de

[1] oculus.com/workrooms
[2] vrchat.com
[3] azure.microsoft.com/services/kinect-dk

tion pipeline that produces avatar streams with twice the maximum capture frame rate of the employed RGBD camera model, while maintaining scene coverage and temporal consistency. Our capture configuration is guided by two observations relating to avatar reconstruction with multiple RGBD cameras. Firstly, we observe that by *spatially* distributing cameras around the capture subject, greater scene coverage can be achieved, reducing holes and inaccuracies caused when parts of the surface are occluded or poorly observed. This observation in itself is not novel and is commonly seen in camera configurations that produce 360-degree avatar reconstructions [10, 11, 38, 45, 52]. Our second observation is that by *temporally* offsetting some of the available cameras by half of the interval between capture frames, the effective capture frame rate of a set of cameras can be increased to twice the capture frame rate of a set of temporally aligned cameras. We combine these observations to create a *spatio-temporally* offset capture configuration, where cameras are assigned to one of two groups that are spatially *and* temporally offset from each other, as shown in Figure 2. The temporal offset means that the two capture groups provide color and depth images to the reconstruction pipeline in an alternating manner. If surfaces reconstructed from different capture groups were rendered at consecutive frames, disturbing flickering artefacts would occur, as reconstructions from different groups are likely to exhibit view-dependent differences. We therefore take inspiration from existing temporal volumetric fusion methods [11, 36] by blending surface contributions from previous frames into the current frame, producing a temporally consistent volumetric avatar stream.

The contributions described in this work are:

- a novel volumetric avatar capturing configuration that, as shown through evaluation, increases the effective capture frame rate and scene coverage;
- a real-time reconstruction pipeline for use with the introduced capture configuration, which leverages a temporal fusion approach to produce temporally consistent avatar geometry streams;
- a method for spatio-temporal blending of color contributions from each camera to maintain temporally consistent texturing of the volumetric avatar.

We evaluated our approach with two groups of four Microsoft Azure Kinect cameras. Our approach effectively doubles the frame rate of the volumetric avatar reconstruction from 30 to 60 Hz and delivers smoothly moving and consistent volumetric avatar streams.

2 Related Work

Research contributions addressing reconstruction of volumetric avatars can be categorized into *offline* and *real-time* approaches. Offline methods target playback of so-called Free-Viewpoint Video after reconstruction [7, 18, 35, 39], and are therefore unsuitable for telepresence applications, for which avatar reconstruction must be achieved in real-time to support interactivity between users.

One class of real-time techniques fits scanned template geometry or human priors to RGBD input [46], and includes contributions that are able to reconstruct complete avatar geometry from monocular input [5, 19, 21, 22, 30, 48, 51, 53, 54]. Although reducing the number of capture cameras is desirable, these methods do not generalize to scene topology changes, such as the spontaneous use of props.

Real-time template-free methods achieve smooth reconstruction quality by leveraging temporal volumetric fusion to blend surface geometry observed in previous frames with that of the current frame [10, 11, 20, 24, 36, 38]. A prerequisite of temporal fusion is knowledge of the deformation undergone by the reconstructed surface between frames. Deformation estimation is typically framed as an energy minimization problem, within which the parameters of a non-rigid deformation field are found that minimize energy terms quantifying poor alignment between the deformed and target surface, and undesirable types of deformation, namely non-regular and non-rigid deformation [43]. Successful deformation approaches are often guided by point-to-point correspondences between the source and target geometry, such as those obtained from SIFT features [24]. Correspondence identification has been accelerated by matching RGB image patches using decision forests [11], or approximating the transformation of a mesh into the spectral domain where correspondences can be more efficiently identified [10]. Guo et al. [20] use the intrinsic appearance of the avatar to estimate motion, in a joint approach that estimates motion and albedo simultaneously.

The state-of-the-art approach to real-time volumetric avatar reconstruction, by Dou et al. [10], employs active IR stereo depth sensors that can achieve a frame rate of 200 FPS and a depth map resolution of 1280×1024 pixels [14]. However, since the hardware proposed in the aforementioned work is not widely available, many approaches focus instead on reconstructing 3D scenes using commodity RGBD sensors. The release of the Microsoft Kinect sensor motivated research into methods that could create smooth, complete reconstructions of 3D scenes using commercially available sensors [37], and subsequently yielded approaches for reconstructing dynamic scenes [36], although reconstruction was prone to failure when large inter-frame motions were present. Reconstruction approaches that only require a single commodity RGBD sensor as input are cost effective, but either rely on human priors [5, 30, 53] or suffer from occlusion issues inherent when viewing a scene from a single perspective [15, 20, 27].

Reconstruction methods that use multiple commodity RGBD cameras construct more complete scene geometry than single-camera methods by fusing point clouds obtained from multiple perspectives. Approaches used to combine surfaces observed from different cameras include overlaying or zippering multiple triangle meshes [4, 32, 33], fitting local surfaces using moving least squares [34], and integrating depth observations into a volume [1] that implicitly encodes the surface as a Truncated Signed Distance Function (TSDF) [9]. More recent methods developed for multiple commodity RGBD cameras typically use human priors [54], or are heavily assisted by learning-based geometry and texture inference [44, 52]. The acquisition and preparation of training data required to make use of these techniques is a barrier for some users, and training reconstruction pipelines with data from scenes containing only human actors can make reconstruction methods less generalizable to telepresence scenarios involving props.

The referenced approaches to avatar reconstruction with multiple commodity RGBD cameras cannot achieve the frame rates possible in the state-of-the-art real-time volumetric avatar reconstruction methods due to the limitations of the RGBD cameras themselves: the Azure Kinect, for example, has a maximum capture frame rate of 30 FPS. In this work, we address this limitation by proposing a novel capture configuration and reconstruction pipeline that increases the frame rate of the reconstructed volumetric avatar stream from 30 Hz to 60 Hz when using the Azure Kinect.

3 System Description

We propose a novel approach to reconstructing real-time volumetric avatars with commodity RGBD cameras. The core idea of our approach is the conceptual separation of a set of synchronizable RGBD cameras into two groups, which we refer to as *capture groups*. By temporally offsetting the center of exposure of one group by half of the time interval between camera frames at maximum capture frame rate, the capture groups provide sets of color and depth images to the reconstruction pipeline in an alternating manner at double the maximum capture frame rate of an individual sensor. Instead of spatially aligning pairs of cameras from different capture groups, all cameras are distributed evenly around the capture volume to achieve high scene coverage, reducing occlusions in complex scenes.

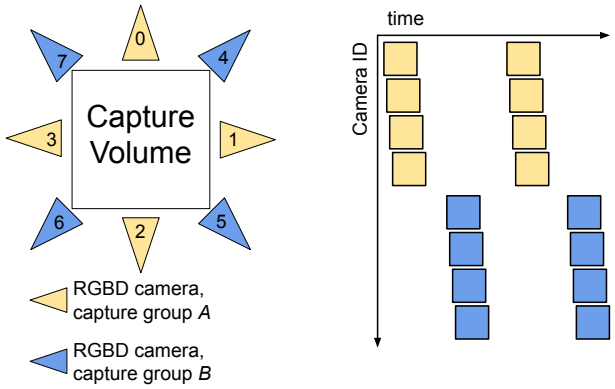

Figure 2: RGBD camera positioning and conceptual grouping used in our proposed avatar reconstruction system is shown on the left. Cameras are divided into two capture groups that provide images to a reconstruction server in an alternating manner. The image on the right represents the images captured from each group as yellow and blue squares respectively, and illustrates the temporal offset between the centers of exposure of cameras in each group. Cameras within a group are slightly offset to avoid interference (see Section 3.1.2)

.

Since the volumetric avatar is reconstructed from alternating sets of views at every reconstruction frame, the resulting geometry is likely to exhibit flickering as view-dependent artifacts repeatedly appear and disappear. To mitigate such artifacts and maintain temporal consistency, geometry and color information computed in previous frames is fused into the current reconstruction frame.

In Section 3.1, we detail the hardware and software configuration used to capture RGBD images for our avatar reconstruction system. In Section 3.2, our reconstruction pipeline is described.

Preliminaries. We denote the number of available RGBD cameras as N. The cameras are assigned to one of two capture groups, A and B, each consisting of $\frac{N}{2}$ cameras. While the proposed approach is suitable for any even number of cameras $N > 1$, we use $N = 8$ to enable each group of four cameras to provide 360-degree coverage of the capture volume, as shown in Figure 2. Cameras are arranged such that neighboring cameras belong to different capture groups. The capture configuration used for the evaluation of our approach employs eight Azure Kinect units. Although we refer to some specific features of the Azure Kinect in our text, our approach is applicable to any set of RGBD cameras that support synchronization.

3.1 RGBD Image Capture

This section describes how the available RGBD cameras should be arranged and configured to capture depth and color images for our volumetric avatar reconstruction pipeline.

3.1.1 Calibration and Registration

We use the factory intrinsic parameters for the color and depth sensors, as well as the factory extrinsic parameters encoding the rigid transformation between the color and depth coordinate systems. To register the cameras into a shared coordinate system, we employ a tracked checkerboard, as described by Beck et al. [3], which enables calculation of the world space positions of the checkerboard pattern's corners. Corresponding positions in depth sensor space are obtained by locating 2D corner points in the infrared image captured by the depth sensor, and back-projecting those points into 3D using the depth sensor's intrinsic parameters. For the registration of each camera, we obtain depth and world space points at 8 checkerboard positions inside the capture volume. A least-squares estimation of transformation parameters between the depth and world space points then provides the extrinsic parameters for each depth sensor.

3.1.2 Synchronization

In order to temporally offset one capture group from the other, it is necessary to instruct one capture group to capture with a fixed delay with respect to the other group. This is possible with the Azure Kinect, which supports programmatic configuration of the synchronization between cameras. One camera is chosen as the master that provides a synchronization signal to subordinate cameras. In general multi-Kinect configurations, cameras should operate with 160 μs offsets between their respective centers of exposure, to avoid interference between cameras[4]. In addition to offsetting the center of exposure of each camera by multiples of 160 μs, we delay captures from cameras in capture group B by 16667 μs, which corresponds to half the time interval between frames at 30 FPS capture frame rate, the maximum supported by Azure Kinect cameras. The resulting synchronization offsets for all cameras are shown in Table 1.

Camera	Group	Role	Delay w.r.t. master
0	A	Master	0 μs
1	A	Subordinate	160 μs
2	A	Subordinate	320 μs
3	A	Subordinate	480 μs
4	B	Subordinate	16667 μs
5	B	Subordinate	16827 μs
6	B	Subordinate	16987 μs
7	B	Subordinate	17147 μs

Table 1: Synchronization parameters used in our spatio-temporally offset capture configuration for eight cameras.

3.1.3 Capture Server

We implement a capture server that configures the available Kinect cameras according to the synchronization parameters shown in Table 1. The server runs two asynchronous processes: one retrieves depth and color images from cameras in alternating capture groups, and one transmits image data to a local or remote reconstruction server.

3.2 Volumetric Avatar Reconstruction

Our approach is dependent on a reconstruction pipeline that can consume color and depth images from alternating capture groups and produce temporally consistent volumetric avatars. The reconstruction pipeline consists of several sequential processing stages, shown in Figure 3. In this section, we give an overview of the stages, before describing them in detail in Sections 3.2.2 to 3.2.5.

3.2.1 Reconstruction Pipeline Overview

The reconstruction pipeline is executed once per reconstruction frame, with the current reconstruction frame denoted as t_r. The input to the pipeline is a set of color and depth images captured at t_r.

In the first stage of the pipeline, the depth images are processed to remove redundant data and counteract noise, producing a set of filtered depth maps. This stage also generates auxiliary textures including *filled silhouette* and *visual hull* textures that are necessary for fusion of surfaces reconstructed from different capture groups.

Next, a surface representing the current frame is calculated by integrating the filtered depth images into a Truncated Signed Distance Field (TSDF), henceforth referred to as the *data volume*.

In order to create a temporally consistent volumetric avatar stream, our pipeline blends surface information from previous frames into the data volume. To allow corresponding parts of the current and

[4] docs.microsoft.com/azure/kinect-dk/multi-camera-sync

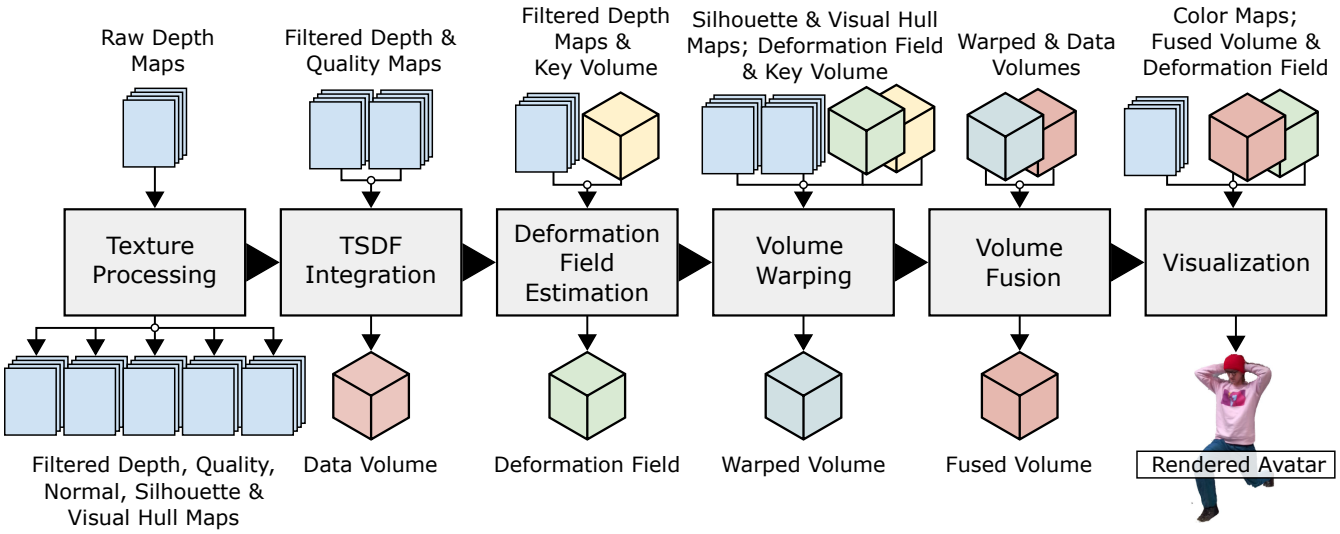

Figure 3: Overview of our volumetric avatar reconstruction pipeline as described in Section 3.2.1.

previous surfaces to be blended, it is necessary to estimate the surface deformation that occurred between frames t_{r-1} and t_r. We adopt concepts from the work by Dou et al. [11] to estimate the deformation field, deform the surface reconstructed at the previous frame, and fuse the deformed surface into the data volume.

The deformation field estimation stage constructs an optimization problem to calculate a deformation field consisting of a sparse set of transformations that warp the surface reconstructed at t_{r-1} (stored in the *key volume*) to align with the filtered depth images from t_r.

The resulting deformation field is then used to deform (or "warp") the surface represented by the key volume, producing a surface that should align with the surface in the data volume, referred to as the *warped volume*. During this stage, misaligned surfaces are detected and excluded from integration into the warped volume using the auxiliary maps generated in the texture processing stage.

After the warped volume has been created, the *warped* and *data* volumes are fused together to produce the final reconstructed surface.

Finally, color texture is applied to the reconstructed surface. To ensure that the color is also temporally consistent, we propose a texturing approach that blends contributions from color images captured at frames t_{r-1} and t_r. To allow the correct parts of the texture from the respective frames to be blended, the deformation field is used to find the corresponding points in t_{r-1} and t_r.

3.2.2 Texture Processing

The first stage of the pipeline is a texture processing stage that applies cleaning and filtering operations to the incoming depth images captured at the reconstruction frame t_r, as well as creating a number of auxiliary maps that are required in subsequent pipeline stages.

The background is removed from the depth images by excluding depth observations that are unprojected to positions outside of a fixed bounding box. Since depth images are known to exhibit random depth error with approximately Guassian distribution [29], a bilateral filter is used to smooth surfaces in the depth images. Normal maps are generated by estimating per-pixel normal vectors using the central differences method. Per-pixel quality maps are then calculated as shown by Alexiadis et al. [1], with values that are based on the angle between the viewing direction and the surface normal, and the the world-space distance between a pixel and its neighbors.

We introduce two additional texture maps, the *refined silhouette* and *visual hull* textures, that aid fusion of surfaces observed from different capture groups by excluding misaligned surfaces from the warped volume during the volume warping stage.

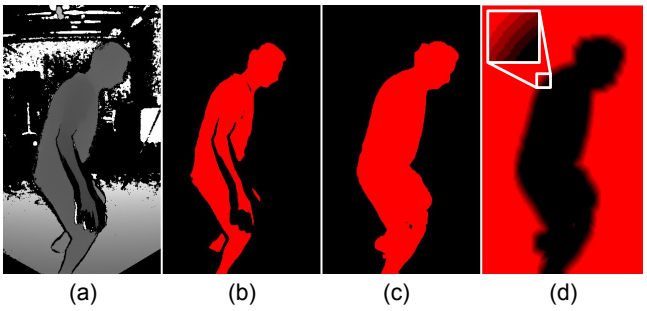

Figure 4: *Refined silhouette* and *visual hull* images are created in the texture processing stage. The raw depth map (a) exhibits holes around the arms. Computing a silhouette directly from the depth image would therefore be incorrect (b). By rendering points created by unprojecting depth observations from the other depth images from the same time frame t_r and applying a closing operation, as described in Section 3.2.2, a refined silhouette without large holes is created (c). A *visual hull* texture is created for a depth image (a) by inverting the filled silhouette (b) and successively applying a kernel that creates borders with increasing value around pixels with a value below 1, leading to a stepped gradient (d, see Section 3.2.2).

A simple binary silhouette texture could be created by classifying pixels with a valid depth observation as part of the silhouette. However, depth images may exhibit holes where surfaces are poorly observed due to motion, viewing angle, or material properties. If those holes propagate to the silhouette texture, the alignment check may incorrectly treat some parts of the warped surface as misaligned, meaning that those surfaces will not be fused into the final result, and that the resulting avatar stream will be less temporally consistent. To avoid this issue, a *refined silhouette* is created. Firstly, an initial base silhouette is created by classifying pixels with valid depth as part of the silhouette. A point cloud is then generated from the other depth images captured at the same frame, by unprojecting depth observations into world space. This point cloud is subsequently rendered on top of the base silhouette, with pixels that the point cloud is projected to becoming part of the silhouette. Finally, morphological closing is applied to the silhouette, with a dilation kernel of size 5×5 pixels, and an erosion kernel of size 3×3. This process has the effect

of closing erroneous holes in the silhouette, as shown in Figure 4. The asymmetry between the dilation and erosion kernel sizes means that the silhouette is expanded. In the volume warping and fusion stages, the expanded silhouette allows TSDF voxels outside the surface to be fused into the final result (Section 3.2.5).

The refined silhouette texture addresses the holes in depth images that can be observed when employing commodity RGBD cameras. Depth cameras may have relatively long exposure times, meaning that depth in areas of movement cannot be reliably estimated. Such issues are less likely to occur when obtaining depth information from custom stereo camera pairs [11] or from sensors with a higher frame rate [10]. Our approach is therefore a novel contribution in the context of reconstruction with commodity RGBD cameras.

Visual hull textures are required to assess the alignment of the warped implicit surface in the TSDF warping and fusion stages (see Section 3.2.5). In the visual hull texture, pixels inside the silhouette are assigned a value of 0, while pixels far outside the silhouette are assigned a value of 1. Pixels close to the silhouette receive a value in range [0,1], where pixels further from the silhouette receive a higher value. The visual hull textures are created from the refined silhouette textures. First, an inverted copy of the silhouette is created with pixels inside the silhouette being assigned value 0. Then, the inverted silhouette is expanded by successive applications of a kernel that creates a border around any pixels with value below 1. Each application of the kernel adds a border with higher pixel values than the last, yielding a stepped gradient from 0 to 1 as pixels further from the silhouette are encountered. An example of a visual hull texture is shown in Figure 4.

3.2.3 TSDF Integration

The filtered depth images from the reconstruction frame t_r are integrated into a volume as a TSDF, following the method introduced by Curless and Levoy [9]. Voxels are projected into depth image space of each sensor, enabling mapping of the 3D voxel position $\mathbf{x}$ to 2D depth image coordinates $\Pi(\mathbf{x})$. The depth values at $\Pi(\mathbf{x})$ in each depth map are weighted and combined using the quality values at the corresponding positions. The resulting volume is comprised of voxels carrying value pairs (d, w), where d is the distance from the center of the voxel to the nearest surface, clamped such that $D_{min} < d < D_{max}$, and w is a weight quantifying the quality of depth observations used to calculate the distance d.

To avoid calculating signed distances for voxels in empty space, we calculate the occupancy of the volume in a brick-wise manner. The volume is subdivided into bricks consisting of 10×10×10 voxels. A conservative brick-wise occupancy of the volume is calculated during the texture processing pass by classifying any bricks within a fixed distance of a valid depth observation as occupied. We compute a signed distance only for voxels in occupied bricks.

3.2.4 Deformation Field Estimation

This component estimates a deformation field that corresponds to the motion that occurred between the frame t_{r-1} and the frame t_r. A correct deformation field enables non-rigid deformation, or *warping*, of the implicit surface encoded in the key volume to align with that encoded by the data volume, meaning that surfaces from the key volume can be fused into the data volume.

We note that there are many deformation field estimation approaches described in the related literature, and that the challenge of detecting a dense deformation field in real time is still not completely solved. We also note that our proposed reconstruction pipeline is not dependent on any specific deformation field estimation approach. Any implementation that can estimate deformation at a frame rate of at least 60 Hz would be suitable for use in our pipeline. As the aim of this work is not to develop an improved deformation estimation method, we implement deformation field estimation based on the Embedded Deformation (ED) paradigm [43] that underpins motion estimation in two influential recent works: Fusion4D [11] and Motion2Fusion [10]. We summarize our implementation of the ED paradigm in the following paragraph.

The ED paradigm parametrizes the deformation field by defining rotation $\mathbf{R} \in \mathbb{R}^{3\times3}$ and translation $\mathbf{t} \in \mathbb{R}^3$ at a sparse set of *ED nodes* on the deforming surface. ED node positions are obtained from the key volume with a variant of the Marching Cubes algorithm [31] that produces one vertex from each volume cell that contains an iso-surface. The volume resolution is adjusted to obtain approximately $2k$ ED nodes for one volumetric avatar. The ED paradigm estimates deformation by defining and evaluating energy terms. Regularization terms penalize undesirable types of deformation, and data terms penalize deformation that does not align the input surface with the target surface. The sum of the energy terms serves as a cost function when optimizing for rotation and translation parameters. Following recent implementations of Sumner et al.'s framework [11, 12], we employ two regularization terms, evaluated at each ED node: one to encourage $\mathbf{R}$ to be a rotation matrix instead of an affine transformation, and one to encourage neighboring ED nodes to have similar transformations. We also use a point-to-plane *projective correspondence* data term that penalizes misalignments between the deformed surface and the depth images observed at t_r [11, 24].

Initialization of ED node Translations. A key challenge when estimating the deformation field is tracking large inter-frame motions caused by fast movement. Other approaches make use of strong point-to-point correspondences, such as robust matches between keypoints in color images [11, 24], to provide an initial estimate of each ED node's transformation parameters, which are subsequently refined. Our approach differs, in that it leverages the body tracking functionality provided by the Azure Kinect Body Tracking SDK (BTSDK)[5], which provides reliable tracking of fast movements and has been used to animate rigged avatars in previous works [16, 50]. We obtain a body pose from one camera, and calculate the change in rotation and translation since frame t_{r-1} for each joint. We initialize ED node translations from joint motions by mapping ED nodes to their closest joint J_c in frame t_{r-1}. The inter-frame rotation of J_c around its parent J_p is combined with the inter-frame translation of J_p to calculate the ED node's initial translation.

The BTSDK runs on the capture server, which transmits pose data to the reconstruction server along with the color and depth images. Since the BTSDK requires ∼30 ms to produce a pose estimation for one image, we calculate a pose using one camera from group *A*, and derive poses for group *B*'s frames through linear interpolation.

We note that the use of the BTSDK restricts our deformation field estimation implementation in some respects. The implementation is limited to RGBD cameras that support pose estimation, and the interpolation approach means that ED node initialization may be incorrect for frames captured by group *B* if the movement direction changes suddenly. Furthermore, ED nodes belonging to non-avatar surfaces may not be correctly initialized, although small deformations can be recovered by the optimization process. While other approaches to ED node initialization [10, 11] are likely to be more effective, the use of the BTSDK suffices to demonstrate the spatio-temporally offset capture approach.

Non-Linear Optimization. Recent reconstruction approaches have shown that the energy minimization problem formed in the ED paradigm can be solved iteratively using the Levenberg-Marquardt (LM) algorithm [10–12]. We follow this approach, allowing up to 5 iterations of the LM solver per frame. The linear solve to determine the parameter step within each LM iteration is executed using a Preconditioned Conjugate Gradient (PCG) solver, which runs for 10 iterations per LM iteration.

[5] docs.microsoft.com/azure/kinect-dk/body-sdk-setup

3.2.5 TSDF Warping and Fusion

Given a deformation field that approximates the motion between frames t_r and t_{r-1}, the surface implicitly encoded in the key volume can be warped to align with the data volume. The result of the warping stage is a TSDF volume referred to as the *warped volume*. We perform TSDF volume warping and fusion in a similar manner to Dou et. al [11], and give a summary of the similar parts of the warping and fusion stages in the following paragraph.

For each voxel in the key volume with signed distance $|d| < D_{max}$ and position $\mathbf{x}$, a warped position $\tilde{\mathbf{x}}$ is calculated by taking a weighted average of transformations from the nearest ED nodes. Each warped voxel is added to the warped volume by integrating signed distance values into the eight closest voxels to $\tilde{\mathbf{x}}$. The distance integrated into each voxel is not d, but is a corrected distance based on the voxel's gradient, to account for cases when $\tilde{\mathbf{x}}$ does not lie in the center of a voxel. Warped voxels are checked for misalignment before being integrated into the warped volume. A misalignment error is calculated for each warped voxel, by determining how far $\tilde{\mathbf{x}}$ lies outside the visual hull of the surface observed at frame t_r, and only those with error below a fixed threshold are integrated. After the warped stage, voxels from the warped volume with $|d| < D_{max}$ are fused into the data volume by calculating a weighted average of d from corresponding voxels in each volume.

Our misalignment error calculation approach differs from the reference [11]. In order to create temporally consistent reconstructions, any holes in the surface reconstructed from one capture group must be filled by the surface from the other. Holes can only be filled when voxels on either side of the surface are integrated into the volume, creating a zero-crossing. Therefore, some voxels that lie just outside of the surface must be fused into the volume, which we allow by using dilated silhouette and visual hull textures. Voxels which may otherwise have been given a high misalignment error and excluded from the data volume instead receive a low error if they are just outside the surface. While the referenced warping implementation [11] calculates misalignment from a 3D visual hull texture, we use an approximation of the visual hull calculated by expanding a refined silhouette texture, described in Section 3.2.2.

The key volume is overwritten by the contents of the data volume, such that the reconstructed result of frame t_r serves as the key volume for frame t_{r+1}. While it is common for volumetric fusion methods to maintain a canonical volume that is incrementally completed [10, 11, 24, 36], thereby building a representation of the observed geometry over many frames, we find that refreshing the key volume at each frame is sufficient to avoid most flickering artefacts, and means that one volume warping operation can be omitted, and surface topology changes are immediately reflected in the key volume.

Finally, geometry can be extracted from the TSDF volume, either by employing the Marching Cubes algorithm [31] to extract an explicit triangle mesh representation, or by using a ray-casting approach to visualize the zero level-set surface.

3.3 Texturing

To apply temporally consistent textures to the reconstructed volumetric avatar geometry, we blend color contributions from cameras in both capture groups. Without careful lighting or color calibration of each RGBD camera, images from different cameras are likely to exhibit noticeable visual differences. Reconstruction methods that combine textures from multiple cameras therefore use spatial blending approaches to achieve smooth transitions between parts of the surface that are textured from different color cameras [13, 28, 38]. In our approach, texturing alternating frames from different camera groups is likely to introduce distracting high-frequency temporal color changes, even if spatial blending is implemented. Furthermore, small inaccuracies in the intrinsic and extrinsic parameters of each camera may lead to incorrect mapping of textures to surfaces, which could also induce high-frequency color changes.

To solve this problem, we introduce a temporal extension to the spatial blending approach introduced in the *Holoportation* system [38] that blends colors from images captured at the reconstruction frame t_r with colors captured at the previous frame t_{r-1}. Since inter-frame movement may have occurred between t_r and t_{r-1}, it is not sufficient to transform the position $\mathbf{p}_r$ of the fragment to be textured into the texture space of cameras from both capture groups. To obtain valid color information from frame t_{r-1}, the position $\mathbf{p}_{r-1}$ of the fragment at frame t_{r-1} must be obtained. We obtain $\mathbf{p}_{r-1}$ with the help of an inverse warping volume calculated during the warping stage. Given $\mathbf{p}_{r-1}$, the texture space positions of the fragment in frame t_{r-1} can be calculated, and the color can be retrieved and blended with color contributions from frame t_r. The blending operation is based on the angle difference between the surface normal and the camera viewpoint directions, and a visibility test to check whether $\mathbf{p}_r$ is visible from each camera to prevent ghosting artefacts.

4 Evaluation

We evaluate our approach with eight Azure Kinect cameras, positioned as shown in Figure 2, and configured according to the synchronization parameters stated in Section 3.1.2. Each camera captures and transmits JPEG color images with a resolution of 1280×720 pixels and depth images with a resolution of 640×576 pixels. Although the capture and reconstruction servers can run simultaneously on the same machine, we record the captured sequences to enable comparison with simulated capture configurations, and run the reconstruction pipeline on a second machine in our evaluation. The capture server, to which 8 Azure Kinect cameras were connected, has an Intel Xeon(R) CPU E5-2687W v4 processor at 3.00 GHz, 377 GB of RAM and an NVIDIA Quadro RTX 6000 graphics card. The volumetric avatar reconstruction pipeline was run on a PC that was equipped with an Intel Core i9-9900X CPU processor at 3.50GHz, 125 GB of RAM and an NVIDIA GeForce RTX 3090 graphics card.

Implementation details. Deformation estimation was implemented using CUDA, while the other reconstruction stages were implemented with the OpenGL graphics API. The sparse block matrix $J^T J$ and the vector $J^T f$, required to estimate the parameters of the deformation field, are evaluated directly to avoid evaluating the entire Jacobian matrix, as explained by Zollhöfer et al. [53]. The PCG solver, also necessary for deformation field estimation, is adapted from public code [17] and is implemented with cuBLAS and custom sparse matrix multiplication kernels. Our implementation allows the reconstruction pipeline to run at 60 Hz, which is the effective capture frame rate when our proposed capture configuration is used. We report the run-time performance of the main processing stages of our reconstruction pipeline in Table 2, as well as the total reconstruction time per frame, which includes some additional time for miscellaneous CPU processes. We note that pre-reconstruction processes, namely receiving, decompressing, and uploading image data to the GPU, are performed asynchronously. These processes, which are necessary for any reconstruction approach, do not contribute to the reconstruction time, but do introduce latency corresponding to two frames (32.2 ms). Pose estimation (Section 3.2.4) and asynchronous transmission of image data by the capture server (Section 3.1.3) also introduce 30 ms of latency between retrieving image data and transmitting to the reconstruction server. The reconstruction pipeline incurs a total latency of 77 ms plus network transmission time.

Capture volume. Our capture volume spans 2 meters along each cardinal axis, containing cubical voxels with side length of 1 cm. To avoid performing TSDF integration, warping, and fusing operations on all voxels in the capture volume, we decompose the volume into bricks of 10^3 voxels, and maintain brick-level occupancy information for the key and data volumes.

Evaluation sequences. Our pipeline was evaluated using three sequences, each capturing different types of human movement. The

Stage	Average time
Texture Processing	4.1 ms
TSDF Integration	0.2 ms
Deformation Field Estimation	3.6 ms
Volume Warping	2.5 ms
Volume Fusing	0.9 ms
Visualization	1.0 ms
Total	**14.5 ms**

Table 2: Mean processing times for each reconstruction stage.

sequences were recorded as color and depth image streams, and have a duration of 10 seconds. In the *Wave* dataset, the subject is static apart from waving one arm. In the *Sway* dataset, the subject stands on one foot and sways from side to side, therefore inducing a slow full-body movement. In the *Move* dataset, the subject moves around more quickly inside the capture space. Avatar reconstructions obtained from the recorded sequences are shown in Figure 5.

Figure 5: Reconstructed volumetric avatars from the evaluation sequences: *Wave* (top left), *Sway* (bottom), and *Move* (top right).

4.1 Comparison Configurations

To demonstrate that our proposed capture configuration and reconstruction pipeline is beneficial for reconstruction of volumetric avatars with synchronizable commodity RGBD cameras, we compare volumetric avatars reconstructed with our pipeline against avatars reconstructed with images from alternative capture configurations that are possible with the same number of RGBD cameras. We refer to our capture configuration, detailed in Section 3, as the *Offset Capture Groups* (OCG) configuration. The capture configurations that we compare against are described in the following paragraphs.

Spatially Aligned Capture Groups (SACG). Instead of distributing all cameras evenly around the capture volume as shown in Figure 2, each camera from group *B* could be spatially aligned with a camera from group *A*, as shown in Figure 6 (left), such that groups *A* and *B* provide very similar sets of views. By temporally offsetting the center of exposure of cameras in *B*, as in the OCG approach, the effective reconstruction frame rate can be doubled. The SACG configuration would be expected to provide temporally consistent results, as view-dependent artifacts would remain the same in consecutive frames. We aim to show that by offsetting the capture groups spatially, as well as temporally, and employing our reconstruction pipeline, we increase scene coverage and achieve improved reconstruction accuracy compared to the SACG configuration, while maintaining a comparable temporal consistency.

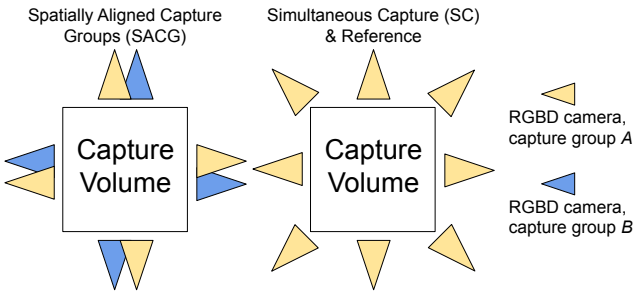

Figure 6: Comparison configurations. The SACG configuration features two groups of cameras that are spatially aligned, but temporally offset. The SC and *reference* configurations consist of eight cameras that are spatially distributed, but temporally aligned.

Reference. To assess the reconstruction accuracy and temporal consistency of the OCG and SACG configurations, we compare them against a ground truth reconstruction, created from color and depth images from eight RGBD cameras in each reconstruction frame.

Simultaneous Capture (SC). One further alternative capture configuration is the SC configuration, where eight spatially distributed cameras capture images simultaneously, as shown in Figure 6 (right). This configuration does not allow the frame rate of the avatar reconstruction to be increased, but should produce a temporally consistent volumetric avatar, since all frames are reconstructed from the same set of cameras. The reconstructed surfaces from the SC configuration are identical to those from the reference configuration, with half the frame rate. Therefore, the SC configuration is not included in the evaluation, since the evaluation methods used are based on comparisons with geometry created with the reference configuration. However, we provide a visual comparison with our method in the video figure.

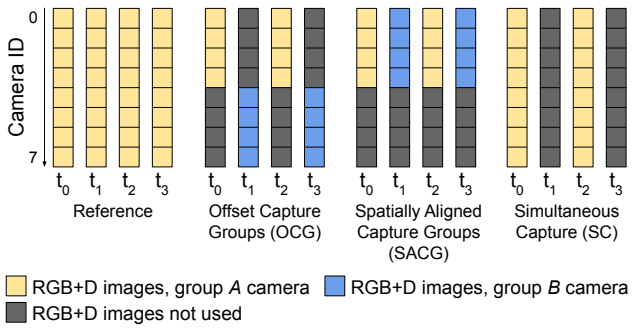

Figure 7: To create evaluation sequences that can be reconstructed as if captured with our proposed configuration or any of the comparison configurations, a sequence is recorded by eight cameras capturing simultaneously at 30 FPS. Images from some cameras can be removed in each frame to simulate all of the required configurations.

Creating evaluation sequences. To compare the OCG and SACG configurations against the reference configuration at 60 Hz, 16 cameras are required: eight without a temporal offset to compare with reconstructions from capture group *A*, and eight with a temporal offset to compare with reconstructions from capture group *B*. Since simultaneous capture from 16 cameras is not possible with our current hardware setup, we instead capture a reference stream with eight cameras at 30 Hz. OCG and SACG configurations are then simulated by removing images from each frame, as illustrated in Figure 7. To simulate our OCG approach, images from alternating groups are removed from each frame. To simulate the SACG

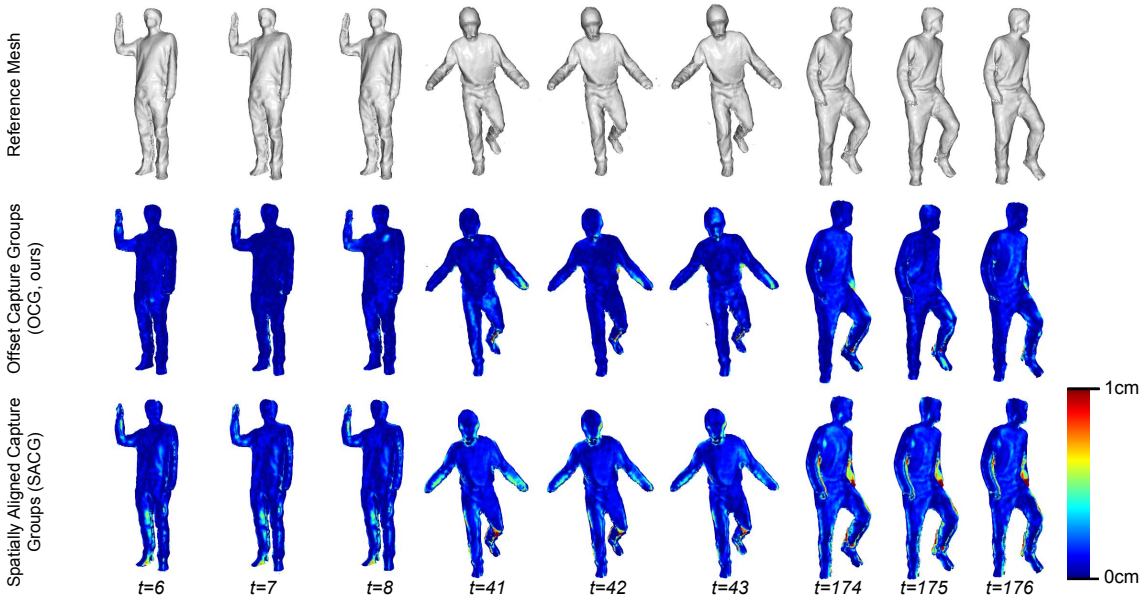

Figure 8: Volumetric avatar meshes reconstructed from our proposed OCG configuration (middle row) and the comparison SACG configuration (bottom row) are compared to meshes reconstructed with the reference configuration (top row). Per-vertex distances to the reference mesh are visualized, with red areas corresponding to a distance of 1 cm. The reconstructions from the OCG configuration exhibit less error overall, and fewer areas with large errors, because the spatially offset groups mean that surfaces are reconstructed from more perspectives.

configuration, images from cameras in group B are removed from each frame. To simulate the SC configuration, alternating frames are removed from the stream entirely. The *reference* configuration simply uses all eight simultaneously captured images to reconstruct each frame. Evaluating with a 30 Hz reference means that the OCG and SACG configurations increase reconstruction frame rate from 15 Hz to 30 Hz, instead of 30 Hz to 60 Hz. Evaluating at half the potential frame rate poses a greater challenge for the reconstruction pipeline, since inter-frame motions are likely to be larger, potentially leading to a less precise estimation of the deformation field and thus larger errors than for the real use case of 30 Hz to 60 Hz.

4.2 Qualitative Comparisons

In this section, we present visual impressions of the reconstruction quality achieved by our approach. We also refer the reader to our video figure in the supplemental material to observe the effect of increasing the reconstruction frame rate.

In Figure 8, volumetric avatar geometry reconstructed with the OCG (middle), SACG (bottom), and reference (top) configurations is presented. Surfaces were extracted as triangle meshes using the Marching Cubes algorithm [31]. To assess the reconstruction quality, meshes extracted from the OCG and SACG configurations are compared against meshes extracted with the reference configuration. Per-vertex distance from the reference mesh is calculated, mapped to the displayed colour scale, and rendered as vertex colors, with red areas indicating a reconstruction error of 1 cm or more. It can be seen that meshes reconstructed from the OCG configuration have lower overall error, as well as fewer areas with large errors. This shows that incorporating depth images from eight different perspectives into each surface, enabled by the temporal volumetric fusion approach used in our proposed reconstruction pipeline, leads to more accurate surface reconstruction, even when half of the views are captured in a temporally offset manner.

4.3 Quantitative Comparisons

In this section, we evaluate the temporal consistency of avatar streams reconstructed with our proposed pipeline and compare against an alternative capture configuration. We also assess the reconstruction error of reconstructed avatars against avatars reconstructed with a reference configuration.

4.3.1 Temporal Consistency

Our proposed volumetric avatar reconstruction approach, with spatio-temporally offset camera groups, could be prone to exhibiting temporally inconsistent surfaces and flickering artefacts due to view-dependent artefacts changing at each frame. To evaluate the temporal consistency of the volumetric avatar streams reconstructed by our system, we use the Temporal Consistency Metric (TCM) proposed by Yao et al. [47]. In its original context of evaluating video processing techniques, the metric compares the temporal consistency of a processed video stream V with the temporal consistency of the raw input video stream O. We adopt the measure to evaluate the temporal consistency of reconstructed avatar streams from the OCG and SACG configurations (V) with respect to streams reconstructed with the *reference* configuration (O). The TCM computes a temporal consistency value for V in the range [0,1], where larger values indicate more temporal consistency, and a score of 1 means that V is as temporally consistent as O.

To calculate a TCM value for an evaluation sequence, we render volumetric avatar streams using TSDF ray-marching at a resolution of 1024x1024 pixels. We compute mean TCM across all pairs of consecutive frames. Differences between the images are calculated by first converting the color images to grayscale images according to ITU recommendation ITU-R BT.601-7 [25], and taking the sum of the squared per-pixel differences.

The results of the temporal consistency evaluation are shown in Figure 9. The OCG configuration receives TCM values of 0.50-0.72, while the SACG configuration has TCM values of 0.53-0.83. For the *Wave* dataset, the OCG configuration is clearly less temporally consistent, but for the *Sway* and *Move* datasets, the OCG configuration is only slightly less temporally consistent than the SACG configuration (although differences were statistically significant for all sequences, when tested with the Mann-Whitney U Test). The SACG configuration should achieve a TCM approaching 1, that

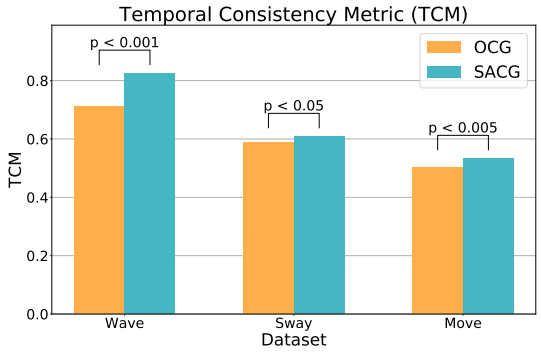

Figure 9: Temporal consistency metric (TCM) for color images reconstructed with OCG (ours) and SACG capture configurations.

is, be nearly as temporally consistent as the reference, because it should not suffer from alternating view-dependent artefacts, due to the fact that all reconstructed frames are captured from the same set of perspectives. The discrepancy between the expectation and the observed values could be explained by temporal sensor noise that commodity RGBD sensors are known to suffer from. The reference is less likely to suffer from temporal noise because more cameras are fused to create the surface, averaging out sensor noise. For two of our three evaluation datasets, it can be seen that the volumetric avatar streams produced by our pipeline are nearly as temporally coherent as those produced from the SACG configuration.

4.3.2 Reconstruction Error

In Figure 10, a quantitative evaluation of the reconstruction error is presented for each of the evaluation sequences. Reconstructions from the reference, OCG and SACG configurations are rendered from multiple viewpoints using TSDF ray-marching to create depth images. Per-pixel depth difference from the corresponding reference images is calculated for images created from the offset and spatially aligned configurations. A mean per-pixel depth difference per frame is calculated, excluding per-pixel differences greater than 30 mm, as large differences are likely to result from comparison of different surfaces rather than reconstruction error. Our analysis shows that geometry reconstructed with our proposed capture technique exhibits less reconstruction error than the SACG configuration.

5 Conclusion and Future Work

In this work, we introduced a novel technique for capturing and reconstructing volumetric avatars with synchronizable commodity RGBD cameras that doubles the capture and reconstruction frame rate of a set of available cameras from 30 Hz to 60 Hz. This is achieved by employing a capture configuration comprised of two spatio-temporally offset capture groups that provide images to a volumetric avatar reconstruction server in an alternating manner. Since the spatio-temporally offset capture groups provide images captured from different sets of perspectives at alternating frames, potentially leading to unpleasant flickering artefacts, we propose an accompanying reconstruction pipeline based on previous work on volumetric temporal fusion that is capable of fusing surfaces computed in previous reconstruction frames into the current reconstruction frame, thereby producing temporally consistent volumetric avatars streams. Our evaluation, in which we compare our proposed configuration against capture configurations possible with the same number of RGBD cameras, reveals that avatar streams reconstructed with our proposed pipeline exhibit fewer surface-to-surface differences to a reference reconstruction stream than a capture configuration comprised of temporally offset, but not spatially offset capture groups, while exhibiting a comparable level of temporal consistency.

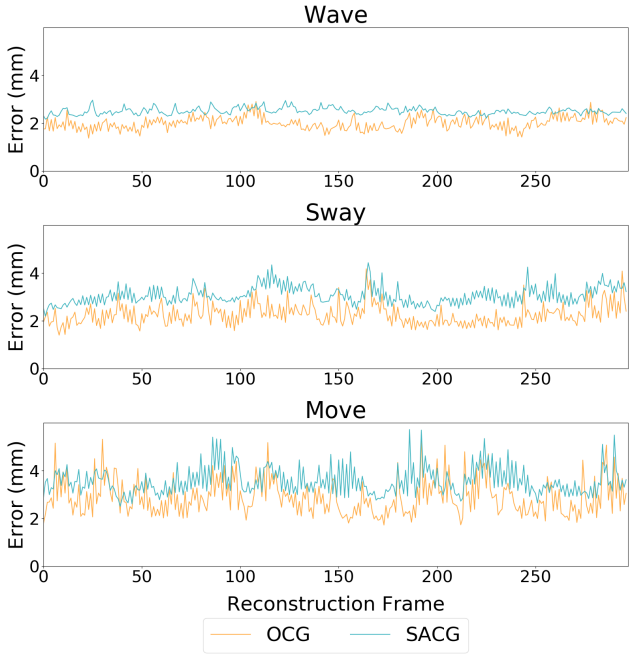

Figure 10: Reconstruction error per-frame for evaluation sequences.

Since our temporal volumetric fusion implementation relies on body tracking to initialize the deformation field, the pipeline is less generalizable to arbitrary scenes. This could be addressed by incorporating a method for detecting strong point-to-point correspondences between frames, as shown in previous works [10, 11].

Our proposed configuration enables volumetric avatars captured with Azure Kinect cameras to be reconstructed at 60 Hz, twice the maximum frame rate of a single camera, which creates generally smoother motions of avatars in virtual reality applications. However, the run-time performance of the avatar reconstruction pipeline is currently limited to 60 Hz. Reconstructing at the refresh rate of the display, typically 72 or 90Hz on modern HMDs, would eliminate motion judder. To this end, our approach could be extended to more capture groups, further increasing the reconstruction frame rate. Acceleration of the reconstruction pipeline would be necessary, which may be afforded by an increase in processing power in future GPUs. Extension to more capture groups requires investigation of the appropriate number and position of cameras in each group.

Acknowledgments

This work is funded by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) under the project ID 444532506, SPP2236 - AUDICTIVE - Auditory Cognition in Interactive Virtual Environments, the German Federal Ministry of Education and Research (BMBF) under grant 16SV8716 (project Goethe-Live-3D) and 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 group at Bauhaus-Universität Weimar for their support.

References

[1] 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. doi: 10.1109/IVMSPW.2013.6611939

[2] S. Aseeri and V. Interrante. The influence of avatar representation on interpersonal communication in virtual social environments. *IEEE Transactions on Visualization and Computer Graphics*, 27(5):2608–2617, 2021. doi: 10.1109/TVCG.2021.3067783

[3] S. Beck and B. Froehlich. Volumetric calibration and registration of multiple rgbd-sensors into a joint coordinate system. In *2015 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 89–96, 2015. doi: 10.1109/3DUI.2015.7131731

[4] S. Beck, A. Kunert, A. Kulik, and B. Froehlich. Immersive group-to-group telepresence. *IEEE Transactions on Visualization and Computer Graphics*, 19(4):616–625, 2013. doi: 10.1109/TVCG.2013.33

[5] F. Bogo, M. J. Black, M. Loper, and J. Romero. Detailed Full-Body Reconstructions of Moving People from Monocular RGB-D Sequences. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2300–2308. IEEE, dec 2015. doi: 10.1109/ICCV.2015.265

[6] S. Cho, S.-w. Kim, J. Lee, J. Ahn, and J. Han. 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. doi: 10.1109/VR46266.2020.00020

[7] 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. In *ACM Transactions on Graphics*, 2015. doi: 10.1145/2766945

[8] E. Cuervo, K. Chintalapudi, and M. Kotaru. Creating the perfect illusion: What will it take to create life-like virtual reality headsets? In *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*, HotMobile '18, p. 7–12. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3177102.3177115

[9] 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, SIGGRAPH 1996*, 1996. doi: 10.1145/237170.237269

[10] M. Dou, P. Davidson, S. R. Fanello, S. Khamis, A. Kowdle, C. Rhemann, V. Tankovich, and S. Izadi. Motion2Fusion: Real-time volumetric performance capture. In *ACM Transactions on Graphics*, 2017. doi: 10.1145/3130800.3130801

[11] 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 Transactions on Graphics*, 35(4):1–13, jul 2016. doi: 10.1145/2897824.2925969

[12] M. Dou, J. Taylor, H. Fuchs, A. Fitzgibbon, and S. Izadi. 3d scanning deformable objects with a single rgbd sensor. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 493–501, 2015. doi: 10.1109/CVPR.2015.7298647

[13] R. Du, M. Chuang, W. Chang, H. Hoppe, and A. Varshney. Montage4d: Interactive seamless fusion of multiview video textures. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '18. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3190834.3190843

[14] S. R. Fanello, J. Valentin, C. Rhemann, A. Kowdle, V. Tankovich, P. Davidson, and S. Izadi. Ultrastereo: Efficient learning-based matching for active stereo systems. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6535–6544, 2017. doi: 10.1109/CVPR.2017.692

[15] W. Gao and R. Tedrake. Surfelwarp: Efficient non-volumetric single view dynamic reconstruction. *ArXiv*, abs/1904.13073, 2018.

[16] M. Gonzalez-Franco, Z. Egan, M. Peachey, A. Antley, T. Randhavane, P. Panda, Y. Zhang, C. Y. Wang, D. F. Reilly, T. C. Peck, A. S. Won, A. Steed, and E. Ofek. Movebox: Democratizing mocap for the microsoft rocketbox avatar library. In *2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pp. 91–98, 2020. doi: 10.1109/AIVR50618.2020.00026

[17] A. Guldemond. `https://github.com/alexguldemond/ssorai_pcg`, June 2019.

[18] K. Guo, P. Lincoln, P. Davidson, J. Busch, X. Yu, M. Whalen, G. Harvey, S. Orts-Escolano, R. Pandey, J. Dourgarian, D. Tang, A. Tkach, A. Kowdle, E. Cooper, M. Dou, S. Fanello, G. Fyffe, C. Rhemann, J. Taylor, P. Debevec, and S. Izadi. The relightables: Volumetric performance capture of humans with realistic relighting. *ACM Transactions on Graphics*, 38(6), nov 2019. doi: 10.1145/3355089.3356571

[19] K. Guo, F. Xu, Y. Wang, Y. Liu, and Q. Dai. Robust non-rigid motion tracking and surface reconstruction using l0 regularization. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 3083–3091, 2015. doi: 10.1109/ICCV.2015.353

[20] K. Guo, F. Xu, T. Yu, X. Liu, Q. Dai, and Y. Liu. Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera. *ACM Trans. Graph.*, 36(4), jun 2017. doi: 10.1145/3072959.3083722

[21] M. Habermann, W. Xu, M. Zollhoefer, G. Pons-Moll, and C. Theobalt. Deepcap: Monocular human performance capture using weak supervision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2020.

[22] M. Habermann, W. Xu, M. Zollhöfer, G. Pons-Moll, and C. Theobalt. LiveCap: Real-time human performance capture from monocular video. *ACM Transactions on Graphics*, 38(2), 2019. doi: 10.1145/3311970

[23] P. Heidicker, E. Langbehn, and F. Steinicke. Influence of avatar appearance on presence in social vr. *2017 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 233–234, 2017.

[24] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In B. Leibe, J. Matas, N. Sebe, and M. Welling, eds., *Computer Vision – ECCV 2016*, pp. 362–379. Springer International Publishing, Cham, 2016.

[25] ITU. Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios. Recommendation BT.601-6, International Telecommunication Union, Geneva, March 2011.

[26] D. Jo, K.-H. Kim, and G. J. Kim. Effects of avatar and background representation forms to co-presence in mixed reality (mr) tele-conference systems. In *SIGGRAPH ASIA 2016 Virtual Reality Meets Physical Reality: Modelling and Simulating Virtual Humans and Environments*, SA '16. Association for Computing Machinery, New York, NY, USA, 2016. doi: 10.1145/2992138.2992146

[27] C. Kozlov, M. Slavcheva, and S. Ilic. Patch-based non-rigid 3d reconstruction from a single depth stream. In *2018 International Conference on 3D Vision (3DV)*, pp. 42–51, 2018. doi: 10.1109/3DV.2018.00016

[28] A. Kreskowski, S. Beck, and B. Froehlich. Output-sensitive avatar representations for immersive telepresence. *IEEE Transactions on Visualization and Computer Graphics*, 28(7):2697–2709, 2022. doi: 10.1109/TVCG.2020.3037360

[29] G. Kurillo, E. Hemingway, M. L. Cheng, and L. Cheng. Evaluating the accuracy of the azure kinect and kinect v2. *Sensors*, 22, 4 2022. doi: 10.3390/s22072469

[30] C. Liu, A. Wang, C. Bu, W. Wang, and H. Sun. Human motion tracking with less constraint of initial posture from a single rgb-d sensor. *Sensors*, 21(9), may 2021. doi: 10.3390/s21093029

[31] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987*, pp. 163–169, 1987. doi: 10.1145/37401.37422

[32] A. Maimone, X. Yang, N. Dierk, A. State, M. Dou, and H. Fuchs. General-purpose telepresence with head-worn optical see-through displays and projector-based lighting. In *2013 IEEE Virtual Reality (VR)*, pp. 23–26, 2013. doi: 10.1109/VR.2013.6549352

[33] S. Meerits, V. Nozick, and H. Saito. Real-time scene reconstruction and triangle mesh generation using multiple RGB-D cameras. *Journal of Real-Time Image Processing*, 16(6):2247–2259, 2019. doi: 10.1007/s11554-017-0736-x

[34] S. Meerits, D. Thomas, V. Nozick, and H. Saito. FusionMLS: Highly dynamic 3D reconstruction with consumer-grade RGB-D cameras. *Computational Visual Media*, 4(4):287–303, dec 2018. doi: 10.1007/s41095-018-0121-0

[35] W. Morgenstern, A. Hilsmann, and P. Eisert. Progressive non-rigid registration of temporal mesh sequences. *Proceedings - CVMP 2019: 16th ACM SIGGRAPH European Conference on Visual Media Production*, 2019. doi: 10.1145/3359998.3369411

[36] R. A. Newcombe, D. Fox, and S. M. Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:343–352, 2015. doi: 10.1109/CVPR.2015.7298631

[37] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th*

IEEE International Symposium on Mixed and Augmented Reality, pp. 127–136, 2011. doi: 10.1109/ISMAR.2011.6092378

[38] 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*, UIST '16, p. 741–754. Association for Computing Machinery, New York, NY, USA, 2016. doi: 10.1145/2984511.2984517

[39] F. Prada, M. Kazhdan, M. Chuang, A. Collet, and H. Hoppe. Spatiotemporal atlas parameterization for evolving meshes. *ACM Trans. Graph.*, 36(4), jul 2017. doi: 10.1145/3072959.3073679

[40] D. Roth, J.-L. Lugrin, D. Galakhov, A. Hofmann, G. Bente, M. E. Latoschik, and A. Fuhrmann. Avatar realism and social interaction quality in virtual reality. In *2016 IEEE Virtual Reality (VR)*, pp. 277–278, 2016. doi: 10.1109/VR.2016.7504761

[41] F. A. Smit, R. van Liere, and B. Fröhlich. The design and implementation of a vr-architecture for smooth motion. In *Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology*, VRST '07, p. 153–156. Association for Computing Machinery, New York, NY, USA, 2007. doi: 10.1145/1315184.1315212

[42] A. Steed and R. Schroeder. *Collaboration in Immersive and Non-immersive Virtual Environments*, pp. 263–282. Springer International Publishing, Cham, 2015. doi: 10.1007/978-3-319-10190-3_11

[43] R. W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 2007. doi: 10.1145/1275808.1276478

[44] X. Suo, Y. Jiang, P. Lin, Y. Zhang, M. Wu, K. Guo, and L. Xu. Neuralhumanfvv: Real-time neural volumetric human performance rendering using rgb cameras. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6222–6233, 2021. doi: 10.1109/CVPR46437.2021.00616

[45] 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.*, 37(6), dec 2018. doi: 10.1145/3272127.3275096

[46] L. Xu, Z. Su, L. Han, T. Yu, Y. Liu, and L. Fang. UnstructuredFusion: Realtime 4D Geometry and Texture Reconstruction Using Commercial RGBD Cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(10), oct 2020. doi: 10.1109/TPAMI.2019.2915229

[47] C.-H. Yao, C.-Y. Chang, and S.-Y. Chien. Occlusion-aware video temporal consistency. In *Proceedings of the 25th ACM International Conference on Multimedia*, MM '17, p. 777–785. Association for Computing Machinery, New York, NY, USA, 2017. doi: 10.1145/3123266.3123363

[48] M. Ye and R. Yang. Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2353–2360, 2014. doi: 10.1109/CVPR.2014.301

[49] B. Yoon, H.-i. Kim, G. A. Lee, M. Billinghurst, and W. Woo. The effect of avatar appearance on social presence in an augmented reality remote collaboration. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 547–556, 2019. doi: 10.1109/VR.2019.8797719

[50] K. Yu, G. Gorbachev, U. Eck, F. Pankratz, N. Navab, and D. Roth. Avatars for teleconsultation: Effects of avatar embodiment techniques on user perception in 3d asymmetric telepresence. *IEEE Transactions on Visualization and Computer Graphics*, 27(11):4129–4139, 2021. doi: 10.1109/TVCG.2021.3106480

[51] Q. Zhang, B. Fu, M. Ye, and R. Yang. Quality dynamic human body modeling using a single low-cost depth camera. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 676–683, 2014. doi: 10.1109/CVPR.2014.92

[52] Y. Zheng, R. Shao, Y. Zhang, T. Yu, Z. Zheng, Q. Dai, and Y. Liu. Deepmulticap: Performance capture of multiple characters using sparse multiview cameras. In *IEEE Conference on Computer Vision (ICCV 2021)*, 2021.

[53] M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, and M. Stamminger. Real-time non-rigid reconstruction using an RGB-D camera. In *ACM Transactions on Graphics*, 2014. doi: 10.1145/2601097.2601165

[54] X. Zuo, S. Wang, J. Zheng, W. Yu, M. Gong, R. Yang, and L. Cheng. Sparsefusion: Dynamic human avatar modeling from sparse rgbd images. *Trans. Multi.*, 23:1617–1629, jan 2021. doi: 10.1109/TMM.2020.3001506