Montage4D: Real-time Seamless Fusion and Stylization 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

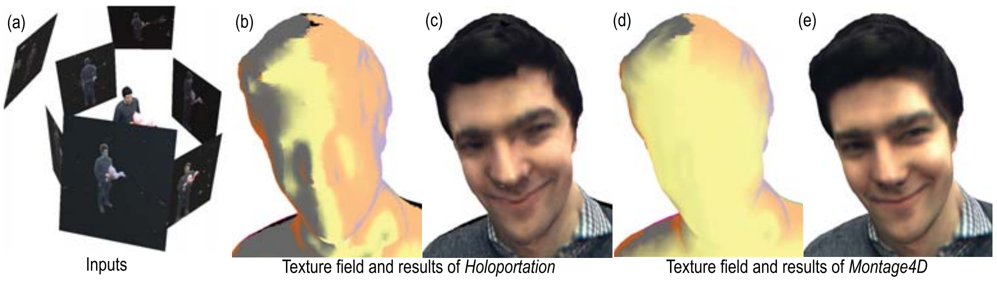
(a)
(b)
(c)
(d)
(e)
Inputs
Texture field and results of *Holoportation*
Texture field and results of *Montage4D*

Figure 1. The Montage4D algorithm seamlessly stitches multiview video textures onto dynamic meshes 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. Furthermore, we present our preliminary exploration towards real-time stylization and relighting to empower the Holoportation users to interactively stylize live 3D content. We envision a wide range of applications for *Montage4D*, including immersive telepresence for business, training, and live entertainment.

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 9.

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 , Montage4D, 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 9, our system achieves significantly higher visual quality at interactive rates compared to the state-of-the-art *Holoportation* system. In addition to the Montage4D pipeline [Du et al. 2018], we present two real-time shaders to stylize and relight the reconstructed model. As compressing and streaming 4D performance [Tang et al. 2018] become practical in real time, Holoportation users may use our stylization techniques to render sketchy scenes or relight performance in mixed reality. Please refer to `www.montage4d.com` for the supplementary video, slides, and code.

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 and real-time stylization.

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.*[1993; 1994] 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 [Lok 2001; Vlasic et al. 2008; Sankaranarayanan et al. 2009; Cagniart et al. 2010; Patro et al. 2011; Xu et al. 2011; Casas et al. 2013; Collet et al. 2015; Du et al. 2016; Prada et al. 2016; Prada et al. 2017a; Boukhayma and Boyer 2018; Huang et al. 2018], 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* [Newcombe et al. 2011; Izadi 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 spatiotemporal 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 employs the Fusion4D algorithm [Dou et al. 2016] for generating temporally consistent polynomial meshes from 8 depth cameras. The meshes, RGB videos, and audio streams are transferred from a fusion server with two NVIDIA TITAN X graphics cards to a dedicated rendering machine with an NVIDIA GTX 1080. To achieve real-time performance for texturing, their system blends multi-view videos textures according to the dot product between surface normals and the camera viewpoint directions. Finally, it offloads the rendering results to VR or AR headsets.

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 [Fanello et al. 2017b; Fanello et al. 2017a; Guo et al. 2018], 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 spatiotemporal 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 [Mitchell 2000; Grinspun et al. 2006; do Goes et al. 2015]. Geodesics on smooth sur-

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

faces 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 $\mathcal{O}\left(n^2 \log n\right)$, and the average is close to $\mathcal{O}\left(n^{1.5}\right)$ [Surazhsky et al. 2005; Bommes and Kobbelt 2007]. 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 $\mathcal{O}\left(n^2\right)$ space like the previous approaches. Kapoor [1999] proposes a sophisticated approach for the "single source, single destination" case in $\mathcal{O}\left(n \log^2 n\right)$ 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:

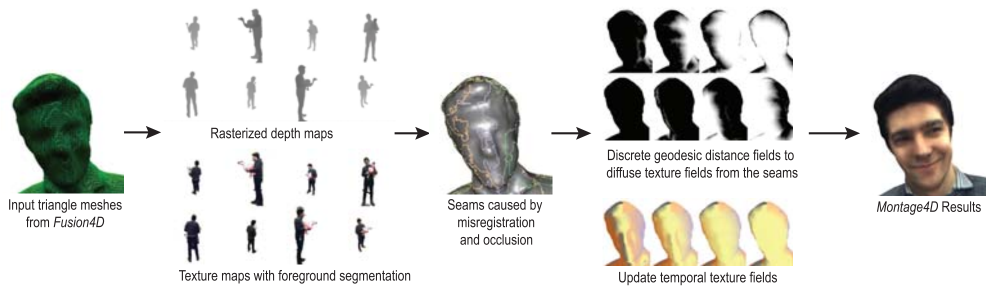
Input triangle meshes from *Fusion4D*
Rasterized depth maps
Texture maps with foreground segmentation
Seams caused by misregistration and occlusion
Discrete geodesic distance fields to diffuse texture fields from the seams
Update temporal texture fields
Montage4D Results

Figure 2. The workflow of the Montage4D rendering pipeline.

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 *Fusion4D* system reconstructs polygonal meshes from 8 pods of depth cameras at around 25-30 frames per second. 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.

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_{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_{th} camera, and α 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

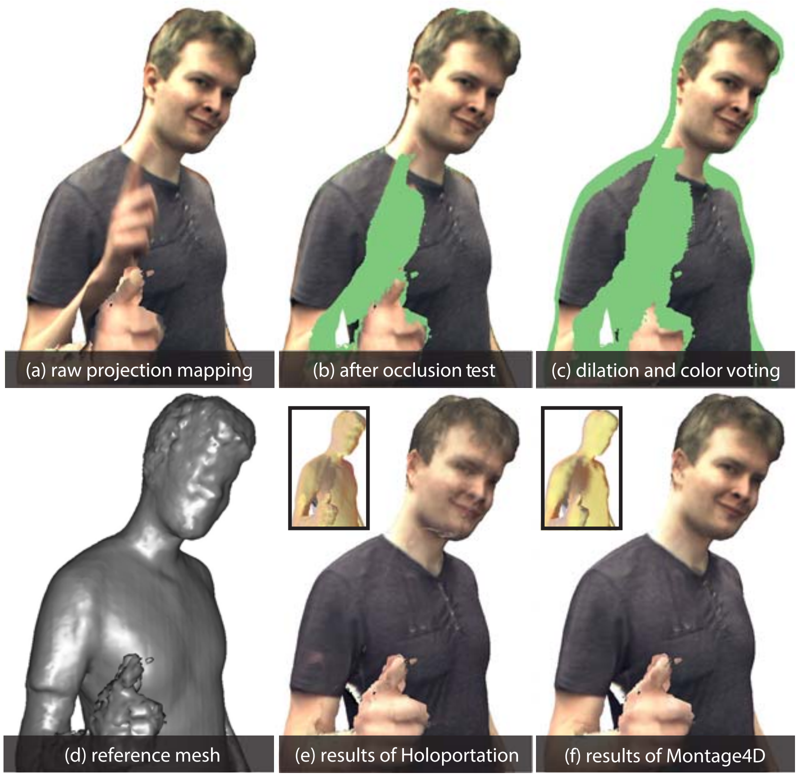
(a) raw projection mapping
(b) after occlusion test
(c) dilation and color voting
(d) reference mesh
(e) results of Holoportation
(f) results of Montage4D

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.

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 mit-

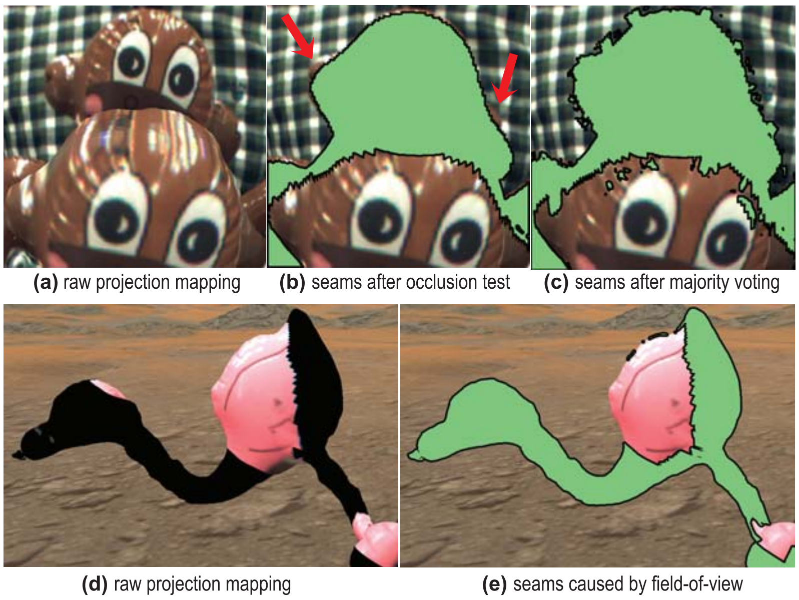
(a) raw projection mapping
(b) seams after occlusion test
(c) seams after majority voting
(d) raw projection mapping
(e) seams caused by field-of-view

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.

igate 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

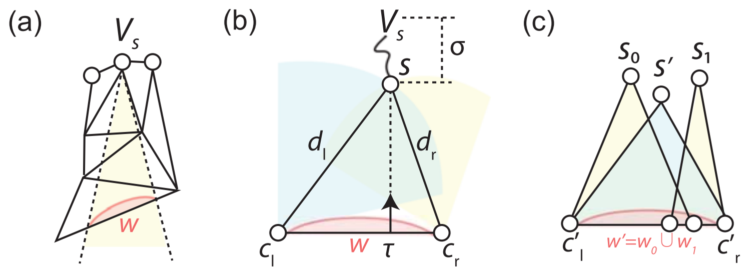

Figure 5. Illustration of computing the approximate geodesics. (a) shows the concept of the geodesic window from a single source vertex. (b) shows the components within a window. (c) shows the merging process of two overlapping windows for approximation.

process the triangles adjacent to the seams, using a propagation procedure by calculating the geodesics directly on the GPU.

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 s, the direction of the geodesic path τ, 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 τ_0, τ_2 agree with each other, and the estimated geodesic lengths are within a bounded error: $|\mathcal{D}(w_0) - \mathcal{D}(w_1)| < \varepsilon \; (\varepsilon = 0.01)$.

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.

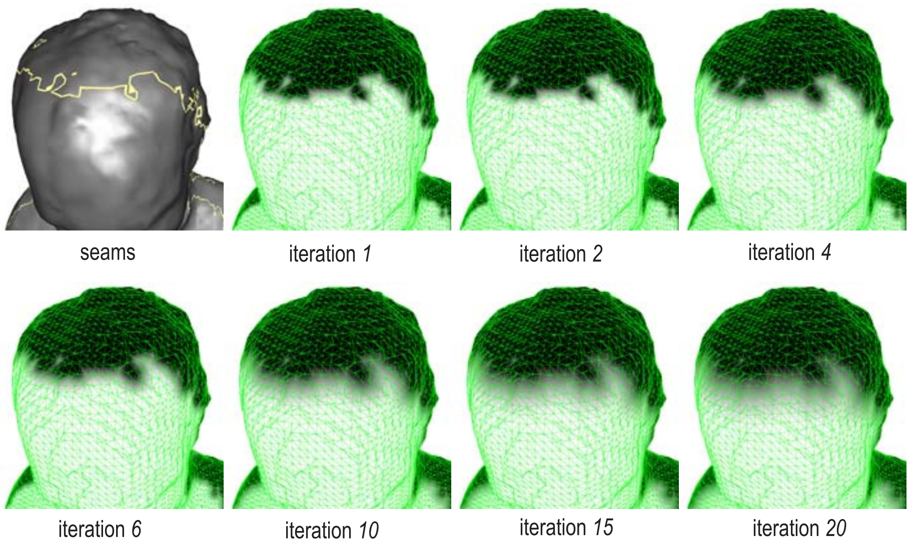

Figure 6. Examples of the initial seam triangles and the propagation process for updating the geodesic distance field.

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:

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

where, $\mathscr{V}_{\mathbf{v}}$ 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 of the i_{th} camera, and α 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 λ 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-

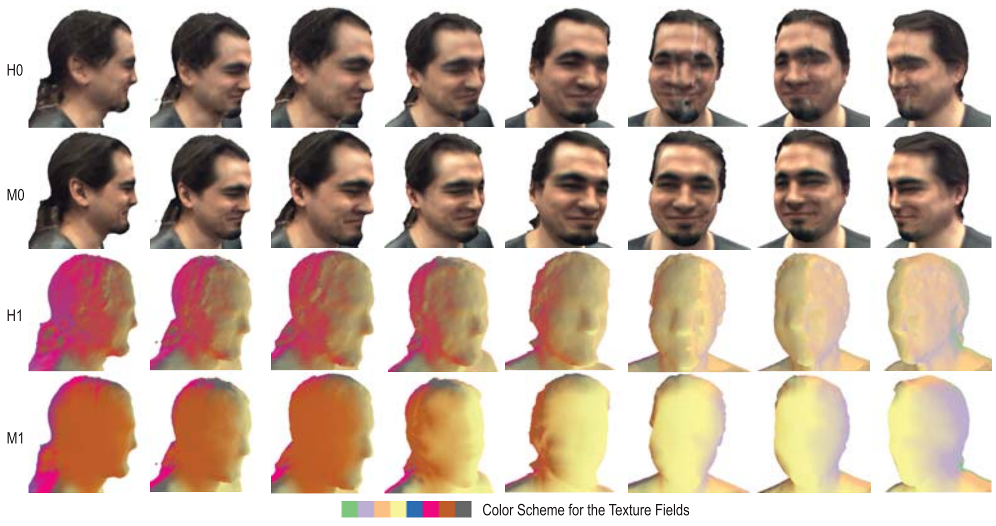

Figure 7. Spatiotemporal comparison of the rendered results (H0, M0) and corresponding texture fields (H1, M1) for *Holoportation* (H0, H1) and *Montage4D* (M0, M1) across 8 viewpoints and 40 frames. As shown in the figures, *Montage4D* takes advantage of view-dependent rendering while mitigating visible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

weighted blending to fill in the black pixels. We discuss the limitations of this compromise in Section 7. Figure 7 shows comparative results between *Holoportation* and *Montage4D*. Holoportation typically mixes three or more views for each vertex in the texture field, while Montage4D favors the frontal views and blends the seam with the help of a geodesic distance field. In Figure 8, we show an example of temporal adaptation within a fraction of a second after the user quickly switches the viewpoint from the side to the front. Note that both the texture quality and color balance transition smoothly over time to avoid abrupt changes in the rendering results.

5. Experimental Results

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. Beyond the buffers for normals, positions, and texture coordinates in the original Holoportation system, we introduce extra buffers to flag seam triangles and store the geodesic distance fields and texture weights per vertex. The seam identification stage is executed with 2^{12} blocks (2^8 threads per block, one triangle per thread) while per-vertex operations are executed with 2^{10} blocks (2^8 threads per block, one vertex per thread). Very large block sizes, such as 2^{10} threads per block, may significantly di-

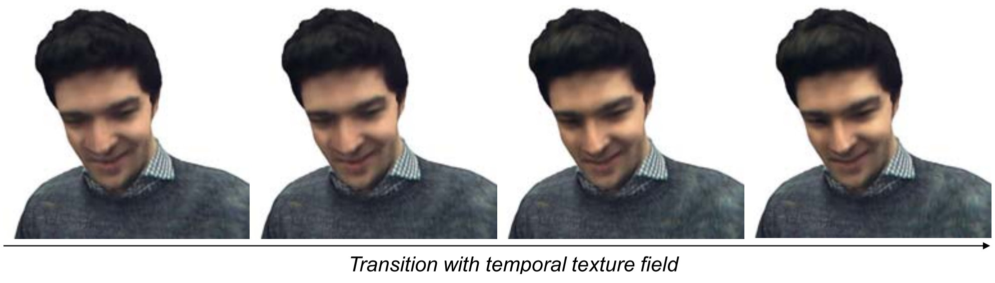

Figure 8. Temporal transition within half a second after an abrupt change in viewpoint. Note that texture quality gradually improves as the texture fields progressively favor the frontal view. In addition, the color balance adjusts slightly from cold to warm, due to the different settings of white balance in the video textures.

minish the performance. To reduce the size of the buffers, we store read-only vectors such as positions and normals as 16-bit floating numbers and decode them directly in the shaders.

5.1. Comparison with the Holoportation Approach

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 2048×2048 pixels. The average frame rate of the video textures is around 25 frames per second (FPS). The detailed statistics of the datasets is listed in Table 1. As in the *Holoportation* project, all incoming data is decompressed using the LZ4 algorithm prior to its ingestion in the rendering pipeline.

Table 1. Statistics of the datasets, including the average number of vertices and triangles per frame, the total size of the datasets (compressed with the LZ4 algorithm), and the total length of the multiview videos. Note that the average frame rate of the input videos ranges from 21 to 28 FPS, depending on the bandwidth.

dataset	#frames	#vertices / frame	#triangles / frame	size (GB)	duration (s)
Timo	837	131K	251K	5.29	33.6
Yury	803	132K	312K	5.29	32.6
Sergio	837	215K	404K	6.96	39.8
Girl	1192	173K	367K	7.43	42.4
Julien	599	157K	339K	4.52	25.5

First, we conduct a cross-validation experiment over the five datasets between

Table 2. Comparison between *Holoportation* and *Montage4D* in cross-validation experiments. Note that Montage4D outperforms the Holoportation approach while maintaining an average frame rate of over 100 Hz, which exceeds the maximum refresh rate of the current generation of VR headsets such as Oculus Rift.

Dataset	*Holoportation*				*Montage4D*			
	RMSE	PSNR	SSIM	FPS	RMSE	PSNR	SSIM	FPS
Timo	5.63%	38.60dB	0.9805	227.2	3.27%	40.23dB	0.9905	135.0
Yury	5.44%	39.20dB	0.9695	222.8	3.01%	40.52dB	0.9826	130.5
Sergio	7.74%	29.84dB	0.9704	186.8	4.21%	30.09dB	0.9813	114.3
Girl	7.16%	36.28dB	0.9691	212.56	3.73%	36.73dB	0.9864	119.4
Julien	12.63%	33.94dB	0.9511	215.18	6.71%	35.05dB	0.9697	120.6

Table 3. Timing comparison between *Holoportation* and *Montage4D* for a new frame of geometry. Geometry and textures are streamed at around 21 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

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, peak signal-to-noise ratio (PSNR), and the structural similarity (SSIM) [Wang et al. 2004][2]. The results are shown in Table 2. 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 9. We also show the input meshes and repre-

[2]A toolkit for comparing image quality: `https://github.com/ruofeidu/ImageQualityCompare`.

sentative 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 most of the misregistration 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 3, 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.

5.2. Comparison with the Floating Textures Approach

We further compare our rendering results with two other blending algorithms: Filtered Blending [Eisemann and Magnor 2007] and Floating Textures [Eisemann et al. 2008][3].

We compiled and ran the implementations of the above approaches with OpenGL 4.5 using an *NVIDIA GTX 1080* graphics card. Since their implementations are offline approaches taking several seconds per frame to load the models and textures into the GPU memory, we only compare the rendering results for static frames visually in Figure 10. We use a black background since their optical flow algorithm assumes that black represents motionless pixels.

The Filtered Blending algorithm runs smoothly at over 60 FPS, while the Floating Textures algorithm runs at between 7 and 22 FPS, depending on the target resolution ($1K \times 1K$ to $2K \times 2K$) and number of iterations within the optical flow computation (we use the default parameters, 30 iterations).

As shown in Figure 10, the Filtered Blending algorithm produces ghosting effects on human faces. The Floating Textures algorithm improves results significantly by using optical flow to address screen-space misalignment. While an elegant approach, it sometimes fails to produce visually convincing results due to several reasons:

First, the optical flow in Floating Textures is calculated in screen space instead of across the 3D mesh. As a result, view morphing leads to visual appearance errors. Second, the algorithm does not employ dilated depth maps, color voting strategies, or seam diffusion in the 3D space (the soft visibility map used by Floating Textures is a diffusion process in screen space), thereby introducing artifacts caused by inaccurate

[3]The source code is available at: https://sourceforge.net/projects/floatingtexture/

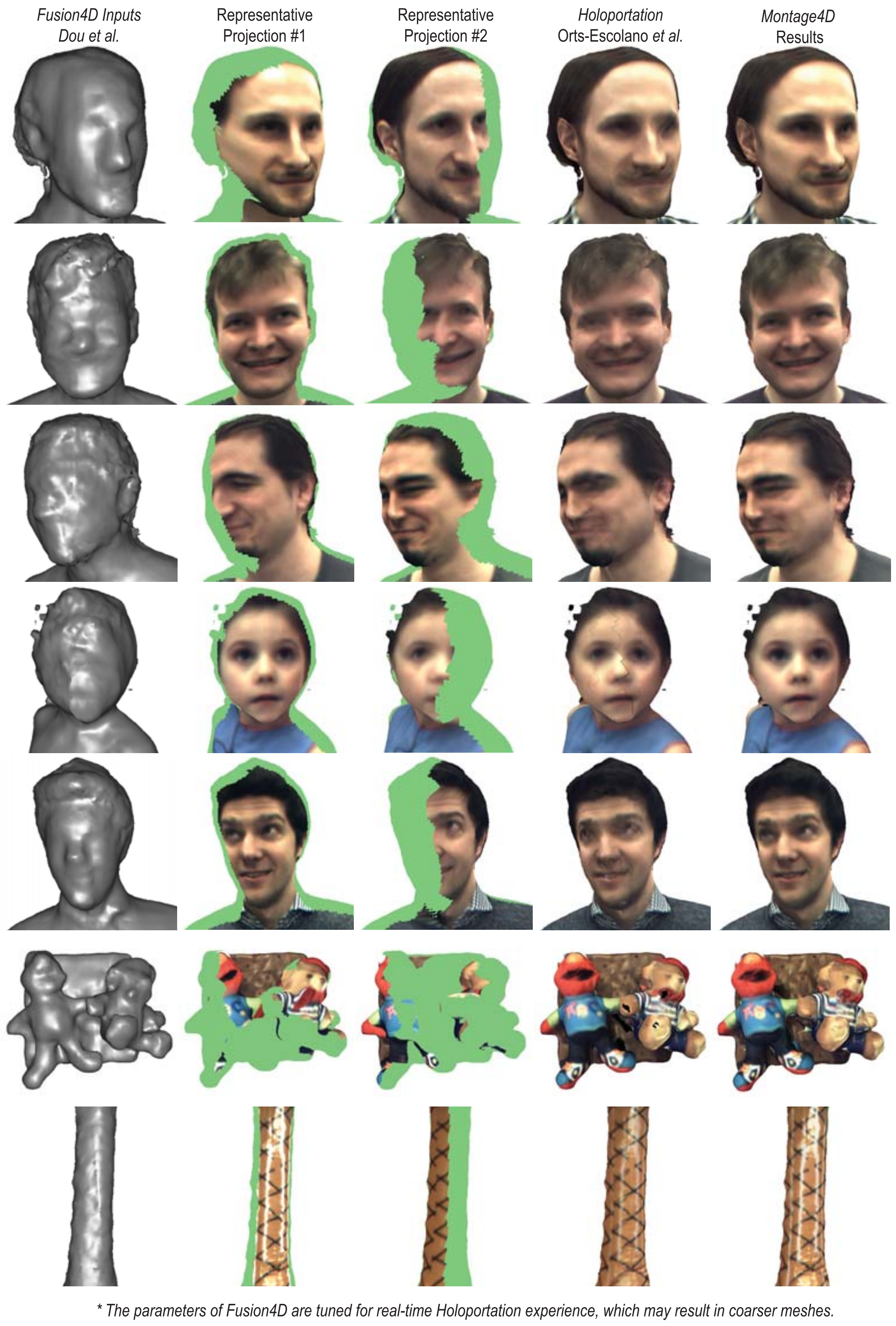

Figure 9. Comparison with the Holoportation approach. From left to right: the input mesh generated by Fusion4D, two representative back-projected images, and the rendering results from Holoportation and our Montage4D system.

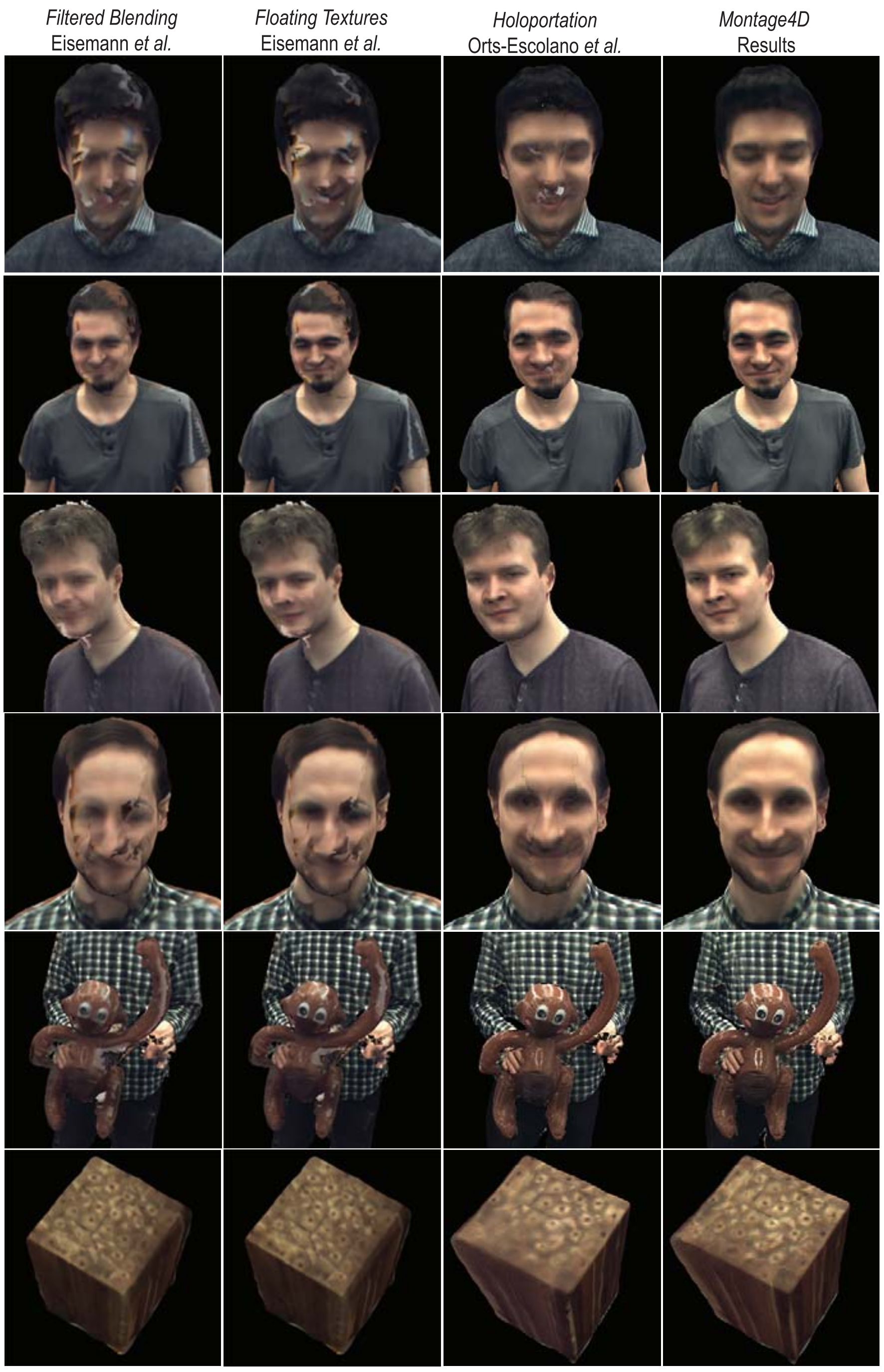

Figure 10. Comparison of different texturing algorithms. From left to right: Filtered Blending, Floating Textures, Holoportation, and our Montage4D system.

geometry. Finally, the algorithm does not take temporal coherence into account. Consequently, appearance can change significantly with an abrupt change in viewpoint. In contrast, we compute the target texture fields for each vertex, and use temporal texture fields to update these, making the temporal animation smooth.

6. Real-time Stylization

Image and video filters have been widely applied in social media platform such as Instagram and Facebook. We envision that real-time stylization may be a popular component in future live reconstruction platforms. In this section, we introduce two real-time stylization methods which enhance visual quality: sketchy stippling and relighting.

6.1. Sketchy Stippling

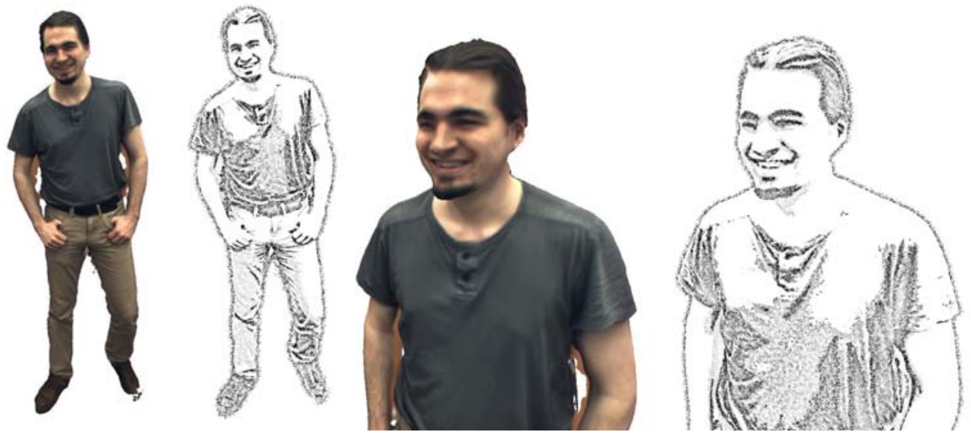

Figure 11. Results before and after applying the sketchy stippling effects.

Stippling is an artistic style which uses small dots for drawing a picture. The density of the dots naturally corresponds to the intensity of the image. Previous stippling methods have employed weighted centroidal Voronoi diagrams [Secord 2002], Voronoi relaxation [Pastor et al. 2003], or feature-guided approaches [Kim et al. 2008]. However, prior arts typically require multiples passes or hardly run in real time at a high resolution. Here, we present a real-time sketchy stippling approach, which combines both sketchy and stippling styles with only two texture lookups. Our approach only require a single screen-space postprocessing pass. We show two examples of sketchy stippling results in Figure 11.

Our sketchy stippling shader consists of four steps, as shown in Figure 12. We open source the code with detailed comments and present a live demo at ShaderToy: `https://www.shadertoy.com/view/ldSyzV`.

1. Set up the input buffer with the output from the *Montage4D* pipeline (the first texture lookup).

2. Generate a dotted blurred image of the input buffer, then invert its color space. We sample the stipples randomly based on the screen coordinates with the second texture lookup.

3. Composite the intermediate results by computing the color dodge between the input buffer and the dotted blurred image. This step preserves the stippling dots in the previous step and shades the dark regions with sketchy effects.

4. Compute the grayscale image and boost the contrast via power functions.

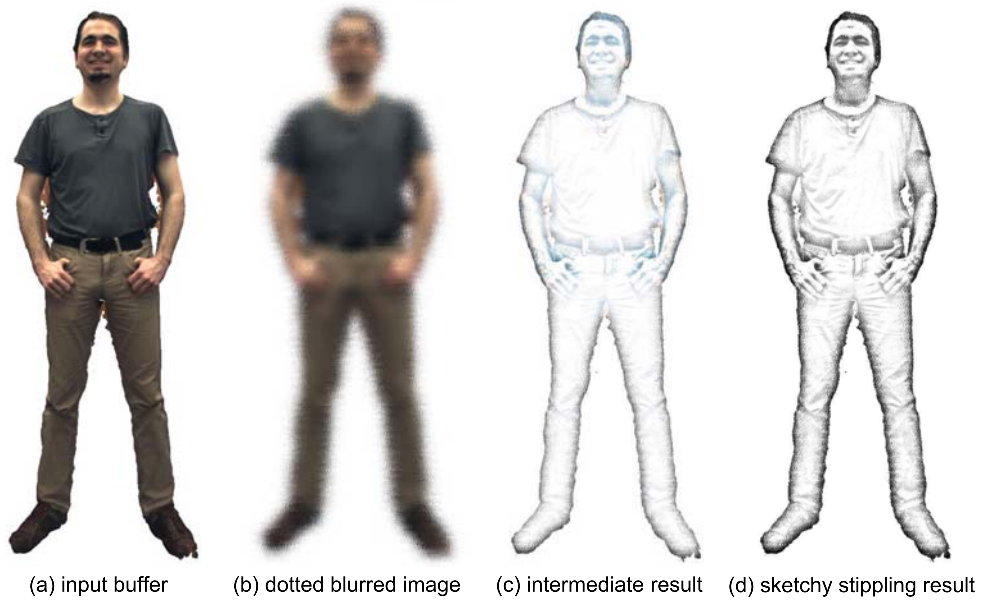

Figure 12. The step-by-step results of our sketchy stippling stylization approach with two texture lookups (one for the original buffer, the other for the blurred image).

We further tested our stylization technique with both Holoportation and Montage4D renderer. As shown in Figure 13(c) and (d), Montage4D mitigates the visual seams shown in Holoportation even with the gradient-based stylization shaders.

6.2. Relighting

When applying *Montage4D* to live telepresence, it is usually preferred that the lighting condition of the 3D models agrees with the virtual environments. Here we present our initial efforts for relighting live streamed 3D models.

First, we convert the normal vector $\hat{\mathbf{n}}$ per vertex to the normal vector in the view-space $\hat{\mathbf{n}}_{xy}$, where $\mathbf{V}^{-1}\mathbf{M}^{-1}$ is the inverse model view matrix:

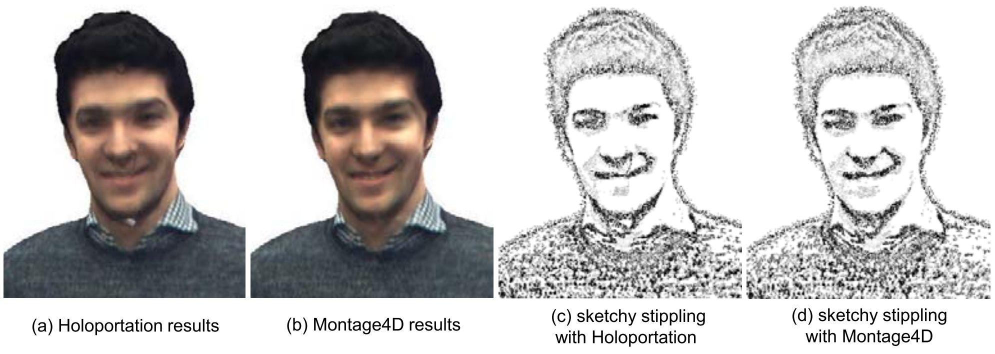
(a) Holoportation results
(b) Montage4D results
(c) sketchy stippling with Holoportation
(d) sketchy stippling with Montage4D

Figure 13. Stylization with the original Holoportation renderer suffers from visual seams, as shown near the nose and mouth in (c); Montage4D mitigates such artifacts in (d).

$$\hat{\mathbf{n}}_{xy} = \mathbf{V}^{-1}\mathbf{M}^{-1}\hat{\mathbf{n}} \tag{6}$$

Next, we sample the light L from a precomputed light probe buffer P, as shown in Figure 14. In practice, the light probe buffer can be reconstructed from the first three orders of the spherical harmonics coefficients [Green 2003], which are sampled from the environment maps.

$$\mathbf{L} = \text{texture}\left(P, \hat{\mathbf{n}}_{xy}\right) \tag{7}$$

Finally, for each pixel, we use a non-linear blending approach (Equation 8) to mix the final color $\mathbf{F}$ with the light color $\mathbf{L}$ and the texture color $\mathbf{T}$. Let $C_{\mathbf{X}}$ be one of the red, green, and blue channels in color $\mathbf{X}$. We lighten the model if $C_{\mathbf{L}}$ has a high intensity and darken the model if $C_{\mathbf{L}}$ has a low intensity. The resulting color is unaffected if $C_{\mathbf{L}}$ is of 50% intensity. We use a power function to lighten or darken the model. A simple demo is presented at `https://www.shadertoy.com/view/XldcDM`.

$$C_{\mathbf{F}} = C_{\mathbf{T}} + |1 - 2C_{\mathbf{L}}|\,(C_{\mathbf{T}}^{1.5-C_{\mathbf{L}}} - C_{\mathbf{T}}), C \in \{\text{Red, Green, Blue}\} \tag{8}$$

For future work, if the intrinsic color of each vertex could be computed in real time, a simple multiplication of the reflectance and shading will composite the resulting color. One may take advantage of intrinsic video [Meka et al. 2016] and material estimation [Meka et al. 2018] to achieve this.

7. 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.

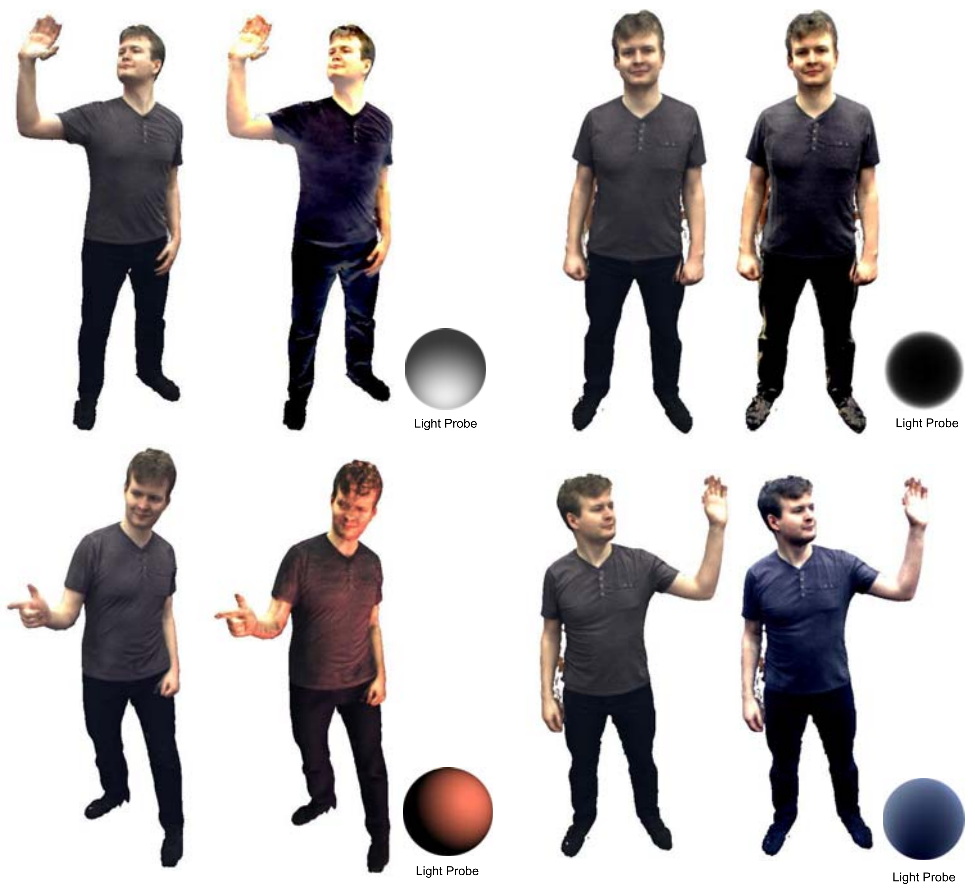

Figure 14. Results before and after the interactive relighting pass with different light probes. Note that the black trousers and shoes are lightened with more details in the top row, while the shirts are lightened with different colors in the bottom row.

7.1. Inaccurate Geometries

As shown in Figure 15(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.

7.2. Missing Texture Fields

Figure 15(b) shows the challenging issue caused by insufficient reliable colors. Such problems may be solved by user-guided in-painting 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.

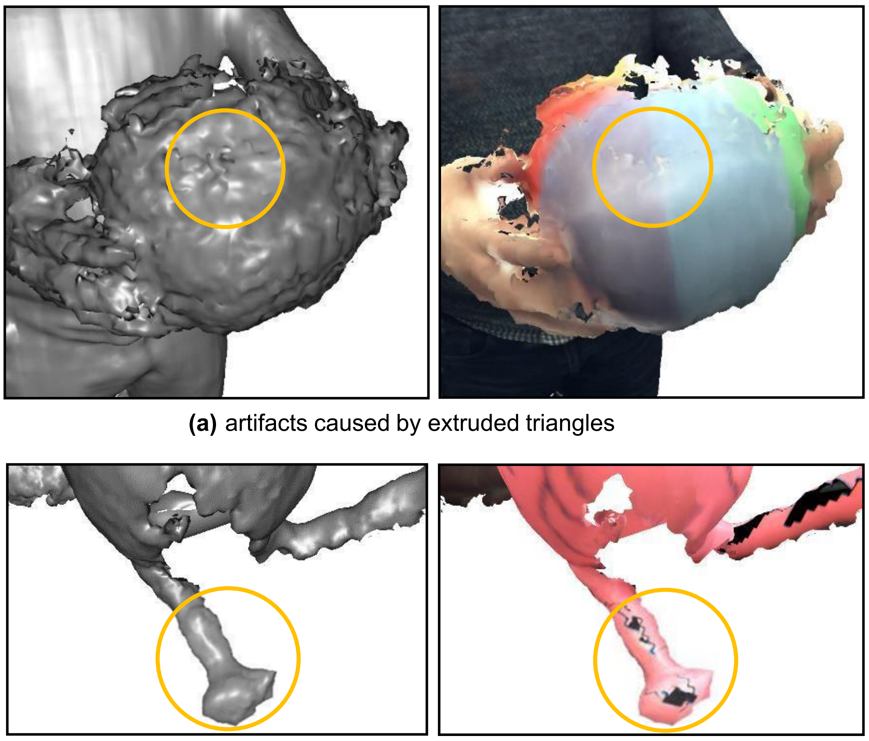
(a) artifacts caused by extruded triangles
(b) holes caused by insufficient reliable colors

Figure 15. Limitations of our approach. Extruded triangles and highly-occluded spots may still cause artifacts.

8. 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, we have integrated the Montage4D pipeline with the Mobile Holoportation[4] project.

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

[4]Mobile Holoportation: `https://www.microsoft.com/en-us/research/project/holoportation-3`

flow algorithms over the mesh surface [Prada et al. 2016; Prada et al. 2017b] to further optimize the texture fields. As discussed in Section 5, we identify some challenges with the screen-based optical flow approach [Eisemann and Magnor 2007; Eisemann et al. 2008]. 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 [Shirley et al. 2011; Chajdas et al. 2011; Mavridis and Papaioannou 2011; Heitz et al. 2013; Crassin et al. 2015], or Poisson blending [Pérez et al. 2003] over the 3D space [Chuang et al. 2009] to eliminate the artifacts. To further increase the frame rate with small camera movement, it may be desirable to morph multiview textures in image using with z-buffers [Darsa et al. 1997; Bista et al. 2017]. In addition, we would like to investigate adaptive and efficient optical flow algorithms over the mesh surface [Prada et al. 2016]. With recent advances in deep neural networks, future work may investigate how to design and train self-supervised deep architectures [Martin-Brualla et al. 2018] to render novel views from fewer cameras while preserving high-frequency details. Such systems should also focus on generalization with a limited amount of training data.

We envision our algorithm to be useful for many virtual and augmented reality applications, such as remote business meetings, medical training, and mixed reality social media platforms [Du 2018].

Acknowledgements

We appreciate the entire Holoportation Team including Shahram Izadi, Sameh Khamis, Mingsong Dou, Yury Degtyarev, Philip Davidson, Sean Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, Marek Kolwalski, Qiyu (Zoey) 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

ALLÈNE, C., PONS, J.-P., AND KERIVEN, R. 2008. Seamless Image-Based Texture Atlases Using Multi-Band Blending. In *19th International Conference on Pattern Recognition*, IEEE, 1–4. URL: `http://doi.org/10.1109/ICPR.2008.4761913`. 6

ALLIEZ, P., MEYER, M., AND DESBRUN, M. 2002. Interactive Geometry Remeshing. *ACM Transactions on Graphics (TOG) 21*, 3, 347–354. URL: `http://doi.org/10.1145/566570.566588`. 23

BISTA, S., DA CUNHA, I. L. L., AND VARSHNEY, A. 2017. Kinetic Depth Images: Flexible Generation of Depth Perception. *The Visual Computer 33*, 10 (October), 1357–1369. URL: http://doi.org/10.1007/s00371-016-1231-2. 25

BOMMES, D., AND KOBBELT, L. 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. 3, 7

BOUKHAYMA, A., AND BOYER, E. 2018. Surface Motion Capture Animation Synthesis. *IEEE Transactions on Visualization and Computer Graphics*. URL: http://doi.org/10.1109/TVCG.2018.2831233. 4

CAGNIART, C., BOYER, E., AND ILIC, S. 2010. Probabilistic Deformable Surface Tracking From Multiple Videos. In *ECCV'10 Proceedings of the 11th European Conference on Computer Vision: Part IV*, Springer, Springer, 326–339. URL: http://doi.org/10.1007/978-3-642-15561-_24. 4

CASAS, D., TEJERA, M., GUILLEMAUT, J.-Y., AND HILTON, A. 2013. Interactive Animation of 4D Performance Capture. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 19*, 5, 762–773. URL: http://doi.org/10.1145/2159616.2159633. 4

CASAS, D., VOLINO, M., COLLOMOSSE, J., AND HILTON, A. 2014. 4d Video Textures for Interactive Character Appearance. *Computer Graphics Forum 33*, 2, 371–380. URL: http://doi.org/10.1145/2775292.2775297. 6

CHAJDAS, M. G., MCGUIRE, M., AND LUEBKE, D. 2011. Subpixel Reconstruction Antialiasing for Deferred Shading. In *Proceedings of Symposium on Interactive 3D Graphics and Games (I3D)*, ACM, 15–22. URL: http://doi.org/10.1145/1944745.1944748. 25

CHUANG, M., LUO, L., BROWN, B. J., RUSINKIEWICZ, S., AND KAZHDAN, M. 2009. Estimating the Laplace-Beltrami Operator by Restricting 3D Functions. *Computer Graphics Forum 28*, 5, 1475–1484. URL: http://doi.org/1735633. 25

COLLET, A., CHUANG, M., SWEENEY, P., GILLETT, D., EVSEEV, D., CALABRESE, D., HOPPE, H., KIRK, A., AND SULLIVAN, S. 2015. High-Quality Streamable Free-Viewpoint Video. *ACM Transactions on Graphics (TOG) 34*, 4, 69. URL: http://doi.org/10.1145/2766945. 2, 4, 5, 6

CRASSIN, C., MCGUIRE, M., FATAHALIAN, K., AND LEFOHN, A. 2015. Aggregate G-Buffer Anti-Aliasing. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games (I3D)*, ACM, 109–119. URL: http://doi.org/10.1145/2699276.2699285. 25

DARSA, L., COSTA, B., AND VARSHNEY, A. 1997. Navigating Static Environments Using Image-Space Simplification and Morphing. In *Proceedings of the Symposium on 3D Interactive Graphics*, I3D, ACM, 25 – 34. URL: http://doi.org/10.1145/253284.253298. 25

DE AGUIAR, E., STOLL, C., THEOBALT, C., AHMED, N., SEIDEL, H.-P., AND THRUN, S. 2008. Performance Capture From Sparse Multi-View Video. *ACM Transactions on Graphics (TOG) 27*, 3, 98. URL: http://doi.org/10.1145/1399504.1360697. 4

DEBEVEC, P., GORTLER, S., MCMILLAN, L., SZELISKI, R., AND BREGLER, C. 1998. Image-Based Modeling and Rendering. *SIGGRAPH 98 Course Notes for Course 15*. URL: http://doi.org/10.1109/VR.2003.1191174. 5

DEBEVEC, P., YU, Y., AND BORSHUKOV, G. 1998. Efficient View-Dependent Image-Based Rendering With Projective Texture-Mapping. In *Rendering Techniques*. Springer, 105–116. URL: http://doi.org/10.1007/978-3-7091-6453-_10. 5

DO GOES, F., DESBRUN, M., AND TONG, Y. 2015. Vector Field Processing on Triangle Meshes. In *SIGGRAPH Asia 2015 Courses*, ACM, 17. URL: http://doi.org/10.1145/2818143.2818167. 6

DOU, M., KHAMIS, S., DEGTYAREV, Y., DAVIDSON, P., FANELLO, S. R., KOWDLE, A., ESCOLANO, S. O., RHEMANN, C., KIM, D., TAYLOR, J., ET AL. 2016. Fusion4D: Real-Time Performance Capture of Challenging Scenes. *ACM Transactions on Graphics (TOG) 35*, 4, 114. URL: http://doi.org/10.1145/2897824.2925969. 2, 5, 8

DOU, M., DAVIDSON, P., FANELLO, S. R., KHAMIS, S., KOWDLE, A., RHEMANN, C., TANKOVICH, V., AND IZADI, S. 2017. Motion2fusion: Real-Time Volumetric Performance Capture. *ACM Transactions on Graphics (TOG) 36*, 6, 246. URL: http://doi.org/10.1145/3130800.3130801. 5, 23

DU, R., BISTA, S., AND VARSHNEY, A. 2016. Video Fields: Fusing Multiple Surveillance Videos Into a Dynamic Virtual Environment. In *Proceedings of the 21st International Conference on Web3D Technology*, ACM, Web3D, ACM, 165–172. URL: http://doi.org/10.1145/2945292.2945299. 4

DU, R., CHUANG, M., CHANG, W., HOPPE, H., AND VARSHNEY, A. 2018. Montage4D: Interactive Seamless Fusion of Multiview Video Textures. In *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*, ACM, I3D, ACM, 124–133. URL: http://doi.org/10.1145/3190834.3190843. 3

DU, R. 2018. *Fusing Multimedia Data Into Dynamic Virtual Environments*. PhD thesis, University of Maryland, Colelge Park. 25

EISEMANN, M., AND MAGNOR, M. A. 2007. Filtered Blending: a New, Minimal Reconstruction Filter for Ghosting-Free Projective Texturing With Multiple Images. In *VMV*, VMV, 119–126. URL: https://graphics.tu-bs.de/publications/Eisemann07FB. 17, 25

EISEMANN, M., DE DECKER, B., MAGNOR, M., BEKAERT, P., DE AGUIAR, E., AHMED, N., THEOBALT, C., AND SELLENT, A. 2008. Floating Textures. *Computer Graphics Forum 27*, 2, 409–418. URL: http://doi.org/10.1111/j.1467-8659.2008.01138.x. 6, 17, 25

FANELLO, S. R., VALENTIN, J., KOWDLE, A., RHEMANN, C., TANKOVICH, V., CILIBERTO, C., DAVIDSON, P., AND IZADI, S. 2017. Low Compute and Fully Parallel Computer Vision With HashMatch. In *IEEE International Conference on Computer Vision (ICCV)*, IEEE, 3894–3903. URL: http://doi.org/10.1109/ICCV.2017.418. 5

FANELLO, S. R., VALENTIN, J., RHEMANN, C., KOWDLE, A., TANKOVICH, V., DAVIDSON, P., AND IZADI, S. 2017. UltraStereo: Efficient Learning-Based Matching for Active Stereo Systems. In *2017 IEEE Conference OnComputer Vision and Pattern Recognition (CVPR)*, IEEE, 6535–6544. URL: `http://doi.org/10.1109/CVPR.2017.692`. 5

FUCHS, H., AND NEUMANN, U. 1993. A Vision of Telepresence for Medical Consultation and Other Applications. In *Sixth International Symposium of Robotics Research*, IFRR, 555–571. URL: `http://doi.org/10.1023/A`. 4

FUCHS, H., BISHOP, G., ARTHUR, K., MCMILLAN, L., BAJCSY, R., LEE, S., FARID, H., AND KANADE, T. 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. URL: `http://doi.org/10.1145/1026776.1026790`. 4

FURUKAWA, Y., AND PONCE, J. 2008. Dense 3D Motion Capture From Synchronized Video Streams. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 1–8. URL: `http://doi.org/10.1109/CVPR.2009.5206868`. 4

GAL, R., WEXLER, Y., OFEK, E., HOPPE, H., AND COHEN-OR, D. 2010. Seamless Montage for Texturing Models. *Computer Graphics Forum 29*, 2, 479–486. URL: `http://doi.org/10.1111/j.1467-8659.2009.01617.x`. 5

GOLDLUECKE, B., AND MAGNOR, M. 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. URL: `http://doi.org/10.1109/CVPR.2004.1315053`. 4

GREEN, R. 2003. Spherical Harmonic Lighting: the Gritty Details. In *Archives of the Game Developers Conference*, vol. 56, GDC, 4. 22

GRINSPUN, E., DESBRUN, M., POLTHIER, K., SCHRÖDER, P., AND STERN, A. 2006. Discrete Differential Geometry: an Applied Introduction. *ACM SIGGRAPH Course 7*, 1–139. URL: `http://doi.org/10.1145/3245634`. 6

GUO, K., XU, F., WANG, Y., LIU, Y., AND DAI, Q. 2015. Robust Non-Rigid Motion Tracking and Surface Reconstruction Using L0 Regularization. In *Proceedings of the IEEE International Conference on Computer Vision*, IEEE, IEEE, 3083–3091. URL: `http://doi.org/10.1109/ICCV.2015.353`. 4

GUO, K., XU, F., YU, T., LIU, X., DAI, Q., AND LIU, Y. 2017. Real-Time Geometry, Albedo, and Motion Reconstruction Using a Single RGB-D Camera. *ACM Transactions on Graphics (TOG) 36*, 3, 32. URL: `http://doi.org/10.1145/3083722`. 4

GUO, K., TAYLOR, J., FANELLO, S., TAGLIASACCHI, A., DOU, M., DAVIDSON, P., KOWDLE, A., AND IZADI, S. 2018. TwinFusion: High Framerate Non-Rigid Fusion Through Fast Correspondence Tracking. In *2018 International Conference on 3D Vision (3DV)*, IEEE, 596–605. 5

HEITZ, E., NOWROUZEZAHRAI, D., POULIN, P., AND NEYRET, F. 2013. Filtering Color Mapped Textures and Surfaces. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*, ACM, 129–136. URL: `http://doi.org/2448196`. 25

HUANG, Z., LI, T., CHEN, W., ZHAO, Y., XING, J., LEGENDRE, C., LUO, L., MA, C., AND LI, H. 2018. Deep Volumetric Video From Very Sparse Multi-View Performance Capture. In *Proceedings of the European Conference on Computer Vision*, ECCV, 336–354. URL: `http://doi.org/10.1145/1399504.1360697`. 4

IZADI, S., KIM, D., HILLIGES, O., MOLYNEAUX, D., NEWCOMBE, R., KOHLI, P., SHOTTON, J., HODGES, S., FREEMAN, D., DAVISON, A., AND FITZGIBBON, A. 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*, ACM, 559–568. URL: `http://doi.org/10.1109/ISMAR.2011.6092378`. 2, 4

JANKO, Z., AND PONS, J.-P. 2009. Spatio-Temporal Image-Based Texture Atlases for Dynamic 3-D Models. In *IEEE 12th International Conference on Computer Vision Workshops*, IEEE, 1646–1653. URL: `http://doi.org/10.1109/ICCVW.2009.5457481`. 6

KANADE, T., RANDER, P., AND NARAYANAN, P. 1997. Virtualized Reality: Constructing Virtual Worlds From Real Scenes. *IEEE Multimedia 4*, 1, 34–47. URL: `http://doi.org/10.1109/93.580394`. 4

KANAI, T., AND SUZUKI, H. 2001. Approximate Shortest Path on a Polyhedral Surface and Its Applications. *Computer-Aided Design 33*, 11, 801–811. URL: `http://doi.org/10.1145/1044731.1044733`. 7

KAPOOR, S. 1999. Efficient Computation of Geodesic Shortest Paths. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, ACM, 770–779. URL: `http://doi.org/10.1145/301250.301449`. 7

KIM, D., SON, M., LEE, Y., KANG, H., AND LEE, S. 2008. Feature-Guided Image Stippling. *Computer Graphics Forum 27*, 4, 1209–1216. URL: `http://doi.org/10.1111/j.1467`. 20

LANTHIER, M., MAHESHWARI, A., AND SACK, J.-R. 1997. Approximating Weighted Shortest Paths on Polyhedral Surfaces. In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*, ACM, 274–283. URL: `http://doi.org/10.1145/262839.263101`. 7

LEMPITSKY, V., AND IVANOV, D. 2007. Seamless Mosaicing of Image-Based Texture Maps. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 1–6. URL: `http://doi.org/10.1109/CVPR.2007.383078`. 6

LOK, B. 2001. Online Model Reconstruction for Interactive Virtual Environments. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics (I3D)*, ACM, 69–72. URL: `http://doi.org/10.1145/364338.364364`. 4

MARTIN-BRUALLA, R., PANDEY, R., YANG, S., PIDLYPENSKYI, P., TAYLOR, J., VALENTIN, J., KHAMIS, S., DAVIDSON, P., TKACH, A., LINCOLN, P., KOWDLE, A., RHEMANN, C., GOLDMAN, D. B., KESKIN, C., SEITZ, S., IZADI, S., AND FANELLO, S. 2018. Lookingood: Enhancing performance capture with real-time neural re-rendering. In *SIGGRAPH Asia 2018 Technical Papers*, ACM, New York, NY, USA, SIGGRAPH Asia '18, ACM, 255:1–255:14. URL: http://doi.acm.org/10.1145/3272127.3275099. 25

MARTINEZ, D., VELHO, L., AND CARVALHO, P. C. 2004. Geodesic Paths on Triangular Meshes. In *Proceedings of Computer Graphics and Image Processing*, IEEE, 8. URL: http://doi.org/10.1109/SIBGRA.2004.1352963. 7

MAVRIDIS, P., AND PAPAIOANNOU, G. 2011. High Quality Elliptical Texture Filtering on GPU. In *Symposium on Interactive 3D Graphics and Games (I3D)*, ACM, 23–30. URL: http://doi.org/10.1145/1944745.1944749. 25

MEKA, A., ZOLLHÖFER, M., RICHARDT, C., AND THEOBALT, C. 2016. Live Intrinsic Video. *ACM Transactions on Graphics (TOG) 35*, 4, 109. URL: http://doi.org/10.1145/2897824.2925907. 22

MEKA, A., MAXIMOV, M., ZOLLHÖFER, M., CHATTERJEE, A., RICHARDT, C., AND THEOBALT, C. 2018. Live Intrinsic Material Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE. 22

MITCHELL, J. S., MOUNT, D. M., AND PAPADIMITRIOU, C. H. 1987. The Discrete Geodesic Problem. *SIAM Journal on Computing 16*, 4, 647–668. URL: http://doi.org/10.1145/1559755.1559761. 7

MITCHELL, J. S. 2000. Geometric Shortest Paths and Network Optimization. *Handbook of Computational Geometry 334*, 633–702. URL: http://doi.org/10.1007/978-1-4613-0303-_10. 6

NARAYAN, K. S., AND ABBEEL, P. 2015. Optimized Color Models for High-Quality 3D Scanning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2503–2510. URL: http://doi.org/10.1109/IROS.2015.7353717. 6

NARAYANAN, P., RANDER, P. W., AND KANADE, T. 1998. Constructing Virtual Worlds Using Dense Stereo. In *Sixth International Conference on Computer Vision (ICCV)*, IEEE, 3–10. URL: http://doi.org/939185. 4

NEWCOMBE, R. A., IZADI, S., HILLIGES, O., MOLYNEAUX, D., KIM, D., DAVISON, A. J., KOHI, P., SHOTTON, J., HODGES, S., AND FITZGIBBON, A. 2011. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, IEEE, 127–136. URL: http://doi.org/10.1109/ISMAR.2011.6092378. 4

NEWCOMBE, R. A., FOX, D., AND SEITZ, S. M. 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. URL: http://doi.org/10.1109/CVPR.2015.7298631. 2, 4

Orts-Escolano, S., Rhemann, C., Fanello, S., Chang, W., Kowdle, A., Degtyarev, Y., Kim, D., Davidson, P. L., Khamis, S., Dou, M., 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. URL: http://doi.org/10.1145/2984511.2984517. 2, 5

Pastor, O. M., Freudenberg, B., and Strothotte, T. 2003. Real-Time Animated Stippling. *IEEE Computer Graphics and Applications 23*, 4, 62–68. URL: http://doi.org/10.1109/MCG.2003.1210866. 20

Patro, R., Ip, C. Y., Bista, S., and Varshney, A. 2011. Social Snapshot: a System for Temporally Coupled Social Photography. *IEEE Computer Graphics and Applications 31*, 1, 74–84. URL: http://doi.org/10.1109/MCG.2010.107. 4

Pérez, P., Gangnet, M., and Blake, A. 2003. Poisson Image Editing. *ACM Transactions on Graphics (TOG) 22*, 3, 313–318. URL: http://doi.org/10.1145/882262.882269. 25

Prada, F., Kazhdan, M., Chuang, M., Collet, A., and Hoppe, H. 2016. Motion Graphs for Unstructured Textured Meshes. *ACM Transactions on Graphics (TOG) 35*, 4, 108. URL: http://doi.org/10.1145/2897824.2925967. 4, 5, 25

Prada, F., Kazhdan, M., Chuang, M., Collet, A., and Hoppe, H. 2017. Spatiotemporal Atlas Parameterization for Evolving Meshes. *ACM Transactions on Graphics (TOG) 36*, 4, 58. URL: http://doi.org/10.1145/3072959.3073679. 4, 5

Prada, F., Kazhdan, M., Chuang, M., Collet, A., and Hoppe, H. 2017. Spatiotemporal Atlas Parameterization for Evolving Meshes. *ACM Transactions on Graphics (TOG) 36*, 4, 58. URL: http://doi.org/10.1145/3072959.3073679. 25

Qin, Y., Han, X., Yu, H., Yu, Y., and Zhang, J. 2016. Fast and Exact Discrete Geodesic Computation Based on Triangle-Oriented Wavefront Propagation. *ACM Transactions on Graphics (TOG) 35*, 4, 13. URL: http://doi.org/10.1145/2897824.2925930. 7

Qu, L., and Meyer, G. W. 2006. Perceptually Driven Interactive Geometry Remeshing. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games (I3D)*, ACM, 199–206. URL: http://doi.org/10.1145/1111411.1111447. 23

Sankaranarayanan, A. C., Patro, R., Turaga, P., Varshney, A., and Chellappa, R. 2009. Modeling and Visualization of Human Activities for Multicamera Networks. *EURASIP Journal on Image and Video Processing 2009*, 259860. URL: http://doi.org/10.1016/j.patrec.2012.07.005. 4

Secord, A. 2002. Weighted Voronoi Stippling. In *Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering*, ACM, 37–43. URL: http://doi.org/10.1145/508530.508537. 20

Shirley, P., Aila, T., Cohen, J., Enderton, E., Laine, S., Luebke, D., and McGuire, M. 2011. A Local Image Reconstruction Algorithm for Stochastic Rendering. In *Symposium on Interactive 3D Graphics and Games (I3D)*, ACM, PAGE–5. URL: http://doi.org/10.1145/1944745.1944747. 25

SURAZHSKY, V., SURAZHSKY, T., KIRSANOV, D., GORTLER, S. J., AND HOPPE, H. 2005. Fast Exact and Approximate Geodesics on Meshes. *ACM Transactions on Graphics (TOG) 24*, 3, 553–560. URL: http://doi.org/10.1145/1186822.1073228. 7, 12

TANG, D., MINGSONG DOU, P. L., DAVIDSON, P., GUO, K., TALYOR, J., FANELLO, S., KESKIN, C., KOWDLE, A., BOUAZIZ, S., IZADI, S., AND TAGLIASACCHI, A. 2018. Real-Time Compression and Streaming of 4D Performance. *ACM Transactions on Graphics (TOG)*. 3

TOWLES, H., CHEN, W.-C., YANG, R., KUM, S.-U., KELSHIKAR, H. F. N., MULLIGAN, J., DANIILIDIS, K., FUCHS, H., HILL, C. C., MULLIGAN, N. K. J., ET AL. 2002. 3D Tele-Collaboration Over Internet2. In *International Workshop on Immersive Telepresence*, Juan Les Pins, France, 6. URL: http://doi.org/10.1109/ICDCS.2008.47. 4

VINEET, V., WARRELL, J., AND TORR, P. H. 2014. Filter-Based Mean-Field Inference for Random Fields With Higher-Order Terms and Product Label-Spaces. *International Journal of Computer Vision 110*, 3, 290–307. URL: http://doi.org/10.1007/s11263. 8

VLASIC, D., BARAN, I., MATUSIK, W., AND POPOVIĆ, J. 2008. Articulated Mesh Animation From Multi-View Silhouettes. *ACM Transactions on Graphics (TOG) 27*, 3, 97. URL: http://doi.org/10.1145/1399504.1360696. 4

VOLINO, M., CASAS, D., COLLOMOSSE, J. P., AND HILTON, A. 2014. Optimal Representation of Multi-View Video. In *Proceedings of the British Machine Vision Conference*, BMVA Press, BMVC. URL: http://doi.org/10.1109/ICC.2017.7996612. 6

WANG, Z., BOVIK, A. C., SHEIKH, H. R., AND SIMONCELLI, E. P. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing 13*, 4, 600–612. URL: http://doi.org/10.1109/TIP.2003.819861. 16

XU, F., LIU, Y., STOLL, C., TOMPKIN, J., BHARAJ, G., DAI, Q., SEIDEL, H.-P., KAUTZ, J., AND THEOBALT, C. 2011. Video-Based Characters: Creating New Human Performances From a Multi-View Video Database. *ACM Transactions on Graphics (TOG) 30*, 4, 32. URL: http://doi.org/10.1145/1964921.1964927. 4

XU, W., SALZMANN, M., WANG, Y., AND LIU, Y. 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. URL: http://doi.org/10.1109/ICCV.2015.252. 4

YE, M., AND YANG, R. 2014. Real-Time Simultaneous Pose and Shape Estimation for Articulated Objects Using a Single Depth Camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2345–2352. URL: http://doi.org/10.1109/CVPR.2014.301. 4

YU, T., GUO, K., XU, F., DONG, Y., SU, Z., ZHAO, J., LI, J., DAI, Q., AND LIU, Y. 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, ACM. URL: http://doi.org/10.1109/ICCV.2017.104. 4

ZHANG, Q., FU, B., YE, M., AND YANG, R. 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. URL: `http://doi.org/10.1109/CVPR.2014.92`. 4

ZHOU, Q.-Y., AND KOLTUN, V. 2014. Color Map Optimization for 3D Reconstruction With Consumer Depth Cameras. *ACM Transactions on Graphics (TOG) 33*, 4, 155. URL: `http://doi.org/10.1145/2601097.2601134`. 6

ZHOU, K., WANG, X., TONG, Y., DESBRUN, M., GUO, B., AND SHUM, H.-Y. 2005. TextureMontage: Seamless Texturing of Arbitrary Surfaces From Multiple Images. *ACM Transactions on Graphics (TOG) 24*, 3, 1148–1155. URL: `http://doi.org/10.1145/1073204.1073325`. 6, 23

ZITNICK, C. L., KANG, S. B., UYTTENDAELE, M., WINDER, S., AND SZELISKI, R. 2004. High-Quality Video View Interpolation Using a Layered Representation. *ACM Transactions on Graphics (TOG) 23*, 3, 600–608. URL: `http://doi.org/10.1145/1015706.1015766`. 4

Index of Supplemental Materials

Videos, slides, and shader code for stylization are published at `www.montage4d.com`.

Author Contact Information

Ruofei Du
University of Maryland, College Park.
me@duruofei.com
http://www.duruofei.com

Ming Chuang
PerceptIn Inc.
mingchuang82@gmail.com
http://www.cs.jhu.edu/˜ming

Wayne Chang
Microsoft Research.
wechang@microsoft.com
http://microsoft.com/research/people/wechang

Hugues Hoppe
Google LLC.
hhoppe@gmail.com
http://hhoppe.com

Amitabh Varshney
University of Maryland, College Park.
varshney@umiacs.umd.edu
http://cs.umd.edu/˜varshney

Ruofei Du, Ming Chuang, Wayne Chang, Hugues Hoppe, Amitabh Varshney, Montage4D: Real-time Seamless Fusion and Stylization of Multiview Video Textures, *Journal of Computer Graphics Techniques (JCGT)*, vol. 8, no. 1, 1–1, 2019
http://jcgt.org/published/0003/01/01/

Received:	2018-08-01		
Recommended:	2018-11-09	Corresponding Editor:	Marc Olano
Published:	2019-1-14	Editor-in-Chief:	Marc Olano