# Montage4D: Interactive Seamless Fusion of Multiview Video Textures

Ruofei Du*
University of Maryland, College Park

Ming Chuang
PerceptIn Inc.

Wayne Chang
Microsoft Research, Redmond

Hugues Hoppe
Google Inc.

Amitabh Varshney
University of Maryland, College Park

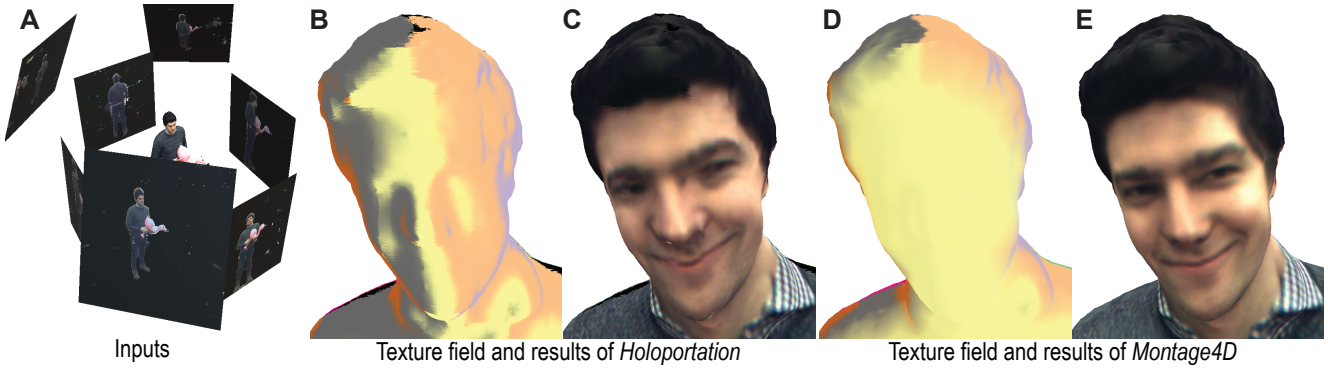


**Figure 1: The Montage4D algorithm stitches multiview video textures onto dynamic meshes seamlessly and at interactive rates. (A) inputs: dynamic triangle meshes reconstructed by the Fusion4D algorithm, multiview video textures, and camera poses; (B) texture field blend weights in Holoportation, based on surface normals, majority voting, and dilated depth discontinuities; (C) resulting Holoportation merged texture; (D) our improved texture fields, which favor the dominant view, ensure temporal consistency, and reduce seams between camera views; (E) the resulting Montage4D merged texture.**

## ABSTRACT

The commoditization of virtual and augmented reality devices and the availability of inexpensive consumer depth cameras have catalyzed a resurgence of interest in spatiotemporal performance capture. Recent systems like *Fusion4D* and *Holoportation* address several crucial problems in the real-time fusion of multiview depth maps into volumetric and deformable representations. Nonetheless, stitching multiview video textures onto dynamic meshes remains challenging due to imprecise geometries, occlusion seams, and critical time constraints. In this paper, we present a practical solution towards real-time seamless texture montage for dynamic multiview reconstruction. We build on the ideas of dilated depth discontinuities and majority voting from *Holoportation* to reduce ghosting effects when blending textures. In contrast to their approach, we determine the appropriate blend of textures per vertex using view-dependent rendering techniques, so as to avert fuzziness caused by the ubiquitous normal-weighted blending. By leveraging geodesics-guided diffusion and temporal texture fields, our algorithm mitigates spatial occlusion seams while preserving temporal consistency. Experiments demonstrate significant enhancement in rendering quality, especially in detailed regions such as faces. We envision a wide range of applications for *Montage4D*, including immersive telepresence for business, training, and live entertainment.

## CCS CONCEPTS

• **Computing methodologies** **Image-based rendering**; *Computational photography*;

## KEYWORDS

texture montage, 3D reconstruction, texture stitching, view-dependent rendering, discrete geodesics, projective texture mapping, differential geometry, temporal texture fields

**ACM Reference Format:**
Ruofei Du, Ming Chuang, Wayne Chang, Hugues Hoppe, and Amitabh Varshney. 2018. Montage4D: Interactive Seamless Fusion of Multiview Video Textures. In *I3D '18: Symposium on Interactive 3D Graphics and Games, May 4–6, 2018, Montreal, QC, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3190834.3190843

*This work is conducted at *Microsoft Research, Redmond*, and *Augmentarium, Department of Computer Science*, and *the Institute for Advanced Computer Studies (UMIACS)* at *University of Maryland, College Park.*


*I3D '18, May 4–6, 2018, Montreal, QC, Canada*

ACM ISBN 978-1-4503-5705-0/18/05…$15.00
https://doi.org/10.1145/3190834.3190843

# 1 INTRODUCTION

With recent advances in consumer-level virtual and augmented reality, several dynamic scene reconstruction systems have emerged, including *KinectFusion* [Izadi et al. 2011], *DynamicFusion* [Newcombe et al. 2015], *Free-Viewpoint Video* [Collet et al. 2015], and *Holoportation* [Orts-Escolano et al. 2016]. Such 4D reconstruction technology is becoming a vital foundation for a diverse set of applications such as 3D telepresence for business, live concert broadcasting, family gatherings, and remote education.

Among these systems, *Holoportation* is the first to achieve real-time, high-fidelity 4D reconstruction without any prior knowledge of the imaged subjects. The success of this system builds upon the breakthrough of fast non-rigid alignment algorithms in fusing multiview depth streams into a volumetric representation by the *Fusion4D* system [Dou et al. 2016]. Although *Holoportation* is able to mitigate a variety of artifacts using techniques such as normal-weighted blending and multilevel majority voting, some artifacts persist. In a previous user study on *Holoportation* [Orts-Escolano et al. 2016], around 30% of the participants did not find that the reconstructed model real compared with a real person. We believe that this is a significant challenge that must be addressed before telepresence can be embraced by the masses. We also note that the user feedback about visual quality was much less positive than other aspects (speed and usability). This is caused by the blurring and visible seams in the rendering results, especially on human faces, as shown in Figure 1, 7, and 8.

*Blurring.* Loss of detail arises because of two reasons. First, texture projection from the camera to the geometry suffers from registration errors, causing visible seams. Second, normal-weighted blending of the different views with different appearance attributes (specular highlights and inconsistent color calibration) leads to an inappropriate mixing of colors and therefore blurring or ghosting.

*Visible Seams.* We further characterize visible seams into: (1) *projection seams* caused by inaccurate estimation of camera parameters , (2) *misregistration seams* caused by imprecise reconstruction of geometry with shrinking/bulging surface patches, and (3) *occlusion seams* arise out of discontinuous texture transitions across the field of view of multiple cameras and self-occlusions. In a static and indoor setting, we suppose the projection matrices are correct, since both the extrinsics and intrinsics of the cameras can be perfectly calibrated.

In this paper, we address both blurring and visible seams and achieve seamless fusion of video textures at interactive rates. Our algorithm estimates the misregistration and occlusion seams based on the self-occlusion from dilated depth discontinuities, multi-level majority voting, foreground segmentation, and the field-of-view of the texture maps. To achieve a smooth transition from one view to another, we compute geodesic distance fields [Bommes and Kobbelt 2007] from the seams, to spatially diffuse the texture fields to the visible seams. In order to prevent view-dependent texture weights from rapidly changing with the viewpoints, we extend the scalar texture field as shown in Figure 1(C) to a temporally changing field to smoothly update the texture weights. As shown in Figure 1(D) and 8, our system achieves significantly higher visual quality at interactive rates compared to the state-of-the-art *Holoportation* system.

In summary, the main contributions of our work are:

- formulation and quantification of the misregistration and occlusion seams for fusing multiview video textures,
- use of equidistance geodesics from the seams based on discrete differential geometry concepts to diffuse texture fields,
- temporal texture fields to achieve temporal consistency of the rendered imagery, and
- a fast computational pipeline for high-fidelity, seamless video-based rendering, enabling effective telepresence.

# 2 RELATED WORK

We build upon a rich literature of prior art on image-based 3D reconstruction, texture stitching, and discrete geodesics.

## 2.1 Image-based 3D Reconstruction

Image-based 3D reconstruction has been researched extensively in the past decades. The pioneering work of Fuchs *et al.* [1994; 1993] envisioned that a patient on the operating table could be acquired by a sea of structured-light cameras, and a remote doctor could conduct medical teleconsultation with a head-mounted display. Kanade *et al.* [1997] invented one of the earliest systems that uses a dome of cameras to generate novel views via triangulated depth maps. Its successor, *3D Dome* [Narayanan et al. 1998], reconstructs explicit surfaces with projected texture. Towles *et al.* [2002] achieve real-time 3D telepresence over networks using 3D point clouds. Goldluecke *et al.* [2004] adopt spatiotemporal level sets for volumetric reconstruction. Furukawa *et al.* [2008] reconstruct deformable meshes by optimizing traces of vertices over time. While compelling, it takes two minutes on a dual Xeon 3.2 GHz workstation to process a single frame. De *et al.* [2008] present a system that reconstructs space-time coherent geometry with motion and textural surface appearance of actors performing complex and rapid moves. However, this also suffers from slow processing speed (approximately 10 minutes per frame), largely due to challenges in stereo matching and optimization. Since then, a number of advances have been made in dealing with video constraints and rendering quality [Cagniart et al. 2010; Casas et al. 2013; Collet et al. 2015; Du et al. 2016; Lok 2001; Patro et al. 2011; Prada et al. 2016, 2017a; Vlasic et al. 2008; Xu et al. 2011], but rendering dynamic scenes in real time from video streams has remained a challenge. Zitnick *et al.* [2004] present an efficient rendering system which interpolates the adjacent two views with a boundary layer and video matting. However, they consider a 2.5D layered representation for the scene geometry rather than a general mesh model that can be viewed from all directions. Their work inspires us with the computation of depth discontinuity and seam diffusion.

With recent advances in consumer-level depth sensors, several reconstruction systems can now generate dynamic point-cloud geometries. *KinectFusion* [Izadi et al. 2011; Newcombe et al. 2011] is the first system that tracks and fuses point clouds into dense meshes using a single depth sensor. However, the initial version of *KinectFusion* can not handle dynamic scenes. The systems developed by Ye *et al.* [2014] and Zhang *et al.* [2014] are able to reconstruct non-rigid motion for articulated objects, such as human bodies and

animals. Further advances by Newcombe *et al.* [2015] and Xu *et al.* [2015] have achieved more robust dynamic 3D reconstruction from a single Kinect sensor by using warp-fields or subspaces for the surface deformation. Both techniques warp a reference volume non-rigidly to each new input frame. Guo *et al.* [2015; 2017] and Yu *et al.* [2017] have realized real-time geometry, albedo, and motion reconstruction using a single RGB-D camera. However, the reconstructed scenes still suffer from the occlusion issues since the data comes from a single depth sensor. In addition, many 3D reconstruction systems rely on a volumetric model that is used for model fitting, which is limited in accommodating fast movement and major shape changes.

Collet *et al.* [2015] have demonstrated the *Free-Viewpoint Video*, an offline pipeline to reconstruct dynamic textured models in a studio setup with 106 cameras. However, it requires controlled lighting, calibration, and approximately 28 minutes per frame for reconstruction, texturing, and compression. Furthermore, Prada *et al.* [2016; 2017a] present a unified framework for evolving the mesh triangles and the spatio-temporal parametric texture atlas. Nonetheless, the average processing time for a single frame is around 80 seconds, which is not yet applicable for real-time applications.

Orts *et al.* [2016] present *Holoportation*, a real-time pipeline to capture dynamic 3D scenes by using multiple RGBD cameras. This system is highly robust to sudden motion and large changes in meshes. To achieve real-time performance, their system blends multi-view textures according to the dot product between surface normals and the camera viewpoint directions.

Our system extends the *Holoportation* system and solves the problems of fuzziness caused by normal-weighted blending, visible seams caused by misregistration and occlusion, while ensuring temporal consistency of the rendered images.

In the state-of-the-art work by Dou *et al.* [2017] with depth maps generated up to 500Hz [2017a; 2017b], a detail layer is computed to capture the high-frequency details and atlas mapping is applied to improve the color fidelity. Our rendering system is compatible with the new fusion pipeline, by integrating the computation of seams, geodesic fields, and view-dependent rendering modules.

## 2.2 Texture Stitching

View-dependent texture-mapping on the GPU has been widely applied for reconstructed 3D models since the seminal work by Debevec *et al.* [1998a; 1998b]. However, seamlessly texturing an object by stitching RGB images remains a challenging problem due to inexact geometry, varying lighting conditions, as well as imprecise calibration matrices.

Previous work has considered using global optimization algorithms to improve color-mapping fidelity in static models. For example, Gal *et al.* [2010] present a multi-label graph-cut optimization approach that assigns compatible textures to adjacent triangles to minimize the seams on the surface. In addition to the source images, their algorithm also searches over a set of local image transformations that compensate for geometric misalignment using a discrete labeling algorithm. While highly creative and elegant, their approach takes 7 to 30 minutes to process one frame on a mesh with 10,000 to 18,000 triangles. Markov Random Field (MRF) optimization-based approaches [Allène et al. 2008; Janko and Pons 2009; Lempitsky and Ivanov 2007] are also similarly time intensive. To reduce the seams caused by different lighting conditions, Zhou *et al.* [2005] introduce *TextureMontage*, which automatically partitions the mesh and the images, driven solely by feature correspondences. *TextureMontage* integrates a surface texture in-painting technique to fill in the remaining charts of the surface with no corresponding texture patches. However, their approach takes over 30 minutes per frame to process. Zhou *et al.* [2014] optimize camera poses in tandem with non-rigid correction functions for all images at the cost of over 30 minutes per frame. Narayan *et al.* [2015] jointly optimize a non-linear least squares objective function over camera poses and a mesh color model at the cost of one to five minutes per frame. They incorporate 2D texture cues, vertex color smoothing, and texture-adaptive camera viewpoint selection into the objective function.

A variety of optical-flow-based approaches have been used to eliminate blurring and ghosting artifacts. For example, Eisemann *et al.* [2008] introduce *Floating Texture*, a view-dependent rendering technique with screen-based optical-flow running at 7-22 frames per second.[1] Casas *et al.* [2014] extend their online alignment with spatio-temporal coherence running at 18 frames per second. Volino *et al.* [2014] employs a surface-based optical flow alignment between views to eliminate blurring and ghosting artifacts. However, the major limitation of optical-flow-based approaches are twofold. First, surface specularity [Eisemann et al. 2008], complex deformations, poor color calibration and low-resolution of the textures [Casas et al. 2014] present challenges in the optical flow estimation. Second, even with GPU computation, the computational overhead of optical flow is still a limitation for real-time rendering. This overhead increases even further with more cameras.

In studio settings, Collet *et al.* [2015] have found that with diffused lighting condition and precisely reconstructed surface geometry, direct image projection followed by normal-weighted blending of non-occluded images yields sufficiently accurate results. However, for real-time reconstruction systems with a limited number of cameras, the reconstructed geometries are often imperfect.

Our work focuses on improving the texture fusion for such real-time applications. Building upon the pioneering research above as well as the work of several others, our approach is able to process over 130,000 triangles at over 100 frames per second.

## 2.3 Geodesic Distance Fields

The field of discrete geodesics has witnessed impressive advances over the last decade [do Goes et al. 2015; Grinspun et al. 2006; Mitchell 2000]. Geodesics on smooth surfaces are the straightest and locally shortest curves and have been widely used in a variety of graphics applications such as optimal movement of an animated subject. Mitchell *et al.* [1987] devise an exact algorithm for computing the *"single source, all destinations"* geodesic paths. For each edge, their algorithm maintains a set of tuples (windows) for the exact distance fields and directions, and updates the windows with a priority queue like the *Dijkstra* algorithm. However, the worst running time could be $O\left(n^2 \log n\right)$, and the average is close to $O\left(n^{1.5}\right)$ [Bommes and Kobbelt 2007; Surazhsky et al. 2005].

[1] We tested *Floating Texture* on a GTX 1080 under target resolution of $1024 \times 1024$ and $2048 \times 2048$.

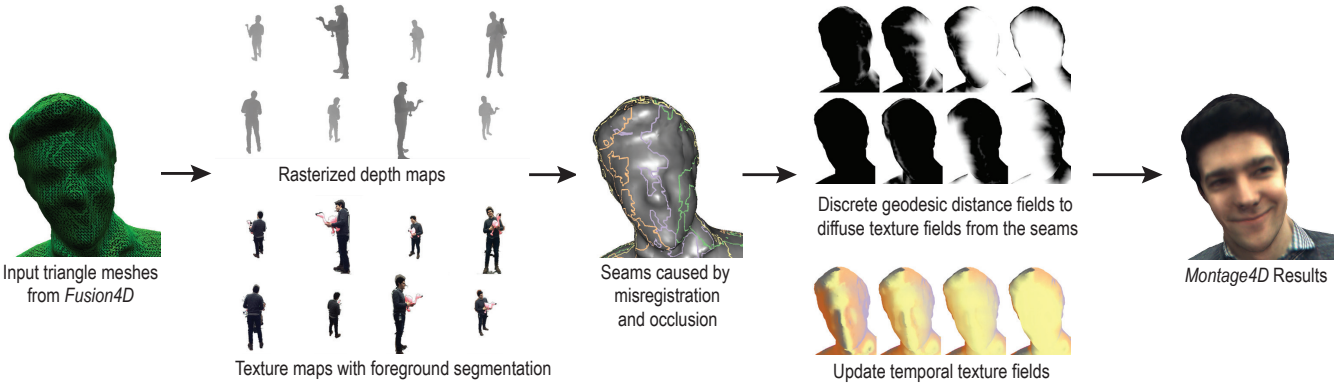


**Figure 2: The workflow of the Montage4D rendering pipeline.**

Recently, Qin *et al.* [2016] proposes a 4-15 times faster algorithm using window pruning strategies. However, their algorithm aims for the exact geodesic paths and requires $O\left(n^2\right)$ space like the previous approaches. Kapoor [1999] proposes a sophisticated approach for the "single source, single destination" case in $O\left(n \log^2 n\right)$ time. As for approximate geodesics, Lanthier [1997] describes an algorithm that adds many extra edges into the mesh. Kanai and Suzuki [2001] and Martinez *et al.* [2004] use iterative optimization to converge the geodesic path locally. However, their methods require a large number of iterations.

In this work, we compute geodesics distance fields for weighting the texture fields, so as to assign low weight near the seams and progressively larger weight up to some maximum distance away from the seams. Our goal is to solve the geodesics problem for the *"multiple sources, all destinations"*. Bommes *et al.* [2007] have introduced an accurate algorithm for computation of geodesic distance fields. In this paper, we follow a variant of the efficient algorithm developed by Surazhsky *et al.* [2005] to measure the approximation of the geodesics fields in $O(n \log n)$ time for a small number of vertices (**seam** vertices are approximately 1% of the total vertices) in a few iterations (typically $15 - 20$).

## 3 SYSTEM OVERVIEW

In this section we present the workflow of the *Montage4D* system as shown in Figure 2:

(1) **Streaming of Meshes and Videos**: Our system streams polygonal meshes and video textures from a reconstruction server that runs the *Fusion4D* pipeline [Dou et al. 2016]. The calibration parameters for projective mapping from camera to model space are only transferred once with the initial frame.
(2) **Rasterized depth maps and segmented texture maps**: For each frame, *Montage4D* estimates rasterized depth maps from each camera's viewpoint and perspective in parallel on the GPU. The video textures are processed with a background subtraction module, using the efficient real-time algorithm performing mean field inference [Vineet et al. 2014].
(3) **Seam identification with dilated depth discontinuities**: The renderer estimates the dilated depth discontinuities from the rasterized depth maps, which are bounded by an estimated reconstruction error $e$. This is crucial for reducing ghosting artifacts, which arise when missing geometry and self-occlusion cause incorrect color projection onto surfaces. The renderer uses the texture maps to calculate the seams due to each camera's limited field of view.
(4) **Geodesic fields**: After the seam identification stage, the renderer calculates the geodesic distance field from the seams to neighboring vertices. This distance field is used to nonlinearly modulate the texture fields, ensuring spatial smoothness of the resulting texture fields.
(5) **Temporal texture fields**: Using the parameters of the rendering camera, the renderer also computes the view-dependent weights of each texture. However, should an abrupt jump in viewpoint occur, the texture weights field can change rapidly. To overcome this challenge, *Montage4D* employs the concept of temporal texture weights so that texture weights transition smoothly over time.
(6) **Color synthesis and post-processing**: We fuse the sampled color using the temporal texture fields for each pixel in screen space. Our system also provides an optional post-processing module for screen-space ambient occlusion.

## 4 ALGORITHMS

In this section, we describe the how we elaborate each step of *Montage4D*.

### 4.1 Formulation and Goals

For each frame, given a triangle mesh and $N$ video texture maps $\mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_N$ streamed from the dedicated *Fusion4D* servers, our goal is to assign for each mesh vertex $\mathbf{v}$ a vector $(\mathscr{T}_\mathbf{v}^1, \ldots, \mathscr{T}_\mathbf{v}^N)$ of scalar texture weights. Let the **texture field** $\mathscr{T}$ denote the piecewise linear interpolation of these vectors over the triangle mesh. For each non-occluded vertex $\mathbf{v} \in \mathbb{R}^3$, we calculate a pair of corresponding $(u, v)$ coordinates for each texture map using back-projection. Finally, the resulting color $c_\mathbf{v}$ is fused using the *normalized* texture field $\mathscr{T}_\mathbf{v}$ at vertex $\mathbf{v}$:

$$c_\mathbf{v} = \sum_{i=1}^{N} c_\mathbf{v}^i \cdot \mathscr{T}_\mathbf{v}^i = \sum_{i=1}^{N} \text{texture}\left(\mathbf{M}_i, u, v\right) \cdot \mathscr{T}_\mathbf{v}^i \tag{1}$$

In order to achieve high-quality rendering, we need to take the following factors into consideration:

(1) **Smoothness**: The transition between the texture fields of adjacent vertices should be smooth, because human perception is especially sensitive to texture discontinuities.
(2) **Sharpness**: The rendered image should preserve the fine-scale detail of the input textures. However, due to imprecisely reconstructed geometry, fusing all the textures onto the mesh usually results in blurring or ghosting artifacts.
(3) **Temporal Consistency**: The texture fields should vary smoothly over time as the mesh changes and as a user's viewpoint changes.

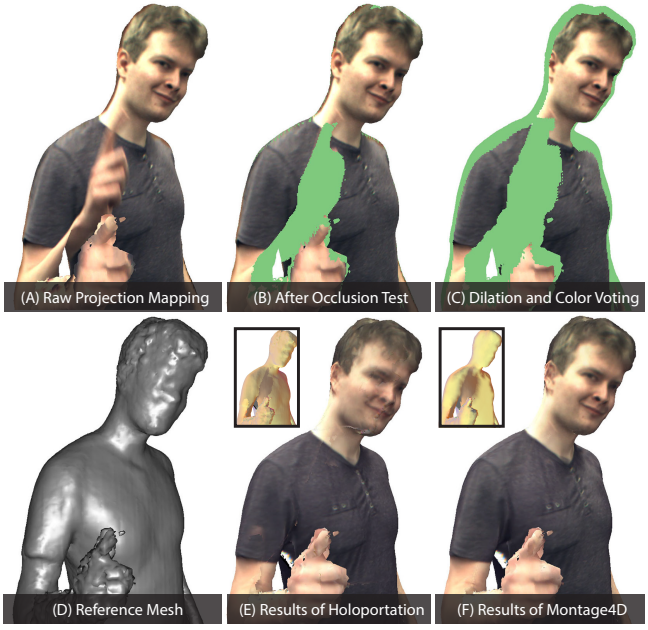


**Figure 3: This figure shows how texture weight fields improve the rendering quality compared to the baseline approach. *Holoportation* removes several ghosting artifacts by taking advantage of dilated depth maps and majority voting algorithm (top row), however, the rendering still suffers from fuzziness and visible seams (bottom row). (A) shows the raw projection mapping result from an input video texture, (B) shows the culling result after the occlusion test, (C) shows the culling result after using dilated depth maps and majority voting algorithm, (D) shows the input mesh, (E) and (F) respectively shows the rendering results from the baseline approach and our algorithm, together with the corresponding texture weight fields for comparison.**

## 4.2 Normal Weighted Blending with Dilated Depth Maps and Coarse-to-Fine Majority Voting Strategy

Our baseline approach is derived from the real-time implementation in the *Holoportation* project. This approach uses normal-weighted blending of non-occluded textures, together with a coarse-to-fine majority voting strategy. For each vertex $\mathbf{v}$, the texture field $\mathscr{T}_{\mathbf{v}}^{i}$ for the $i_{\text{th}}$ view is defined as

$$\mathscr{T}_{\mathbf{v}}^{i} = \mathscr{V}_{\mathbf{v}} \cdot \max\left(0, \hat{\mathbf{n}}_{\mathbf{v}} \cdot \hat{\mathbf{v}}_{i}\right)^{\alpha}, \tag{2}$$

where $\mathscr{V}_{\mathbf{v}}$ is a visibility test using dilated depth maps and multi-level majority voting algorithm introduced later, $\hat{\mathbf{n}}_{\mathbf{v}}$ is the smoothed normal vector at vertex $\mathbf{v}$, $\hat{\mathbf{v}}_{i}$ is the view direction of the $i_{\text{th}}$ camera, and $\alpha$ determines the smoothness of the transition, and favors the frontal views. This approach determines the texture fields purely based on the geometry, which may have missing or extruded triangles. The resulting texture fields may favor completely different views, thus introducing visible seams.

In order to remove the ghosting effect, we adopt the method from the *Holoportation* project, which uses a dilated depth map to detect the occluded regions as shown in Figure 3(C), thus removing many artifacts caused by inexact geometries: For each input view, we create a rasterized depth map of the surface and identify depth discontinuities using a filter radius determined by $\epsilon = 4$ pixels. Then, when rendering the surface mesh, within the pixel shader, we look up each depth map to see if the point lies within the discontinuous region. If such a discontinuity is found, we set $\mathscr{T}_{\mathbf{v}}^{i} = 0$.

In addition, we also adopt the same multi-level majority voting strategy. For a given vertex $\mathbf{v}$ and texture map $\mathbf{M}_i$, we search from coarse to fine levels, the sampled color $c_{\mathbf{v}}^{i}$ is trusted if at least half of the visible views (we denote the number of visible views as $X$) agree with it in the *Lab* color space, here $\delta = 0.15$:

$$\sum_{j=1, j\neq i}^{N} \left( \left| c_{\mathbf{v}}^{i} - c_{\mathbf{v}}^{j} \right| < \delta \right) \geq \left\lfloor \frac{X}{2} \right\rfloor \tag{3}$$

Although the dilated depth maps and multilevel majority voting strategy can mitigate most of the ghosting effects in real time (Figure 3(C)) the rendering results still suffer from blurring and visible seams, as shown in Figure 3(E).

## 4.3 Computing Misregistration and Occlusion Seams

Our algorithm identifies each triangle as a misregistration or occlusion seam when any of the following three cases occur:

(1) **Self-occlusion**: One or two vertices of the triangle are occluded in the dilated depth map while the others are not.
(2) **Majority voting**: The triangle vertices have different results in the majority voting process, which may be caused by either misregistration or self-occlusion.
(3) **Field of View**: One or two triangle vertices lie outside the camera's field of view or in the subtracted background region while the rest are not.

Some of these examples are shown in Figure 4.

For the datasets acquired for real-time telepresence applications we have observed the fraction of seam triangles to be less than 1%. This observation has guided us to process the triangles adjacent to the seams, using a propagation procedure by calculating the geodesics directly on the GPU.

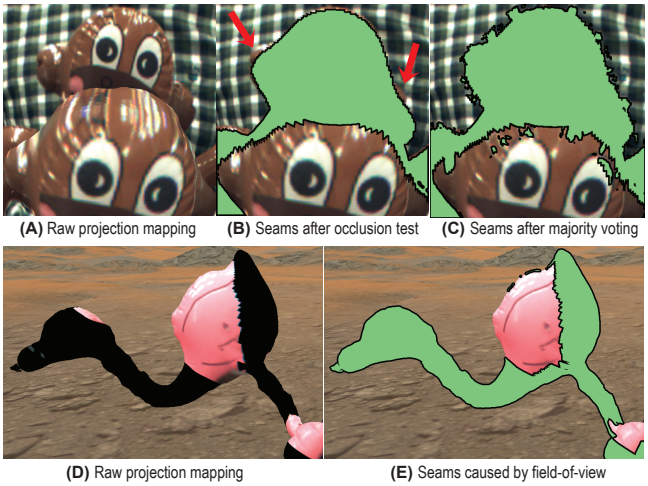


**Figure 4: Examples of misregistration and occlusion seams. (A) shows the raw projection mapping result of a monkey toy in front of a plaid shirt, (B) shows the seams after the occlusion test with dilated depth maps, and (C) shows the seams after the majority voting test. Note that while (B) fails to remove some ghosting artifacts from the monkey toy, (C) removes most of them. (D) shows another projection onto a crane toy, (E) shows the seams identified by the field-of-view test.**

## 4.4 Discrete Geodesic Distance Field for Diffusing Seams

We efficiently diffuse the texture fields using the geodesic distance fields, by making a tradeoff between accuracy and efficiency of the resulting diffusion. We follow a variant of the highly efficient approximation algorithm described in [Surazhsky et al. 2005], by computing the geodesics distance fields from a set of vertices rather than a single vertex as follows:

Let $S$ be a piecewise planar surface defined by the triangle mesh. We define the geodesic distance function as $\mathcal{D}(\cdot) : S \mapsto \mathbb{R}$. In an earlier stage, we extracted the vertices from the seam triangles $V_s \in S$ as the source vertices. For any point $p \in S$, the algorithm returns the length of the geodesic path $\mathcal{D}(p)$ from $p$ back to the closest seam vertex $v \in V_s$. We iteratively diffuse across the triangles from the seams towards the non-occluded triangles.

As illustrated in Figure 5, for each edge $e$, we maintain a small number of windows $\mathbf{w}(e)$ consisting of a pair of coordinates $(c_l, c_r)$ (counterclockwise), the corresponding geodesic distance $(d_l, d_r)$ to the closest pseudosource source $s$, the direction of the geodesic path $\tau$, and the geodesic length $\sigma = \mathcal{D}(s)$. The position of $s$ can be calculated by intersecting two circles. As suggested by [Surazhsky et al. 2005], when propagating a window $w_1(e)$ with an existing window $w_0(e)$ on the same edge, we try to merge the two windows $w' \leftarrow w_0(e) \cup w_1(e)$, if the directions $\tau_0, \tau_2$ agree with each other, and the estimated geodesic lengths are within a bounded error: $|\mathcal{D}(w_0) - \mathcal{D}(w_1)| < \varepsilon$.

In order to achieve interactive rates for rendering, we march at most $k = 15$ triangles from the seams in $K = 20$ iterations. In this propagation process, we maintain two windows per edge and discard the rest. We chose the parameter $k < K$ so that each vertex's minimum geodesic distance field could be updated from the vertices that are $K - k$ edges away. As Figure 6 shows, this compromise gives us visually pleasing results for diffusing the texture fields spatially near the seams.

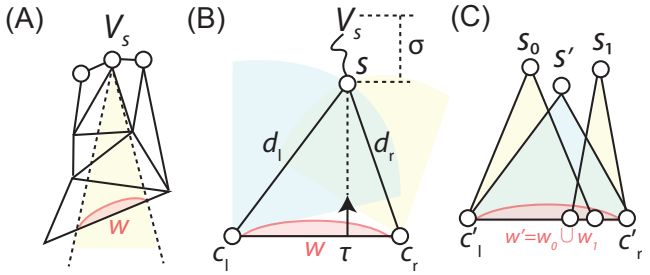


**Figure 5: Illustration of computing the approximate geodesics. (A) shows the concept of the geodesic window from a single source vertex. (B) shows the components within a window. (C) shows the merging process of two overlapping windows for approximation.**

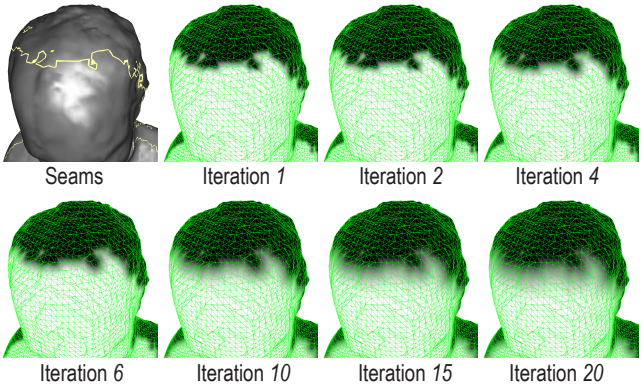


**Figure 6: Examples of the initial seam triangles and the propagation process for updating the geodesic distance field.**

## 4.5 Temporal Texture Fields

To prevent the texture weights from changing too fast during view transitions, we use *target texture fields* and *temporal texture fields*. The target texture fields are determined using view-dependent texture weights and occlusion seams:

$$\mathbf{T}_{\mathbf{v}}^{i} = \mathcal{V}_{\mathbf{v}} \cdot g^{i} \cdot \gamma_{\mathbf{v}}^{i} \cdot \max(0, \hat{\mathbf{v}} \cdot \hat{\mathbf{v}}_i)^{\alpha}, \tag{4}$$

where, $\mathcal{V}_{\mathbf{v}}$ is the original visibility test at vertex $\mathbf{v}$ with dilated depth maps and multi-level majority voting, $g^i$ is a normalized global visibility score of each view, which is calculated by the number of visible vertices from each view. Therefore, $g^i$ reduces weights for less significant views. $\gamma_{\mathbf{v}}^{i}$ is the minimum length of the equidistance geodesics to the seams for the texture map $\mathbf{M}_i$, $\hat{\mathbf{v}}$ is the view vector from the current user's camera to the vertex $\mathbf{v}$, $\hat{\mathbf{v}}_i$ is the view vector

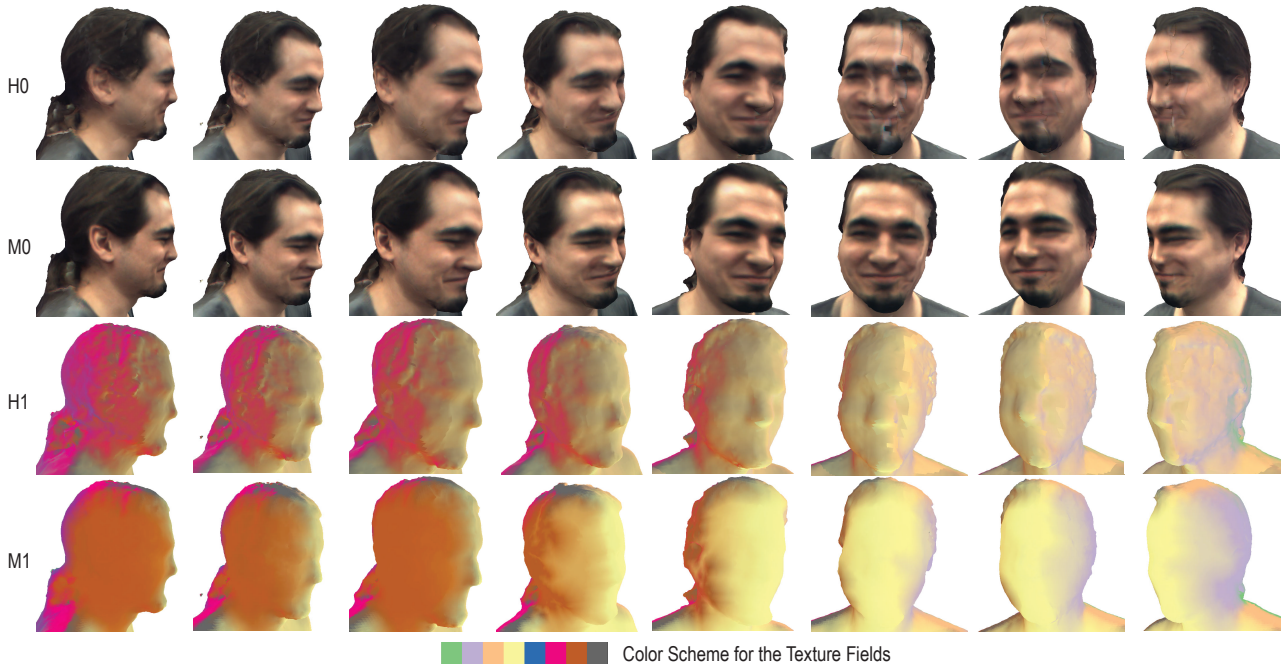


**Figure 7: Spatiotemporal comparison of the rendered results (H0, M0) and corresponding texture fields (H1, M1) for *Holoportation* (H0, H1) and *Montage4D* (M0, M1) across 8 viewpoints and 40 frames. As shown in the figures, *Montage4D* takes advantage of view-dependent rendering while mitigating visible seams. In addition, temporal texture weights facilitate smooth transitions in space and time. Please see the accompanying video for a temporal comparison.**

of the $i_{\text{th}}$ camera, and $\alpha$ determines the smoothness of the transition. We use temporal texture fields to handle the temporal artifacts as follows:

$$\mathscr{T}_{\mathbf{v}}^{i}(t) = (1-\lambda)\mathscr{T}_{\mathbf{v}}^{i}(t-1) + \lambda \mathbf{T}_{\mathbf{v}}^{i}(t), \tag{5}$$

where, $\mathscr{T}_{\mathbf{v}}^{i}(t)$ represents the temporal texture field at vertex $\mathbf{v}$ at frame $t$ and the time constant $\lambda$ determines the transition rate of the texture fields: $\lambda = 0.05$.

We normalize the texture fields and fuse the sampled colors using the Equation 1. For highly occluded regions, if $\sum_i \gamma_{\mathbf{v}}^{i} < 1$, we preserve the result from normal-weighted blending to fill in the black pixels. We discuss the limitations of this compromise in Section 6. Figure 7 shows comparative results between *Holoportation* and *Montage4D*.

## 5 EXPERIMENTAL RESULTS

### 5.1 Comparison with the Holoportation Approach

We implement our rendering pipeline using the multi-pass compute, vertex, and fragment shaders with *DirectX 11*, and conduct quantitative analysis on a commodity workstation with a *GeForce GTX 1080* graphics card with 8 GB frame buffers. We evaluate our results with five recorded datasets with a *Fusion4D* program running in the background to feed the reconstructed meshes and video textures to *Montage4D*. These datasets cover a wide range of subjects, including children, adults, and air-inflated toys with specular highlights. Each dataset contains at least 500 frames, and each frame contains at least 130, 000 vertices, 250, 000 triangles, and 8 video texture maps at the resolution of $2048 \times 2048$ pixels. The average frame rate of the video textures is 25 frames per second (FPS). As in the *Holoportation* project, all incoming data is decompressed using the LZ4 algorithm prior to its ingestion in the rendering pipeline.

First, we conduct a cross-validation experiment over the five datasets between the ground truth image from each camera's perspective and the rendering results of the *Holoportation* or *Montage4D* renderer. We demonstrate the quantitative results using the average of the root mean square error (RMSE) of the RGB color values, the structural similarity (SSIM) [Wang et al. 2004], and peak signal-to-noise ratio (PSNR). The results are shown in Table 1. We can see that *Montage4D* achieves higher image quality (lower RMSE, higher SSIM and PSNR) while maintaining interactive frame rates for virtual reality applications.

Next, we visually compare the quality and sharpness of the rendered images from novel views, as illustrated in Figure 8. We also show the input meshes and representative back-projected images. Although the approach taken in the *Holoportation* project is able to render textured meshes smoothly and eliminates most of the ghosting artifacts, it often fails to preserve the fine details such as human faces. In contrast, the *Montage4D* renderer preserves the details from the dominating views using view-dependent texture weights and transitions smoothly using the temporal texture fields. Meanwhile, the diffusion process in *Montage4D* is able to remove

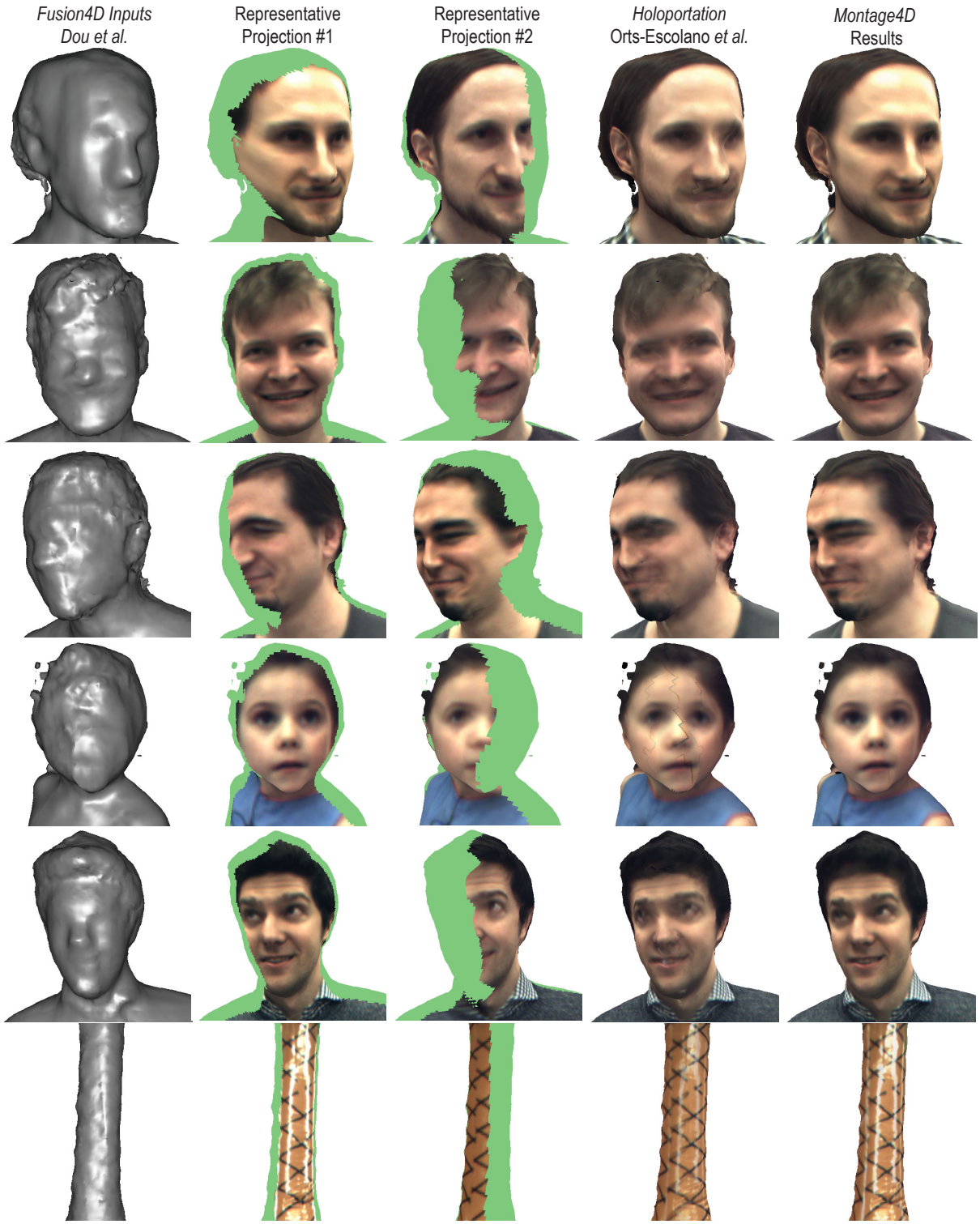


**Figure 8: Comparison with the Holoportation approach. From left to right: the input mesh generated by Fusion4D, two representative back-projected images, and the rendering results from Holoportation and our Montage4D system.**

**Table 1: Comparison between *Holoportation* and *Montage4D* in cross-validation experiments**

| Dataset | Frames | #vertices / frame | #triangles / frame | *Holoporation* | | | | *Montage4D* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | RMSE | SSIM | PSNR | FPS | RMSE | SSIM | PSNR | FPS |
| Timo | 837 | 131K | 251K | 5.63% | 0.9805 | 38.60dB | 227.2 | 3.27% | 0.9905 | 40.23dB | 135.0 |
| Yury | 803 | 132K | 312K | 5.44% | 0.9695 | 39.20dB | 222.8 | 3.01% | 0.9826 | 40.52dB | 130.5 |
| Sergio | 837 | 215K | 404K | 7.74% | 0.9704 | 29.84dB | 186.8 | 4.21% | 0.9813 | 30.09dB | 114.3 |
| Girl | 1192 | 173K | 367K | 7.16% | 0.9691 | 36.28dB | 212.56 | 3.73% | 0.9864 | 36.73dB | 119.4 |
| Julien | 526 | 157K | 339K | 12.63% | 0.9511 | 33.94dB | 215.18 | 6.71% | 0.9697 | 35.05dB | 120.6 |

**Table 2: Timing comparison between *Holoportation* and *Montage4D* for a new frame of geometry. Geometry and textures are streamed at around 30 fps.**

| Procedure | Timing (ms) | |
|---|---|---|
| | Holoportation | Montage4D |
| Communication between CPU and GPU | 4.83 | 9.49 |
| Rendering and Texture Sampling | 0.11 | 0.30 |
| Rasterized Depth Maps calculation | 0.14 | 0.13 |
| Seams Identification | N/A | 0.01 |
| Approximate Geodesics estimation | N/A | 0.31 |
| Other events | 0.12 | 0.18 |
| Total | 5.11 | 10.40 |

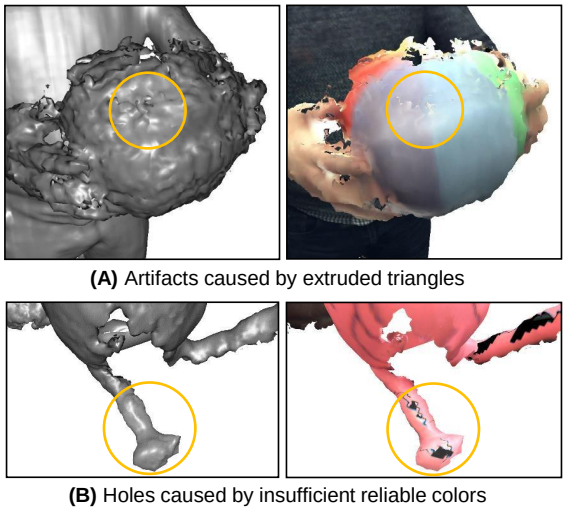


**Figure 9: Limitations of our approach. Extruded triangles and highly-occluded spots may still cause artifacts.**

most of the mis-registration and occlusion seams that occur in the representative back-projections.

Additionally, we use the *Unity profiler* to analyze and compare the timing for a typical frame of the *Sergio* dataset. As shown in Table 2, the main difference between the two approaches is data transfer time between CPU and GPU. In addition to copying buffers for vertex indices and positions, the *Montage4D* system also transfers compute buffers for geodesics, texture fields, and seam factors, which induces a small overhead over the original approach. However, dispatching the diffusion kernels does not impact the frame rate much and the overall timing is still satisfactory for interactive applications.

## 6 LIMITATIONS

Even though we have demonstrated a real-time pipeline for seamlessly fusing multiview videos with dynamic meshes, our system is not without limitations as discussed next.

### 6.1 Inaccurate Geometries

As shown in Figure 9(A), our system suffers artifacts resulting from the extruded triangles reconstructed during very fast motion. It should be possible to use a **remeshing** algorithm [Alliez et al. 2002; Qu and Meyer 2006] to tackle such problems. With the state-of-the-art *Motion2Fusion* reconstruction pipeline [Dou et al. 2017], such artifacts may be eliminated with more accurate geometries.

### 6.2 Missing Texture Fields

Figure 9(B) shows the challenging issue caused by insufficient reliable colors. Such problems may be solved by user-guided inpainting and seamless cloning, which are proposed in the offline *TextureMontage* system [Zhou et al. 2005]. However, for interactive applications, it will be ideal if one could achieve such interpolation with minimal overhead without the user's intervention.

## 7 CONCLUSION AND FUTURE WORK

In this paper, we have presented *Montage4D*, an interactive and real-time solution to blend multiple video textures onto dynamic meshes with nearly indiscernible view transitions. We improve on previous *Holoportation* renderer by adopting view-dependent rendering, seam identification, diffusion based on geodesic distance fields, and smooth transition using temporal texture fields. Our technique offers sharper images than previous interactive texturing algorithms, allowing users to observe fine facial expressions for immersive telepresence and communication. Recently, in collaboration with the *Mobile Holoportation* team, we have already integrated the *Montage4D* pipeline with the interactive live streaming platform *Mixer*[2].

[2]Mixer: https://mixer.com

In the future, we would like to further integrate the *Montage4D* texturing pipeline with the cloud-based scene acquisition servers. By incorporating the user's view directions, the acquisition servers could progressively synthesize a compact view-dependent video texture atlas directly on the client side, thus greatly reducing the bandwidth requirement. We would like to investigate adaptive and efficient optical flow algorithms over the mesh surface [Prada et al. 2016, 2017b] to further optimize the texture fields. In the supplementary materials, we identify some challenges with the screen-based optical flow approach [Eisemann et al. 2008; Eisemann and Magnor 2007]. We observe that surface specularity and poor color calibration may result visible artifacts using screen-space optical flow. One may take advantage of real-time texture filtering algorithms such as [Chajdas et al. 2011; Crassin et al. 2015; Heitz et al. 2013; Mavridis and Papaioannou 2011; Shirley et al. 2011], or Poisson blending [Pérez et al. 2003] over the 3D space [Chuang et al. 2009] to eliminate the artifacts. In addition, we would like to investigate adaptive and efficient optical flow algorithms over the mesh surface [Prada et al. 2016].

We envision our algorithm to be useful for many virtual and augmented reality applications, such as remote business meetings, medical training, and live social events.

## ACKNOWLEDGMENTS

We appreciate the entire Holoportation Team including Shahram Izadi, Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, Marek Kolwalski, Qiyu Chen, Spencer Fowers, Jeff Foster, and Ben Cutler at Microsoft Research, Redmond for providing the data and the pipeline of *Fusion4D* and *Holoportation*. We would also like to thank the anonymous reviewers for the insightful comments on the manuscript.

This work has been supported in part by the NSF Grants 14-29404, 15-64212, and the State of Maryland's MPower initiative. Any opinions, findings, conclusions, or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the research sponsors.

## REFERENCES

Cédric Allène, Jean-Philippe Pons, and Renaud Keriven. 2008. Seamless image-based texture atlases using multi-band blending. In *19th International Conference on Pattern Recognition*. IEEE, 1–4. https://doi.org/10.1109/ICPR.2008.4761913

Pierre Alliez, Mark Meyer, and Mathieu Desbrun. 2002. Interactive geometry remeshing. *ACM Transactions on Graphics (TOG)* 21, 3 (2002), 347–354. https://doi.org/10.1145/566570.566588

David Bommes and Leif Kobbelt. 2007. Accurate computation of geodesic distance fields for polygonal curves on triangle meshes. In *Computer Cision, Graphics and Visualization Workshop*, Vol. 7. VMV, 151–160.

Cedric Cagniart, Edmond Boyer, and Slobodan Ilic. 2010. Probabilistic deformable surface tracking from multiple videos. In *ECCV'10 Proceedings of the 11th European Conference on Computer Vision: Part IV*. Springer, 326–339. https://doi.org/10.1007/978-3-642-15561-1_24

Dan Casas, Margara Tejera, Jean-Yves Guillemaut, and Adrian Hilton. 2013. Interactive animation of 4D performance capture. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 19, 5 (2013), 762–773. https://doi.org/10.1109/TVCG.2012.314

Dan Casas, Marco Volino, John Collomosse, and Adrian Hilton. 2014. 4D video textures for interactive character appearance. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 371–380.

Matthäus G Chajdas, Morgan McGuire, and David Luebke. 2011. Subpixel reconstruction antialiasing for deferred shading. In *Proceedings of Symposium on Interactive 3D Graphics and Games (I3D)*. ACM, 15–22. https://doi.org/10.1145/1944745.1944748

Ming Chuang, Linjie Luo, Benedict J Brown, Szymon Rusinkiewicz, and Michael Kazhdan. 2009. Estimating the Laplace-Beltrami Operator by Restricting 3D Functions. In *Computer graphics forum*, Vol. 28. Wiley Online Library, 1475–1484. https://doi.org/10.1111/j.1467-8659.2009.01524.x

Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. 2015. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 69. https://doi.org/10.1145/2766945

Cyril Crassin, Morgan McGuire, Kayvon Fatahalian, and Aaron Lefohn. 2015. Aggregate G-buffer anti-aliasing. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games (I3D)*. ACM, 109–119.

Edilson De Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. 2008. Performance capture from sparse multi-view video. *ACM Transactions on Graphics (TOG)* 27, 3 (2008), 98. https://doi.org/10.1145/1360612.1360697

Paul Debevec, Steven Gortler, Leonard McMillan, Richard Szeliski, and Chris Bregler. 1998a. Image-based modeling and rendering. *SIGGRAPH 98 Course Notes for Course* 15 (1998).

Paul Debevec, Yizhou Yu, and George Borshukov. 1998b. Efficient view-dependent image-based rendering with projective texture-mapping. In *Rendering Techniques*. Springer, 105–116.

Fernando do Goes, Mathieu Desbrun, and Yiying Tong. 2015. Vector field processing on triangle meshes. In *SIGGRAPH Asia 2015 Courses*. ACM, 17. https://doi.org/10.1145/2818143.2818167

Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. 2017. Motion2fusion: real-time volumetric performance capture. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 246. https://doi.org/10.1145/3130800.3130801

Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. 2016. Fusion4D: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 114. https://doi.org/10.1145/2897824.2925969

Ruofei Du, Sujal Bista, and Amitabh Varshney. 2016. Video Fields: Fusing Multiple Surveillance Videos into a Dynamic Virtual Environment. In *Proceedings of the 21st International Conference on Web3D Technology*. ACM, 165–172. https://doi.org/10.1145/2945292.2945299

Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson De Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. 2008. Floating Textures. *Computer Graphics Forum* 27, 2 (2008), 409–418. https://doi.org/10.1111/j.1467-8659.2008.01138.x

Martin Eisemann and Marcus A Magnor. 2007. Filtered Blending: A new, minimal Reconstruction Filter for Ghosting-Free Projective Texturing with Multiple Images.. In *VMV*. 119–126.

Sean Ryan Fanello, Julien Valentin, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, Carlo Ciliberto, Philip Davidson, and Shahram Izadi. 2017a. Low Compute and Fully Parallel Computer Vision with HashMatch. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 3894–3903. https://doi.org/10.1109/ICCV.2017.418

Sean Ryan Fanello, Julien Valentin, Christoph Rhemann, Adarsh Kowdle, Vladimir Tankovich, Philip Davidson, and Shahram Izadi. 2017b. UltraStereo: Efficient Learning-based Matching for Active Stereo Systems. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 6535–6544. https://doi.org/10.1109/CVPR.2017.692

Henry Fuchs, Gary Bishop, Kevin Arthur, Leonard McMillan, Ruzena Bajcsy, Sang Lee, Hany Farid, and Takeo Kanade. 1994. Virtual Space Teleconferencing Using a Sea of Cameras. In *Proc. First International Conference on Medical Robotics and Computer Assisted Surgery (MRCAS)*, Vol. 26. TR94-033, 7.

H Fuchs and U Neumann. 1993. A vision of telepresence for medical consultation and other applications. In *Sixth International Symposium of Robotics Research*. IFRR, 555–571.

Yasutaka Furukawa and Jean Ponce. 2008. Dense 3D Motion Capture from Synchronized Video Streams. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1–8. https://doi.org/10.1109/CVPR.2009.5206868

Ran Gal, Yonatan Wexler, Eyal Ofek, Hugues Hoppe, and Daniel Cohen-Or. 2010. Seamless montage for texturing models. *Computer Graphics Forum* 29, 2 (2010), 479–486. https://doi.org/10.1111/j.1467-8659.2009.01617.x

Bastian Goldluecke and Marcus Magnor. 2004. Space-Time Isosurface Evolution for Temporally Coherent 3D Reconstruction. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1. IEEE, I–350.

Eitan Grinspun, Mathieu Desbrun, Konrad Polthier, Peter Schröder, and Ari Stern. 2006. Discrete differential geometry: an applied introduction. *ACM SIGGRAPH Course* 7 (2006), 1–139.

Kaiwen Guo, Feng Xu, Yangang Wang, Yebin Liu, and Qionghai Dai. 2015. Robust non-rigid motion tracking and surface reconstruction using l0 regularization. In

*Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 3083–3091. https://doi.org/10.1109/ICCV.2015.353

Kaiwen Guo, Feng Xu, Tao Yu, Xiaoyang Liu, Qionghai Dai, and Yebin Liu. 2017. Real-Time Geometry, Albedo, and Motion Reconstruction Using a Single RGB-D Camera. *ACM Transactions on Graphics (TOG)* 36, 3 (2017), 32. https://doi.org/10.1145/3083722

Eric Heitz, Derek Nowrouzezahrai, Pierre Poulin, and Fabrice Neyret. 2013. Filtering color mapped textures and surfaces. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*. ACM, 129–136. https://doi.org/10.1145/2448196.2448217

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. 2011. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST)*. ACM, 559–568. https://doi.org/10.1145/2037826.2037857

Zsolt Janko and Jean-Philippe Pons. 2009. Spatio-temporal image-based texture atlases for dynamic 3-D models. In *IEEE 12th International Conference on Computer Vision Workshops*. IEEE, 1646–1653. https://doi.org/10.1109/ICCVW.2009.5457481

Takeo Kanade, Peter Rander, and PJ Narayanan. 1997. Virtualized Reality: Constructing Virtual Worlds from Real Scenes. *IEEE Multimedia* 4, 1 (1997), 34–47.

Takashi Kanai and Hiromasa Suzuki. 2001. Approximate shortest path on a polyhedral surface and its applications. *Computer-Aided Design* 33, 11 (2001), 801–811. https://doi.org/10.1109/GMAP.2000.838256

Sanjiv Kapoor. 1999. Efficient computation of geodesic shortest paths. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*. ACM, 770–779. https://doi.org/10.1145/301250.301449

Mark Lanthier, Anil Maheshwari, and Jörg-Rüdiger Sack. 1997. Approximating weighted shortest paths on polyhedral surfaces. In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*. ACM, 274–283. https://doi.org/10.1145/262839.263101

Victor Lempitsky and Denis Ivanov. 2007. Seamless mosaicing of image-based texture maps. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1–6. https://doi.org/10.1109/CVPR.2007.383078

Benjamin Lok. 2001. Online model reconstruction for interactive virtual environments. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics (I3D)*. ACM, 69–72. https://doi.org/10.1145/364338.364364

Dimas Martınez, Luiz Velho, and Paulo Cezar Carvalho. 2004. Geodesic paths on triangular meshes. In *Proceedings of Computer Graphics and Image Processing*. IEEE, 8. https://doi.org/10.1109/SIBGRA.2004.1352963

Pavlos Mavridis and Georgios Papaioannou. 2011. High quality elliptical texture filtering on GPU. In *Symposium on Interactive 3D Graphics and Games (I3D)*. ACM, 23–30. https://doi.org/10.1145/1944745.1944749

Joseph SB Mitchell. 2000. Geometric shortest paths and network optimization. *Handbook of Computational Geometry* 334 (2000), 633–702. http://www.ams.sunysb.edu/~jsbm/papers/survey.ps.gz

Joseph SB Mitchell, David M Mount, and Christos H Papadimitriou. 1987. The discrete geodesic problem. *SIAM J. Comput.* 16, 4 (1987), 647–668.

Karthik S Narayan and Pieter Abbeel. 2015. Optimized color models for high-quality 3D scanning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2503–2510. https://doi.org/10.1109/IROS.2015.7353717

PJ Narayanan, Peter W Rander, and Takeo Kanade. 1998. Constructing Virtual Worlds Using Dense Stereo. In *Sixth International Conference on Computer Vision (ICCV)*. IEEE, 3–10. https://doi.org/10.1109/ICCV.1998.710694

Richard A Newcombe, Dieter Fox, and Steven M Seitz. 2015. DynamicFusion: Reconstruction and Tracking of Non-Rigid Scenes in Real-Time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 343–352. https://doi.org/10.1109/CVPR.2015.7298631

Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 127–136. https://doi.org/10.1109/ISMAR.2011.6092378

Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. 2016. Holoportation: Virtual 3D Teleportation in Real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST)*. ACM, 741–754. https://doi.org/10.1145/2984511.2984517

Rob Patro, Cheuk Yiu Ip, Sujal Bista, and Amitabh Varshney. 2011. Social Snapshot: A system for temporally coupled social photography. *IEEE Computer Graphics and Applications* 31, 1 (2011), 74–84.

Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. *ACM Transactions on graphics (TOG)* 22, 3 (2003), 313–318. https://doi.org/10.1145/882262.882269

Fabián Prada, Misha Kazhdan, Ming Chuang, Alvaro Collet, and Hugues Hoppe. 2016. Motion graphs for unstructured textured meshes. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 108. https://doi.org/10.1145/2897824.2925967

Fabián Prada, Misha Kazhdan, Ming Chuang, Alvaro Collet, and Hugues Hoppe. 2017a. Spatiotemporal atlas parameterization for evolving meshes. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 58. https://doi.org/10.1145/3072959.3073679

Fabián Prada, Misha Kazhdan, Ming Chuang, Alvaro Collet, and Hugues Hoppe. 2017b. Spatiotemporal atlas parameterization for evolving meshes. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 58. https://doi.org/10.1145/3072959.3073679

Yipeng Qin, Xiaoguang Han, Hongchuan Yu, Yizhou Yu, and Jianjun Zhang. 2016. Fast and exact discrete geodesic computation based on triangle-oriented wavefront propagation. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 13. https://doi.org/10.1145/2897824.2925930

Lijun Qu and Gary W Meyer. 2006. Perceptually driven interactive geometry remeshing. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games (I3D)*. ACM, 199–206. https://doi.org/10.1145/1111411.1111447

Peter Shirley, Timo Aila, Jonathan Cohen, Eric Enderton, Samuli Laine, David Luebke, and Morgan McGuire. 2011. A local image reconstruction algorithm for stochastic rendering. In *Symposium on Interactive 3D Graphics and Games (I3D)*. ACM, PAGE–5. https://doi.org/10.1145/1944745.1944747

Vitaly Surazhsky, Tatiana Surazhsky, Danil Kirsanov, Steven J Gortler, and Hugues Hoppe. 2005. Fast exact and approximate geodesics on meshes. *ACM transactions on graphics (TOG)* 24, 3 (2005), 553–560. https://doi.org/10.1145/1073204.1073228

Herman Towles, Wei-Chao Chen, Ruigang Yang, Sang-Uok Kum, Henry Fuchs Nikhil Kelshikar, Jane Mulligan, Kostas Daniilidis, Henry Fuchs, Carolina Chapel Hill, Nikhil Kelshikar Jane Mulligan, et al. 2002. 3D tele-collaboration over Internet2. In *International Workshop on Immersive Telepresence*. Juan Les Pins, France, 6.

Vibhav Vineet, Jonathan Warrell, and Philip HS Torr. 2014. Filter-based mean-field inference for random fields with higher-order terms and product label-spaces. *International Journal of Computer Vision* 110, 3 (2014), 290–307.

Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. 2008. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics (TOG)* 27, 3 (2008), 97. https://doi.org/10.1145/1360612.1360696

Marco Volino, Dan Casas, John Collomosse, and Adrian Hilton. 2014. Optimal Representation of Multiple View Video. In *Proceedings of the British Machine Vision Conference*. BMVA Press. https://doi.org/10.5244/C.28.8

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612.

Feng Xu, Yebin Liu, Carsten Stoll, James Tompkin, Gaurav Bharaj, Qionghai Dai, Hans-Peter Seidel, Jan Kautz, and Christian Theobalt. 2011. Video-based characters: creating new human performances from a multi-view video database. *ACM Transactions on Graphics (TOG)* 30, 4 (2011), 32. https://doi.org/10.1145/2010324.1964927

Weipeng Xu, Mathieu Salzmann, Yongtian Wang, and Yue Liu. 2015. Deformable 3D Fusion: From Partial Dynamic 3D Observations to Complete 4D Models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2183–2191. https://doi.org/10.1109/ICCV.2015.252

Mao Ye and Ruigang Yang. 2014. Real-Time Simultaneous Pose and Shape Estimation for Articulated Objects Using a Single Depth Camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2345–2352.

Tao Yu, Kaiwen Guo, Feng Xu, Yuan Dong, Zhaoqi Su, Jianhui Zhao, Jianguo Li, Qionghai Dai, and Yebin Liu. 2017. BodyFusion: Real-time Capture of Human Motion and Surface Geometry Using a Single Depth Camera. In *The IEEE International Conference on Computer Vision (ICCV)*. ACM. https://doi.org/10.1109/ICCV.2017.104

Qing Zhang, Bo Fu, Mao Ye, and Ruigang Yang. 2014. Quality dynamic human body modeling using a single low-cost depth camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 676–683. https://doi.org/10.1109/CVPR.2014.92

Kun Zhou, Xi Wang, Yiying Tong, Mathieu Desbrun, Baining Guo, and Heung-Yeung Shum. 2005. TextureMontage: Seamless texturing of arbitrary surfaces from multiple images. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 1148–1155. https://doi.org/10.1145/1073204.1073325

Qian-Yi Zhou and Vladlen Koltun. 2014. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 155. https://doi.org/10.1145/2601097.2601134

C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. 2004. High-quality video view interpolation using a layered representation. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 600–608. https://doi.org/10.1145/1015706.1015766