

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 non-rigid 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.

87 2 Related Work

88 **Multi-view Performance Capture:** Many compelling offline per-
 89 formance capture systems have been proposed. Some specifi-
 90 cally model complex human motion and dynamic geometry, in-
 91 cluding people with general clothing, possibly along with pose
 92 parameters of an underlying kinematic skeleton (see [Theobalt
 93 et al. 2010] for a full review). Some methods employ variants
 94 of shape-from-silhouette [Waschbüsch et al. 2005] or active or pas-
 95 sive stereo [Starck and Hilton 2007]. Template-based approaches
 96 deform a static shape model such that it matches a human [de Aguiar
 97 et al. 2008; Vlasic et al. 2008; Gall et al. 2009] or a person’s cloth-
 98 ing [Bradley et al. 2008]. Vlasic et al. [2009] use a sophisticated
 99 photometric stereo light stage with multiple high-speed cameras
 100 to capture geometry of a human at high detail. Dou et al. [2013]
 101 capture precise surface deformations using an eight-Kinect rig, by
 102 deforming a human template, generated from a KinectFusion scan,
 103 using embedded deformation [Sumner et al. 2007]. Other methods
 104 jointly track a skeleton and the nonrigidly deforming surface [Vlasic
 105 et al. 2008; Gall et al. 2009].

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

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

135 **Real-time Approaches:** Only recently have we seen real-time non-
 136 rigid reconstruction systems appear. Approaches fall into three
 137 categories. *Single object parametric* approaches focus on a single
 138 object of interest, e.g. face, hand, or body, which is parametrized
 139 ahead of time in an offline manner, and tracked or deformed to fit the
 140 data in real-time. Compelling real-time reconstructions of nonrigid
 141 articulated motion (e.g. [Ye et al. 2013; Stoll et al. 2011; Zhang et al.
 142 2014]) and shape (e.g. [Ye et al. 2013; Ye and Yang 2014]) have
 143 been demonstrated. However by their very nature, these approaches
 144 rely on strong priors based on either pre-learned statistical models,
 145 articulated skeletons, or morphable shape models, prohibiting cap-
 146 ture of arbitrary scenes or objects. Often the parametric model is
 147 not rich enough to capture challenging poses or all types of shape
 148 variation. For human bodies, even with extremely rich offline shape
 149 and pose models [Bogo et al. 2015], reconstructions can suffer from
 150 the effect of uncanny valley [Mori et al. 2012]; and clothing or hair

151 can prove problematic [Bogo et al. 2015].

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

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

189 3 System Overview

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

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

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

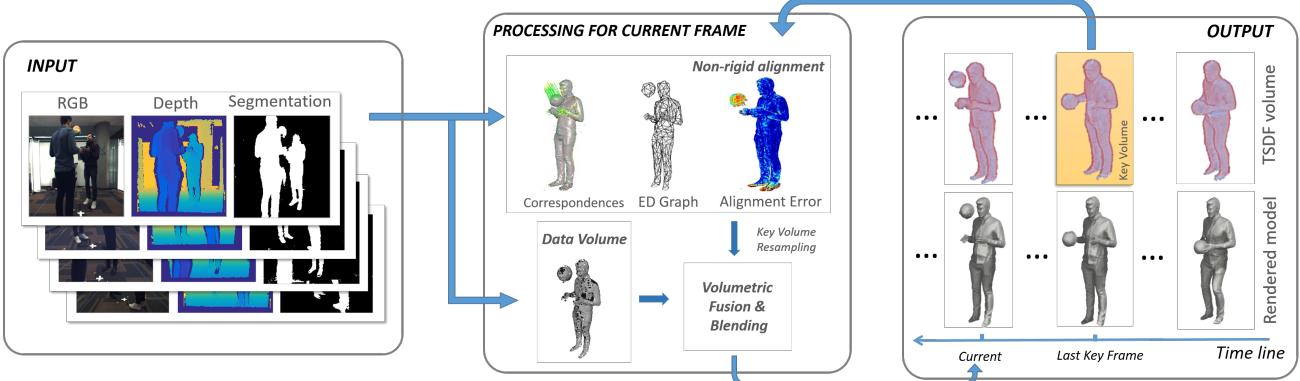


Figure 2: The Fusion4D pipeline. Please see text in Sec. 3 for details.

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

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

226 4 Raw Depth Acquisition and Preprocessing

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

239 A segmentation step follows the depth computation algorithm, where
 240 2D silhouettes of the regions of interest are produced. The segmenta-
 241 tion mask plays a crucial role in estimating the visual hull constraint
 242 (see Sec. 5.2.3) that helps ameliorate issues with missing data in the
 243 input depth and ensures that foreground data is not deleted from the
 244 model. Our segmentation also avoids the need for a green screen
 245 setup as in [Collet et al. 2015] and allows capture in natural and
 246 realistic settings. In our pipeline we employed a simple background
 247 model (using both RGB and depth cues) that does not take into
 248 account temporal consistency. This background model is used to
 249 compute unary potentials by considering pixel-wise differences with
 250 the current scene observation. We then use a dense Conditional Ran-
 251 dom Field (CRF) [Krähenbühl and Koltun 2011] model to enforce
 252 smoothness constraints between neighboring pixels. Due to our real-
 253 time requirements, we use an approximate GPU implementation
 254 similar to [Vineet et al. 2012].

255 5 Nonrigid Motion Field Estimation

256 In each frame we observe N depthmaps, $\{\mathbb{D}_n\}_{n=1}^N$ and N fore-
 257 ground masks, $\{\mathbb{S}_n\}_{n=1}^N$. As is common [Curless and Levoy 1996;
 258 Newcombe et al. 2011; Newcombe et al. 2015], we accumulate
 259 this depth data into a non-parametric surface represented implicitly
 260 by a truncated signed distance function (TSDF) or volume \mathbb{V} in
 261 some “reference frame” (which we denote as *key volume*). This
 262 allows efficient alignment and allows for all the data to be averaged
 263 into a complete surface with greatly reduced noise. Further, the
 264 zero crossings of the TSDF can be easily located to extract a high
 265 quality mesh¹ $\mathbf{V} = \{\mathbf{v}_m\}_{m=1}^M \subseteq \mathbb{R}^3$ with corresponding normals
 266 $\{\mathbf{n}_m\}_{m=1}^M$. The goal of this section is to show how to estimate a
 267 deformation field that warps the key volume \mathbb{V} or the mesh \mathbf{V} to
 268 align with the raw depth maps $\{\mathbb{D}_n\}_{n=1}^N$. We typically refer \mathbb{V} or \mathbf{V}
 269 as model, and $\{\mathbb{D}_n\}_{n=1}^N$ as data.

270 5.1 Deformation Model

271 Following [Li et al. 2009] and [Dou et al. 2015] we choose the
 272 embedded deformation (ED) model of [Sumner et al. 2007] to para-
 273 meterize the nonrigid deformation field. Before processing each
 274 new frame, we begin by uniformly sampling a set of K “ED nodes”
 275 within the reference volume by sampling locations $\{\mathbf{g}_k\}_{k=1}^K \subseteq \mathbb{R}^3$
 276 from the mesh \mathbf{V} extracted from this volume. Every vertex \mathbf{v}_m in
 277 that mesh is then “skinned” to its closest ED nodes $\mathcal{S}_m \subseteq \{1, \dots, K\}$
 278 using a set of fixed skinning weights $\{w_k^m : k \in \mathcal{S}_m\} \subseteq [0, 1]$
 279 calculated as $w_k^m = \frac{1}{Z} \exp(-\|\mathbf{v}_m - \mathbf{g}_k\|^2 / 2\sigma^2)$, where Z is a nor-
 280 malization constant ensuring that, for each vertex, these weights
 281 add to one. Here σ defines the effective radius of the ED nodes,
 282 which we set as $\sigma = 0.5d$, where d is the average distance between
 283 neighboring ED nodes after the uniform sampling.

284 We then represent the local deformation around each ED node \mathbf{g}_k
 285 using an affine transformation $A_k \in \mathbb{R}^{3 \times 3}$ and a translation $\mathbf{t}_k \in
 286 \mathbb{R}^3$. In addition, a global rotation $R \in SO(3)$ and translation
 287 $T \in \mathbb{R}^3$ are added. The set $G = \{R, T\} \cup \{A_k, \mathbf{t}_k\}_{k=1}^K$ fully
 288 parameterizes the deformation that warps any point $\mathbf{v} \in \mathbb{R}^3$ to

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

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

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

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

290 and normalization is applied afterwards.

291 5.2 Energy Function

292 To estimate the parameters G , we formulate an energy function
 293 $E(G)$ that penalizes the misalignment between our model and the
 294 observed data, regularizes the types of allowed deformations and
 295 encodes other priors and constraints. The energy function

$$E(G) = \lambda_{\text{data}} E_{\text{data}}(G) + \lambda_{\text{hull}} E_{\text{hull}}(G) + \lambda_{\text{corr}} E_{\text{corr}}(G) + \\ \lambda_{\text{rot}} E_{\text{rot}}(G) + \lambda_{\text{smooth}} E_{\text{smooth}}(G) \quad (3)$$

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

297 5.2.1 Data Term

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

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

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

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

324 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
 325 slight notational abuse we simply use $\tilde{\mathbf{v}}$ and $\tilde{\mathbf{n}}$ to represent the
 326 warped points and normals); $\Gamma_n(\mathbf{v}) = P_n(\Pi_n(\mathbf{v}))$, with $\Pi_n : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ projecting a point into the n 'th depth map and $P_n : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ back-projecting the corresponding pixel in \mathbb{D}_n into 3D;
 327 and $\mathcal{V}_n(G) \subseteq \{1, \dots, M\}$ are vertex indices that are considered to
 328 be “visible” in view n when the model is deformed using G . In
 329 particular, we consider a vertex to be visible if
 330

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

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

334 Although (5) is an approximation to (4), it offers a variety of key ben-
 335 efits. First, the use of a point-to-plane term is a well known strategy
 336 to speed up convergence [Chen and Medioni 1992]. Second, the use
 337 of a “projective correspondence” avoids the expensive minimization
 338 in (4). Lastly, the visibility set $\mathcal{V}_n(G)$ is explicitly computed to be
 339 robust to outliers which avoids employing a robust data term here
 340 that often slows Gauss-Newton like methods [Zach 2014]. Interest-
 341 ingly, the last two points interfere with the differentiability of (5) as
 342 $P_n(\Pi_n(\tilde{\mathbf{v}}))$ jumps as the projection crosses pixel boundaries and
 343 $\mathcal{V}(G)$ undergoes discrete modifications as G changes. Nonetheless,
 344 we use a further approximation (see Sec. 5.3) at each Gauss-Newton
 345 iteration whose derivative both exists everywhere and is more effi-
 346 cient to compute.

347 5.2.2 Regularization Terms

348 As the deformation model above could easily represent unreason-
 349 able deformations, we follow [Dou et al. 2015] by deploying two
 350

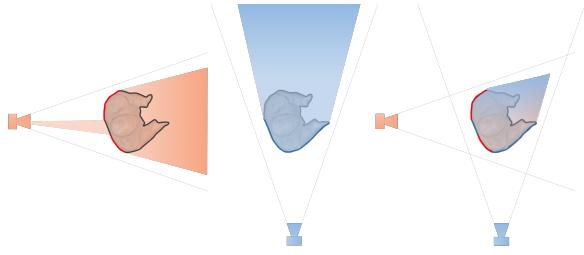


Figure 3: An illustration of visual hull in our optimization. Left: the first camera’s visual hull (shaded region) is defined by the foreground segmentation and the observed data (red line on the surface). In this case, a hole on foreground causes the hull to extend all the way to the camera. Our energy 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.

351 regularization terms to restrict the class of allowed deformations.
 352 The first term

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

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

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

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

362 5.2.3 Visual Hull Term

363 The data term above only constrains the deformation when the
 364 warped model is close to the data. To see why this is problem-
 365 atic, let us assume momentarily the best case scenario where we
 366 happen to have a perfect model that should be able to fully “explain”
 367 the data. If a piece of the model is currently being deformed to a
 368 location outside the truncation threshold of depth maps, the gradient
 369 will be zero. Another, more fundamental issue, is that a piece of
 370 the model that is currently unobserved (e.g. a hand hidden behind a
 371 user’s back) is allowed to enter free-space. This occurs despite the
 372 fact that we *know* that free-space should not be occupied as we have
 373 *observed* it to be free. Up until now, other methods [Newcombe et al.
 374 2015] have generally ignored this constraint, or equivalently their
 375 model has only been forced to explain the foreground data while
 376 ignoring “negative” background data.

377 To address this, we formulate an additional energy term that en-
 378 codes the constraint that the deformed model lies within the visual
 379 hull. The visual hull is a concept used in shape-from-silhouette
 380 space-carving reconstruction techniques [Kutulakos and Seitz 2000].
 381 Typically in 2D it is defined as the intersection of the cones cut-out
 382 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

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

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 volume², 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

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

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 $\{u_{nf}^{prev}, u_{nf}\}_{f=1}^{N_f}$ between pixels in the current frame and the previous frame. For each match (u_{nf}^{prev}, u_{nf}) we can find a corresponding point $\mathbf{q}_{nf} \in \mathbb{R}^3$ in the reference frame using

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

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 energy term

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

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

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

463 5.3 Optimization

464 In this section, we show how to rapidly and robustly minimize
 465 $E(G)$ on the GPU to obtain an alignment between the model and
 466 the current frame. To this end, we let $\mathbf{X} \in \mathbb{R}^D$ represent the
 467 concatenation of all the parameters and let each entry of $\mathbf{f}(\mathbf{X}) \in \mathbb{R}^C$
 468 contain each of the C unsquared terms (*i.e.* the residuals) from
 469 the energy above so that $E(G) = \mathbf{f}(\mathbf{X})^\top \mathbf{f}(\mathbf{X})$. In this form, the
 470 problem of minimizing $E(G)$ can be seen as a standard sparse non-
 471 linear least squares problem which can be solved by approaches
 472 based on the Gauss-Newton algorithm. We handle the robust terms
 473 using the square-rooting technique described in [Engels et al. 2006;
 474 Zach 2014].

475 For each frame we initialize all the parameters from the motion
 476 field of the previous frame. We then fix the ED nodes parame-
 477 ters $\{A_k, \mathbf{t}_k\}_{k=1}^K$ and estimate the global rigid motion parameters
 478 $\{R, T\}$ using projective iterative closest point (ICP) [Rusinkiewicz
 479 and Levoy 2001]. Next we fix the global rigid motion parameters and
 480 estimate the ED nodes parameters. The details of the optimization
 481 are presented in the following sections.

482 5.3.1 Computing a Step Direction

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

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

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

496 **Per-Iteration Approximation** In order to deal with the non-
 497 differentiability of $E_{\text{data}}(G)$ and improve performance, at the start
 498 of each iteration we can take a copy of the current set of parameters
 499 $G_0 \leftarrow G$ to create a differentiable approximation to $E_{\text{data}}(G)$ as

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

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

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

514 Further, $J^\top J$ itself is a sparse matrix composed of non-zero blocks
 515 $\{\mathbf{h}_{ij} \in \mathbb{R}^{12 \times 12} : i, j \in \{1, \dots, K\} \wedge i \sim j\}$ created by ordering
 516 parameter blocks from K ED nodes, where $i \sim j$ denotes that the
 517 i 'th and j 'th ED nodes simultaneously contribute to at least one
 518 residual. The (i, j) 'th block can be computed as

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

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

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

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

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

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

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

554 5.4 Implementation Details

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

563 The sparsity of $J^\top J$ is largely determined by two parameters: $|\mathcal{S}_m|$,
 564 the number of neighboring ED nodes that a surface vertex m is
 565 skinned to, and $|\mathcal{N}_k|$, the number of neighboring ED nodes that an
 566 ED node k is connected to for the regularization cost term. We let
 567 $|\mathcal{S}_m| = 4 \forall m$ and $|\mathcal{N}_k| = 8 \forall k$ in our experiments, resulting in
 568 ~15K non-zero $J^\top J$ blocks.

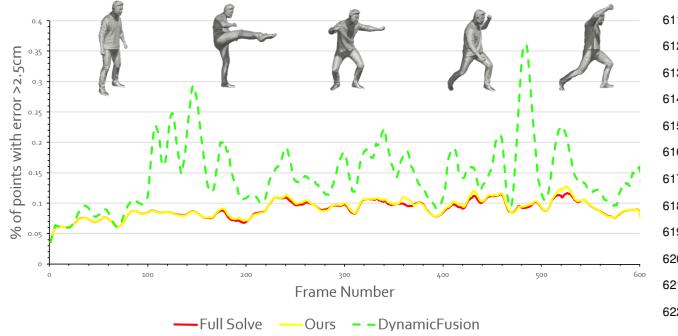


Figure 4: Solver convergence over a sequence for a fixed number of iterations: Green dashed line demonstrates an approximate evaluation of $\mathbf{J}^\top \mathbf{J}$. Red line shows an exact Cholesky solve. Our method is shown in yellow and shows similar convergence behavior as the exact method, with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

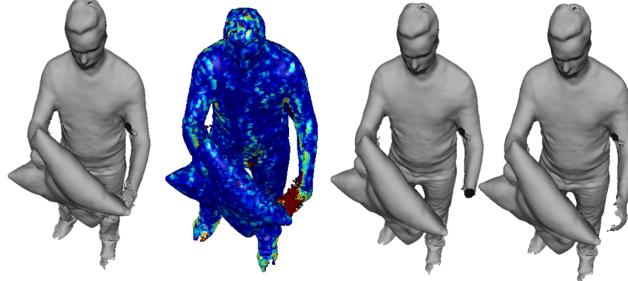


Figure 6: Volume blending. Left to right: reference surface; non-rigid alignment residual showing topology change; extracted surface at the warped reference; extracted surface from final blended volume

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

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

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

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

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

³Marching 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.

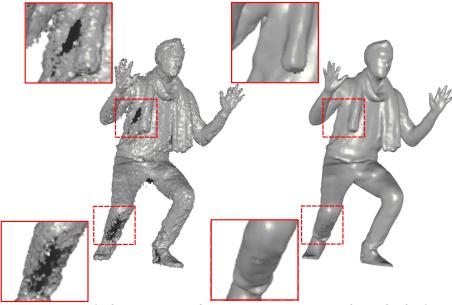


Figure 8: A comparison of the input data as a point cloud (left), the fused live data without nonrigid alignment (center), and the output of our system (right).



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

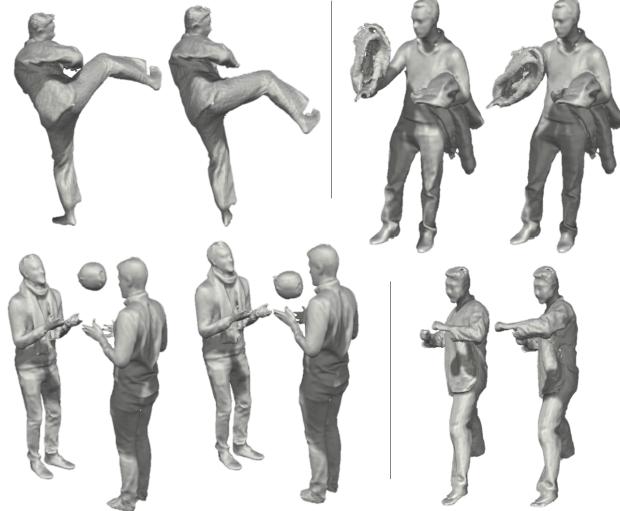


Figure 10: Our approach is robust to fast motions.

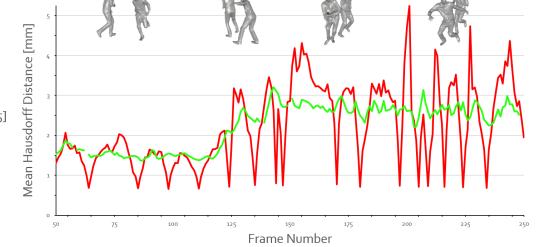


Figure 11: Quantitative comparison with [Collet et al. 2015]

tems would fail. Fig. 10 depicts typical situations where the small motion assumption does not hold. This robustness is 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 $> 5\text{mm}$. 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

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-

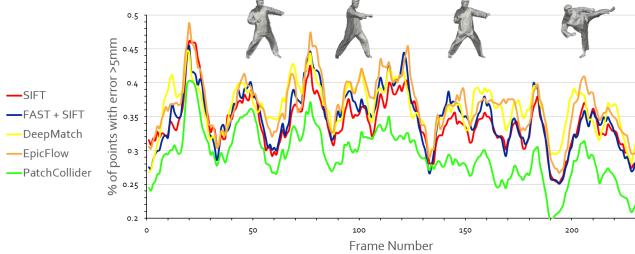


Figure 12: Quantitative comparisons of different correspondence methods: SIFT, FAST+SIFT, DeepMatch, EpicFlow and Global Patch Collider. We computed the residual error and reported the percentage of vertexes with error $> 5\text{mm}$. The method proposed in Sec. 5.2.4 achieved the best score with only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

[799] [Revaud et al. 2015] and our extension of Global Patch Collider [800] [Wang et al. 2016] described in Sec. 5.2.4. Quantitative results on [801] this fast motion sequence are reported in Fig. 12. The best results [802] are obtained by our method with 29% outliers, then SIFT (34%), [803] FAST+SIFT (34%), DeepMatch (36%) and EpicFlow (36%). Most [804] of the error occurred in regions where very large motion is present: [805] a qualitative comparison is depicted in Fig. 13.

806 7.4 Nonrigid Reconstruction Comparisons

In Fig. 15, we compare to the dataset of [Collet et al. 2015] for a [807] sequence with extremely high motions. The figure compares renderings [808] of the original meshes and multiple reconstructions, where red [809] corresponds to a fitting error of 15mm. In particular, we compare [810] our method with [Zollhöfer et al. 2014] and [Newcombe et al. 2015], [811] showing our superior reconstructions in these challenging situations. [812] We also show results and distance metrics for the method of [Collet [813] et al. 2015] which is an offline technique with a runtime of about [814] 30 minutes per frame on the CPU, and runs with 30 more cameras [815] than our system. In a more quantitative analysis (Fig. 11) we plot [816] the error over the input mesh for our method and [Collet et al. 2015], [817] which shows that our algorithm can match the motion and fine scale [818] details exhibited in this sequence. Our approach shows qualitatively [819] similar results but with a system that is about 4 orders of magnitude [820] faster, allowing for true real-time performance capture. [821]

Finally, multiple qualitative comparisons among different state of [822] the art methods are shown in Fig. 14. These sequences exhibits all [823] classical situations where online methods fail, such as large motions [824] and topology changes. Again our real-time reconstruction methods [825] correctly retrieves the non rigid shapes for any of these scenarios. [826] Please also see accompanying video figure. [827]

828 8 Limitations

Even though we demonstrated one of the first methods for real-time [829] nonrigid reconstruction from multiple views, showing reconstruction [830] of challenging scenes, our system is not without limitations. Given [831] the tight real-time constraint (33ms/frame) of our approach, we [832] rely on temporal coherence of the RGBD input stream making [833] the processing at 30Hz a necessity. If the frame rate is too low [834] or frame-to-frame motion is too large, either the frame-to-frame [835] correspondences would be inaccurately estimated or the nonrigid [836] alignment would fail to converge given the tight time budget. In [837] either case our method might lose tracking. In both scenarios our [838] system does fall back to the live fused data. However, as shown [839] in Fig. 16 the volume blending can look noisy as new data is first [840] being fused. Another issue in our current work is robustness to [841]

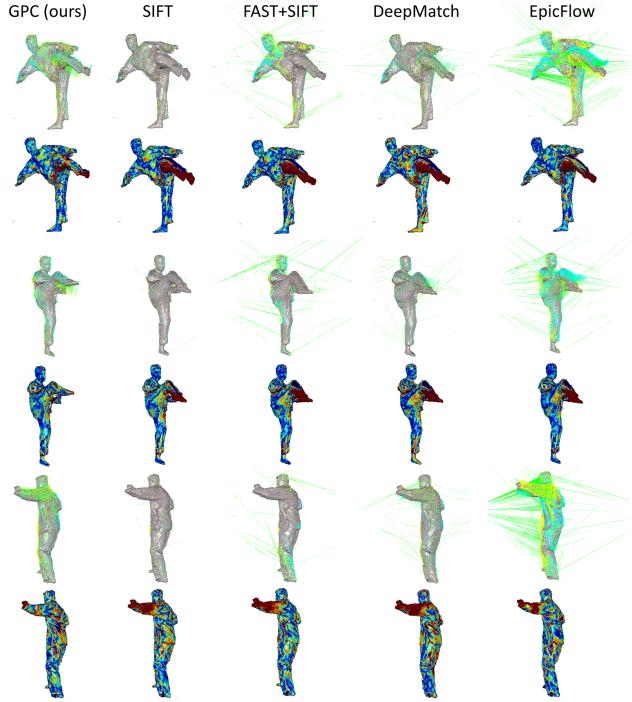


Figure 13: Qualitative comparisons of correspondence algorithms. We show the detected correspondences (green lines) between the previous frame (yellow points) and current frame (cyan points). GPC shows less residual error in fast motion regions, whereas current state of the art algorithms (DeepMatch, 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).

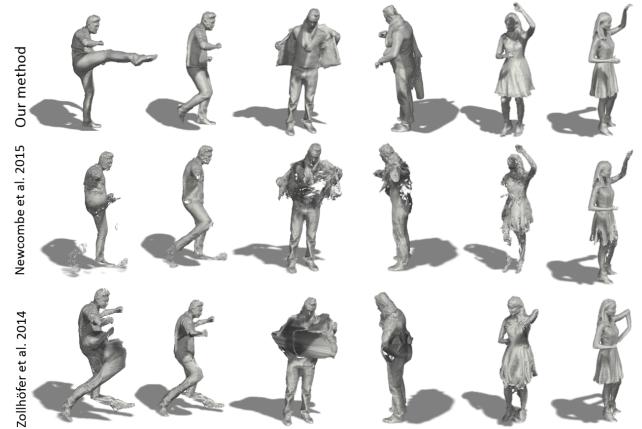


Figure 14: Qualitative comparisons with state of the art approaches.

842 segmentation errors. Large segmentation errors, if there is missing
843 depth data for instance, can lead to incorrect visual hull estimation.
844 This can cause some noise to be integrated into the model as shown
845 in Fig. 16. Finally, any small nonrigid alignment errors can cause
846 slight oversmoothing of the model at times e.g. Fig. 16. We deal
847 with topology change by refreshing correspondence voxels. This
848 strategy works in general, but has artifacts when one object slides
849 over another surface, e.g., unzipping a jacket. To solve the topology
850 problem intrinsically, a nonrigid matching algorithm that explicitly
851 handles topology changes needs to be designed.

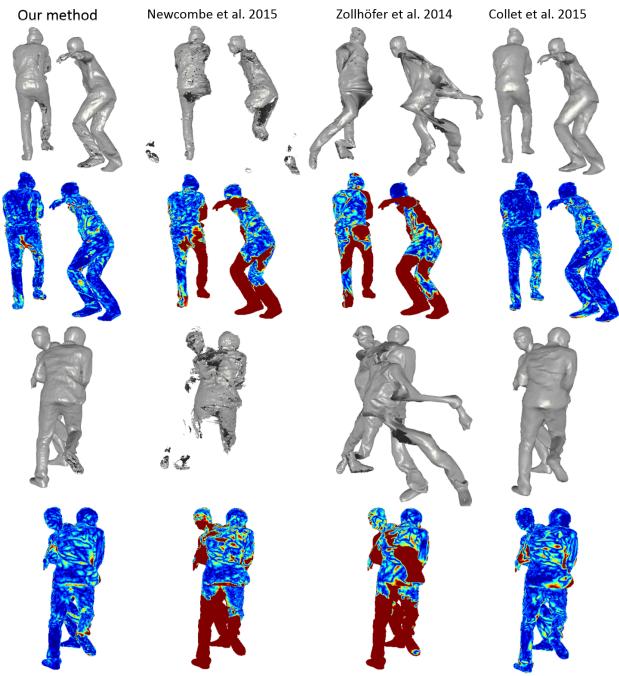


Figure 15: Qualitative comparisons with the high quality offline system of [Collet et al. 2015].



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.

References

- 868 BEELER, T., HAHN, F., BRADLEY, D., BICKEL, B., BEARDSLEY,
869 P., GOTSMAN, C., SUMNER, R. W., AND GROSS, M. 2011.
870 High-quality passive facial performance capture using anchor
871 frames. *ACM Transactions on Graphics (TOG)* 30, 4, 75.
- 873 BLEYER, M., RHEMANN, C., AND ROTHER, C. 2011. Patchmatch
874 stereo: Stereo matching with slanted support windows. In *Proc.
875 BMVC*, vol. 11, 1–11.
- 876 BOGO, F., BLACK, M. J., LOPER, M., AND ROMERO, J. 2015.
877 Detailed full-body reconstructions of moving people from monocular
878 RGB-D sequences. In *ICCV*, 2300–2308.
- 879 BOJSEN-HANSEN, M., LI, H., AND WOJTAN, C. 2012. Tracking
880 surfaces with evolving topology. *ACM Trans. Graph.* 31, 4, 53.
- 881 BRADLEY, D., POPA, T., SHEFFER, A., HEIDRICH, W., AND
882 BOUBEKEUR, T. 2008. Markerless garment capture. *ACM TOG
883 (Proc. SIGGRAPH)* 27, 3, 99.
- 884 CAGNIART, C., BOYER, E., AND ILIC, S. 2010. Free-form mesh
885 tracking: a patch-based approach. In *Proc. CVPR*.
- 886 CHEN, Y., AND MEDIONI, G. 1992. Object modelling by registra-
887 tion of multiple range images. *CVIU* 10, 3, 144–155.
- 888 CHEN, J., BAUTEMBACH, D., AND IZADI, S. 2013. Scalable
889 real-time volumetric surface reconstruction. *ACM TOG*.
- 890 COLLET, A., CHUANG, M., SWEENEY, P., GILLETT, D., EVSEEV,
891 D., CALABRESE, D., HOPPE, H., KIRK, A., AND SULLIVAN,
892 S. 2015. High-quality streamable free-viewpoint video. *ACM
893 TOG* 34, 4, 69.
- 894 CURLESS, B., AND LEVOY, M. 1996. A volumetric method for
895 building complex models from range images. In *Proceedings of
896 the 23rd annual conference on Computer graphics and interactive
897 techniques*, ACM, 303–312.
- 898 DE AGUIAR, E., STOLL, C., THEOBALT, C., AHMED, N., SEIDEL,
899 H.-P., AND THRUN, S. 2008. Performance capture from sparse
900 multi-view video. *ACM TOG (Proc. SIGGRAPH)* 27, 1–10.
- 901 DOU, M., FUCHS, H., AND FRAHM, J.-M. 2013. Scanning and
902 tracking dynamic objects with commodity depth cameras. In
903 *Proc. ISMAR*, IEEE, 99–106.
- 904 DOU, M., TAYLOR, J., FUCHS, H., FITZGIBBON, A., AND IZADI,
905 S. 2015. 3d scanning deformable objects with a single rgbd
906 sensor. In *CVPR*.
- 907 ENGELS, C., STEWÉNIUS, H., AND NISTÉR, D. 2006. Bundle
908 adjustment rules. *Photogrammetric computer vision* 2, 124–131.
- 909 GALL, J., STOLL, C., DE AGUIAR, E., THEOBALT, C., ROSEN-
910 HAHN, B., AND SEIDEL, H.-P. 2009. Motion capture using joint
911 skeleton tracking and surface estimation. In *Proc. CVPR*, IEEE,
912 1746–1753.
- 913 GUO, K., XU, F., WANG, Y., LIU, Y., AND DAI, Q. 2015. Robust
914 non-rigid motion tracking and surface reconstruction using ℓ_0
915 regularization. In *ICCV*, 3083–3091.
- 916 KRÄHENBÜH, P., AND KOLTUN, V. 2011. Efficient inference in
917 fully connected crfs with gaussian edge potentials. *NIPS*.
- 918 KUTULAKOS, K. N., AND SEITZ, S. M. 2000. A theory of shape
919 by space carving. *IJCV*.
- 920 LI, H., ADAMS, B., GUIBAS, L. J., AND PAULY, M. 2009. Robust
921 single-view geometry and motion reconstruction. *ACM TOG*.
- 922 LOWE, D. G. 2004. Distinctive image features from scale-invariant
923 keypoints. *IJCV*.

9 Conclusions

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

- 924 MITRA, N. J., FLÖRY, S., OVSJANIKOV, M., GELFAND, N., 981
 925 GUIBAS, L. J., AND POTTMANN, H. 2007. Dynamic geometry 982
 926 registration. In *Proc. SGP*, 173–182.
- 927 MORI, M., MACDORMAN, K. F., AND KAGEKI, N. 2012. The 985
 928 uncanny valley [from the field]. *Robotics & Automation Magazine,* 986
IEEE 19, 2, 98–100.
- 930 NEWCOMBE, R. A., IZADI, S., HILLIGES, O., MOLYNEAUX, D., 987
 931 KIM, D., DAVISON, A. J., KOHLI, P., SHOTTON, J., HODGES, 988
 932 S., AND FITZGIBBON, A. 2011. KinectFusion: Real-time dense 989
 933 surface mapping and tracking. In *Proc. ISMAR*, 127–136.
- 934 NEWCOMBE, R. A., FOX, D., AND SEITZ, S. M. 2015. Dy- 990
 935 namicfusion: Reconstruction and tracking of non-rigid scenes in 991
 936 real-time. In *CVPR*, 343–352.
- 937 PONS-MOLL, G., TAYLOR, J., SHOTTON, J., HERTZMANN, A., 992
 938 AND FITZGIBBON, A. 2015. Metric regression forests for corre- 993
 939 spondence estimation. *IJCV 113*, 3, 163–175.
- 940 PRADEEP, V., RHEMANN, C., IZADI, S., ZACH, C., BLEYER, 994
 941 M., AND BATHICHE, S. 2013. MonoFusion: Real-time 3D 995
 942 reconstruction of small scenes with a single web camera. In *Proc.* 996
ISMAR, IEEE, 83–88.
- 944 REVAUD, J., WEINZAEPFEL, P., HARCHAOUI, Z., AND SCHMID, 997
 945 C. 2015. Epicflow: Edge-preserving interpolation of correspon- 998
 946 dences for optical flow. *CVPR*.
- 947 ROSTEN, E., AND DRUMMOND, T. 2005. Fusing points and lines 999
 948 for high performance tracking. In *ICCV*.
- 949 RUSINKIEWICZ, S., AND LEVOY, M. 2001. Efficient variants of 1000
 950 the icp algorithm. In *3DIM*, 145–152.
- 951 SHOTTON, J., GLOCKER, B., ZACH, C., IZADI, S., CRIMINISI, 1001
 952 A., AND FITZGIBBON, A. 2013. Scene coordinate regression 1002
 953 forests for camera relocalization in rgbd images. In *CVPR*.
- 954 SMOLIC, A. 2011. 3d video and free viewpoint videofrom capture 1003
 955 to display. *Pattern recognition 44*, 9, 1958–1968.
- 956 STARCK, J., AND HILTON, A. 2007. Surface capture for 1004
 957 performance-based animation. *Computer Graphics and Applica-*
958 tions 27, 3, 21–31.
- 959 STOLL, C., HASLER, N., GALL, J., SEIDEL, H., AND THEOBALT, 1005
 960 C. 2011. Fast articulated motion tracking using a sums of gaus- 1006
 961 sians body model. In *Proc. ICCV*, IEEE, 951–958.
- 962 SUMNER, R. W., SCHMID, J., AND PAULY, M. 2007. Embedded 1007
 963 deformation for shape manipulation. *ACM TOG 26*, 3, 80.
- 964 TEVS, A., BERNER, A., WAND, M., IHRKE, I., BOKELOH, M., 1008
 965 KERBER, J., AND SEIDEL, H.-P. 2012. Animation cartography- 1009
 966 intrinsic reconstruction of shape and motion. *ACM TOG*.
- 967 THEOBALT, C., DE AGUIAR, E., STOLL, C., SEIDEL, H.-P., AND 1010
 968 THRUN, S. 2010. Performance capture from multi-view video. In 1011
Image and Geometry Processing for 3D-Cinematography, R. Ron-
 969 fard and G. Taubin, Eds. Springer, 127ff.
- 971 VINEET, V., WARRELL, J., AND TORR, P. H. S. 2012. Filter-based 1012
 972 mean-field inference for random fields with higher-order terms 1013
 973 and product label-spaces. In *ECCV*.
- 974 VLASIC, D., BARAN, I., MATUSIK, W., AND POPOVIĆ, J. 2008. 1014
 975 Articulated mesh animation from multi-view silhouettes. *ACM 1015
 976 TOG (Proc. SIGGRAPH)*.
- 977 VLASIC, D., PEERS, P., BARAN, I., DEBEVEC, P., POPOVIC, J., 1016
 978 RUSINKIEWICZ, S., AND MATUSIK, W. 2009. Dynamic shape 1017
 979 capture using multi-view photometric stereo. *ACM TOG (Proc.*
SIGGRAPH Asia 28, 5, 174.
- 980 WAND, M., ADAMS, B., OVSJANIKOV, M., BERNER, A., 1018
 981 BOKELOH, M., JENKE, P., GUIBAS, L., SEIDEL, H.-P., AND 1019
 982 SCHILLING, A. 2009. Efficient reconstruction of nonrigid shape 1020
 983 and motion from real-time 3D scanner data. *ACM TOG*.
- 985 WANG, S., FANELLO, S. R., RHEMANN, C., IZADI, S., AND 1021
 986 KOHLI, P. 2016. The global patch collider. *CVPR*.
- 987 WASCHBÜSCH, M., WÜRMLIN, S., COTTING, D., SADLO, F., 1022
 988 AND GROSS, M. 2005. Scalable 3D video of dynamic scenes. In 1023
Proc. Pacific Graphics, 629–638.
- 990 WEI, L., HUANG, Q., CEYLAN, D., VOUGA, E., AND LI, H. 1024
 991 2015. Dense human body correspondences using convolutional 1025
 992 networks. *arXiv preprint arXiv:1511.05904*.
- 993 WEINZAEPFEL, P., REVAUD, J., HARCHAOUI, Z., AND SCHMID, 1026
 994 C. 2013. Deepflow: Large displacement optical flow with deep 1027
 995 matching. In *ICCV*.
- 996 YE, M., AND YANG, R. 2014. Real-time simultaneous pose 1028
 997 and shape estimation for articulated objects using a single depth 1029
 998 camera. In *CVPR*, IEEE.
- 999 YE, M., ZHANG, Q., WANG, L., ZHU, J., YANG, R., AND GALL, 1030
 1000 J. 2013. A survey on human motion analysis from depth data. 1031
 1001 In *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and*
1002 Applications. Springer, 149–187.
- 1003 ZACH, C. 2014. Robust bundle adjustment revisited. In *Computer 1033
 1004 Vision—ECCV 2014*. Springer, 772–787.
- 1005 ZENG, M., ZHENG, J., CHENG, X., AND LIU, X. 2013. Template- 1034
 1006 less quasi-rigid shape modeling with implicit loop-closure. In 1035
 1007 *Proc. CVPR*, IEEE, 145–152.
- 1008 ZHANG, Q., FU, B., YE, M., AND YANG, R. 2014. Quality 1036
 1009 dynamic human body modeling using a single low-cost depth 1037
 1010 camera. In *CVPR*, IEEE, 676–683.
- 1011 ZOLLHÖFER, M., NIESSNER, M., IZADI, S., RHEMANN, C., 1038
 1012 ZACH, C., FISHER, M., WU, C., FITZGIBBON, A., LOOP, C., 1039
 1013 THEOBALT, C., ET AL. 2014. Real-time non-rigid reconstruction 1040
 1014 using an rgbd camera. *ACM TOG*.