Motion2Fusion: Real-time Volumetric Performance Capture

MINGSONG DOU, PHILIP DAVIDSON*, SEAN RYAN FANELLO*, SAMEH KHAMIS*, ADARSH KOWDLE*, CHRISTOPH RHEMANN*, VLADIMIR TANKOVICH*, and SHAHRAM IZADI,

perceptiveIO

Fig. 1. 3D Captures with Our System. *We propose a new volumetric performance capture pipeline that is robust to fast motions and topology changes and is capable of reconstructing challenging scenes at 3x the speed of the previous state-of-the-art.*

We present Motion2Fusion, a state-of-the-art 360 performance capture system that enables *real-time* reconstruction of arbitrary non-rigid scenes. We provide three major contributions over prior work: 1) a new non-rigid fusion pipeline allowing for far more faithful reconstruction of high frequency geometric details, avoiding the over-smoothing and visual artifacts observed previously. 2) a high speed pipeline coupled with a machine learning technique for 3D correspondence field estimation reducing tracking errors and artifacts that are attributed to fast motions. 3) a backward and forward non-rigid alignment strategy that more robustly deals with topology changes but is still free from scene priors. Our novel performance capture system demonstrates real-time results nearing 3x speed-up from previous state-of-the-art work on the exact same GPU hardware. Extensive quantitative and qualitative comparisons show more precise geometric and texturing results with less artifacts due to fast motions or topology changes than prior art.

CCS Concepts: • **Computing methodologies** → **Motion capture**; *Reconstruction*;

Additional Key Words and Phrases: nonrigid, real-time, 4D reconstruction, multi-view

ACM Reference Format:
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. 1, 1, Article 246 (October 2017), 16 pages.
https://doi.org/10.475/123_4

*Authors equally contributed to this work.

XXXX-XXXX/2017/10-ART246 $15.00
https://doi.org/10.475/123_4

1 INTRODUCTION

The ability to model the complex pose, shape, appearance and motions of humans in 3D is a well studied problem in computer graphics and computer vision. Indeed the sub-field of performance capture, volumetric or free-view point video and non-rigid reconstruction has amassed considerable research, ranging from high-end studio capture leveraging multiple cameras [Collet et al. 2015a] to more commodity multi-view setups [Dou et al. 2016; Orts-Escolano et al. 2016] or even lighter weight solutions [Innmann et al. 2016; Newcombe et al. 2015].

Recently the potential for supporting this type of performance capture in *real-time* has unlocked the potential for exciting new applications for gaming [Mehta et al. 2017; Shotton et al. 2013], telepresence [Orts-Escolano et al. 2016] or augmented and virtual reality [Orts-Escolano et al. 2016]. Systems that have focused on

Fig. 2. Screenshots from Holoportation [Orts-Escolano et al. 2016] and Fusion4D [Dou et al. 2016] videos. *Geometric over-smoothing leads to uncanny reconstructions especially of faces (top row). Fast motions are supported but by degrading to noisy data samples in that region c.f. the scarf of the user (bottom right). Topological changes are also handled by localized resetting of the data c.f. the ball (bottom left).*

general non-rigid scene reconstructions, free from priors or templates, have shown particular potential, removing the over-fitting to human-only scenarios or prescriptive shape or motions. These *general* approaches allow for non-rigid reconstructions where humans can interact with objects, perform free-form motions, and even support topology changes [Dou et al. 2016].

The state-of-the-art in this type of *real-time volumetric performance capture* can be broken down into methods that support a single RGBD sensor [Guo et al. 2017; Innmann et al. 2016; Newcombe et al. 2015; Zollhöfer et al. 2014] or multiple sensors [Dou et al. 2016]. Despite the challenge of the single sensor case, given only a partial view of the scene and occlusions, these systems have shown impressive results but with careful and orchestrated motions [Innmann et al. 2016; Newcombe et al. 2015; Zollhöfer et al. 2014], avoiding fast and large topology changes [Guo et al. 2017; Innmann et al. 2016; Newcombe et al. 2015] or relying heavily on online generated template priors [Zollhöfer et al. 2014]. Multi-camera systems whilst requiring more costly hardware and setup, have recently demonstrated incredibly robust performance capture in real-time [Dou et al. 2016]. In particular, the Fusion4D method [Dou et al. 2016] which is the underlying performance capture technology of the Holoportation system [Orts-Escolano et al. 2016], has shown robust reconstructions even under large topological changes and large frame-to-frame motions.

Despite this, Fusion4D and hence Holoportation, suffers from many problems as illustrated in Fig. 2 and Fig 18. While the reconstructions are compelling given real-time performance, there is a clear lack of geometric detail, in particular over-smoothing of high frequency details. For example, faces appear extremely over-smoothed, and once textured the reconstructions exhibit a clear uncanny feeling. Second, during large motions, Fusion4D fails considerably. This is less apparent than other template or single reference volume systems [Guo et al. 2017; Innmann et al. 2016; Newcombe et al. 2015; Zollhöfer et al. 2014], as the model gracefully degrades to data only in regions of high alignment error. However, this effectively means in regions of tracking failure, the reconstruction is representative of the noisy data only, with loss of temporal consistency. The issue of tracking failures also relates to Fusion4D's handling of topology changes. While the system is clearly robust to many challenging scenarios with topological changes, it deals with it through this notion of partially resetting the model in the areas where topological changes occur. So, for example, in a region where two interlocked hands separate, the system first resets the reconstruction in this region, and then rapidly accumulates the model once the topology change occurs. This again breaks temporal consistency and falls back to noisier data briefly within this region.

In this paper we address these three challenges of over-smoothing of geometric details, degradation of quality where fast motions occur, and handling topological changes by model resetting. To deal with these issues we must build a new volumetric performance capture system from the ground-up with the following properties:

- To significantly improve over-smoothing of geometric details we create a new non-rigid alignment strategy with both geometric and photometric terms, with learned 3D correspondences predicted using an efficient spectral embedding technique, and finer scale alignment. This is coupled with a fusion strategy that can go beyond the resolution of voxel-only reconstructions with an overlaid geometric detail layer, and more faithful texturing using a compact embedded atlas.
- To deal with fast motions we create a non-rigid reconstruction pipeline with 3x the performance of Fusion4D using the same GPU hardware. This improved speed is due to a faster matching strategy using a more compact and robust parameterization. This is coupled with a novel machine learning technique for 3D correspondence estimation used to initialize our frame-to-model alignment avoiding certain frame-to-frame errors.
- To deal more effectively with topological changes our non-rigid alignment strategy is composed of both forward and backward passes using geodesic skinning to extract and refine correspondences for alignment. This allows for far more effective handling of topological changes than in Fusion4D.

Our system, called Motion2Fusion, runs at almost 100fps using high-end but consumer-available graphics hardware, works in arbitrary challenging scenes with topological changes or large motions without scene priors, and gracefully degrades from multi-view scenarios to single camera reconstructions. Extensive quantitative and qualitative comparisons show more precise geometric and texturing results with less artifacts due to fast motions or topology changes than prior techniques [Dou et al. 2016; Innmann et al. 2016; Newcombe et al. 2015; Zollhöfer et al. 2014].

2 RELATED WORK

Multi-view performance capture has been an active area of research for decades. We refer the reader to [Theobalt et al. 2010; Ye et al. 2013] for full reviews. The majority of previous works use offline

techniques with a focus on very high quality reconstructions, often requiring all data frames a-priori and considerable compute, making real-time scenarios prohibitive. The state-of-the-art offline systems offer incredible temporally consistent results, handling certain topology changes and arbitrary scene reconstruction, using active and passive 2D/3D cameras, chroma keying, and expensive hardware setups, but at frame rates that are orders of magnitude slower than real-time [Collet et al. 2015b].

When focusing on real-time methods, techniques either focus on parametric tracking of human bodies e.g. [Mehta et al. 2017], hands e.g.[Tan et al. 2016; Taylor et al. 2016] or faces e.g. [Cao et al. 2013; Thies et al. 2016], but often avoid arbitrary non-rigid shape reconstruction. The work of [Zollhöfer et al. 2014] uses an online rigidly captured template, which is non-rigidly tracked over time with a detail layer for accumulating some high frequency details. The work of [Cao et al. 2015] specializes to an online face model and shows incredibly detailed reconstructions from only an RGB sensor. Other works use skeletal priors and template models for impressive real-time reconstruction of scenes including humans and animals [Ye and Yang 2014]. These systems however clearly fail when under significant scene and topology changes, where the template model becomes inconsistent with the observed data.

DynamicFusion [Newcombe et al. 2015] was one of the first real-time systems that removed the need for such an explicit template prior, using a reference volumetric model that was non-rigidly warped to each new input frame, with data samples being fused into the reference over time. However, large topological changes and more complex motions led to tracking failures. VolumeDeform [Innmann et al. 2016] extends this general approach of warping an evolving reference volume over time, adding a color term and densely computing the warp field across all voxels. The work of [Guo et al. 2015] extends the notion of geometric and color based warping of the reference volume by modeling geometry, surface albedo and appearance. The per-frame non-rigid warp field is computed with an additional low-frequency lighting term that generates impressive tracking results.

To overcome these limitations of single view real-time systems, and create a practical performance capture system, Fusion4D [Dou et al. 2016] focused on real-time multi-view scenarios, with the notion of key volumes and partial fusion of data and reference volumes to handle topological changes and tracking failures, and a learning-based frame-to-frame 2D correspondence field estimation method [Wang et al. 2016] to handle fast motions. However as seen in the Fusion4D and Holoportation projects [Dou et al. 2016; Orts-Escolano et al. 2016], there is significant over-smoothing of geometric details, degradation of quality where fast motions occur, and topological changes are handled by a hard reset of the model within evolving regions.

In the remainder of this paper, we address each of these limitations of related work, and present a new pipeline for real-time performance capture. Our system does not require a template prior, removes the over-smoothing inherent in Fusion4D using a more precise alignment strategy, with finer scale fusion and texturing. We also support faster motions using an efficient 100fps reconstruction pipeline with robust correspondence estimation which moves beyond 2D and frame-to-frame estimation per view seen previously [Dou et al. 2016; Wang et al. 2016]. Finally, we present a new *two-way* non-rigid matching technique for handling topological changes more gracefully.

3 METHOD

We first describe the high-level components of our new pipeline as illustrated in Figure 3, which achieves significantly better performance and higher quality than previous real-time performance capture systems. Our performance capture system takes RGBD images as input from one or more viewpoints, and generates textured meshes as output. We use active IR stereo cameras to generate depth maps at high-speeds of 200fps [Fanello et al. 2017a,b] with additional color data. As in [Dou et al. 2016] we maintain both a data volume and evolving key (or reference) volume.

The core of our system solves for the non-rigid alignment parameters that warp the mesh at the key volume to the data frames. The accuracy of this alignment typically suffers from two major challenging scenarios: fast motions and surface topology changes. We extend current non-rigid matching work to handle both cases. To tackle tracking failures caused by fast motion, first we significantly improve the alignment frame rate by simplifying the non-rigid parameterization and improving the performance of the non-linear solve (see Sec. 3.2) . We can align full-body captures at 100+fps and upper-body captures at 200+fps. At such an unprecedented frame rate the motion between adjacent input frames is reduced. In most cases initializing the alignment parameters with the values from the previous frame causes the non-linear solver to converge much faster at this frame rate (see Sec. 4).

To further increase the alignment reliability, we learn a spectral embedding to predict the 3D correspondences between the reference mesh and the data mesh (see Sec. 3.1). These correspondences provide strong constraints that speed up the convergence further even if the initialization was poor due to fast motion. Additionally, we use a dense color term in the optimization to penalize any mismatch between the mesh colors and the input RGB data, effectively dealing with drift in areas with finer geometry e.g, faces (see Sec. 3.2.2). To better deal with surface topology changes and finer scale alignment, we run both forward matching (reference to data) and backward matching (data to reference) to find the dense correspondences between the reference and data, which are then used in the residuals during the final non-rigid optimization (see Sec. 3.2.1).

After the reference is aligned with the data, we fuse the data into the reference volume and also warp the reference to the data in order to generate a high fidelity output. We maintain a Truncated Signed Distance Function (TSDF) volume [Curless and Levoy 1996] at the reference for volumetric fusion (see Sec. 3.4). In addition to a TSDF value at each voxel, we also store the voxel color. For memory and computational efficiency, the volume is based on a two level hierarchy. Marching cubes is used to extract a triangle mesh with per-vertex color from the volume. With this approach our voxel resolutions are approximately limited to 5mm at frame-rate, however, we further fuse high frequency details using a novel 2D fusion step (see Sec. 3.4) which allows a finer scale 2D geometric displacement map to be encoded on-top of the coarser mesh.

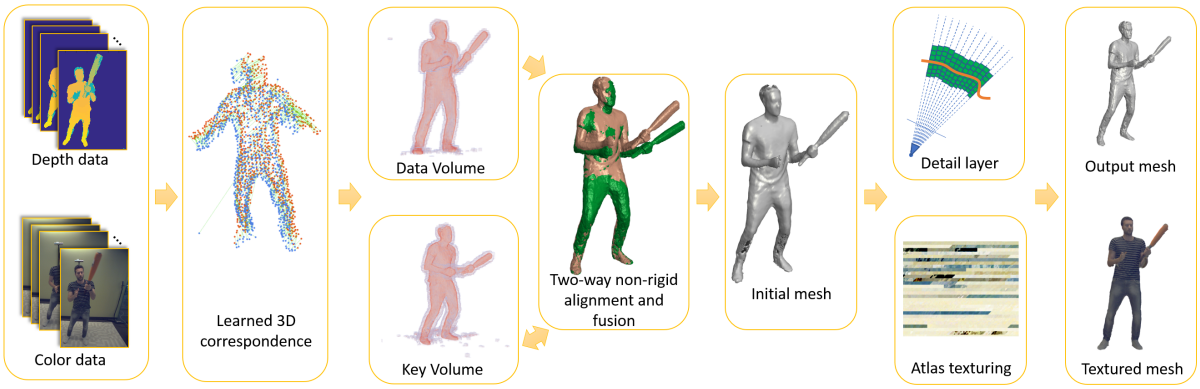

Fig. 3. System Pipeline. *We first compute RGBD frames from one or more sensors. We maintain both a data volume and an evolving key (or reference) volume. First a correspondence field from the data to the key volume is estimated using a learned approximation of a spectral embedding of the extracted points. This is used to initialize a first phase non-rigid matching performed bidirectionally between reference and model to refine the dense correspondence field. These correspondences are then used as residuals in the final non-rigid matching phase, after which the data is fused with the model encoding the overall shape. Afterwards the high frequency details and the texture mapping are incorporated.*

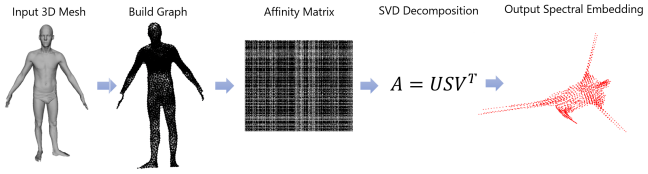

Fig. 4. Spectral Embedding Algorithm. *Starting from an input mesh the affinity matrix of the vertices is computed. The spectral analysis of the graph provides a new space which is robust to non-rigid transformations.*

Finally, we rasterize the output mesh triangles onto a plane and blend in the colors from the input. The generated atlas is UV mapped to the mesh geometry and a high fidelity reconstruction is rendered. Our system is capable of running at 100fps on a single GPU, in comparison to Fusion4D or Holoportation [Orts-Escolano et al. 2016] that required two GPUs and ran at 30fps with the exact same Nvidia Titan X hardware.

3.1 Learning the Spectral Embedding

While our system tracks at high frame-rates, relying purely on the previous frame can result in tracking failures, to address this we find model-to-data 3D correspondences using a novel learned spectral embedding approach, which provides strong constraints at later nonrigid matching stage (Section 3.2). This approach contrasts greatly from Fusion4D and Holoportation [Dou et al. 2016; Orts-Escolano et al. 2016], where the focus is on finding image patch correspondences [Bailer et al. 2015; Fischer et al. 2015; Wang et al. 2016; Zagoruyko and Komodakis 2015; Žbontar and LeCun 2015] from one viewpoint, and frame-to-frame only. With these 2D techniques their generalization to 3D space is not straightforward. This ultimately means that the correspondence estimation in Fusion4D suffers from frame-to-frame drift, occlusions due to each viewpoint being independently computed, and other projective errors. We instead learn to find correspondences directly between 3D meshes. Recently, deep 3D shape descriptors have been also proposed [Xie et al. 2015], but they cannot be used to find precise correspondences due to computational requirements, also their application is mostly limited to full shape retrieval rather than keypoints detection. An orthogonal trend is to consider 3D meshes in a graph representation and perform matching in this domain. A lot of computer vision efforts [Leordeanu et al. 2009; Zaslavskiy et al. 2009; Zhou and la Torre 2012] focus on solving this hard optimization problem leading to solutions that are not feasible in real-time and in general do not cope well with non-rigid deformations.

Our work borrows from the spectral embedding literature [Jain and Zhang 2006], where the general idea is to remap a 3D mesh into a new domain which gives a high degree of invariance to non-rigid deformations. Finding correspondences in this new space becomes a more tractable problem and they can be used as initialization for the rest of the pipeline. Specifically our approach starts from the input mesh (or 3D point cloud) and builds an affinity matrix $\mathbf{A} = \exp(-\frac{\mathbf{D}^2}{2\sigma^2})$ for every node (3D point) in the graph (mesh). The affinity matrix encodes the similarity of every node in the graph based on their distances $\mathbf{D}$, which approximates a Kernel function to encode non-linearities in the data. It has been shown that when the Geodesic distance is used, the spectral analysis of the affinity matrix is very robust to non-rigid transformation [Jain and Zhang 2006]. The affinity matrix $\mathbf{A}$ is then decomposed using the SVD decomposition in $\mathbf{A} = \mathbf{USV}^{\mathrm{T}}$ and the first $k-1$ eigenvectors $\mathbf{u}_2, \ldots, \mathbf{u}_k$ are used as new space where correspondences search becomes easier[1]. In Fig. 4 we depict the whole pipeline. Notably, the output space does not change when the subject drastically changes her pose. The main limitation of this approach is the computational cost to build the affinity matrix and perform the SVD decomposition. Although approximated solution exists [Woolfe et al. 2008], they still exceed our computational budget; moreover this method does not handle topology changes.

[1]Note we discard the first eigenvector which is associated with the eigenvalue 0 and is a constant vector.

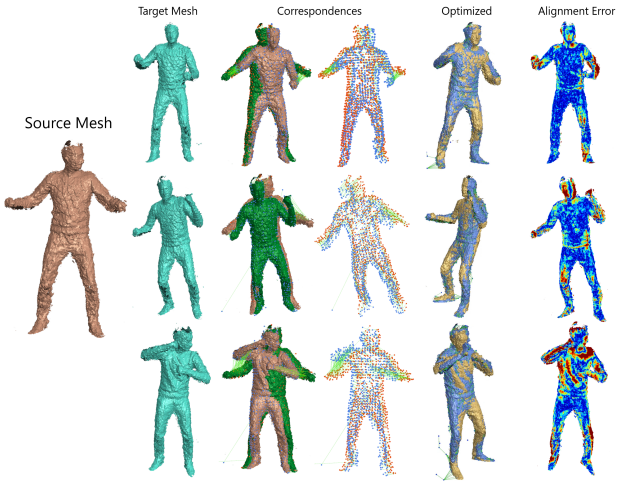

Fig. 5. Sparse Correspondences via Learned Spectral Embedding. *We visualize the correspondences and the alignment error obtained using our learning method.*

Our idea is to learn to approximate the embedded points starting from a set of 3D points $\mathbf{X} \in \mathrm{R}^{n\times 3}$ representing the vertices of the current mesh. Indeed, from a machine learning perspective, we can relate the affinity matrix $\mathbf{A}$ to a non-linear mapping of our input space, similar to a Kernel function. The embedded space therefore can be seen as target output that we want to approximate efficiently, avoiding the explicit computation of the affinity matrix and its SVD decomposition. In this setting we can cast the problem as a regression task. In particular, given the set of 3D vertices $\mathbf{X} \in \mathrm{R}^{n\times 3}$ we want to learn a function $f(\mathbf{X}) = \mathbf{Y} \in \mathrm{R}^{k\times n}$, where $\mathbf{Y}$ are the first k eigenvectors of the affinity matrix $\mathbf{A}$. This allows us to avoid both building the matrix $\mathbf{A}$ and performing its SVD decomposition.

Ideally, to keep low the amount of computation required, we would employ a simple linear model of the form $f(\mathbf{X}) = \mathbf{Y} = \mathbf{XW}$, where $\mathbf{W}$ are the model parameters. However, as discussed above, the mapping from $\mathbf{X}$ to the embedded space $\mathbf{Y}$ is strongly non-linear, due to the presence of the affinity matrix $\mathbf{A}$ and its SVD decomposition. To deal with such non-linearities we first have to use a non-linear mapping $\phi(\mathbf{X}) \in \mathrm{R}^{n\times F}$, which remaps the data to a higher dimensional space. In detail, we want to learn a non-linear mapping $\phi(\mathbf{X})$ of the point cloud and a regression function $\mathbf{W}$ that minimizes the loss:

$$\arg\min_{\mathbf{W}} \ \mathcal{L}(\ \phi(\mathbf{X})\ \mathbf{W}, \mathbf{Y}\ ) + \Gamma(\mathbf{W}) \tag{1}$$

where $\mathbf{Y} \in \mathrm{R}^{k\times n}$ is our embedded space and $\Gamma(\mathbf{W})$ is a regularization term to avoid overfitting, which we set to be the ℓ_2-norm of $\mathbf{W}$. The non-linearity ϕ is crucial to learn such a complex space and, as discussed, the typical approach is to compute the affinity (Kernel) matrix $\mathbf{A}$ of the data. Since our goal is indeed to avoid such computation, we choose to approximate the Kernel matrix via random features [Rahimi and Recht 2007] and set $\phi(\mathbf{X}) = \exp(i\frac{\mathbf{XF}_1}{\sigma_f})$, where $\mathbf{F}_1 \in \mathrm{R}^{3\times F}$ is randomly generated from a Gaussian distribution.

Applying any learned function directly to a point cloud is not trivial since it strongly depends on the order of the points and their arrangement in the space. Therefore we take inspiration from [Qi et al. 2016] and we use the concept of spatial pooling [Lazebnik et al. 2006] to summarize the whole point cloud and capture both global and local structures. In particular we perform an additional non-linear mapping $\Psi = \exp(i\frac{\mathbf{XF}_2}{\sigma_f})$ and compute its *max-pooling* over all the points n to obtain a single compact descriptor $\psi = max_i \mathbf{\Psi}_i \in \mathrm{R}^F$ where we set the same number of random features F. In order to catch both local and global information for each point $\mathbf{x}$ in the pointcloud $\mathbf{X}$ we aggregate the descriptors $\phi(\mathbf{x})$ and ψ with a simple subtraction $\mathbf{\Phi} = \phi(\mathbf{x}) - \psi$. Other forms of aggregation such as concatenation or max are possible, but we did not find any substantial difference.

Training Time. At training time, we collect point clouds $\mathbf{X}_1, \ldots \mathbf{X}_N$, which we randomly subsample 256 points from each point cloud. Each point cloud is a particular subject performing a particular pose. We compute the embedded space $\mathbf{Y}_1, \ldots \mathbf{Y}_N$ using the slow offline process proposed in [Jain and Zhang 2006]. Then we remap the input point clouds to $\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_N$ and we minimize the following loss function:

$$\arg\min_{\mathbf{W}} \sum_i \|\mathbf{Y}_i - \mathbf{\Phi}_i\mathbf{W}\|^2 + \lambda\|\mathbf{W}\|^2 \tag{2}$$

This is a standard least-squares problem and it can be solved in close form with respect to the variables $\mathbf{W}$.

Runtime. At run time, given two meshes $\mathcal{M}_1$ and $\mathcal{M}_2$ we randomly subsample them to 256 3D coordinates $\mathbf{X}_1$ and $\mathbf{X}_2$, we compute their non-linear mappings $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$ and generate the embedded spaces $\mathbf{Y}_1 = \mathbf{\Phi}_1\mathbf{W}$ and $\mathbf{Y}_2 = \mathbf{\Phi}_2\mathbf{W}$. In this new space, which is robust to non-rigid deformation, we perform a simple rigid alignment to retrieve the correspondences and use these in the rest of the pipeline for the initialization of the correspondence field. See Figure 5 for a visualization of the final learned correspondences.

3.2 Non-rigid Alignment

Similar to recent work [Dou et al. 2016; Li et al. 2009, 2013; Newcombe et al. 2015], we use an embedded deformation (ED) graph-based method for non-rigid matching [Sumner et al. 2007]. Given a reference mesh, ED nodes are uniformly sampled from the vertices. We represent the i-th node location as $\mathbf{g_i}$. Additionally, an ED node is associated with a set of parameters that represent the deformation it imposes in its local neighborhood. Neighboring ED nodes are connected together to form an ED graph, and we use G to collectively represent the deformation parameters and ED node locations on the ED graph. Each mesh vertex is "skinned" to K neighboring ED nodes (with $K = 4$ in our experiments), so that the mesh would be deformed according to the given parameters of an ED graph.

The task of non-rigid matching is to find the ED graph parameters that "perfectly" deform the reference mesh to fit the data. This is achieved by solving an energy minimization problem with the Levenberg-Marquardt (LM) algorithm. Our energy function is a combination of data terms and regularization terms. For the data terms, we penalize geometry misalignments, color inconsistencies, visual hull violations [Dou et al. 2016], and sparse correspondences disagreement (from Sec. 3.1). For the regularization terms, we added a smoothness term to enforce that two connected ED nodes having

similar parameters. We also constrain the local deformation at each ED node to be as rigid as possible to minimize distortion. At each LM step, the Hessian approximation matrix J^TJ and first order gradient $J^T\mathbf{f}$ are evaluated from the energy terms, where J is the Jacobian matrix of all residuals $\mathbf{f}$. An update vector $\mathbf{h}$ on ED parameters is solved from the normal equation $(J^TJ+\lambda\mathbf{I})\mathbf{h} = -\mathbf{J^Tf}$. J^TJ is a sparse block matrix, and we solve the normal equation with a PCG method optimized for the block structure of J^T.

ED Parameterization. The deformation at each ED node can be parameterized as an affine transformation which has 12 parameters (a 3×3 matrix A together with a 3D vector $\mathbf{t}$). Using Linear Blend Skinning (LBS), the transformation applied to warp a vertex $\mathbf{v}$ at the reference to $\tilde{\mathbf{v}}$ at the target is denoted by $\tilde{\mathbf{v}}(G) = \sum_{k=1}^{K} w_i(A_k(\mathbf{v}-\mathbf{g_k})+\mathbf{g_k}+\mathbf{t_k})$. Practically, a regularization is added on A as a soft constraint to enforce it as close to rotation matrix as possible. In DynamicFusion [Newcombe et al. 2015], A is forced to be a rotation matrix by parameterizing the deformation at each ED node as a dual quaternion, and approximate Dual Quaternion Blending (DQB) [Kavan et al. 2007] is instead used in the warp function. Compared with linear blending of rigid deformations, dual quaternion blending avoids mesh collapse artifacts, at the cost of more expensive compute.

In this paper, we represent A as a quaternion $\mathbf{q}$ without explicitly forcing $\mathbf{q}$ to be unitary, instead we treat $\|\mathbf{q}\| = \mathbf{1}$ as a soft constraint. Essentially, this semi-rigid parameterization is a trade-off between a full affine deformation (12-DOF) and a rigid deformation (6-DOF) by adding a uniform scale factor in addition to the rigid parameters, resulting in a 7-DOF transformation per ED node. To warp a vertex $\mathbf{v}$, we perform:

$$\tilde{\mathbf{v}}(G) = \sum_k w_i(R(\mathbf{q_k})(\mathbf{v}-\mathbf{g_k})+\mathbf{g_k}+\mathbf{t_k}), \tag{3}$$

where $R(\cdot)$ converts a quaternion to a rotation matrix. Since we do not explicitly force $\mathbf{q}$ to be unitary, $R(\mathbf{q})$ becomes a rotation matrix multiplied by a scalar. Note that we use linear blending instead of quaternion blending. This is partially to stay within our computational budget and maintain a very high frame rate, but also, in practice the mesh collapse artifacts associated with LBS were not an issue due to the high frame rate we maintain.

Compared with a full affine parameterization the semi-rigid representation is computationally more efficient at both the evaluation of J^TJ and later during the optimization. The block size of the sparse block matrix J^TJ is 7×7, so roughly only one third of the compute and memory operations is required compared with the 12×12 block matrix in the affine parameterization case. Also the CUDA implementation needs less shared memory and can achieve higher kernel occupancy. Overall, in our implementation, this lead to a 4x performance improvement compared to recent work like Fusion4D [Dou et al. 2016]. Unlike the dual-quaternion parameterization in DynamicFusion [Newcombe et al. 2015], each ED node "rotates" vertex $\mathbf{v}$ around its location $\mathbf{g}$, which makes the optimization problem numerically more stable. Also, no singularity point exists in quaternion space, while the underlying Rodriguez rotation vector would suffer from the singularity problem requiring the delta technique to handle the issue [Newcombe et al. 2015].

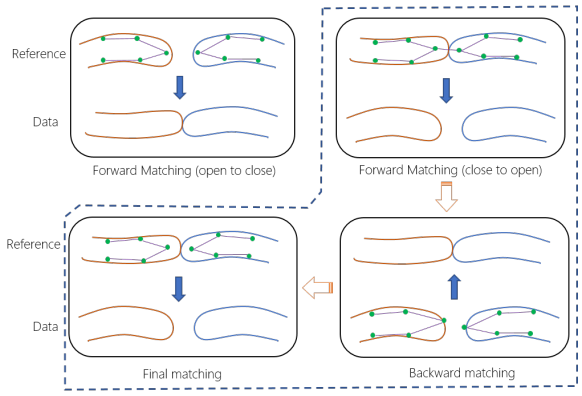

Fig. 6. Two-way Non-rigid Matching. *Forward matching would fail when aligning a reference mesh with closed topology to a data mesh with open topology. Our two-way matching approach properly handles this case.*

3.2.1 Two-Way Non-rigid Matching . One failure case for the approach detailed so far is surface topology changes. As illustrated in Figure 6, the ED graph-based alignment will not have any issues aligning a reference surface with open topology to data with closed topology (i.e, surface merging), but it will fail the other way around (i.e, surface splitting). This is due to both the vertex-to-node skinning data term and the node-to-node connectivity regularization term. By construction, there are no semantic labels associated with either the ED nodes or the vertices. That means that a vertex is inevitably skinned to whichever ED nodes fall within its local neighborhood. For instance, when a hand touches the face, vertices on the face can be partially skinned to ED nodes on the hand. Moreover, face ED nodes and hand ED nodes might be connected through regularization terms. However, this does not occur when the mesh has an open topology to begin with.

The above observation leads us to develop a novel two-way non-rigid matching strategy. We run both forward matching, aligning the reference to the data and then backward matching, aligning the data to the reference. From this two phase alignment we identify the best point-to-point correspondences between reference and data, which are then used for the final high fidelity non-rigid matching.

Initial Forward Matching. We start by matching the reference to the data. As in Fusion4D [Dou et al. 2016; Newcombe et al. 2015], the data term we use to measure the misalignment between the reference mesh and the input depth maps using the projective point-to-plane distance:

$$E_{\text{data}}(G) = \sum_m \sum_n \delta_{mn}\tilde{\mathbf{n}}^T(\tilde{\mathbf{v}}_m - \Gamma_n(\tilde{\mathbf{v}}_m)), \tag{4}$$

where $\tilde{\mathbf{v}}$ is the warped vertex as defined in Equation 3; m and n are the indices of the vertex and the depth map respectively, and $\Gamma_n(\mathbf{v})$ represents the corresponding projective 3D point of $\mathbf{v}$ from the n-th depth map. δ_{mn} represents the visibility test for $\tilde{\mathbf{v}}_m$ where $\delta_{mn} = 1$ if it is visible in the n-th depth map, and $\delta_{mn} = 0$ otherwise. We additionally integrate the visual hull term [Dou et al. 2016] and the learned correspondence term from Sec. 3.1. To deal with drift we add a color term as well (see Sec. 3.2.2).

After the forward alignment is established, we pair each vertex $\mathbf{v_m}$ on the reference surface with its corresponding point $\mathbf{p_m^{fwd}}$ on the data mesh (volumetrically fused from the input depth maps). We let $\mathbf{p_m^{fwd}}$ denote the closest surface point of $\tilde{\mathbf{v}}_m$, and we discard a corresponding pair if their distance is greater than 2mm. This step culminates in a correspondence set $\{(\mathbf{v_m}, \mathbf{p_m^{fwd}})\}$.

Backward Matching. After the forward matching, we use Equation 3 to warp the ED nodes of the reference to the data pose ($\mathbf{g} \rightarrow \mathbf{g} + \mathbf{t}$). We then update the node-node connectivity based on the mesh topology at the data. The updated ED graph is then used for the backward matching, where we find the correspondences on the data for all reference vertices: $\{(\mathbf{v_m}, \mathbf{p_m^{bwd}})\}$. The key here is to re-use the ED graph instead of resampling a new ED graph from the data mesh. This way we preserve the correct ED node connectivity during the alignment of a reference mesh with open topology to input data with closed topology, where the ED graph from the reference would have the right connectivity while the resampled ED graph at the data would have the wrong one. Instead of aligning a mesh to depth maps and color images as we did in the forward matching, we align the data mesh to the reference TSDF volume $\mathbb{V}(\cdot)$, therefore the data term measuring the misalignment has a different formula:

$$E_{\text{tsdf}}(G) = \sum_m |\bar{\mathbb{V}}(\tilde{\mathbf{v}})|^2. \tag{5}$$

where $\mathbb{V}(\cdot)$ only defines the signed distance values at fixed regular lattice points, while $\bar{\mathbb{V}}(\cdot)$ is a continuous volume field sampled through the trilinear interpolation of $\mathbb{V}(\cdot)$ at any given point. The parameters at each ED node for the backward matching are initialized as the inverse of the forward transformation at that node: $\mathbf{q}^{\text{bwd}} = \mathbf{q}^{-1}$ and $\mathbf{t}^{\text{bwd}} = -\mathbf{t}$.

Final Matching. After forward and backward matching, we have two sets of dense correspondences. A vetex on the reference mesh might have one correspondence from forward matching and one from backward matching. We pick the best correspondence (closest) for a reference vertex (if any exists): $\{(\mathbf{v_m}, \mathbf{p_m})\}$. If the forward correspondence is chosen for a reference vertex $\mathbf{v}$ we conclude that this vertex has a reliable correspondence at the data given the reference mesh topology. However, if the backward correspondence is chosen for $\mathbf{v}$, we then update its skinning results (both ED node set and weights) to match the topology at the data. We perform the final matching to align the reference to the data with the updated graph connectivity and vertex skinning, using the extracted correspondences as residuals:

$$E_{\text{corr}} = \sum_m \|\tilde{\mathbf{v}}_m - \mathbf{p}_m\|^2 \tag{6}$$

Two-way no-nrigid matching has two additional matching operations compared to the standard forward matching, but both backward matching and the final matching are computationally lightweight due to simplified energy function with good (in some case perfect) initialization from previous steps.

3.2.2 Color Term. The point-to-plane-distance data term used in the forward matching does not constrain a vertex's movement along the tangent plane, leading to *drift* when the surfaces have finer geometry or lack geometric features altogether (see figure 7). To handle drift we add an energy term to measure the photo-consistency between per-vertex colors on the reference mesh and the observed color for the mesh vertices from the input color images $\{\mathbf{I}_n\}_{n=1}^N$. This new term is formulated as:

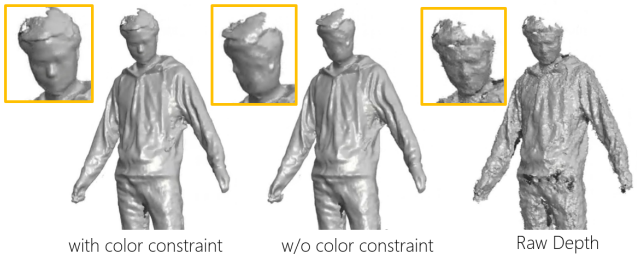

Fig. 7. Color term. *When geometry lacks details giving rise to ambiguity, matching can fail. Incorporating a color term ameliorates this issue.*

$$E_{\text{clr}} = \sum_m \sum_n \delta_{mn} \|\mathbf{I}_n\left(\Pi_n(\tilde{\mathbf{v}}_m)\right) - \mathbf{c}_m)\|^2, \tag{7}$$

where $\Pi_n(\tilde{\mathbf{v}})$ projects the warped reference vertex projected to the n-th color image space (2D image coordinate), δ_{mn} is the visibility term as in Equation 4, and $\mathbf{c}_m$ represents the 3D color vector of the m-th vertex on the reference mesh. In practice, we found that collapsing the 3D color into a single intensity (grayscale) value barely degrades the alignment quality but dramatically reduces the computation requirement for the $J^T J$ evaluation. To further reduce the compute we collapse multiple residuals on the same vertex into one, incorporating instead the residuals:

$$E_{\text{clr}} = \sum_m \left(\sum_n w_{mn} \bar{I}_n(\Pi_n(\tilde{\mathbf{v}}_m)) - \bar{c}_m) \right)^2, \tag{8}$$

where $\bar{I}$ is an input intensity image and $\bar{c}$ is the per-vertex intensity value, and w_{mn} is the weight on each input view where $\sum_n w_{mn} = 1$. The weight w_{mn} takes into account visibility and viewing angle.

3.3 Skinning

From Equation 3 we see that a vertex $\mathbf{v}$ is warped using a set of transformations $\mathbf{q_k}$ and $\mathbf{t_k}$. Under the same real-time constraints previous work opted to use the k-nearest ED nodes [Dou et al. 2016; Newcombe et al. 2015]. This was done by simply searching a small set of neighbors that lie in the same 3D space partition [Dou et al. 2016]. This approach completely ignores the topology of the shape, assigning vertices to nodes that are close in Euclidean space but very far on the mesh manifold. This results in artifacts that cause the mesh to stretch and distort counter-intuitively during non-rigid alignment.

A more principled approach is to choose the closest ED nodes to that vertex on the mesh manifold, replacing the Euclidean distance with the geodesic distance. This can be shown to avoid the deformation artifacts but at the expense of expensive computation [Li et al. 2009]. We address this issue using a extremely fast approximation for geodesic skinning. We first find the nearest ED nodes to each vertex using the Euclidean distance [Dou et al. 2016]. For each ED node we then solve for the heat equilibrium over the surface, where the

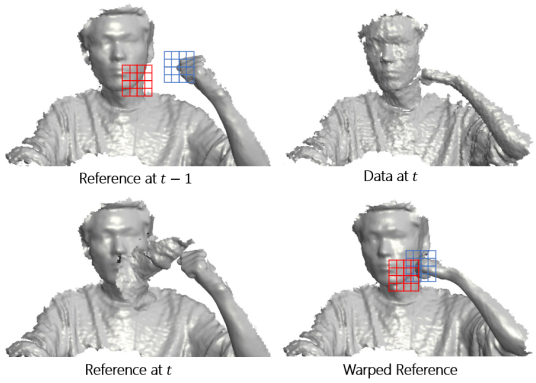

Fig. 8. The Space Compression Problem and the Double Surface Artifact. *Both the red voxels on the face and the blue voxels around the hand will project onto the same sets of depth pixels (face pixels) after warping, so the face would be fused into both areas, causing the double surface artifact.*

heat is transferred from the ED node to only the vertices to which this node was selected as one of its k-nearest ED nodes [Baran and Popović 2007]. This culminates in solving the linear system for ED node i:

$$(\mathbf{D} - t\Delta)\mathbf{w}_i = \delta_i, \tag{9}$$

where $\mathbf{D}$ is a diagonal matrix where D_{ii} is the area of vertex i (one third the area of all the triangles incident on vertex i), Δ is the discrete surface Laplacian calculated using the cotangent formula [Meyer et al. 2002] and using a half-edge data structure for the intermediate representation, t is a temperature hyperparameter, and δ_i is a one-hot vector indicating the vertex that represents the ED node. This also led us to sample ED nodes on the mesh manifold by choosing an ED node as the medoid of a set of vertices instead of the centroid as in previous work.

The resulting linear systems are small and independent. We proceed to solve them in parallel with a custom Gauss-Seidel implementation. When the recovered solution w_{ij} is zero for a vertex j, it is then determined to be too distant on the manifold and is subsequently detached from ED node i, effectively avoiding most distortion artifacts. We only apply this strategy to mesh surface warping. To stay within computational budget we still employ the Euclidean distance during volume warping. The details are in the next section.

3.4 Data Fusion and Detail Layer

We employ the idea of key frames, in a strategy similar to [Collet et al. 2015a; Dou et al. 2016], to increase the overall robustness of the data fusion. Initially we set the first frame as the key frame, and we perform non-rigid alignment between the key frame and the new data frame, so that data can be fused into the key frame. When the alignment error is past a threshold at a data frame, we set this data frame as the new key frame. We use a TSDF volume as the underlying data structure for fusion [Curless and Levoy 1996; Newcombe et al. 2011]. In addition to fusion at the key frame, where data is fused into the key volume $\mathbb{V}^k$, the key volume is also selectively warped to the data pose according to the alignment error and blended with the live data. We will detail the novel parts of our volumetric fusion and volume warp algorithm in the following sections.

Volume operations are expensive both computationally and memory-wise, so we cannot afford a volume with a prohibitively high voxel resolution. In our experiments we set the voxel resolution to 5mm. To preserve the finer geometric details we also maintain a light-weight 2.5D volume in the depth map space. The details from the 2.5D volume are superimposed on the the mesh extracted from the regular volume.

Key Volume Fusion and Space Compression. Using the (forward) warp function (Equation 3) we can transform and then fuse the input depth maps from the current data frame into the key volume. Each voxel $\mathbf{x}$ of the key volume is first warped to the data pose $\tilde{\mathbf{x}}$, and then $\tilde{\mathbf{x}}$ is projected to each depth map to pick up a depth value. The depth value is then "fused" by updating the TSDF value at $\mathbf{x}$ [Curless and Levoy 1996]. In the non-rigid case voxels from one surface could end up warped towards another surface that is sufficiently close, resulting in the space between the two surfaces to be compressed (see Figure 8). Depth pixels corresponding to one surface could then be erroneously selected by voxels from both surfaces, causing the "double surface" artifact. We refer to this issue as the "space compression" problem. Since this additional phantom surface tends to show up at some distance away from the real surface, this problem can be mitigated by adjusting the weight of a new TSDF value according to its distance to the surface, so that TSDF values corresponding to the phantom surface are suppressed by the existing TSDF values in the key volume. However, this does not fully resolve the problem, as the "double" surface will eventually appear if the two surfaces stay at this pose for a long enough time.

This space compression problem can be solved by updating the TSDF value of voxel $\mathbf{x}$ if and only if the depth pixel that $\tilde{\mathbf{x}}$ projects to is coming from the same surface. Intuitively, surface points from the same neighborhood should be skinned to a similar set of ED nodes with similar weights. Thus, given two points and their skinned ED nodes and skinning weights: $\{I_k, w_k^I\}_{k=1}^K$ and $\{J_k, w_k^J\}_{k=1}^K$, we can assume they come from the same surface if:

$$\sum_i \sum_j \delta(I_i, J_j) \min(w_i^I, w_j^J) > C, \tag{10}$$

where $\delta(I_i, J_j) = 1$ if $I_i = I_j$ and $\delta(I_i, J_j) = 0$ otherwise. We set the threshold $C = 0.8$ in our experiments.

Voxels of the key volume can be trivially skinned to their closest ED nodes in the Euclidean sense. However, depth pixels of the data frame cannot be directly skinned to the ED nodes at the reference pose. Warping ED nodes to the data pose first before skinning depth pixels to them is also problematic, because the association between a point to an ED node could change if the pose changes. Thus, a surface point on a face could be skinned to ED nodes on a hand when that hand touches the face. Instead, we warp the mesh of the key frame (key mesh) to the data pose (for only vertices with high alignment scores), and project the warped mesh to the depth maps. We assume that a depth pixel with depth d is associated with a surface point $\mathbf{v}$ if the warped $\tilde{\mathbf{v}}$ projects onto the pixel and the depth difference between the projected value and the pixel value

is smaller than 10mm. We then skin the depth pixel to the same ED nodes with the same weights as its associated surface point. If we cannot find a corresponding surface point for a depth pixel, we skin it directly to the warped ED nodes. With both voxel $\mathbf{v}$ and the depth pixel that $\tilde{\mathbf{v}}$ falls onto are skinned to ED nodes properly, we use Equation 10 to test if the depth pixel can be fused into voxel $\mathbf{v}$.

Volume Warp and Collision Detection As introduced in Section 3.2.1, the final warp function would warp the key volume towards the data. Voxels in the key volume sit on regular lattice grid points, but after the forward warp, we need to re-sample the TSDF values and weights at the non-uniform lattice grid of the data volume. Also, we need to handle the voxel collision problem since voxels corresponding to different surface parts might be warped to the same location. Instead of treating each voxel individually, we apply the warp function to each volumetric cube (with eight voxels at its corners) on the lattice grid. After warping, the eight voxels can also become non-cuboid. We discard cubes that are distorted dramatically and assume the rest approximately have the shape of a cuboid. We then perform a rasterization process on each warped cuboid to detect all the lattice points of the data volume that sit inside it. We trilinearly interpolate the TSDF value and weight for each lattice point.

A data voxel can get TSDF values from multiple cubes when a collision happens. This can be handled with a custom atomic operation in our implementation. When we write a new TSDF value d^{new} to a voxel with an old TSDF value d^{old}, we set $d = d^{\text{new}}$ if either d^{old} is invalid or $|d^{\text{new}}| < |d^{\text{old}}|$; and set $d = d^{\text{old}}$ otherwise.

Detail Layer. In addition to the regular voxel grid with 5mm resolution, we also use a finer fusion step that captures higher frequency geometric details. Unlike the regular volume, the finer grid lives in the depth map space. At each grid point (i.e., depth pixel), there are multiple voxels distributed along the view ray going through the grid point, and these voxels locate around a point with depth value $\bar{d}$, representing the depth uncertainty around $\bar{d}$, where $\bar{d}$ is determined by projecting the mesh from the regular volume to the depth map. As in TSDF regular volume, each voxel here also has a TSDF value and a weight. This 2.5D volume preserves all the data that is encapsulated in the input depth maps as the grid has the same spatial resolution as the incoming depth map. We use a higher resolution along the viewing ray (depth difference between adjacent voxels along the viewing rays) and set it to 2mm in our implementation. Figure 9(top) shows the locations of the voxels (in green) of one line of depth pixels in 3D space. The results at the bottom of the figure illustrate the details we recover. Note fine details like the bump from the lettering on the t-shirt and the jacket pocket; both are over-smoothed when the detail layer is not used.

The volume operation on this 2.5D volume is efficient. First, all voxels at the same grid point have the same skinning parameters, so the warping operation is computationally cheap. Second, marching cubes on this volume is as trivial as finding the zero crossings at each pixel since the pixel connectivity is a simple 4-way grid.

3.5 Atlas Mapping

With a higher detailed model, a new method for texturing is also desired. We have access to the per-vertex colors that we use to aid in tracking fine details, which prior work used in the final output [Dou et al. 2016]. However, tying the color sampling to the geometry resolution gives rise to an unfavorable trade-offs. On the one hand, increasing the voxel resolution to improve the color fidelity significantly slows down our pipeline and increases its memory footprint, and on the other hand lowering the color sampling resolution produces blurred results.

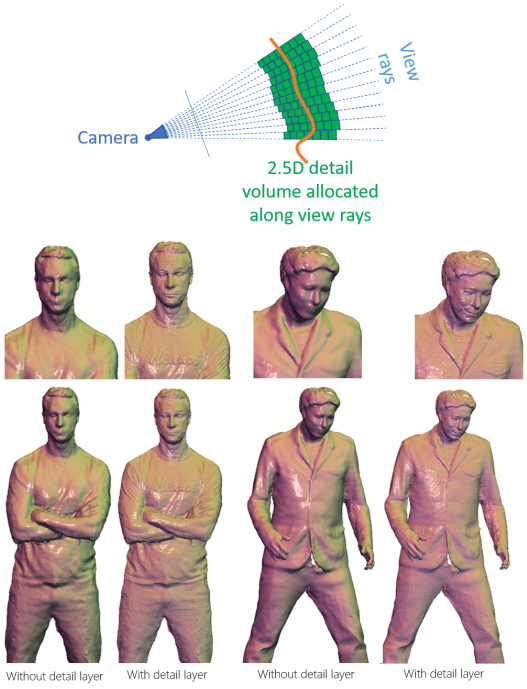

Fig. 9. Detail Layer. *We build a 2.5D volume to capture the high frequency details that can be lost during the matching process (top). Incorporating these details recovers subtle features (e.g, lettering on the t-shirt, jacket pocket) and produces a higher fidelity output.*

It became clear that generating an atlas map would avoid this trade-off. The literature on atlas mapping is vast. To enable planar embedding, a surface has to be first either cut through a seam (e.g, [Sheffer and Hart 2002]) or segmented into charts (e.g, [Zhou et al. 2004]). After applying one or more cuts, one can construct a parameterization to map the "flat" mesh segments onto a plane, minimizing angle distortions (e.g, [Lévy et al. 2002]), distance distortions (e.g, [Sander et al. 2001]), or any other distortion metric. The unwrapped charts have to additionally be packed into a single rectangular atlas image.

To stay within our computational budget we chose to construct a simple per-triangle atlas map. This decouples color fidelity and geometry resolution without sacrificing the frame rate. Inspired by [Soucy et al. 1996], we construct a half-pixel corrected triangular atlas, where triangles are independently paired (back to back) and mapped onto a fixed size square region. Additionally, by further contracting the triangles by a half-pixel we can support bilinear interpolation of the atlas texels in OpenGL without any bleeding artifacts [Carr and Hart 2002]. Figure 10 compares rendering with

Fig. 10. Per-vertex Color Compared to Atlas Mapping. *With the voxel resolution fixed, per-vertex colors look blurred and suffer from block artifacts. Increasing the voxel resolution to alleviate this issue is computationally inefficient. On the other hand, atlas mapping decouples geometry resolution from color quality.*

per-vertex colors and with our atlas mapping approach at the same geometry resolution. It is apparent that the atlas preserves more details whereas the per-vertex texturing results in over smoothing.

Finally, the triangle pixels are projected to the input images and the color across all inputs is blended in. The colors are weighted according to the surface normal and the camera viewpoint direction, favoring frontal views. Note we build the atlas individually for each frame, but we do fuse color temporally into the volume, which gives per-vertex color on the output mesh. Per-vertex color serves two purposes: provides additional constraints for the nonrigid matching in Section 3.2.2 and helps to fill up the texture at the regions blocked from all color cameras.

4 RESULTS

In the following, we show qualitative results of our methods as well as qualitative and quantitative comparisons to related work.

4.1 Implementation deatails and performance

Motion2Fusion runs on a single GPU (Nvidia Titan X). We eliminated host-device synchronization by using multiple streams and a buffering strategy to maximize the GPU utilization. The system performance largely depends on the volume of the captured object. On average, here is the computation breakdown of each system component for single person full-body capture with eight views: ~1.5ms for preprocessing (e.g., compute normal map, visual hull, and sparse correspondences), ~5ms for nonrigid alignments (3 iterations for forwards matching, and 2 iterations for the backward and final matching respectively), ~3.0ms for data fusion (fusion at key volume, volume warp, etc), ~0.6ms for atlas calculation. Note that some expensive operations (e.g., volume warp, marching cubes on data volume, building atlas) only need to run at the display rendering framerate (60 fps in our case), thus the overall system performance is further improved. On average, for single person eight-view full-body capture, the system runs at around 100 fps, while we achieve 200 fps for upper-body capture with a single person and a single view.

4.2 Qualitative results

We captured a series of live multi-view captures to show the high quality of our reconstruction pipeline. All sequences were captured *live* in *real-time*, with our full pipeline (including depth estimation and non-rigid reconstruction) running at around 100 Hz. We captured many diverse, challenging sequences including people interacting and deforming objects, topology changes and fast motions. Figure 17 shows multiple examples that demonstrate the high level of detail in the reconstruction obtained using Motion2Fusion, that is lacking in other systems.

4.3 Comparisons

Qualitative comparisons. In the remainder of this section, we compare Motion2Fusion to a variety of state-of-the-art real-time non-rigid reconstruction/tracking methods, which include our reimplementations of DynamicFusion ([Newcombe et al. 2015]), VolumeDeform ([Innmann et al. 2016]), Template tracking ([Zollhöfer et al. 2014]) and Fusion4D ([Dou et al. 2016]). We compare the different approaches using standard single camera datasets made available by [Newcombe et al. 2015] and [Innmann et al. 2016] in Figure 11. Similar to the work by [Collet et al. 2015a] and [Dou et al. 2016] we build a system made up of 8 sensors bars that are uniformly placed in a room and calibrated with respect to each other. Each sensor bar provides as output real-time depth maps aligned with color at high frame rates leveraging recent work in high speed depth computation ([Fanello et al. 2017a,b]) that is capable of producing depth maps at speeds as high as 200fps in our setup. We capture a wide array of challenging yet casual non-rigid sequences and use this data to compare our approach with the other approaches in Figure 18.

The prominent failure cases seen in case of prior works fall into two main categories: tracking failures and oversmoothing as shown in Figure 11. In case of complex motions and topology changes approaches that exploit a single fusion volume ([Newcombe et al. 2015]) or tracking a template model ([Zollhöfer et al. 2014]) or densely deform the volume at every voxel ([Innmann et al. 2016]) often break in case of challenging scenes such as topology changes and fast motion, due to tracking failures, which are hard to recover from. We qualitatively demonstrate the efficacy of our proposed formulation in handing these challenges in Figure 11. In case of multi-view data, while the data is more complete, these prior approaches cannot handle topology changes that are prominent in casual motion capture resulting in failures that is clearly observed in Figure 18. Motion2Fusion is however robust to this, and maintains higher fidelity in the final reconstruction.

Fusion4D ([Dou et al. 2016]) tried to address a number of these problems however their approach fails to recover details in the reconstruction, and often breaks in case of fast motion as was observed in Figure 2. We qualitatively compare Motion2Fusion against our re-implementation of Fusion4D ([Dou et al. 2016]). Note in

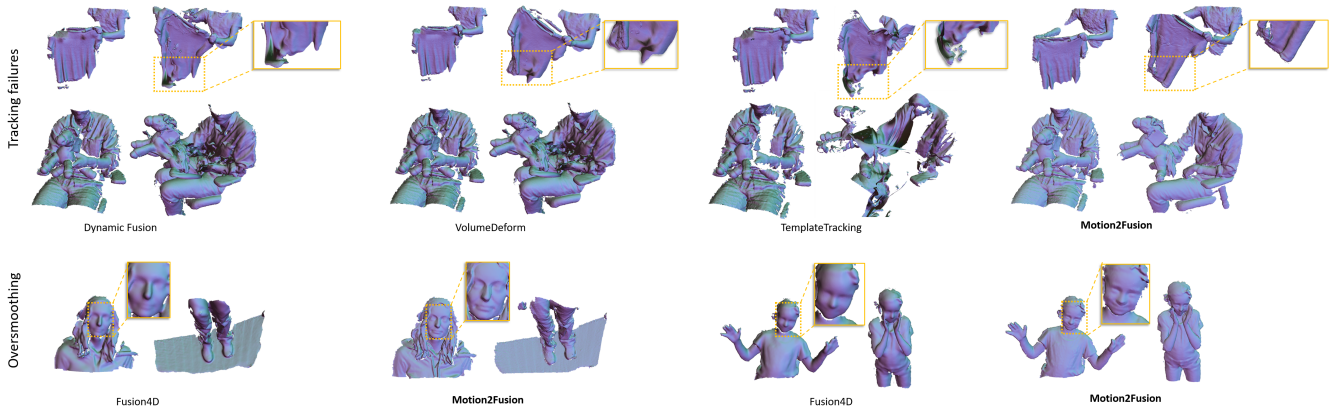

Fig. 11. Comparisons to previous real-time single RGBD systems. *We show a comparison of our work to reimplementations of previous work on public/author provided datasets. These datasets are optimized for single view non-rigid reconstructions, and hence avoid complex motions. Our system is comparable to these RGBD systems, and avoid the over-smoothing in the model as observed by Fusion4D (e.g, [Dou et al. 2016]). This is encouraging given our system is optimized for multi-view scenarios where there are fast motions and topology changes but less occlusions.*

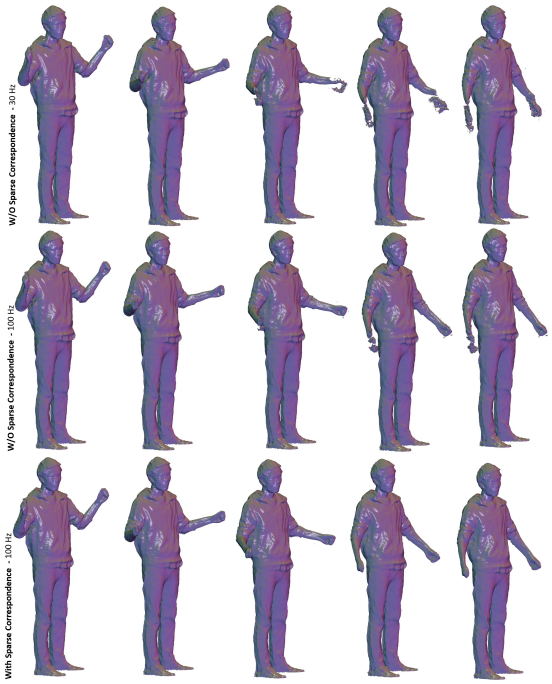

Fig. 12. Fast Motion Comparison. *At 30fps there are clear reconstruction artifacts on the arms of the user observed (1st row - our system running at 30fps without using sparse correpondences). At 100fps, most artifacts disappear (2nd row - our system running at 100fps without using sparse correspondences), and the learned sparse correspondences further improve the tracking quality (3rd row - our system running at 100fps with sparse correspondences).*

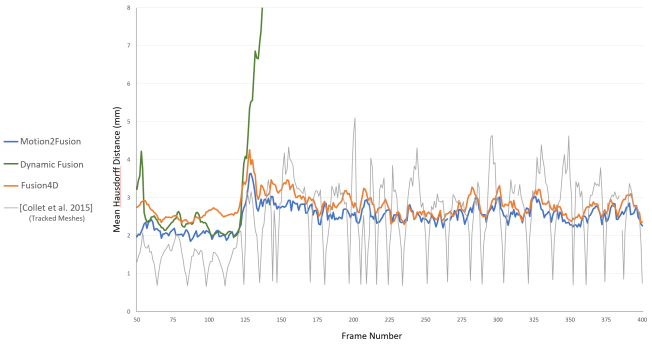

Fig. 13. Ground Truth Error Comparison to Other Approaches. *The mean error reported by our real-time system on the dateset of [Collet et al. 2015a] is on par with the state-of-the-art offline methods.*

Figure 11 the high level of geometric detail we achieve in our reconstruction that is lacking in the Fusion4D result. Further, note the higher level of texture quality due to the use of an atlas which is specifically noticeable when zooming to the face and which is important in virtual and augmented reality telepresence scenarios such as in [Orts-Escolano et al. 2016]. The oversmoothing results in smoothed detail and also results in inaccurate reconstructions in case of topology changes such as the ball being thrown from the hand as shown in Figure 18. We can also see that Fusion4D breaks in case of very fast motions such as the baseball bat shown in Figure 18 whereas Motion2Fusion faster non-rigid pipeline coupled with correspondence estimation and is capable of processing three times the amount of data and maintains the fidelity of fast motion and important details in the reconstruction.

Quantitative comparisons. We perform quantitative comparison across prior approaches using the dataset of [Collet et al. 2015a] similar to the comparison provided by [Dou et al. 2016]. In Figure 13 we show the mean error against the ground truth mesh for several prior approaches. In particular, we show that the fidelity of Motion2Fusion is comparable to offline simplification procedures while avoiding the tracking failures of [Newcombe et al. 2015]. The results reported for [Collet et al. 2015a] are those of the tracked

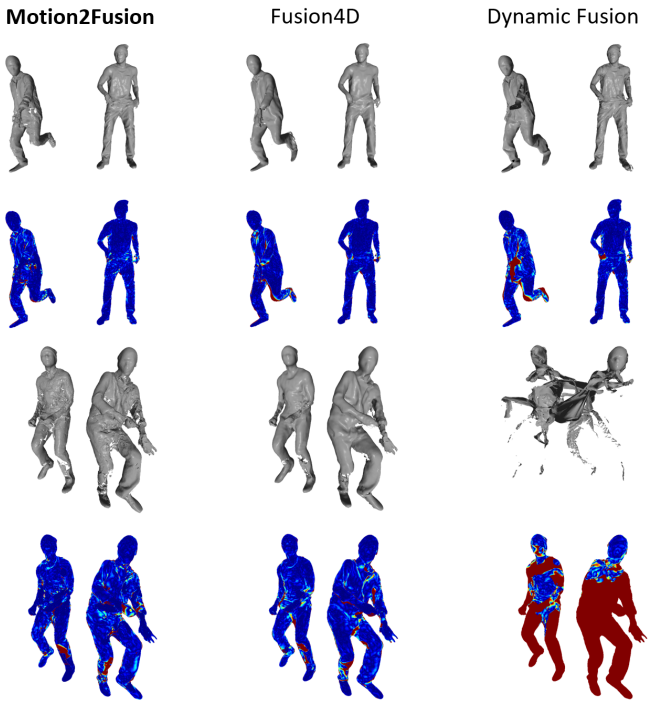

Fig. 14. Error Maps against Ground Truth. *We visualize the error on the reconstruction compared to the ground truth meshes on the dataset of [Collet et al. 2015a].*

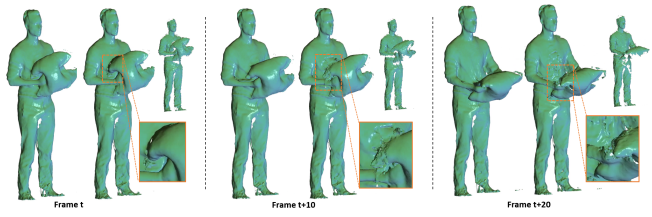

Fig. 15. Surface Topology Change. *We show the recontruction results for a sequence of moving a pillow away from chest. At each frame, the 1st surface shows the result with two-way nonrigid matching; the 2nd surface shows the result with standard forward matching as in Fusion4D; the 3rd one shows the input mesh (fused from multiple depth maps at the same frame) for reference.*

mesh and therefore are comparable to our results, while their simplified meshes uniformly have a mean error of 1mm. In Figure 14 we visualize the projected error of the tracked reconstruction compared to the ground truth mesh. Note the improved performance of our method compared to [Dou et al. 2016] on small details in key areas such as the face and the hands.

5 DISCUSSION

Contribution of system components. Compared with Fusion4D, DynamicFusion and others, our new *nonrigid alignment component* has less tracking failure (Figure 18) and results with a smaller misalignment error (Figure 13 14). More specifically,

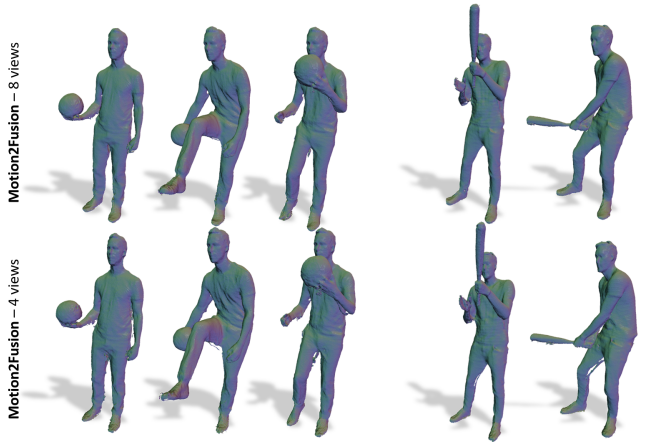

Fig. 16. Robustness to the number of views. *Dropping the number of viewpoints in our multi-view data from 8 views to 4 symmetric views around the object continues to produce very compelling results albeit with some holes appearing on the legs due to missing data.*

- the proposed two-way nonrigid matching algorithm improves the alignments when surface topology change appears as shown in Figure 15;
- the color term provides more constraints when tracking surfaces with subtle geometries (e.g., faces) as shown in Figure 7;
- the learned sparse correspondences and compact nonrigid parameterization allows the system to handle fast motion. The energy terms, including the data term (Equation 4) and the color term (Equation 8), have a smaller convergence basin, and thus optimizing the problem with these terms requires a good initialization to converge. The motion field from last frame is used for initialization, which limits the system to support fast motion. The sparse correspondences directly give the optimization problem a bigger convergence basin, while the compact nonrigid parameterization improves the system performance and enables the system to consume more data to improve tracking stability for fast motion. In Figure 12, we test our system by triggering the cameras at high frame rates of 100fps and the more typically used 30fps. At low framerate fast (casual) motion such as swinging the arms up and down can result many reconstruction artifacts due to tracking failures. Our results show that both high frame rate and the sparse correspondences improve the tracking quality.

Note that above new components help the alignment only for the aforementioned specific cases. The two-way nonrigid matching and the color term do not help in the case of fast motion, while the improved performance does not help in the case of surface topology change.

Our fusion component avoids the space compression problem (Figure 8), and the detail la yer adds more geometry details (Figure 9). The atlas mapping component provides better texture compared with per-vertex color method (Figure 10). More results are shown in the accompany video.

Fig. 17. Qualitative comparison. *Several examples that demonstrate the quality and fidelity of our results despite the challenging input.*

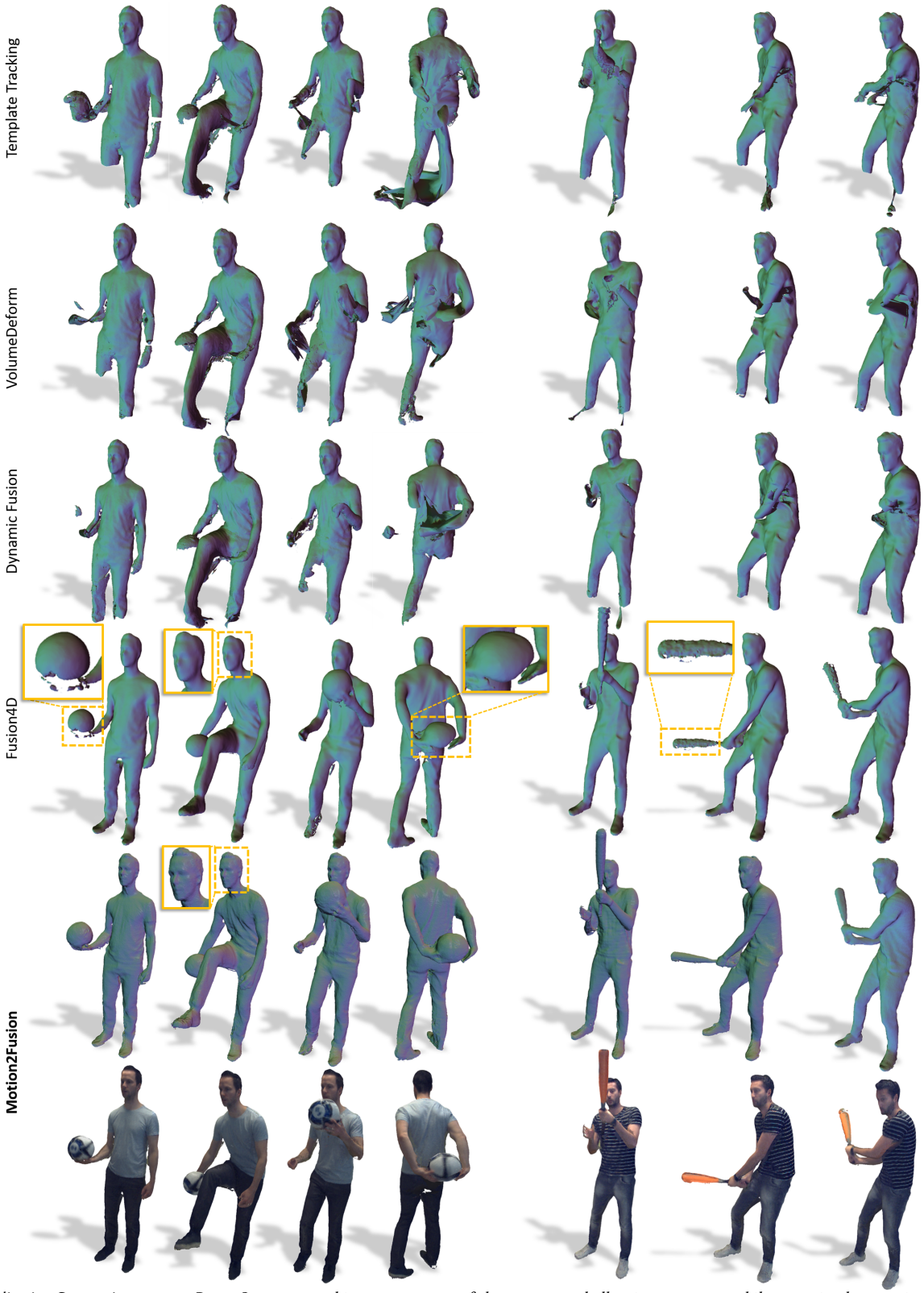

Fig. 18. Qualitative Comparison on our Data. *Our system achieves a new state-of-the-art on our challenging sequences, while our re-implementations of previous work either fails or over-smooths, compromising the fidelity.*

Robustness to number of viewpoints. While on one hand there are a number of offline systems such as [Collet et al. 2015a] that have leverage a huge number of viewpoints. In our multi-view capture system we use 8 different viewpoints similar to [Dou et al. 2016] in order to allow for good coverage of the 3D space they encompass. However, as we show in Figure 11 we achieve good fidelity reconstructions even in case of a single camera albeit not complete. In Figure 16 we show even by halving the number of viewpoints from 8 to 4 views within our setup Motion2Fusion continues to obtain high fidelity detailed reconstructions with minimal missing data such as the legs, demonstrating the robustness of our approach to the number of viewpoints.

Limitations. Despite the unprecedented fidelity we can achieve in real-time, our system still has limitations. For instance, while we skin mesh vertices properly to geodesically close ED nodes, we still skin voxels to ED nodes that are close in the Euclidean sense. We chose this compromise mainly for performance reasons, and it led to some artifacts when the surface topology changes. Although our system runs at a high framerate, it still cannot utilize framerates of some recent depth capture algorithms (e.g, 200fps+ in [Fanello et al. 2014, 2016, 2017a,b]). Our system drops frames when the reconstruction falls behind depth capture, but these dropped frames could be helpful for extremely fast motion capture. Ideally, low-cost motion field computations could be distributed to each depth computation device, and the final reconstruction system collects these motion fields and performs the final refinement and data fusion.

6 CONCLUSION

We presented Motion2Fusion, a state-of-the-art volumetric performance capture system. We have demonstrated the efficacy of Motion2Fusion over prior work in the high fidelity reconstruction of dynamic objects while avoiding over-smoothing and retaining high frequency geometric details. This is key to avoid the uncanny valley in telepresence and VR/AR scenarios. We developed a high speed pipeline coupled with a novel machine learning framework for 3D correspondence field estimation, which reduces tracking error and artifacts induced by fast motion. We also introduce a backward and forward non-rigid alignment strategy to address complex topology changes. Through extensive qualitative and quantitative comparisons we show that Motion2Fusion achieves more precise geometric and texturing results with less artifacts on extremely challenging sequences compared to the previous state-of-the-art. Pushing on more robust reconstructions with less infrastructure is a clear highly impactful area of future work, to bring such technologies to even more commodity end user scenarios.

REFERENCES

Christian Bailer, Bertram Taetz, and Didier Stricker. 2015. Flow Fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *Proceedings of the IEEE International Conference on Computer Vision*. 4015–4023.

Ilya Baran and Jovan Popović. 2007. Automatic Rigging and Animation of 3D Characters. *ACM TOG* 26, 3 (2007), 72.

Chen Cao, Derek Bradley, Kun Zhou, and Thabo Beeler. 2015. Real-time high-fidelity facial performance capture. *ACM TOG* 34, 4 (2015), 46.

Chen Cao, Yanlin Weng, Stephen Lin, and Kun Zhou. 2013. 3D Shape Regression for Real-time Facial Animation. *ACM TOG* 32, 4, Article 41 (2013), 10 pages.

Nathan A Carr and John C Hart. 2002. Meshed atlases for real-time procedural solid texturing. *ACM TOG* 21, 2 (2002), 106–131.

Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. 2015a. High-quality Streamable Free-viewpoint Video. *ACM TOG* (2015).

Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. 2015b. High-quality streamable free-viewpoint video. *ACM TOG* 34, 4 (2015), 69.

Brian Curless and Marc Levoy. 1996. A volumetric method for building complex models from range images. In *SIGGRAPH*. 303–312.

Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, and Shahram Izadi. 2016. Fusion4D: Real-time Performance Capture of Challenging Scenes. *ACM TOG* 35, 4 (2016), 114.

Sean Ryan Fanello, Cem Keskin, Shahram Izadi, Pushmeet Kohli, David Kim, David Sweeney, Antonio Criminisi, Jamie Shotton, Sing Bing Kang, and Tim Paek. 2014. Learning to be a depth camera for close-range human capture and interaction. In *ACM Transactions on Graphics (TOG)*.

Sean Ryan Fanello, Christoph Rhemann, Vladimir Tankovich, A Kowdle, S Orts Escolano, D Kim, and S Izadi. 2016. Hyperdepth: Learning depth from structured light without matching. In *CVPR*.

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 *ICCV*.

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 *CVPR*.

Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. 2015. FlowNet: Learning Optical Flow with Convolutional Networks. In *ICCV*. 2758–2766.

Kaiwen Guo, Feng Xu, Yangang Wang, Yebin Liu, and Qionghai Dai. 2015. Robust Non-Rigid Motion Tracking and Surface Reconstruction Using L0 Regularization. In *ICCV*. 3083–3091.

Kaiwen Guo, Feng Xu, Tao Yu, Xiaoyang Liu, Qionghai Dai, and Yebin Liu. 2017. Real-time Geometry, Albedo and Motion Reconstruction Using a Single RGBD Camera. *ACM Transactions on Graphics (TOG)* (2017).

Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. 2016. VolumeDeform: Real-time volumetric non-rigid reconstruction. In *ECCV*. 362–379.

Varun Jain and Hao Zhang. 2006. Robust 3D Shape Correspondence in the Spectral Domain. In *SMA*. 19–19.

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2007. Skinning with dual quaternions. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*. ACM, 39–46.

S. Lazebnik, C. Schmid, and J. Ponce. 2006. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *CVPR*, Vol. 2. 2169–2178.

Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. 2009. An Integer Projected Fixed Point Method for Graph Matching and MAP Inference. In *NIPS*.

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM TOG* 21, 3 (2002), 362–371.

Hao Li, Bart Adams, Leonidas J Guibas, and Mark Pauly. 2009. Robust single-view geometry and motion reconstruction. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 175.

Hao Li, Etienne Vouga, Anton Gudym, Linjie Luo, Jonathan T Barron, and Gleb Gusev. 2013. 3D self-portraits. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 187.

Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. 2017. VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera. *ACM Transactions on Graphics* 36, 4, 14. https://doi.org/10.1145/3072959.3073596

Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. 2002. Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and mathematics* 3, 2 (2002), 52–58.

Richard A Newcombe, Dieter Fox, and Steven M Seitz. 2015. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 343–352.

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 *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, 127–136.

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*. ACM, 741–754.

Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2016. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*.

Ali Rahimi and Benjamin Recht. 2007. Random Features for Large-scale Kernel Machines. In *NIPS*. 5.

Pedro V Sander, John Snyder, Steven J Gortler, and Hugues Hoppe. 2001. Texture mapping progressive meshes. In *SIGGRAPH*. ACM, 409–416.

Alla Sheffer and John C Hart. 2002. Seamster: inconspicuous low-distortion texture seam layout. In *Visualization*. 291–298.

Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. 2013. Real-time human pose recognition in parts from single depth images. *Commun. ACM* 56, 1 (2013), 116–124.

Marc Soucy, Guy Godin, and Marc Rioux. 1996. A texture-mapping approach for the compression of colored 3D triangulations. *The Visual Computer* 12, 10 (1996), 503–514.

Robert W Sumner, Johannes Schmid, and Mark Pauly. 2007. Embedded deformation for shape manipulation. *ACM TOG* 26, 3 (2007), 80.

David Joseph Tan, Thomas Cashman, Jonathan Taylor, Andrew Fitzgibbon, Daniel Tarlow, Sameh Khamis, Shahram Izadi, and Jamie Shotton. 2016. Fits Like a Glove: Rapid and Reliable Hand Shape Personalization. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, Arran Topalian, Erroll Wood, Sameh Khamis, Pushmeet Kohli, Shahram Izadi, Richard Banks, Andrew Fitzgibbon, and Jamie Shotton. 2016. Efficient and Precise Interactive Hand Tracking Through Joint, Continuous Optimization of Pose and Correspondences. *SIGGRAPH* (2016).

C. Theobalt, E. de Aguiar, C. Stoll, H.-P. Seidel, and S. Thrun. 2010. Performance Capture from Multi-view Video. In *Image and Geometry Processing for 3D-Cinematography*, R. Ronfard and G. Taubin (Eds.). Springer, 127ff.

J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. 2016. Face2Face: Real-time Face Capture and Reenactment of RGB Videos. In *CVPR*.

Shenlong Wang, Sean Ryan Fanello, Christoph Rhemann, Shahram Izadi, and Pushmeet Kohli. 2016. The Global Patch Collider. In *CVPR*. 127–135.

Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert. 2008. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis* 25, 3 (2008), 335–366.

Jin Xie, Yi Fang, Fan Zhu, and Edward Wong. 2015. Deepshape: Deep learned shape descriptor for 3D shape matching and retrieval. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1275–1283.

Mao Ye and Ruigang Yang. 2014. Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera. In *CVPR*. IEEE.

Mao Ye, Qing Zhang, Liang Wang, Jiejie Zhu, Ruigang Yang, and Juergen Gall. 2013. A survey on human motion analysis from depth data. In *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*. Springer, 149–187.

Sergey Zagoruyko and Nikos Komodakis. 2015. Learning to Compare Image Patches via Convolutional Neural Networks. In *CVPR*. 4353–4361.

Mikhail Zaslavskiy, Francis Bach, and Jean-Philippe Vert. 2009. A Path Following Algorithm for the Graph Matching Problem. *PAMI* 31, 12 (2009), 2227–2242.

Jure Žbontar and Yann LeCun. 2015. Computing the stereo matching cost with a convolutional neural network. In *CVPR*. 1592–1599.

F. Zhou and F. De la Torre. 2012. Factorized graph matching. In *CVPR*. 127–134.

Kun Zhou, John Synder, Baining Guo, and Heung-Yeung Shum. 2004. Iso-charts: stretch-driven mesh parameterization using spectral analysis. In *SGP*. 45–54.

Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rhemann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, et al. 2014. Real-time non-rigid reconstruction using an RGB-D camera. *ACM TOG* 33, 4 (2014), 156.