This is a previous version of the article published in ACM Transactions on Graphics. 2016, 35(4): Article 114. doi:10.1145/2897824.2925969

Online Submission ID:

Fusion4D: Real-time Performance Capture of Challenging Scenes

Figure 1: *We present a new method for real-time high quality 4D (i.e. spatio-temporally coherent) performance capture, allowing for incremental nonrigid reconstruction from noisy input from multiple RGBD cameras. Our system demonstrates unprecedented reconstructions of challenging nonrigid sequences, at real-time rates, including robust handling of large frame-to-frame motions and topology changes.*

Abstract

We contribute a new pipeline for live multi-view performance capture, generating temporally coherent high-quality reconstructions in real-time. Our algorithm supports both incremental reconstruction, improving the surface estimation over time, as well as parameterizing the nonrigid scene motion. Our approach is highly robust to both large frame-to-frame motion and topology changes, allowing us to reconstruct extremely challenging scenes. We demonstrate advantages over related real-time techniques that either deform an online generated template or continually fuse depth data nonrigidly into a single reference model. Finally, we show geometric reconstruction results on par with offline methods which require orders of magnitude more processing time and many more RGBD cameras.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Digitizing and Scanning

Keywords: nonrigid, real-time, 4D reconstruction, multi-view

1 Introduction

Whilst *real-time* 3D reconstruction has "come of age" in recent years with the ubiquity of RGBD cameras, the majority of systems still focus on static, non-moving, scenes. This is due to computational and algorithmic challenges in reconstructing scenes under nonrigid motion. In contrast to rigid scenes where motion is encoded by a single 6DoF (six degrees of freedom) pose, the nonrigid case requires solving for orders of magnitude more parameters in real-time. Whereas both tasks must deal with noisy or missing data, and handle occlusions and large frame-to-frame motions, the nonrigid case is further complicated by changing scene topology – e.g. a person removing a worn jacket or interlocked hands separating apart.

Despite these challenges, there is clear value in reconstructing nonrigid motion and surface deformations in *real-time*. In particular, *performance capture*, where multiple cameras are used to reconstruct human motion and shape, and even object interactions, is currently constrained to offline processing: people interact in a scene and then expect hours of processing time before seeing the final result. What if this processing could happen *live* in real-time directly as the performance is happening? This can lead to new real-time experiences such as the ability to watch a remote concert or sporting event live in full 3D, or even the ability to communicate in real-time with remotely captured people using immersive AR/VR displays.

However, despite remarkable progress in offline performance capture over the years (see [Theobalt et al. 2010; Ye et al. 2013; Smolic 2011] for surveys), real-time approaches have been incredibly rare, especially when considering high quality reconstruction of general shape and motion i.e. without a strong prior on the human body. Recent work has demonstrated compelling real-time reconstructions of general nonrigid scenes using a single depth camera [Zollhöfer et al. 2014; Newcombe et al. 2015]. Our motivation, however, differs to these systems as we focus on robust real-time performance capture across *multiple* views. As quantified later in this paper, this prior work cannot meet our requirements for real-time performance capture for two main reasons. First these systems rely on a *reference model* that is used for model fitting e.g. Zollhöfer *et al.* [2014] use a statically captured reference model, i.e. *template*, and Newcombe *et al.* [2015] use a volumetric model that is incrementally updated with new depth input. Ultimately, this reference model regularizes the model fitting, but can also overly constrain it so that major changes in shape and topology are hard to accommodate. Second, these systems find correspondences by assuming small frame-to-frame motions, which makes the nonrigid estimation brittle in the presence of large movements.

We contribute Fusion4D, a new pipeline for live multi-view performance capture, generating temporally coherent high-quality reconstructions in real-time, with several unique capabilities over this prior work: (1) We make no prior assumption regarding the captured scene, operating without a skeleton or template model, allowing reconstruction of arbitrary scenes; (2) We are highly robust to both large frame-to-frame motion and topology changes, allowing reconstruction of extremely challenging scenes; (3) We scale to multi-view capture from multiple RGBD cameras, allowing for performance capture at qualities never before seen in real-time systems.

Fusion4D combines the concept of volumetric fusion with estimation of a smooth deformation field across RGBD views. This enables both incremental reconstruction, improving the surface estimation over time, as well as parameterization of nonrigid scene motion. Our approach robustly handles large frame-to-frame motion by using a novel, fully parallelized, nonrigid registration framework, including a learning-based RGBD correspondence matching regime. It also robustly handles topology changes, by switching between reference models to better explain the data over time, and robustly blending between data and reference volumes based on correspondence estimation and alignment error. We compare to related work and show several clear improvements over real-time approaches that either track an online generated template or fuse depth data into a single reference model incrementally. Further, we show geometric reconstruction results on-par with offline methods which require orders of magnitude more processing time and many more RGBD cameras.

2 Related Work

Multi-view Performance Capture: Many compelling offline performance capture systems have been proposed. Some specifically model complex human motion and dynamic geometry, including people with general clothing, possibly along with pose parameters of an underlying kinematic skeleton (see [Theobalt et al. 2010] for a full review). Some methods employ variants of shape-from-silhouette [Waschbüsch et al. 2005] or active or passive stereo [Starck and Hilton 2007]. Template-based approaches deform a static shape model such that it matches a human [de Aguiar et al. 2008; Vlasic et al. 2008; Gall et al. 2009] or a person's clothing [Bradley et al. 2008]. Vlasic et al. [2009] use a sophisticated photometric stereo light stage with multiple high-speed cameras to capture geometry of a human at high detail. Dou et al. [2013] capture precise surface deformations using an eight-Kinect rig, by deforming a human template, generated from a KinectFusion scan, using embedded deformation [Sumner et al. 2007]. Other methods jointly track a skeleton and the nonrigidly deforming surface [Vlasic et al. 2008; Gall et al. 2009].

Whilst compelling, these multi-camera approaches require considerable compute and are orders of magnitude slower than real-time, also requiring dense camera setups in controlled studios, with sophisticated lighting and/or chroma-keying for background subtraction. Perhaps the high-end nature of these systems is exemplified by [Collet et al. 2015] which uses over 30 RGBD cameras and a large studio setting with green screen and controlled lighting, producing extremely high quality results, but at approximately 30 seconds per frame. We compare to this system later, and demonstrate comparable results in real-time with a greatly reduced set of RGBD cameras.

Accommodating General Scenes: The approach of [Li et al. 2009] uses a coarse approximation of the scanned object as a shape prior to obtain high quality nonrigid reconstructions of general scenes. Others also treat the template as a generally deformable shape without skeleton and use volumetric [de Aguiar et al. 2008] or patch-based deformation methods [Cagniart et al. 2010]. Other nonrigid techniques remove the need for a shape or template prior, but assume small and smooth motions [Zeng et al. 2013; Wand et al. 2009; Mitra et al. 2007]; or deal with topology changes in the input data (e.g., the fusing and then separation of hands) but suffer from drift and over-smoothing of results for longer sequences [Tevs et al. 2012; Bojsen-Hansen et al. 2012]. [Guo et al. 2015; Collet et al. 2015] introduce the notion of keyframe-like transitions in offline nonrigid reconstructions, to accommodate topology changes and tracking failures. [Dou et al. 2015] demonstrate a compelling offline system with nonrigid variants of loop closure and bundle adjustment to create compelling scans of arbitrary scenes without a prior human or template model. All these more general techniques are far from real-time, ranging from seconds to hours per frame.

Real-time Approaches: Only recently have we seen real-time nonrigid reconstruction systems appear. Approaches fall into three categories. *Single object parametric* approaches focus on a single object of interest, e.g. face, hand, or body, which is parametrized ahead of time in an offline manner, and tracked or deformed to fit the data in real-time. Compelling real-time reconstructions of nonrigid articulated motion (e.g. [Ye et al. 2013; Stoll et al. 2011; Zhang et al. 2014]) and shape (e.g. [Ye et al. 2013; Ye and Yang 2014]) have been demonstrated. However by their very nature, these approaches rely on strong priors based on either pre-learned statistical models, articulated skeletons, or morphable shape models, prohibiting capture of arbitrary scenes or objects. Often the parametric model is not rich enough to capture challenging poses or all types of shape variation. For human bodies, even with extremely rich offline shape and pose models [Bogo et al. 2015], reconstructions can suffer from the effect of uncanny valley [Mori et al. 2012]; and clothing or hair can prove problematic [Bogo et al. 2015].

Recently, real-time *template-based* reconstruction of more diverse nonrigidly moving objects was demonstrated [Zollhöfer et al. 2014]. Here an online template model was captured statically, and deformed in real-time to fit the data captured from a novel RGBD sensor. Additionally, displacements on this tracked surface model were computed from the input data and fused over time. Despite impressive real-time results, this work still requires a template to be first acquired rigidly, making it impractical for capturing children, animals or other objects that rarely hold still. Furthermore, the template model is fixed and so any scene topology change will break the fitting. Such approaches also rely heavily on closest point correspondences [Rusinkiewicz and Levoy 2001] and are not robust to large frame-to-frame motions. Finally in both template based and single object parametric approaches the model is *fixed*, and the aim is to deform or articulate the model to explain the data rather than incrementally *reconstruct* the scene. This means that new input data does not refine the reconstructed model over the time.

DynamicFusion [Newcombe et al. 2015] addresses some of the challenges inherent in template-based reconstruction techniques by demonstrating compelling results of *nonrigid volumetric fusion* using a single Kinect sensor. The reference surface model is incrementally updated based on new depth measurements, refining and completing the model over time. This is achieved by warping a reference volume nonrigidly to each new input frame, and fusing depth samples into the model. However, as shown in the supplementary video of this work the frame-to-frame motions are slow and carefully orchestrated, again due to reliance on closest point correspondences. Also, the reliance on a single volume registered to a single point in time means that the current data being captured cannot represent a scene dramatically different from the model. This makes fitting the model to the data and incorporating it back into the model more challenging. Gross inconsistencies between the reference volume and data can result in tracking failures. For example, if the reference model is built with a user's hands fused together, estimation of the deformation field will fail when the hands are seen to separate in the data. In practice, these types of topology changes occur often as people interact in the scene.

3 System Overview

Our work, Fusion4D, attempts to bring aspects inherent in multi-view performance capture systems to real-time scenarios. In so doing, we need to design a new pipeline that addresses the limitations outlined in current real-time nonrigid reconstruction systems. Namely, we need to be robust to fast motions and topology changes and support multi-view input, whilst still maintaining real-time rates.

Fig. 2 shows the main system pipeline. We accumulate our 3D reconstruction in a hierarchical voxel grid and employ volumetric fusion [Curless and Levoy 1996] to denoise the surface over time (Sec. 6). Unlike existing real-time approaches, we use the concept of *key volumes* to deal with radically different surface topologies over time (Sec. 6). This is a voxel grid that maintains the reference model, and ensures smooth nonrigid motions within the key volume sequence, but allows more drastic changes across key volumes. This is conceptually similar to the concept of a keyframe or anchor frame used in nonrigid tracking [Guo et al. 2015; Collet et al. 2015; Beeler et al. 2011], but this concept is extended for online nonrigid volumetric reconstruction.

We take multiple RGBD frames as input and first estimate a segmentation mask per camera (Sec. 4). A dense correspondence field is estimated per separate RGBD frame using a novel learning-based technique (Sec. 5.2.4). This correspondence field is used to initialize the nonrigid alignment, and allows for robustness to fast motions –

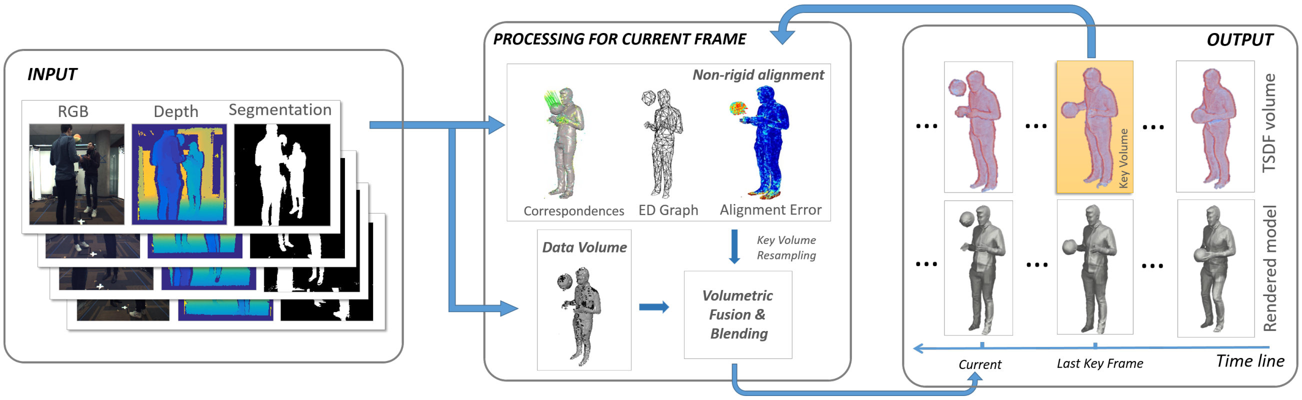

Figure 2: *The Fusion4D pipeline. Please see text in Sec. 3 for details.*

a failure case when closest point correspondences are assumed as in [Zollhöfer et al. 2014; Newcombe et al. 2015].

Next is nonrigid alignment, where we estimate a deformation field to warp the current key volume to the data. We cover the details of this step in Sec. 5. In addition to fusing data into the key (or reference) volume as in [Newcombe et al. 2015], we also fuse the currently accumulated model into the data volume by warping and resampling the key volume. This allows Fusion4D to be more responsive to new data, whilst allowing more conservative model updates. Nonrigid alignment error and the estimated correspondence fields can be used to guide the fusion process, allowing for new data to appear very quickly when occluded regions, topology changes, or tracking failures occur, but also allowing fusion into the model over time.

4 Raw Depth Acquisition and Preprocessing

In terms of acquisition our setup is similar to [Collet et al. 2015], but with a reduced number of cameras and no green screen. Our system, in its most general form, produces N depthmaps using $2N$ monocular infrared (IR) cameras and N RGB images used only to provide texture information. Whereas the setup in [Collet et al. 2015] consists of 106 cameras producing 24 depthmaps, our setup uses only 24 cameras, producing $N = 8$ depthmaps and RGB images. All of our cameras are in a trinocular configuration and have a 1 megapixel output resolution. Depth estimation is carried out using the PatchMatch Stereo algorithm [Bleyer et al. 2011], which runs in real-time on GPU hardware (see [Zollhöfer et al. 2014] and [Pradeep et al. 2013] for more details).

A segmentation step follows the depth computation algorithm, where 2D silhouettes of the regions of interest are produced. The segmentation mask plays a crucial role in estimating the visual hull constraint (see Sec. 5.2.3) that helps ameliorate issues with missing data in the input depth and ensures that foreground data is not deleted from the model. Our segmentation also avoids the need for a green screen setup as in [Collet et al. 2015] and allows capture in natural and realistic settings. In our pipeline we employed a simple background model (using both RGB and depth cues) that does not take into account temporal consistency. This background model is used to compute unary potentials by considering pixel-wise differences with the current scene observation. We then use a dense Conditional Random Field (CRF) [Krähenbüh and Koltun 2011] model to enforce smoothness constraints between neighboring pixels. Due to our real-time requirements, we use an approximate GPU implementation similar to [Vineet et al. 2012].

5 Nonrigid Motion Field Estimation

In each frame we observe N depthmaps, $\{\mathbb{D}_n\}_{n=1}^N$ and N foreground masks, $\{\mathbb{S}_n\}_{n=1}^N$. As is common [Curless and Levoy 1996; Newcombe et al. 2011; Newcombe et al. 2015], we accumulate this depth data into a non-parametric surface represented implicitly by a truncated signed distance function (TSDF) or volume $\mathbb{V}$ in some "reference frame" (which we denote as *key volume*). This allows efficient alignment and allows for all the data to be averaged into a complete surface with greatly reduced noise. Further, the zero crossings of the TSDF can be easily located to extract a high quality mesh[1] $\mathbf{V} = \{\mathbf{v}_m\}_{m=1}^M \subseteq \mathbb{R}^3$ with corresponding normals $\{\mathbf{n}_m\}_{m=1}^M$. The goal of this section is to show how to estimate a deformation field that warps the key volume $\mathbb{V}$ or the mesh $\mathbf{V}$ to align with the raw depth maps $\{\mathbb{D}_n\}_{n=1}^N$. We typically refer $\mathbb{V}$ or $\mathbf{V}$ as model, and $\{\mathbb{D}_n\}_{n=1}^N$ as data.

5.1 Deformation Model

Following [Li et al. 2009] and [Dou et al. 2015] we choose the embedded deformation (ED) model of [Sumner et al. 2007] to parameterize the nonrigid deformation field. Before processing each new frame, we begin by uniformly sampling a set of K "ED nodes" within the reference volume by sampling locations $\{\mathbf{g}_k\}_{k=1}^K \subseteq \mathbb{R}^3$ from the mesh $\mathbf{V}$ extracted from this volume. Every vertex $\mathbf{v}_m$ in that mesh is then "skinned" to its closest ED nodes $\mathcal{S}_m \subseteq \{1, ..., K\}$ using a set of fixed skinning weights $\{w_k^m : k \in \mathcal{S}_m\} \subseteq [0, 1]$ calculated as $w_k^m = \frac{1}{Z}\exp\left(\|\mathbf{v}_m - \mathbf{g}_k\|^2/2\sigma^2\right)$, where Z is a normalization constant ensuring that, for each vertex, these weights add to one. Here σ defines the effective radius of the ED nodes, which we set as $\sigma = 0.5d$, where d is the average distance between neighboring ED nodes after the uniform sampling.

We then represent the local deformation around each ED node $\mathbf{g}_k$ using an affine transformation $A_k \in \mathbb{R}^{3\times3}$ and a translation $\mathbf{t}_k \in \mathbb{R}^3$. In addition, a global rotation $R \in SO(3)$ and translation $T \in \mathbb{R}^3$ are added. The set $G = \{R, T\} \cup \{A_k, \mathbf{t}_k\}_{k=1}^K$ fully parameterizes the deformation that warps any point $\mathbf{v} \in \mathbb{R}^3$ to

$$\mathcal{T}(\mathbf{v}_m; G) = R \sum_{k\in\mathcal{S}_m} w_k^m \left[A_k(\mathbf{v} - \mathbf{g}_k) + \mathbf{g}_k + \mathbf{t}_k\right] + T. \quad (1)$$

Equally, a normal $\mathbf{n}$ will be transformed to

$$\mathcal{T}^{\perp}(\mathbf{n}_m; G) = R \sum_{k\in\mathcal{S}_m} w_k^m A_k^{-T} \mathbf{n}_m, \quad (2)$$

[1] A triangulation is also extracted which we use for rendering.

and normalization is applied afterwards.

5.2 Energy Function

To estimate the parameters G, we formulate an energy function $E(G)$ that penalizes the misalignment between our model and the observed data, regularizes the types of allowed deformations and encodes other priors and constraints. The energy function

$$\begin{aligned} E(G) =& \lambda_{\text{data}} E_{\text{data}}(G) + \lambda_{\text{hull}} E_{\text{hull}}(G) + \lambda_{\text{corr}} E_{\text{corr}}(G) + \\ & \lambda_{\text{rot}} E_{\text{rot}}(G) + \lambda_{\text{smooth}} E_{\text{smooth}}(G) \end{aligned} \tag{3}$$

consists of a variety of terms that we systematically define below.

5.2.1 Data Term

The most crucial portion of our energy formulation is a data term that penalizes misalignments between the deformed model and the data. In its most natural form, this term would be written as

$$\hat{E}_{\text{data}}(G) = \sum_{n=1}^{N} \sum_{m=1}^{M} \min_{\mathbf{x} \in \mathcal{P}(\mathbb{D}_n)} \|\mathcal{T}(\mathbf{v}_m; G) - \mathbf{x}\|^2 \tag{4}$$

where $\mathcal{P}(\mathbb{D}_n) \subseteq \mathbb{R}^3$ extracts a point cloud from depth map $\mathbb{D}_n$. We, however, approximate this using a projective point-to-plane term as

$$E_{\text{data}}(G) = \sum_{n=1}^{N} \sum_{m \in \mathcal{V}_n(G)} \left(\tilde{\mathbf{n}}_m(G)^\top \left(\tilde{\mathbf{v}}_m(G) - \Gamma_n(\tilde{\mathbf{v}}_m(G)) \right) \right)^2 \tag{5}$$

where $\tilde{\mathbf{n}}_m(G) = \mathcal{T}^\perp(\mathbf{n}_m; G)$ and $\tilde{\mathbf{v}}_m(G) = \mathcal{T}(\mathbf{v}_m; G)$ (with slight notational abuse we simply use $\tilde{\mathbf{v}}$ and $\tilde{\mathbf{n}}$ to represent the warped points and normals); $\Gamma_n(\mathbf{v}) = P_n(\Pi_n(\mathbf{v}))$, with $\Pi_n : \mathbb{R}^3 \to \mathbb{R}^2$ projecting a point into the n'th depth map and $P_n : \mathbb{R}^2 \to \mathbb{R}^3$ back-projecting the corresponding pixel in $\mathbb{D}_n$ into 3D; and $\mathcal{V}_n(G) \subseteq \{1, ..., M\}$ are vertex indices that are considered to be "visible" in view n when the model is deformed using G. In particular, we consider a vertex to be visible if

$$\begin{aligned} & \Pi_n(\tilde{\mathbf{v}}_m) \text{ is a valid and visible pixel in view } n \text{ and} \\ & \|\tilde{\mathbf{v}}_m - P_n(\Pi_n(\tilde{\mathbf{v}}_m))\| \le \epsilon_d \text{ and} \\ & \tilde{\mathbf{n}}_m^\top \, P_n^\perp(\Pi_n(\tilde{\mathbf{v}}_m)) < \epsilon_n \end{aligned}$$

where $P_n^\perp : \mathbb{R}^2 \to \mathbb{R}^3$ maps pixels to normal vectors estimated from $\mathbb{D}_n$; ϵ_d and ϵ_n are the truncation thresholds for depth and normal respectively.

Although (5) is an approximation to (4), it offers a variety of key benefits. First, the use of a point-to-plane term is a well known strategy to speed up convergence [Chen and Medioni 1992]. Second, the use of a "projective correspondence" avoids the expensive minimization in (4). Lastly, the visibility set $\mathcal{V}_n(G)$ is explicitly computed to be robust to outliers which avoids employing a robust data term here that often slows Gauss-Newton like methods [Zach 2014]. Interestingly, the last two points interfere with the differentiability of (5) as $P_n(\Pi_n(\tilde{\mathbf{v}}))$ jumps as the projection crosses pixel boundaries and $\mathcal{V}(G)$ undergoes discrete modifications as G changes. Nonetheless, we use a further approximation (see Sec. 5.3) at each Gauss-Newton iteration whose derivative both exists everywhere and is more efficient to compute.

5.2.2 Regularization Terms

As the deformation model above could easily represent unreasonable deformations, we follow [Dou et al. 2015] by deploying two regularization terms to restrict the class of allowed deformations. The first term

$$E_{\text{rot}}(G) = \sum_{k=1}^{K} \|A_k^T A_k - \mathbf{I}\|_F + \sum_{k=1}^{K} (\det(A_k) - 1)^2. \tag{6}$$

encourages each local deformation to be close to a rigid transform. The second encourages the neighboring affine transformations to be similar as

$$E_{\text{smooth}}(G) = \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_k} w_{jk} \rho(\|A_j(\mathbf{g}_k - \mathbf{g}_j) + \mathbf{g}_j + \mathbf{t}_j - (\mathbf{g}_k + \mathbf{t}_k)\|^2) \tag{7}$$

where $w_{jk} = \exp(-\|\mathbf{g}_k - \mathbf{g}_j\|^2 / 2\sigma^2)$ is a smoothness weight that is inversely proportional to the distance between two neighboring ED nodes, and σ is set to be the average distance between all pairs of neighboring ED nodes. Here $\mathcal{N}_k$ denotes the set of ED nodes neighboring node k, and $\rho(\cdot)$ is a robustifier to allow for discontinuities in the deformation field.

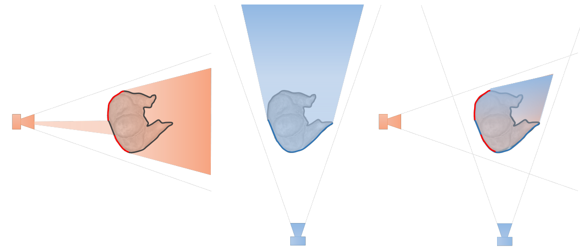

Figure 3: *An illustration of visual hull in our optimization. Left: the first camera's visual hull (shaded region) is defined by the foreground segmentation and the observed data (red line on the surface). In this case, a hole on foreground causes the hull to extend all the way to the camera. Our energy penalizes the surface (*e.g.*, those drawn in black) from erroneously moving outside of the visual hull into known free-space. Middle: A second camera can be added which gives a different visual hull constraint. Right: The intersection of multiple visual hulls yield increasingly strong constraints on where the entire model must lie.*

5.2.3 Visual Hull Term

The data term above only constrains the deformation when the warped model is close to the data. To see why this is problematic, let us assume momentarily the best case scenario where we happen to have a perfect model that should be able to fully "explain" the data. If a piece of the model is currently being deformed to a location outside the truncation threshold of depth maps, the gradient will be zero. Another, more fundamental issue, is that a piece of the model that is currently unobserved (*e.g.* a hand hidden behind a user's back) is allowed to enter free-space. This occurs despite the fact that we *know* that free-space should not be occupied as we have *observed* it to be free. Up until now, other methods [Newcombe et al. 2015] have generally ignored this constraint, or equivalently their model has only been forced to explain the foreground data while ignoring "negative" background data.

To address this, we formulate an additional energy term that encodes the constraint that the deformed model lies within the visual hull. The visual hull is a concept used in shape-from-silhouette space-carving reconstruction techniques [Kutulakos and Seitz 2000]. Typically in 2D it is defined as the intersection of the cones cut-out by the back-projection of an object's silhouette into free-space.

The near-camera side of the back-projected cone that each silhouette generates is cut using the observed depth data (see Fig. 3) before being intersected. In the single viewpoint scenario, where there is much occlusion, the constraint helps push portions of the deformed model corresponding to the visual hull of the true scene into occluded regions. In the multiview case (see Fig. 3), occlusion are less ubiquitous, and the term is able to provide constraints in free space where data is missing. For example, depth sensors often struggle to observe data on a user's hair and yet a multi-view visual hull constraint will still provide a tight bounding box on where the head should lie. Without this term, misalignment will be more pronounced making the accumulation of highly noisy data prohibitive.

The visual hull can be represented as an occupancy volume $\mathbb{H}$ with values of 1 inside the visual hull and 0 outside. Each voxel of $\mathbb{H}$ is projected to each depthmap and set to 0 if it is in the background mask or closer to the camera than the depth pixel that it is projected onto. To be conservative, we set a voxel as occupied if it is in front of an invalid foreground depth pixel. To apply the visual hull constraint in the form of a cost function term, we first calculate an approximate distance transform $\mathcal{H}$ to the visual hull, where the distance would be 0 for space inside the hull. The visual hull term is written as

$$E_{\text{hull}}(G) = \sum_{m=1}^{M} \mathcal{H}(\mathcal{T}(\mathbf{v}_m; G))^2. \tag{8}$$

The exact computation of $\mathcal{H}$ is computationally expensive and unsuitable for a real-time setting. Instead, we approximate $\mathcal{H}$ by applying Gaussian blur to the occupancy volume[2], which is implemented efficiently on the GPU.

5.2.4 Correspondence Term

Finding the 3D motion field of nonrigid surfaces is an extremely challenging task. Approaches relying on non-convex optimization can easily end up in erroneous local optima due to bad starting points caused by noisy and inconsistent input data e.g. due to large motions. A key role is played by the initial alignment of the current input data $\mathbb{D}_n$ and the model. Our aim is therefore to find point-wise correspondences to provide a robust initialization for the solver. Finding reliable matches between images has been exhaustively studied; recently, deep learning techniques have shown superior performance [Weinzaepfel et al. 2013; Revaud et al. 2015; Wei et al. 2015]. However these are computationally expensive, and currently prohibitive for real-time scenarios (even with GPU implementations).

In this paper we extend the recently proposed Global Patch Collider (GPC) [Wang et al. 2016] framework to efficiently generate accurate correspondences for RGBD data. GPC finds correspondences in linear time, avoiding the computation of costly distance functions among all the possible candidates. The method relies on decision trees which have the advantages of being fully parallelizable. Training is performed offline on held-out annotated data, and at test time, the correspondence estimation is fully integrated in the real-time system pipeline. Note, no user subject training is required.

Given two consecutive images I_s and I_t, our target is to find local correspondences between pixel positions. We consider a local patch $\mathbf{x}$ with center coordinate $\mathbf{p}$ from an image I, which is passed through a decision tree until it reaches one terminal node (leaf). The leaf node can be interpreted as a hash key for the image patch. The GPC returns as matches only pixels which end up in the same terminal node. To increase recall multiple trees are run and matches are selected as unique intersections over all the terminal nodes (see [Wang et al. 2016] for details). Correspondence estimation with decision trees is also used in [Pons-Moll et al. 2015; Shotton et al. 2013]. A key difference is that this prior work computes the correspondences with respect to a template model and only for the segmented object of interest. We, on the other hand, do not require a template model and compute the correspondences between two image frames, at a local patch level, and subsequently we are agnostic to the specific objects in the scene at both training and test time.

In [Wang et al. 2016] the authors rely on multi-scale image descriptors in order to ensure robustness to scale and perspective transformation. In this work we extend their method by making use of depth, which gives scale invariance. We also use a different strategy for the match retrieval phase based on a voting scheme. Formally, our split node contains a set of learned parameters $\delta = (\mathbf{u}, \mathbf{v}, \theta)$, where $(\mathbf{u}, \mathbf{v})$ are 2D pixel offsets and θ represents a threshold value. The split function f is evaluated at pixel $\mathbf{p}$ as

$$f(\mathbf{p}; \theta) \quad = \quad \begin{cases} \text{L} & \text{if } I_s(\mathbf{p} + \mathbf{u}/d_s) - I_t(\mathbf{p} + \mathbf{v}/d_t) < \theta \\ \text{R} & \text{otherwise} \end{cases} \tag{9}$$

where I_s and I_t are the two input RGB images and $d_s = \mathbb{D}_s(\mathbf{p})$ and $d_t = \mathbb{D}_t(\mathbf{p})$ are the depth values at the pixel coordinate $\mathbf{p}$. Normalizing these offsets by the depth of the current pixel provide invariance to scaling factors. This kind of pixel difference test is commonly used with decision forest classifiers due to its efficiency and discriminative power [Wang et al. 2016].

During training, we select the split functions to maximize the weighted harmonic mean between precision and recall of the patch correspondences. Ground truth correspondences for training the split function parameters of the decision trees are obtained via the offline but accurate nonrigid bundle adjustment method proposed by [Dou et al. 2015]. We tested different configurations of the algorithm and empirically found that 5 trees with 15 levels give the best trade-off between precision and recall. At test time, when simple pixel differences are used as features, the intersection strategy proposed in [Wang et al. 2016] is not robust due to perspective transformations of RGB images. A single tree does not have the ability to handle all possible image patch transformations. Intersection across multiple trees (as proposed in [Wang et al. 2016]) also fails to retrieve the correct match in the case of RGBD data. Only few correspondences usually belonging to small motion regions are estimated.

We address this by taking the union over all the trees, thus modeling all image transformations. However a simple union strategy generates many false positives. We solve this problem by proposing a voting scheme. Each tree with a unique collision (i.e. a leaf with only two candidates) votes for a possible match, and the one with the highest number of votes is returned. This approach generates much more dense and reliable correspondences even when large motion is present. We evaluate this method in Sec. 7.3.

This method gives us, in the n'th view, a set of F_n matches $\{u^{\text{prev}}_{nf}, u_{nf}\}_{f=1}^{N_f}$ between pixels in the current frame and the previous frame. For each match $(u^{\text{prev}}_{nf}, u_{nf})$ we can find a corresponding point $\mathbf{q}_{nf} \in \mathbb{R}^3$ in the reference frame using

$$\mathbf{q}_{nf} = \underset{\mathbf{v} \in \mathbf{V}}{\operatorname{argmin}} \, \|\Pi_n(\mathcal{T}(\mathbf{v}; G^{\text{prev}})) - u^{\text{prev}}_{nf}\| \tag{10}$$

where G^{prev} are the parameters that deform the reference surface $\mathbf{V}$ to the previous frame. We would then like to encourage these model points to deform to their 3D correspondences. To this end, we employ the the energy term

$$E_{\text{corr}}(G) = \sum_{n=1}^{N} \sum_{f=1}^{F_n} \rho(\|\mathcal{T}(\mathbf{q}_{nf}; G) - P_n(u_{nf})\|^2) \tag{11}$$

where $\rho(\cdot)$ is a robustifier to handle correspondence outliers.

[2]Followed with postprocessing, *i.e.*, applying $1.0 - \mathcal{H}$ and scaling.

5.3 Optimization

In this section, we show how to rapidly and robustly minimize $E(G)$ on the GPU to obtain an alignment between the model and the current frame. To this end, we let $\mathbf{X} \in \mathbb{R}^D$ represent the concatenation of all the parameters and let each entry of $\mathbf{f}(\mathbf{X}) \in \mathbb{R}^C$ contain each of the C unsquared terms (*i.e.* the residuals) from the energy above so that $E(G) = \mathbf{f}(\mathbf{X})^\top \mathbf{f}(\mathbf{X})$. In this form, the problem of minimizing $E(G)$ can be seen as a standard sparse non-linear least squares problem which can be solved by approaches based on the Gauss-Newton algorithm. We handle the robust terms using the square-rooting technique described in [Engels et al. 2006; Zach 2014].

For each frame we initialize all the parameters from the motion field of the previous frame. We then fix the ED nodes parameters $\{A_k, \mathbf{t}_k\}_{k=1}^K$ and estimate the global rigid motion parameters $\{R, T\}$ using projective iterative closest point (ICP) [Rusinkiewicz and Levoy 2001]. Next we fix the global rigid motion parameters and estimate the ED nodes parameters. The details of the optimization are presented in the following sections.

5.3.1 Computing a Step Direction

We compute a step direction $\mathbf{h} \in \mathbb{R}^D$ in the style of the Levenberg-Marquardt (LM) solver on the GPU. At any point X in the search space we solve for

$$(\mathbf{J}^\top \mathbf{J} + \mu \mathbf{I})\mathbf{h} = -\mathbf{J}^\top \mathbf{f} \tag{12}$$

where μ is a damping factor, $\mathbf{J} \in \mathbb{R}^{C \times D}$ is the Jacobian of $\mathbf{f}(\mathbf{X})$ and $\mathbf{f}$ is simply an abbreviation for $\mathbf{f}(\mathbf{X})$ to obtain a step direction $\mathbf{h}$. If the update will lower the energy (*i.e.* $E(\mathbf{X}+\mathbf{h}) < E(\mathbf{X})$ the step is accepted (*i.e.* $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{h}$) and the damping factor is lowered to be more aggressive. When the step is rejected, as it would raise the energy, the damping factor is raised and (12) is solved again. This behaviour can be interpreted as interpolating between an aggressive Gauss-Newton minimization and a robust gradient descent search as lowering the damping factor implicitly downscales the update as a back-tracking line search would.

Per-Iteration Approximation In order to deal with the non-differentiability of $E_{\text{data}}(G)$ and improve performance, at the start of each iteration we can take a copy of the current set of parameters $G_0 \leftarrow G$ to create a differentiable approximation to $E_{\text{data}}(G)$ as

$$\tilde{E}_{\text{data}}(G) = \sum_{n=1}^{N} \sum_{m \in \mathcal{V}_n(G_0)} \Big(\tilde{\mathbf{n}}_m(G_0)^\top \left(\tilde{\mathbf{v}}_m(G) - \Gamma_n(\tilde{\mathbf{v}}_m(G_0)) \right) \Big)^2. \tag{13}$$

In addition to being differentiable, the independence of $\tilde{\mathbf{n}}_m$ greatly simplifies the necessary derivative calculations as the derivative with respect to any parameter in G is the same for any view.

Evaluation of $\mathbf{J}^\top \mathbf{J}$ and $\mathbf{J}^\top \mathbf{f}$ In order to make this algorithm tractable for the large number of parameters we must handle, we bypass the traditional approach of evaluating and storing $\mathbf{J}$ so that it can be reused in the computation of $\mathbf{J}^\top \mathbf{J}$ and $\mathbf{J}^\top \mathbf{f}$. Instead we directly evaluate both $\mathbf{J}^\top \mathbf{J}$ and $\mathbf{J}^\top \mathbf{f}$ given the current parameters $\mathbf{X}$. In our scenario, this approach results in a dramatically cheaper memory footprint while simultaneously minimizing global memory reads and writes. This is because the number of residuals in our problem is orders of magnitude larger than the number of parameters (*i.e.* $C >> D$) and therefore the size of the Jacobian $\mathbf{J} \in \mathbb{R}^{C \times D}$ dwarfs that of $\mathbf{J}^\top \mathbf{J} \in \mathbb{R}^{D \times D}$.

Further, $\mathbf{J}^\top \mathbf{J}$ itself is a sparse matrix composed of non-zero blocks $\{\mathbf{h}_{ij} \in \mathbb{R}^{12 \times 12} : i, j \in \{1, ..., K\} \wedge i \sim j\}$ created by ordering parameter blocks from K ED nodes, where $i \sim j$ denotes that the i'th and j'th ED nodes simultaneously contribute to at least one residual. The (i, j)'th block can be computed as

$$\mathbf{h}_{ij} = \sum_{c \in \mathcal{I}_{ij}} \mathbf{j}_{ci}^\top \mathbf{j}_{cj} \tag{14}$$

where $\mathcal{I}_{ij}$ is the collection of residuals dependent on both parameter block i and j and $\mathbf{j}_{ci}$ is the gradient of c'th residual, $\mathbf{f}_c$, *w.r.t.* i-th parameter block. Note that each $\mathcal{I}_{ij}$ will not change during a step calculation (due to our approximation) so we only need to calculate each index set once. Further, the cheap derivatives of the approximation in (13) ensure that the complexity of computing $\mathbf{J}^\top \mathbf{J}$, although linearly proportional to the number of surface vertices, is independent of the number of cameras.

To avoid atomic operations on the GPU global memory, we let each CUDA block handle one $\mathbf{J}^\top \mathbf{J}$ block and perform reduction on the GPU shared memory. Similarly, $\mathbf{J}^\top \mathbf{f} \in \mathbb{R}^{D \times 1}$ can be divided into K segments, $\{(\mathbf{J}^\top \mathbf{f})_i \in \mathbb{R}^{12 \times 1}\}_{i=1}^K$, with the i'th segment calculated as

$$(\mathbf{J}^\top \mathbf{f})_i = \sum_{c \in \mathcal{I}_i} \mathbf{j}_{ci}^\top \mathbf{f}_c \tag{15}$$

where $\mathcal{I}_i$ contains all the constraints related to ED node i. We assign one GPU block per $(\mathbf{J}^\top \mathbf{f})_i$ and again perform the reduction on shared memory.

Linear Equations Solver Solving the cost function in Eq. (3) amounts to a series of linear solves of the normal equations (Eq. (12)). DynamicFusion [Newcombe et al. 2015] uses a direct sparse Cholesky decomposition. Given their approximation of the data term component of $\mathbf{J}^\top \mathbf{J}$ as a block diagonal matrix this still results in a real-time system. However, we do not wish to compromise the fidelity of the reconstruction by approximating $\mathbf{J}^\top \mathbf{J}$ if we can still optimize the cost function in real-time, so we chose to iteratively solve using preconditioned conjugate gradient (PCG). The diagonal blocks of $\mathbf{J}^\top \mathbf{J}$ are used as the preconditioner.

Our approach to the linear solver is akin to the approach taken by [Zollhöfer et al. 2014], but instead of implementing our solver in terms of $\mathbf{J}\mathbf{f}$ and $\mathbf{J}^\top \mathbf{f}$, we use terms $\mathbf{J}^\top \mathbf{J}$ and $\mathbf{J}^\top \mathbf{f}$. Both approaches can effectively handle a prohibitively large number of residuals, but while [Zollhöfer et al. 2014] template-based approach must scale to a large number of parameters, our approach requires considerably less Jacobian evaluations and therefore is significantly faster. To perform sparse matrix-vector multiplication, a core routine in our system, we use a custom warp-level optimized kernel.

5.4 Implementation Details

In our experiments, we set the volume resolution to be 4mm. Marching cubes then extracts a mesh with around $250K$ vertices. In the multi-camera capture system, each surface vertex might be observed by more than one camera (observed ~3 times in our case). In total the number of residuals C in our experiment is around 1 million, with the data terms and visual hull terms constituting the majority. We sample one ED node every 4cm, which leads to ~$2K$ ED nodes in total, and thus the number of parameters $D \approx 24K$.

The sparsity of $\mathbf{J}^\top \mathbf{J}$ is largely determined by two parameters: $|\mathcal{S}_m|$, the number of neighboring ED nodes that a surface vertex m is skinned to, and $|\mathcal{N}_k|$, the number of neighboring ED nodes that an ED node k is connected to for the regularization cost term. We let $|\mathcal{S}_m| = 4\ \forall m$ and $|\mathcal{N}_k| = 8\ \forall k$ in our experiments, resulting in ~15K non-zero $\mathbf{J}^\top \mathbf{J}$ blocks.

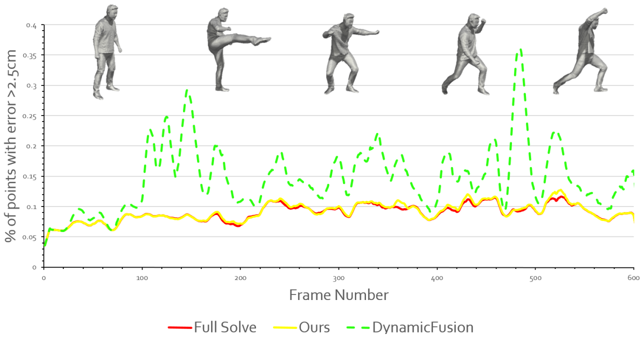

Figure 4: *Solver convergence over a sequence for a fixed number of iterations: Green dashed line demonstrates an approximate evaluation of* $\mathbf{J}^\top\mathbf{J}$. *Red line shows an exact Cholesky solve. Our method is shown in yellow and shows similar convergence behavior as the exact method, with improvements over approximate approaches.*

We run 5 iterations of the LM solver to estimate all the nonrigid parameters, and for each iteration of LM the PCG solver is run for 10 iterations. As shown in Fig. 4, our PCG solver with 10 iterations achieves the same alignment performance as an exact Cholesky solver. It also shows that full $\mathbf{J}^\top\mathbf{J}$ rather than the approximate evaluation (as in [Newcombe et al. 2015]) is important for convergence.

6 Data Fusion

The nonrigid matching stage estimates a deformation field which can be applied to either a volume or a surface to align with the input data in a frame. This alignment can be used, for example, to fuse that data into the volumetric model in order to denoise the model or to deform the model into the current frame for rendering. Indeed, prior work [Dou et al. 2015; Newcombe et al. 2015] defined the first frame as the reference frame (or model), and then incrementally aligned with and fused the data from all subsequent frames. The model is warped into each frame to provide a temporal sequence of reconstructions. This strategy works very well for simple examples (*e.g.*, slow motion, small deformation), but our experiments show that it fails for realistic situations, as shown in our results and supplementary video.

It is difficult, and often impossible, to use a single reference model to explain every possible frame. In an unconstrained and realistic setting, the latter frames might introduce dramatic deformations or even have completely different surface topology (*e.g.*, surfaces that split or merge). These approaches will then struggle as currently used deformation fields do not allow for the discontinuities needed to model this behaviour. Second, it is unrealistic to expect that the nonrigid tracking would never fail, at which point the warped model would not be true to the data.

We approach this problem by redesigning the fusion pipeline. Our gold standard is that the temporal information from the estimated model should never downgrade the quality of the observed data. Put another way, the accumulated model should "upgrade" the data frame, when deemed feasible, by adding accumulated detail or filling in holes caused by occlusion or sensor failures. With this standard in mind, we designed a data fusion pipeline aimed at improving the quality and fidelity of the reconstruction at the data frame by robustly handling realistic surface deformations and tracking failure. There are *two* key features in our pipeline that tackle this goal:

1. **Data Volume.** While previous work maintained a volume for the reference (or the model), which we refer to as $\mathbb{V}^r$, we also maintain a volume at the "data frame" $\mathbb{V}^d$. Following the nonrigid alignment we then fuse the data from the current frame into the reference volume$\mathbb{V}^r$ as in [Newcombe et al. 2011]. We also, however, fuse the reference volume back into the data frame volume $\mathbb{V}^d$. The fusion into $\mathbb{V}^d$ is very selective as to which data from the previously accumulated reference volume is integrated. This allows us to guarantee that the quality of the fused data is never lower than the quality of the observed data in the current frame, even with a poor quality alignment from the reference volume. We then use the fused data volume to extract a high quality reconstruction of the current frame for output, or to reset the reference volume as described below.

2. **Key Volumes.** The key volume strategy allows us to consistently maintain a high quality reference model that handles tracking failures. Instead of simply fixing the reference frame to the first frame, we explicitly handle drastic misalignments by periodically resetting the reference to a fused data volume which we then call a *key volume*. In addition, we detect model-data misalignments and refresh the misaligned voxels using the corresponding voxels from the data volume. Voxel refreshing within a subsequence corresponding to a key volume fixes small scale tracking failures and keeps small data changes from being ignored (*e.g.*, clothes wrinkling). However, when a larger tracking failure occurs (*e.g.*, losing track of an entire arm), refreshing the voxels in the key volume would only replace the arm voxels with empty space. Further, the arm in the data frame will not be reflected in the key volume because no motion field is estimated there to warp the data to the reference. In this case, resetting the reference volume (*i.e.* as a new key volume) would re-enables the tracking and data fusion for the regions that previously lost tracking.

6.1 Fusion at the Data Frame

6.1.1 Volume Warping

We represent the volume as a two level hierarchy similar to [Chen et al. 2013]. As in [Curless and Levoy 1996], each voxel at location $\mathbf{x} \in \mathbb{R}^3$ has a signed distance value and a weight $\langle d, w\rangle$ associated with it, *i.e.*, $\mathbb{V} = (\mathcal{D}, \mathcal{W})$.

At any given iteration we start by sampling a new data volume $\mathbb{V}^d$ from the depth maps. We next *warp* the current reference volume $\mathbb{V}^r$ to this data volume and fuse with the data using the estimated deformation field (see Sec. 5.1 for the details). The ED graph aligns the reference surface $\mathbf{V}^r$ to the data frame. The same forward warping function in Eq. (1) can also be applied to a voxel $\mathbf{x}^r$ in the reference to compute the warped voxel $\tilde{\mathbf{x}}^r = \mathcal{T}(\mathbf{x}^r; G)$. The warped voxel then gets to cast a weighted vote for (*i.e.*, accumulate) its data $\langle d^r, w^r\rangle$ at neighboring voxels within some distance τ on the regular lattice of the data volume. Every data voxel $\mathbf{x}^d$ would then calculate the weighted average of the accumulated data $\langle \bar{d^r}, w^r\rangle$, both SDF value and SDF weight, using the weight $\exp(-\|\tilde{\mathbf{x}}^r - \mathbf{x}^d\|^2/2\sigma^2)$.

Note, this blending (or averaging) is bound to cause some geometric blur. To ameliorate this effect, each reference voxel $\mathbf{x}^r$ does not directly vote for the SDF value it is carrying (*i.e.*, d^r) but for the corrected value $\bar{d^r}$ using the gradient field of the SDF, *i.e.*,

$$\bar{d^r} = d^r + (\tilde{\mathbf{x}}^r - \mathbf{x}^d)^\top\tilde{\Delta},$$

Δ is the gradient at $\mathbf{x}^r$ in the reference. $\tilde{\Delta}$ is the warped gradient using Eq. (2) and approximates the gradient field at the data volume. In other words, $\bar{d^r}$ is the prediction of the SDF value at $\mathbf{x}^d$ given the SDF value and gradient at $\tilde{\mathbf{x}}^r$.

Figure 5: *Left: reference surface. Middle and Right: surfaces from warped volume without and with voxel collision detection.*

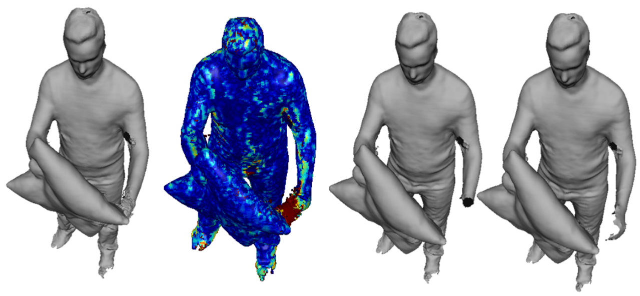

Figure 6: *Volume blending. Left to right: reference surface; nonrigid alignment residual showing topology change; extracted surface at the warped reference; extracted surface from final blended volume*

6.1.2 Selective Fusion

To ensure a high-fidelity reconstruction at the data frame, we need to ensure that each warped reference voxel $\tilde{\mathbf{x}}^r$ will not corrupt the reconstructed result. To this end we perform two tests before fusing in a warped voxel and reject its vote if it fails either.

Voxel Collision. When two model parts move towards each other (*e.g.*, clapping hands), the reference voxels contributing to different surface areas might collide after warping, and averaging the SDF values voted for by these voxels is problematic: in the worst case, the voxels with a higher absolute SDF value will overwhelm the voxels at the zero crossing, leading to a hole in the model (Fig. 5).

To deal with this voxel collision problem, we perform the fusion in two passes. In the first pass, and for any given data voxel $\mathbf{x}^d$, we evaluate all the reference voxels voting at its location and save the reference voxel $\dot{\mathbf{x}}^r$ with the smallest absolute SDF value. In the second pass we reject the vote of any reference voxel $\mathbf{x}^r$ at this location if $|\mathbf{x}^r - \dot{\mathbf{x}}^r| > \eta$.

Voxel Misalignment. We also need to evaluate a proxy error at each reference voxel $\mathbf{x}^r$ to detect if the nonrigid tracking failed so we are able to similarly reject its vote. To do this we first calculate an alignment error at each warped model vertex $\tilde{\mathbf{x}}^r$

$$e_{\tilde{\mathbf{x}}^r} = \begin{cases} |\mathcal{D}^d(\tilde{\mathbf{x}}^r)| & \text{if } \mathcal{H}^d(\tilde{\mathbf{x}}^r) = 0 \\ \min\left(|\mathcal{D}^d(\tilde{\mathbf{x}}^r)|, \mathcal{H}^d(\tilde{\mathbf{x}}^r) \right) & \text{otherwise} \end{cases} \tag{16}$$

where $\mathcal{D}^d$ is the fused TSDF at the data frame, and $\mathcal{H}^d$ is the visual hull distance transform (Sec. 5.2.3). We then aggregate this error at the ED nodes by averaging the errors from the vertices associated with the same ED node. This aggregation process reduces the influence of the noise in the depth data on the alignment error. Finally, we reject any reference voxel if any of its neighboring ED nodes has an alignment error beyond a certain threshold. The extracted surface from $\tilde{\mathbb{V}}^r$ is illustrated in Fig. 6.

6.1.3 Volume Blending

After we fuse the depth maps into a data volume $\mathbb{V}^d$ and warp the reference volume to the data frame forming $\tilde{\mathbb{V}}^r$, the next step is to blend the two volumes $\mathbb{V}^d$ and $\tilde{\mathbb{V}}^r$ to get the final fused volume $\bar{\mathbb{V}}^d$, used for the reconstructed output.[3]

Even after the conservative selective fusion described in the previous section, simply taking a weighted average of the two volumes (*i.e.*, $\bar{d}^d = \frac{\tilde{d}^r \tilde{w}^r + d^d w^d}{\tilde{w}^r + w^d}$) leads to artifacts. This naive blending does not guarantee that the SDF band around the zero-crossing will have a smooth transition of values. This is because boundary voxels that survived the rejection phase will suppress any zero-crossings coming from the data, causing artifacts and lowering the quality at the output.

To handle this problem, we start by projecting the reference surface vertices $\mathbf{V}^r$ to the depth maps. We can then calculate a per-pixel depth alignment error as the difference between the vertex depth d and its projective depth d_{proj}, normalized by a maximum $d_{\max}$. Put together, we calculate

$$e_{\text{pixel}} = \begin{cases} \min\left(1.0, |d - d_{\text{proj}}| \,/\, d_{\max} \right) & \text{if } d_{\text{proj}} \text{ is valid} \\ 1.0 & \text{otherwise.} \end{cases} \tag{17}$$

Each voxel in the data volume $\mathbb{V}^d$ can then have an aggregated average depth alignment error e_{voxel} when projecting it to depth maps. Finally, instead of using the naive blending described above, we use the blending function

$$\bar{d}^d = \frac{\tilde{d}^r \tilde{w}^r (1.0 - e_{\text{voxel}}) + d^d w^d}{\tilde{w}^r (1.0 - e_{\text{voxel}}) + w^d,} \tag{18}$$

downweighting the reference voxel data by its depth misalignment.

6.2 Fusion at the Reference Frame

As in [Newcombe et al. 2015], to update the reference model we warp each reference voxel $\mathbf{x}^r$ to the data frame, project it to the depth maps, and update the TSDF value and weight. This avoids an explicit data-to-model warp. Additionally, we also know the reference voxels $\tilde{\mathbf{x}}^r$ not aligned well to the data from Eq. (16). For these voxels we discard their data and *refresh* it from the data in the current data frame. Finally, we reset the entire volume periodically to the fused data volume $\bar{\mathbb{V}}^d$ (*i.e.*, key volumes) to handle large misalignments that cannot be recovered from by the per-voxel refresh.

7 Results

We now provide results, experiments and comparisons of our real-time performance capture method.

7.1 Live Performance Capture

Our system is fully implemented on the GPU using CUDA. Results of live multi-view scene captures for our test scenes are shown in Figures 1 and 7 as well as in the supplementary material. It is important to stress that all these sequences were captured online and in real-time, including depth estimation and full nonrigid reconstruction. Furthermore, these sequences are captured over long time periods comprising many minutes. We make a strong case for nonrigid alignment in Fig. 8. While volumetrically fusing the live data does produce a more aesthetically appealing result compared to

[3]Marching cubes is applied to this volume to extract the final mesh representation.

Figure 7: *Real-time results captured of Fusion4D, showing a variety of challenging sequences. Please also see accompanying video.*

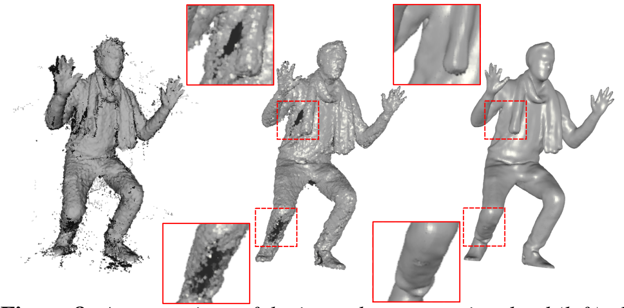

Figure 8: *A comparison of the input data as a point cloud (left), the fused live data without nonrigid alignment (center), and the output of our system (right).*

Figure 9: *Our system is robust to many complex topology changes.*

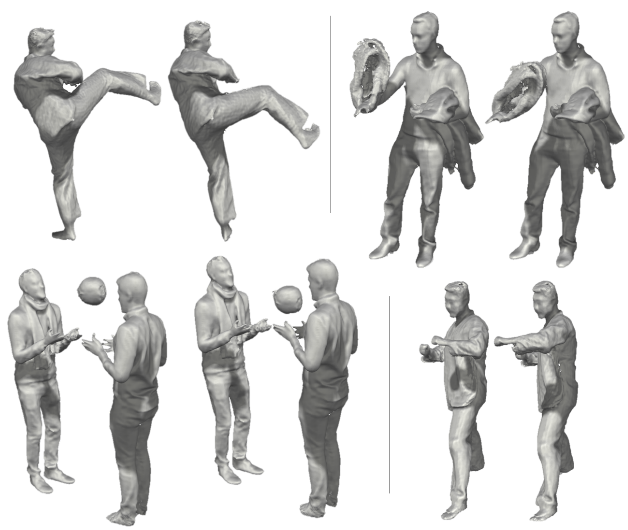

Figure 10: *Our approach is robust to fast motions.*

the input point cloud, it cannot resolve issues arising from missing data (holes) or noise. On the other hand, these issues are significantly ameliorated in the reconstructed mesh with Fusion4D by leveraging temporal information.

We captured a variety of diverse and challenging nonrigidly moving scenes. This includes multiple people interacting, deforming objects, topology changes and fast motions. Fig. 7 shows multiple examples for each of these scenes. Our reconstruction algorithm is able to deal with extremely fast motion, where most online nonrigid sys-

Figure 11: *Quantitative comparison with [Collet et al. 2015]*

tems would fail. Fig. 10 depicts typical situations where the small motion assumption does not hold. This robustness is in due to the ability to estimate fast RGBD correspondences allowing for robust initialization of the ED graph, and also the ability to recover from misalignment errors. In Fig. 9 we show a number of challenging topology changes that our system can cope with in a robust manner. This includes hands being initially reconstructed on the hips of the performer and then moved, and items of clothing being removed, such as a jacket or scarf etc.

Other examples of reconstructions in Fig. 7 and supplementary video, depict clothing changes, taekwondo moves, dancing, animals, moving hair and interaction with objects. For any of these situations the algorithm automatically retrieves the nonrigid reconstruction with real-time performance. Notice also that the method has no shape prior of the object of interest and can easily generalize to non-human models, for example animals or objects.

7.2 Computational Time

Similar to [Collet et al. 2015] the input RGBD and segmentation data is generated on dedicated PCs. Each machine is an Intel Core i7 3.4GHz CPU, 16GB of RAM and it uses two NVIDIA Titan X GPUs. Each PC processes two depthmaps and two segmentation masks in parallel. The total time is $21ms$ and $4ms$ for the stereo matching and segmentation, respectively. Correspondence estimation requires $5ms$ with a parallel GPU implementation. In total each machine uses no more than $30ms$ to generate the input for the nonrigid reconstruction pipeline. RGBD frames are generated in parallel to the nonrigid pipeline, but do introduce 1 frame of latency.

A master PC (another Intel Core i7 3.4GHz CPU, 16GB of RAM, with a single NVIDIA Titan X), aggregates and synchronizes all the depthmaps, segmentation masks and correspondences. Once the RGBD inputs are available, the average processing time to nonrigidly reconstruct is $32ms$ (i.e., 31fps) with $3ms$ for preprocessing (10% of the overall pipeline), $2ms$ (7%) for rigid pose estimation (on average 4 iterations), $20ms$ (64%) for the nonrigid registration (5 LM iterations, with 10 PCG iterations), and $6ms$ (19%) for fusion.

7.3 Correspondence Evaluation

In Sec. 5.2.4 we described our approach to estimating RGBD correspondences. We now evaluate its robustness compared to other state-of-the-art methods. One sequence with very fast motions is considered. In order to compare different correspondence algorithms, we only minimize the $E_{corr}(G)$ term in Eq. 3 and we compute the residual error. We report results as percentage of alignment error between the current observation and the model. In particular, we show the percentage of vertices with error $> 5mm$. We compared different methods: standard SIFT detector and descriptors [Lowe 2004] , a FAST detector [Rosten and Drummond 2005] followed by SIFT descriptors, DeepMatch [Weinzaepfel et al. 2013], EpicFlow

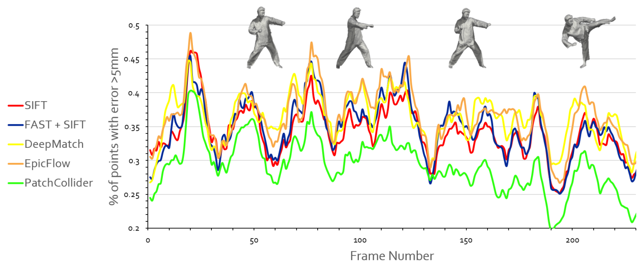

Figure 12: *Quantitative comparisons of different correspondence methods: SIFT, FAST+SIFT, DeepMatch, EpicFlow and Global Patch Collider. We computed the residual error and reported the percentage of vertexes with error $> 5mm$. The method proposed in Sec. 5.2.4 achieved the best score with only 29% outliers.*

[Revaud et al. 2015] and our extension of Global Patch Collider [Wang et al. 2016] described in Sec. 5.2.4. Quantitative results on this fast motion sequence are reported in Fig. 12. The best results are obtained by our method with 29% outliers, then SIFT (34%), FAST+SIFT (34%), DeepMatch (36%) and EpicFlow (36%). Most of the error occurred in regions where very large motion is present: a qualitative comparison is depicted in Fig. 13.

7.4 Nonrigid Reconstruction Comparisons

In Fig. 15, we compare to the dataset of [Collet et al. 2015] for a sequence with extremely high motions. The figure compares renderings of the original meshes and multiple reconstructions, where red corresponds to a fitting error of $15mm$. In particular, we compare our method with [Zollhöfer et al. 2014] and [Newcombe et al. 2015], showing our superior reconstructions in these challenging situations. We also show results and distance metrics for the method of [Collet et al. 2015] which is an offline technique with a runtime of about 30 minutes per frame on the CPU, and runs with 30 more cameras than our system. In a more quantitative analysis (Fig. 11) we plot the error over the input mesh for our method and [Collet et al. 2015], which shows that our algorithm can match the motion and fine scale details exhibited in this sequence. Our approach shows qualitatively similar results but with a system that is about 4 orders of magnitude faster, allowing for true real-time performance capture.

Finally, multiple qualitative comparisons among different state of the art methods are shown in Fig. 14. These sequences exhibits all classical situations where online methods fail, such as large motions and topology changes. Again our real-time reconstruction methods correctly retrieves the non rigid shapes for any of these scenarios. Please also see accompanying video figure.

8 Limitations

Even though we demonstrated one of the first methods for real-time nonrigid reconstruction from multiple views, showing reconstruction of challenging scenes, our system is not without limitations. Given the tight real-time constraint (33ms/frame) of our approach, we rely on temporal coherence of the RGBD input stream making the processing at 30Hz a necessity. If the frame rate is too low or frame-to-frame motion is too large, either the frame-to-frame correspondences would be inaccurately estimated or the nonrigid alignment would fail to converge given the tight time budget. In either case our method might lose tracking. In both scenarios our system does fall back to the live fused data. However, as shown in Fig. 16 the volume blending can look noisy as new data is first being fused. Another issue in our current work is robustness to segmentation errors. Large segmentation errors, if there is missing depth data for instance, can lead to incorrect visual hull estimation. This can cause some noise to be integrated into the model as shown in Fig. 16. Finally, any small nonrigid alignment errors can cause slight oversmoothing of the model at times e.g. Fig. 16. We deal with topology change by refreshing correspondence voxels. This strategy works in general, but has artifacts when one object slides over another surface, *e.g.*, unzipping a jacket. To solve the topology problem intrinsically, a nonrigid matching algorithm that explicitly handles topology changes needs to be designed.

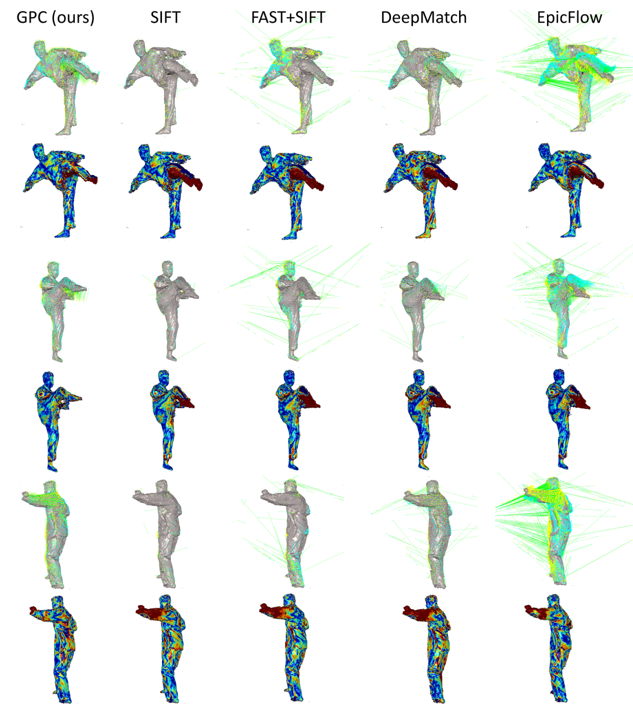

Figure 13: *Qualitative comparisons of correspondence algorithms. We show the detected correspondences (green lines) between the previous frame (yellow points) and current frame (cyan points). GPC shows less residual error in fast motion regions, whereas current state of the art algorithms (DeepMatch, EpicFlow) and traditional correspondence methods (SIFT, FAST) show higher error due to the highest percentage of false positives (FAST, DeepMatch, EpicFlow), or due to the poor recall (SIFT).*

Figure 14: *Qualitative comparisons with state of the art approaches.*

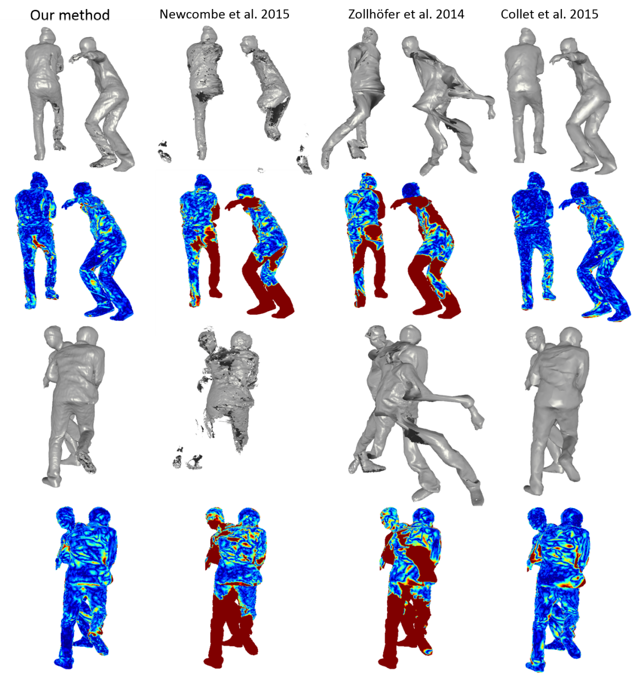

Figure 15: *Qualitative comparisons with the high quality offline system of [Collet et al. 2015].*

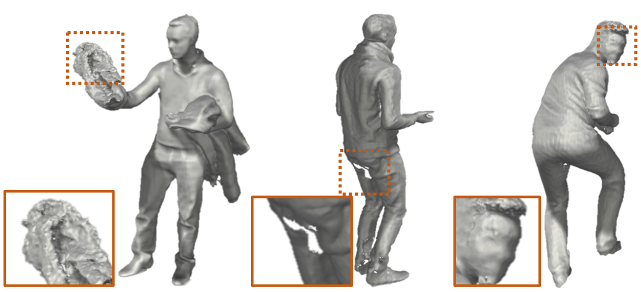

Figure 16: *Current limitations of our system. From left to right: Noisy data when tracking is lost. Holes due to segmentation errors. Oversmoothing due to alignment errors.*

9 Conclusions

We have demonstrated Fusion4D; the first real-time multi-view non-rigid reconstruction system for live performance capture. We have contributed a new pipeline for live multi-view performance capture, generating high-quality reconstructions in real-time, with several unique capabilities over prior work. As shown, our reconstruction algorithm enables both incremental reconstruction, improving the surface estimation over time, as well as parameterizing the nonrigid scene motion. We also demonstrated how our approach robustly handles both large frame-to-frame motion and topology changes. This was achieved using a novel real-time solver, correspondence algorithm, and fusion method. We believe our work can enable new types of live performance capture experiences, such as broadcasting live events including sports and concerts in 3D, and also the ability to capture humans live and have them re-rendered in other geographic locations to enable high fidelity immersive telepresence.

References

Beeler, T., Hahn, F., Bradley, D., Bickel, B., Beardsley, P., Gotsman, C., Sumner, R. W., and Gross, M. 2011. High-quality passive facial performance capture using anchor frames. *ACM Transactions on Graphics (TOG) 30*, 4, 75.

Bleyer, M., Rhemann, C., and Rother, C. 2011. Patchmatch stereo: Stereo matching with slanted support windows. In *Proc. BMVC*, vol. 11, 1–11.

Bogo, F., Black, M. J., Loper, M., and Romero, J. 2015. Detailed full-body reconstructions of moving people from monocular RGB-D sequences. In *ICCV*, 2300–2308.

Bojsen-Hansen, M., Li, H., and Wojtan, C. 2012. Tracking surfaces with evolving topology. *ACM Trans. Graph. 31*, 4, 53.

Bradley, D., Popa, T., Sheffer, A., Heidrich, W., and Boubekeur, T. 2008. Markerless garment capture. *ACM TOG (Proc. SIGGRAPH) 27*, 3, 99.

Cagniart, C., Boyer, E., and Ilic, S. 2010. Free-form mesh tracking: a patch-based approach. In *Proc. CVPR*.

Chen, Y., and Medioni, G. 1992. Object modelling by registration of multiple range images. *CVIU 10*, 3, 144–155.

Chen, J., Bautembach, D., and Izadi, S. 2013. Scalable real-time volumetric surface reconstruction. *ACM TOG*.

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 TOG 34*, 4, 69.

Curless, B., and Levoy, M. 1996. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, 303–312.

de Aguiar, E., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.-P., and Thrun, S. 2008. Performance capture from sparse multi-view video. *ACM TOG (Proc. SIGGRAPH) 27*, 1–10.

Dou, M., Fuchs, H., and Frahm, J.-M. 2013. Scanning and tracking dynamic objects with commodity depth cameras. In *Proc. ISMAR*, IEEE, 99–106.

Dou, M., Taylor, J., Fuchs, H., Fitzgibbon, A., and Izadi, S. 2015. 3d scanning deformable objects with a single rgbd sensor. In *CVPR*.

Engels, C., Stewénius, H., and Nistér, D. 2006. Bundle adjustment rules. *Photogrammetric computer vision 2*, 124–131.

Gall, J., Stoll, C., De Aguiar, E., Theobalt, C., Rosenhahn, B., and Seidel, H.-P. 2009. Motion capture using joint skeleton tracking and surface estimation. In *Proc. CVPR*, IEEE, 1746–1753.

Guo, K., Xu, F., Wang, Y., Liu, Y., and Dai, Q. 2015. Robust non-rigid motion tracking and surface reconstruction using l0 regularization. In *ICCV*, 3083–3091.

Krähenbüh, P., and Koltun, V. 2011. Efficient inference in fully connected crfs with gaussian edge potentials. *NIPS*.

Kutulakos, K. N., and Seitz, S. M. 2000. A theory of shape by space carving. *IJCV*.

Li, H., Adams, B., Guibas, L. J., and Pauly, M. 2009. Robust single-view geometry and motion reconstruction. *ACM TOG*.

Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *IJCV*.

Mitra, N. J., Flöry, S., Ovsjanikov, M., Gelfand, N., Guibas, L. J., and Pottmann, H. 2007. Dynamic geometry registration. In *Proc. SGP*, 173–182.

Mori, M., MacDorman, K. F., and Kageki, N. 2012. The uncanny valley [from the field]. *Robotics & Automation Magazine, IEEE 19*, 2, 98–100.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. 2011. KinectFusion: Real-time dense surface mapping and tracking. In *Proc. ISMAR*, 127–136.

Newcombe, R. A., Fox, D., and Seitz, S. M. 2015. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR*, 343–352.

Pons-Moll, G., Taylor, J., Shotton, J., Hertzmann, A., and Fitzgibbon, A. 2015. Metric regression forests for correspondence estimation. *IJCV 113*, 3, 163–175.

Pradeep, V., Rhemann, C., Izadi, S., Zach, C., Bleyer, M., and Bathiche, S. 2013. MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera. In *Proc. ISMAR*, IEEE, 83–88.

Revaud, J., Weinzaepfel, P., Harchaoui, Z., and Schmid, C. 2015. Epicflow: Edge-preserving interpolation of correspondences for optical flow. *CVPR*.

Rosten, E., and Drummond, T. 2005. Fusing points and lines for high performance tracking. In *ICCV*.

Rusinkiewicz, S., and Levoy, M. 2001. Efficient variants of the icp algorithm. In *3DIM*, 145–152.

Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., and Fitzgibbon, A. 2013. Scene coordinate regression forests for camera relocalization in rgb-d images. In *CVPR*.

Smolic, A. 2011. 3d video and free viewpoint videofrom capture to display. *Pattern recognition 44*, 9, 1958–1968.

Starck, J., and Hilton, A. 2007. Surface capture for performance-based animation. *Computer Graphics and Applications 27*, 3, 21–31.

Stoll, C., Hasler, N., Gall, J., Seidel, H., and Theobalt, C. 2011. Fast articulated motion tracking using a sums of gaussians body model. In *Proc. ICCV*, IEEE, 951–958.

Sumner, R. W., Schmid, J., and Pauly, M. 2007. Embedded deformation for shape manipulation. *ACM TOG 26*, 3, 80.

Tevs, A., Berner, A., Wand, M., Ihrke, I., Bokeloh, M., Kerber, J., and Seidel, H.-P. 2012. Animation cartography-intrinsic reconstruction of shape and motion. *ACM TOG*.

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

Vineet, V., Warrell, J., and Torr, P. H. S. 2012. Filter-based mean-field inference for random fields with higher-order terms and product label-spaces. In *ECCV*.

Vlasic, D., Baran, I., Matusik, W., and Popović, J. 2008. Articulated mesh animation from multi-view silhouettes. *ACM TOG (Proc. SIGGRAPH)*.

Vlasic, D., Peers, P., Baran, I., Debevec, P., Popovic, J., Rusinkiewicz, S., and Matusik, W. 2009. Dynamic shape capture using multi-view photometric stereo. *ACM TOG (Proc. SIGGRAPH Asia) 28*, 5, 174.

Wand, M., Adams, B., Ovsjanikov, M., Berner, A., Bokeloh, M., Jenke, P., Guibas, L., Seidel, H.-P., and Schilling, A. 2009. Efficient reconstruction of nonrigid shape and motion from real-time 3D scanner data. *ACM TOG*.

Wang, S., Fanello, S. R., Rhemann, C., Izadi, S., and Kohli, P. 2016. The global patch collider. *CVPR*.

Waschbüsch, M., Würmlin, S., Cotting, D., Sadlo, F., and Gross, M. 2005. Scalable 3D video of dynamic scenes. In *Proc. Pacific Graphics*, 629–638.

Wei, L., Huang, Q., Ceylan, D., Vouga, E., and Li, H. 2015. Dense human body correspondences using convolutional networks. *arXiv preprint arXiv:1511.05904*.

Weinzaepfel, P., Revaud, J., Harchaoui, Z., and Schmid, C. 2013. Deepflow: Large displacement optical flow with deep matching. In *ICCV*.

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

Ye, M., Zhang, Q., Wang, L., Zhu, J., Yang, R., and Gall, J. 2013. A survey on human motion analysis from depth data. In *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*. Springer, 149–187.

Zach, C. 2014. Robust bundle adjustment revisited. In *Computer Vision–ECCV 2014*. Springer, 772–787.

Zeng, M., Zheng, J., Cheng, X., and Liu, X. 2013. Templateless quasi-rigid shape modeling with implicit loop-closure. In *Proc. CVPR*, IEEE, 145–152.

Zhang, Q., Fu, B., Ye, M., and Yang, R. 2014. Quality dynamic human body modeling using a single low-cost depth camera. In *CVPR*, IEEE, 676–683.

Zollhöfer, M., Niessner, M., Izadi, S., Rhemann, C., Zach, C., Fisher, M., Wu, C., Fitzgibbon, A., Loop, C., Theobalt, C., et al. 2014. Real-time non-rigid reconstruction using an rgb-d camera. *ACM TOG*.