Real-time Non-rigid Reconstruction using an RGB-D Camera

Michael Zollhöfer[1] Matthias Nießner[2] Shahram Izadi[3] Christoph Rhemann[3] Christopher Zach[3] Matthew Fisher[2]
Chenglei Wu[4] Andrew Fitzgibbon[3] Charles Loop[3] Christian Theobalt[4] Marc Stamminger[1]

[1]University of Erlangen-Nuremberg [2]Stanford University [3]Microsoft Research [4]Max Planck Institute for Informatics

Figure 1: *Our system enables the real-time capture of general shapes undergoing non-rigid deformations using a single depth camera.* ***Top left:*** *the object to be captured is scanned while undergoing rigid deformations, creating a base template.* ***Bottom left:*** *the object is manipulated and our method deforms the template to track the object.* ***Top and middle row:*** *we show our reconstruction for upper body, face, and hand sequences being captured in different poses as they are deformed.* ***Bottom row:*** *we show corresponding color and depth data for the reconstructed mesh in the middle row.*

Abstract

We present a combined hardware and software solution for markerless reconstruction of non-rigidly deforming physical objects with arbitrary shape in *real-time*. Our system uses a single self-contained stereo camera unit built from off-the-shelf components and consumer graphics hardware to generate spatio-temporally coherent 3D models at 30 Hz. A new stereo matching algorithm estimates real-time RGB-D data. We start by scanning a smooth template model of the subject as they move rigidly. This geometric surface prior avoids strong scene assumptions, such as a kinematic human skeleton or a parametric shape model. Next, a novel GPU pipeline performs non-rigid registration of live RGB-D data to the smooth template using an extended non-linear as-rigid-as-possible (ARAP) framework. High-frequency details are fused onto the final mesh using a linear deformation model. The system is an order of magnitude faster than state-of-the-art methods, while matching the quality and robustness of many offline algorithms. We show precise real-time reconstructions of diverse scenes, including: large deformations of users' heads, hands, and upper bodies; fine-scale wrinkles and folds of skin and clothing; and non-rigid interactions performed by users on flexible objects such as toys. We demonstrate how acquired models can be used for many interactive scenarios, including re-texturing, online performance capture and preview, and real-time shape and motion re-targeting.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Digitizing and Scanning; I.4.1 [Image Processing]: Digitization and Image Capture—Scanning

Keywords: non-rigid, deformation, shape, surface reconstruction, 3D scanning, stereo matching, depth camera

Links: DL PDF

1 Introduction

Acquiring 3D models of the real-world is a long standing problem in computer vision and graphics. For static scenes, *real-time* reconstruction techniques are now highly mature [Newcombe et al. 2011; Izadi et al. 2011; Nießner et al. 2013]. However, real-time reconstruction of non-rigidly deforming objects remains challenging. The ability to reconstruct the fine-grained non-rigid motions and shape of physical objects in a live and temporally consistent manner opens up many applications. For example, in real-time, a user can re-target their motions and detailed expressions to avatars for gaming or video conferencing. An actor's performance and motions can be captured

ACM Reference Format
Zollhofer, M., Nießner, M., Izadi, S., Rhemann, C., Zach, C., Fisher, M., Wu, C., Fitzgibbon, A., Loop, C., Theobalt, C., Stamminger, M. 2014. Real-time Non-rigid Reconstruction using an RGB-D Camera. ACM Trans. Graph. 33, 4, Article 156 (July 2014), 12 pages. DOI = 10.1145/2601097.2601165 http://doi.acm.org/10.1145/2601097.2601165.

DOI: http://doi.acm.org/10.1145/2601097.2601165

online for live feedback and preview. By reconstructing the detailed motion and shape of surfaces, systems can overlay digital content onto the physical world in more convincing ways; e.g., for virtual clothing, makeup, and other augmented reality applications. Finally, deforming physical objects can become props for digital interaction, further bridging the gap between real and virtual worlds.

Despite considerable advances in the field of non-rigid tracking and reconstruction, there has been limited work on real-time techniques that work on general scenes. In special cases such as hands, faces or full bodies, researchers have demonstrated compelling real-time reconstructions of non-rigid articulated motion [Oikonomidis et al. 2011; Taylor et al. 2012] and shape [Weise et al. 2011; Cao et al. 2013]. However, these rely on strong priors based on either pre-learned statistical models, articulated skeletons, or morphable shape models, prohibiting capture of general scenes. Reconstruction techniques that can handle more general scenes are far from real-time in terms of performance, and need seconds to hours to compute a single frame [Hernández et al. 2007; Liao et al. 2009; Li et al. 2009].

In this paper, we present the first real-time reconstruction system capable of capturing a variety of non-rigid shapes and their deformations. As demonstrated, our entire pipeline from depth acquisition to non-rigid deformation runs at 33ms per frame, orders of magnitude faster than state-of-the-art methods, while achieving reconstruction quality and robustness that approach offline methods with more complex sensor setups.

Our system is markerless, uses a single self-contained stereo camera unit, and consumer graphics hardware. The stereo camera is built from off-the-shelf components and uses a new stereo matching algorithm to generate RGB-D images, with a greater degree of flexibility and accuracy than current consumer depth cameras. Using this camera, a smooth template model of the rigidly moving subject is acquired online. This acts as a geometric and topological prior for non-rigid reconstruction, but avoids strong assumptions about the scanned scene, such as a kinematic skeleton or a parametric shape model (e.g., for faces, hands or bodies) that would limit the generality of our system. Then, for each live RGB-D frame, a novel GPU pipeline performs non-rigid registration to the acquired template model with an as-rigid-as-possible (ARAP) regularizer and integrates detail using a thin shell deformation model on a displacement map [Sorkine and Alexa 2007].

We show precise real-time reconstructions of diverse scenes, including: large deformations of users' heads, hands, and upper bodies; fine-scale wrinkles and folds of skin and clothing; and non-rigid interactions performed by users on flexible objects such as toys. We demonstrate how acquired models can be used for many interactive scenarios, including re-texturing, online performance capture and preview, and real-time shape and motion re-targeting.

The specific contributions of our work are:

- A general, real-time, non-rigid reconstruction pipeline, non-trivially realized on the GPU. While in the spirit of previous non-rigid reconstruction frameworks, in particular Li et al. [2009], our GPU pipeline is orders of magnitude faster, with many algorithmic and implementation differences.
- The creation of a fully automatic real-time non-rigid capture system. This allows novice users to quickly generate template models of arbitrary objects, whose motions and non-rigid deformations can be captured with live user feedback.
- An interactive application of our non-rigid reconstruction pipeline that demonstrates spatio-temporal coherent models for motion and shape re-targeting in video games, performance capture, and augmented reality.
- A lightweight visible light and infrared (IR) stereo camera setup for generating compelling RGB-D input at real-time rates, which allows us to capture higher quality RGB-D data at closer ranges than consumer depth cameras.

2 Related Work

The emergence of depth cameras, such as the Kinect, has spawned new interest in *real-time* rigid 3D scanning as exemplified by systems such as KinectFusion [Newcombe et al. 2011; Izadi et al. 2011]. It is therefore a natural next step to think about online capture of non-rigid scenes using RGB-D cameras. Interestingly, follow-up work based on KinectFusion specifically focused on scanning humans (e.g., for 3D printing or generating avatars) where the user rotates in front of the Kinect while maintaining a roughly rigid pose, e.g., [Weiss et al. 2011; Li et al. 2013a; Tong et al. 2012; Zeng et al. 2013; Helten et al. 2013]. Similar to [Brown and Rusinkiewicz 2007; Weise et al. 2009a], in these *offline* systems non-rigid registration techniques are employed to accommodate for small deviations in the motion between different viewpoints. These systems are motivated by producing a *single* mesh as output from an RGB-D sequence, whereas we wish to continuously reconstruct non-rigid motions during live capture.

Many multi-camera techniques for non-rigid reconstruction of geometry and motion have been proposed. Some are specifically motivated by modeling 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]. Model-based approaches deform a static shape template (obtained by a laser scan or image-based reconstruction) such that it matches a human [de Aguiar et al. 2008; Vlasic et al. 2008; Gall et al. 2009] or a person's apparel [Bradley et al. 2008]. Vlasic et al. [2009] use dynamic photometric stereo in a sophisticated controlled light stage dome with multiple high-speed cameras to capture temporally incoherent 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 non-rigidly deforming surface [Vlasic et al. 2008; Gall et al. 2009], while some 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].

These multi-camera approaches have runtime performances far from real-time, and require dense camera setups in controlled studios, with sophisticated lighting and/or chroma-keying for background subtraction. However, with the availability of consumer depth cameras, other more 'lightweight' camera setups have been proposed [Hernández et al. 2007; Liao et al. 2009; Li et al. 2009; Weise et al. 2011; Valgaerts et al. 2012; Chen et al. 2012; Wu et al. 2013; Garrido et al. 2013]. Ye et al. [2012] capture multi-person performances with three moving Kinects. Furthermore, in special cases, such as for hands, faces and full bodies, researchers have demonstrated compelling *real-time* reconstructions of articulated motion [Oikonomidis et al. 2011; Wei et al. 2012; Taylor et al. 2012] and/or non-rigid shape and motion [Weise et al. 2011; Cao et al. 2013; Helten et al. 2013]. However, these rely on strong priors based on either an offline learned model [Taylor et al. 2012], an articulated skeleton [Oikonomidis et al. 2011] or morphable shape model [Blanz and Vetter 1999; Weise et al. 2011; Helten et al. 2013; Cao et al. 2013], which prohibits capture of general scenes. Additionally, these real-time methods are unable to reconstruct high-frequency shape detail obtained with state-of-the-art offline approaches.

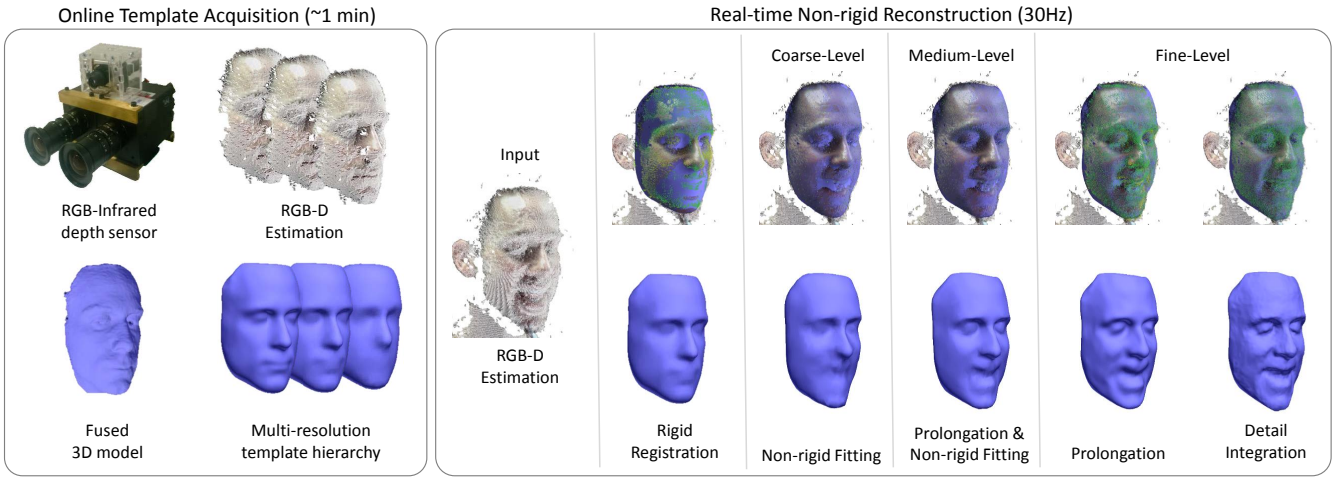


Figure 2: *Main system pipeline.* **Left:** *the initial template acquisition is an online process. Multiple views are volumetrically fused, and a multi-resolution mesh hierarchy is precomputed for the tracking phase.* **Right:** *in the tracking phase, each new frame is rigidly registered to the template, and a sequence of calls to the GPU-based Gauss-Newton optimizer is issued from coarse to fine mesh resolution. At the finest resolution, detail is integrated using a thin-plate spline regularizer on the finest mesh.*

The approach of [Li et al. 2009] uses a coarse approximation of the scanned object as a shape prior to obtain high quality non-rigid reconstructions. Other non-rigid techniques do not require 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 oversmoothing of results for longer sequences [Tevs et al. 2012; Bojsen-Hansen et al. 2012]. These more general techniques are far from real-time, ranging from seconds to hours to compute a single frame.

Our system attempts to hit a 'sweet spot' between methods that can reconstruct general scenes, and techniques that rely on a stronger shape prior (e.g., a blendshape face model or body or hand skeleton), which are beginning to demonstrate real-time performance. To our knowledge, our system is the first that provides real-time performance, several orders of magnitude faster than general methods, but does not require a specific 'baked in' kinematic or shape model. Instead our system allows users to acquire a template online, and use this for *live* non-rigid reconstructions. Our system is simple to use and self-contained, with a single lightweight stereo camera setup, moving closer to commodity or consumer use. Additionally, in terms of reconstructed geometry detail, our method also narrows that gap between offline and online methods. This simplicity and real-time performance however does not come at a significant cost of reconstruction quality (as shown in the results section), and brings us a step closer to high-quality real-time performance capture systems for consumer scenarios, including gaming, home and semi-professional movie and animation production, and human-computer interaction.

3 System Overview

Our system is designed to deal with close range non-rigid reconstructions of single objects, such as faces, hands, upper bodies, or hand held physical objects. The general usage scenario is illustrated in Fig. 1, and the system pipeline in Fig. 2, and comprises two phases: online template acquisition and real-time non-rigid reconstruction.

The first part of the pipeline is a *online template acquisition* phase that takes $\sim$1 minute to perform. The user sits or stands in front of our custom RGB-D sensor (up to 1½ meters away from the sensor). First, the desired object is scanned while undergoing mostly rigid deformations. Immediate feedback is provided during scanning using the volumetric fusion framework of Nießner et al. [2013], from which a triangle mesh model is automatically extracted. The mesh is preprocessed to create a multi-resolution hierarchy to be used in the online phase.

The second phase of our pipeline performs *real-time non-rigid reconstruction*, which produces a deformed mesh at every time step, executing the following three steps at every frame:

1. **Rigid registration** roughly aligns the template to the input data.
2. **Non-rigid surface fitting** by minimization of a fitting energy which combines dense geometric and photometric model-to-data constraints, as well as an as-rigid-as-possible (ARAP) regularizer. The energy is minimized using a new efficient GPU-based Gauss-Newton solver using the preconditioned conjugate gradient method (PCG) in its inner loop. This solver is applied in a coarse-to-fine manner, using the multi-resolution mesh hierarchy prepared at template acquisition. At each level, the fitting energy is optimized at the current resolution using several iterations of Gauss-Newton, and then a *prolongation* step interpolates the solution to the next finer level.
3. **Detail integration** at the finest template level: a thin-shell deformation energy under model-to-data constraints is minimized by solving a linear least squares system for displacements along the model normal at each vertex.

These components are now explained in detail after a description of our custom stereo sensor.

4 Lightweight Active Stereo Sensor

For acquisition, we designed a new *RGB-IR* stereo rig which reconstructs pixel synchronized *RGB-D data* in real-time. The use of a custom depth camera provides us with a greater deal of flexibility than consumer sensors. In particular, our specific scenario requires close range capture at high quality, and most existing sensors are limited in these terms. For comparison, in Sec. 6 we will also present results of our system running with a consumer Kinect camera.

The sensor comprises a fully-calibrated pair of video cameras of resolution 1024×768 providing both RGB and IR images with the same center of projection (by employing a beam splitter). For high-quality depth computation, we employ Kinect-type infrared emitters in order to project a suitable pattern onto the surface to be reconstructed. In contrast to the Kinect sensor in our setup, the emitters are not calibrated with respect to the cameras and can be placed freely to maximize the coverage of the emitted pattern in the scene. For further technical details see the supplemental material.

For real-time depth acquisition we use a patch-match based stereo algorithm inspired by [Bleyer et al. 2011], but in analogy to [Pradeep et al. 2013] we reduce the search space complexity significantly by not estimating local surface normals. Thus, the only unknown to be determined at pixel p (at location (u_p, v_p)) is the scene depth D_p. Further, we deviate from the original propagation schedule of patch-match stereo to achieve better GPU utilization. Our patch-match stereo algorithm can be summarized as follows: the starting phase to initialize the depth map with random samples is followed by four (left-to-right, top-to-bottom, right-to-left, and bottom-to-top) propagation steps, where the matching score of the current depth hypothesis is compared with the one of the respective neighboring pixel, and the better scoring depth value is retained. This propagation strategy allows all rows (or columns) of the image to be easily processed in parallel in the GPU implementation. We use the zero-mean normalized cross-correlation (ZNCC) computed over 7×7 windows as a matching score to assess the similarity of image patches. The advantage of our patch-match stereo implementation is its high speed (100 Hz to estimate 1024×768 depth images), but all patch-match inspired algorithms produce piece-wise constant outputs with an undesired "blocky" appearance. Consequently, we refine the raw depth map produced by our patch-match stereo method using a variational approach, which combines the (local) matching score profile with a global smoothness prior as follows: if we denote the depth map returned by patch-match stereo as $\widehat{D}_p$, then we minimize

$$\mathcal{E}(\mathbf{D}) = \sum_p \alpha_p \left(D_p - \widehat{D}_p\right)^2 + \sum_{(p,q)\in E} \omega_{pq} \left(D_p - D_q\right)^2 \quad (1)$$

with respect to the depth values $\mathbf{D} = [D_1, ..., D_{WH}]$, with E the set of 4-neighbor pixel pairs. We model the local behavior of the matching scores near $\widehat{D}_p$ using a quadratic model, i.e., we fit a quadratic function to the matching scores of $\widehat{D}_p - 1$, $\widehat{D}_p$, and $\widehat{D}_p + 1$. This determines the coefficient α_p. In general, the local quadratic model of matching scores should be a convex parabola, i.e., $\alpha_p > 0$, if $\widehat{D}_p$ is at a (local) minimum. We avoid a non-sensible concave parabola fitted to the matching scores by setting $\alpha_p = 0$ in these cases. Our regularizer prefers smooth depth maps, but we avoid smoothing over depth discontinuities by using a contrast-aware regularization term, i.e., we introduce weights $\omega_{pq} \in [0, 1]$ for neighboring pixels p and q, which are based on strong color edges in the RGB images, $\omega_{pq} = 1/(1 + \beta\|\nabla I^L(u_p, v_p)\|)$. Here $I^L(\cdot)$ denotes the left color image, and β is a tuning parameter always set to 20. The objective in Eq. 1 is quadratic in the unknowns $\mathbf{D}$, and we use a GPU-implemented successive over-relaxation (SOR) solver to obtain the refined depth values. Since we are only interested in estimating and retaining depth for foreground objects, we utilize a simple, color-based background subtraction step to discard depth values corresponding to undesired background.

Our active sensor has a variety of advantages over existing real-time scanners or low-cost depth cameras. The use of active (infrared) illumination allows high-quality shape acquisition without solely relying on the object's texture (as in passive stereo) or distorting the color image (as in some fringe-based techniques). Our stereo setup allows the baseline between the cameras to be modified easily in order to adapt to the observed volume of interest. Changing the baseline in our setup requires a standard geometric calibration procedure to determine the new relative pose between the cameras. This is in contrast to the Kinect camera, which has a fixed baseline and would require a more difficult projector-camera calibration if the baseline is changed. Compared to time-of-flight cameras, it features a much higher depth and image resolution and does not suffer from their well-known systematic data distortions due to light modulation, reflectance dependencies and multi-path light transport [Kolb et al. 2009]. We employ a prototype setup built from standard vision cameras and do not have the same small form factor as mass-manufactured depth cameras. However, in mass production similar form factors and production cost could be achieved while maintaining its technical advantages.

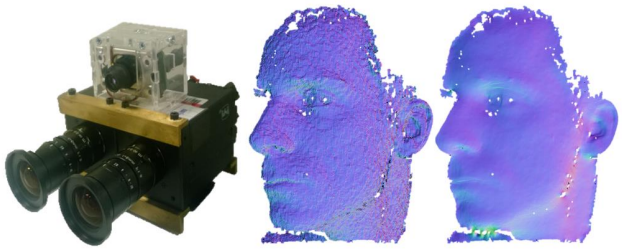


Figure 3: *Left: our active stereo sensor. Middle: patch-match result. Right: Result after variational refinement.*

5 Surface tracking as model fitting

Our template model is a hierarchy of triangle meshes (typically three levels; see Fig. 2) where vertices of a finer level are connected by a space deformation to the next coarser level [Sumner et al. 2007]. This connection is used to apply the prolongation operator (see Sec. 5.2.3). The hierarchy levels are computed through a series of mesh simplification and Laplacian smoothing steps. Note that the transition between template capture and non-rigid tracking is fully automated and seamless, requiring a few seconds to execute.

Each triangle mesh is defined by n vertices $\mathsf{V}^0 = \{\mathbf{v}_i^0 \in \mathbb{R}^3 \mid i = \{1, \ldots, n\}\}$ and m edges. The mesh topology is constant during tracking, and is queried only via the sets $\mathcal{N}_i$, which hold the indices of vertices sharing an edge with vertex i. This allows the use of non-manifold meshes, and indeed we use internal edges on some sequences to add a weak form of volume preservation; i.e., we tetrahedralize the template interior. Any internal vertices have visibility flags (see below) permanently zeroed.

5.1 Energy function

Our goal in surface tracking is to determine, at time t, the 3D positions of the model vertices $\mathsf{V}^t = \{\mathbf{v}_i^t\}_{i=1}^n$, and global rotation and translation $\mathbf{R}^t$, $\mathbf{t}^t$. This will be achieved by running a Gauss-Newton solver on a suitable energy function, using the values $(\mathsf{V}^{t-1}, \mathbf{R}^{t-1}, \mathbf{t}^{t-1})$ from the previous time step as an initial estimate. As each frame is processed otherwise independently, the t superscripts are dropped below.

We are given as input a depth image $\mathbf{d}$ which maps 2D points $\mathbf{u}$ to 3D world points using the sensor output and the known camera calibration information, so $\mathbf{d}(\mathbf{u}) \in \mathbb{R}^3$. We also have data normals $\mathbf{n} : \mathbb{R}^2 \mapsto \mathbb{R}^3$ computed by Sobel filtering on the depth map. These images are evaluated at non-integer locations using bilinear interpolation, and the derivatives $\nabla_{\mathbf{u}}\mathbf{d}(\mathbf{u})$ and $\nabla_{\mathbf{u}}\mathbf{n}(\mathbf{u})$ are therefore also well-defined.

The energy function is a sum of data terms, which encourage every visible model vertex to be as close as possible to the sampled data, and regularizers which control the smoothness of deformations and motion. **Visibility** is defined by a variable η_i associated with each model vertex, and is determined statically before each Gauss-Newton solve by rendering the model under the current parameters. In practice this computes correct visibilities for the majority of data points and a robust kernel in the energy handles the remainder.

5.1.1 Data terms

The **data term** measures the distance to the closest data point, and is written with an explicit minimization over the corresponding 2D image position $\mathbf{u}$:

$$\mathcal{E}_{\text{point}}(\mathsf{V}) = \lambda_{\text{point}} \sum_{i=1}^{n} \eta_i \min_{\mathbf{u}} \psi\left(\frac{\|\mathbf{v}_i - \mathbf{d}(\mathbf{u})\|}{\sigma_d}\right), \tag{2}$$

where σ_d is an estimate of sensor noise, and ψ is a robust kernel similar to Tukey's biweight, but with additional properties, described below. In practice, including the closest-point search within the energy gives a complicated energy surface, so we "lift" the inner minimizers to become search parameters $\mathsf{U} = \{\mathbf{u}_i\}_{i=1}^n$

$$\mathcal{E}_{\text{point}}(\mathsf{V}, \mathsf{U}) = \lambda_{\text{point}} \sum_{i=1}^{n} \eta_i \psi\left(\frac{\|\mathbf{v}_i - \mathbf{d}(\mathbf{u}_i)\|}{\sigma_d}\right). \tag{3}$$

Note that this is exact: $\mathcal{E}_{\text{point}}(\mathsf{V}) = \min_{\mathsf{U}} \mathcal{E}_{\text{point}}(\mathsf{V}, \mathsf{U})$. In essence, we are trading complexity of the energy surface for a $^5/_3$-fold increase in the number of unknowns. It does *not* imply an iterated closest point strategy of alternating minimization over V and U, which is known to have poor convergence properties. Rather, it suggests a simultaneous optimization over all unknowns, which is particularly important near the optimum. Also, as will be shown below, the new unknowns lead to a simple and sparse augmentation of the system Jacobian, so that optimization runtime is only very mildly affected by the increase in problem size. Although one could use the values from the previous timestep as an initial estimate for U, a more effective strategy is described in Sec. 5.3.

To further improve the properties of the energy, we adopt the common strategy of including a point-to-plane term

$$\mathcal{E}_{\text{plane}}(\mathsf{V}, \mathsf{U}) = \lambda_{\text{plane}} \sum_{i=1}^{n} \eta_i \psi\left(\frac{\mathbf{n}(\mathbf{u}_i)^\top (\mathbf{v}_i - \mathbf{d}(\mathbf{u}_i))}{\sigma_n}\right) \tag{4}$$

which allows incorrectly assigned correspondences to "slide" along the surface, thus improving convergence. Again, σ_n is a noise level estimate.

We further encourage vertices to preserve their RGB appearance from frame to frame with a color term

$$\mathcal{E}_{\text{color}}(\mathsf{V}) = \lambda_{\text{color}} \sum_{i=1}^{n} \eta_i \psi\left(\frac{\|\mathbf{I}_i - \mathbf{I}(\pi(\mathbf{v}_i))\|}{\sigma_c}\right), \tag{5}$$

where π is the projection from 3D to image coordinates, known from camera calibration, and $\mathbf{I}_i = \mathbf{I}^{t-1}(\pi(\mathbf{v}_i^{t-1}))$ is a color attached to each vertex from the previous timestep. In our implementation only the intensity channel is used. Here σ_c is the noise level of the RGB sensor.

5.1.2 Robust kernel

A further amelioration of the energy surface is obtained by rewriting the non-convex robust kernel $\psi(e)$ using a similar "lifting" technique as was used for the correspondences. As the wide variety of published robust kernels might indicate, the precise shape of ψ is not of great importance, but it should typically have a standard form: linear or quadratic for small e values, reducing to linear or constant for larger e. We observe that the function

$$\psi(e) = \min_{w} \left(\frac{2w^2e^2}{\tau^2} + (1-w^2)^2\right) = \begin{cases} \frac{e^2}{\tau^2}(2 - \frac{e^2}{\tau^2}) & \text{if } e^2 < \tau^2 \\ 1 & \text{otherwise} \end{cases}$$

has the shape in Fig. 4(a), which has the required properties. τ is a width parameter, normally set to 1. Applying this in our framework again uses the lifting trick, so that terms of the form

$$E(\Theta) = \sum_i \psi(f_i(\Theta)) = \sum_i \min_w \left(2w^2 f_i(\Theta)^2 + (1-w^2)^2\right)$$

become, when lifted to depend on parameters $\mathsf{W} = \{w_i\}_{i=1}^n$

$$E(\Theta, \mathsf{W}) = 2\sum_i w_i^2 f_i(\Theta)^2 + \sum_i (1-w_i^2)^2. \tag{6}$$

Again, the number of parameters increases, but the error function is more amenable to optimization (see also Fig. 4). Note that this is the same weighting used in [Li et al. 2008], but the connection to robust estimation was not made there. We avoid introducing separate W vectors for each energy term by applying the robust kernel to sums of terms per vertex. That is, we replace

$$\sum_{\alpha \in \{\text{point,plane,color}\}} \lambda_\alpha \sum_i \psi(f_i^\alpha(\Theta)) \rightarrow \sum_i \psi\left(\sqrt{\textstyle\sum_\alpha \lambda_\alpha f_i^\alpha(\Theta)^2}\right)$$

where the f^α are the residual terms in (3), (4), (5), so the robust kernel is applied to the sum of the squared residuals, not to each separately.

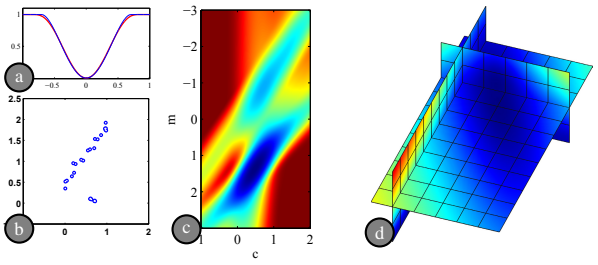


Figure 4: *Robust kernel (Sec. 5.1.2).* **(a)** *Our kernel $\psi(e)$ (blue) has similar shape to the standard Tukey's biweight kernel (red).* **(b)** *A 2D line fitting problem with two minima. Data points $y_i \approx mx_i + c$.* **(c)** *Energy landscape of $f(m,c) = \sum_i \psi(y_i - mx_i - c)$ is complicated.* **(d)** *3D slice through $(2+n)$ dimensional landscape of lifted function $F(m, c, w_1, ..., w_n) = \sum_i w_i^2(y_i - mx_i - c)^2 + (1 - w_i^2)^2$ is simpler. Minimization of lifted F found the global optimum on 82.4% of runs, in contrast to 43.0% on two-parameter f, which also had 20.1% outright failures vs. 0% on lifted.*

5.1.3 Shape regularizer

The geometric prior term $\mathcal{E}_{\text{reg}}$ forces local deformations of the surface to be as close possible to isometry, and thus approximates elastic deformation behavior. The as-rigid-as-possible (ARAP) framework [Sorkine and Alexa 2007] measures deformation between a pair of meshes $\mathsf{V}, \hat{\mathsf{V}}$ as follows

$$\begin{aligned} D(\mathsf{V}, \hat{\mathsf{V}}) &= \min_{\mathbf{R}_1,\ldots,\mathbf{R}_n} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i} \|(\mathbf{v}_i - \mathbf{v}_j) - \mathbf{R}_i(\hat{\mathbf{v}}_i - \hat{\mathbf{v}}_j)\|^2 \\ &= \min_{\mathcal{R}} D_{\text{ARAP}}(\mathsf{V}, \hat{\mathsf{V}}, \mathcal{R}) \end{aligned} \tag{7}$$

In our energy, ARAP controls the deformation of the shape in the current frame from the rigidly transformed initial template $\mathbf{R}\mathsf{V}^0 + \mathbf{t}$ as follows:

$$\mathcal{E}_{\text{reg}}(\mathsf{V}, \mathcal{R}, \mathbf{R}, \mathbf{t}) = \lambda_{\text{reg}} D_{\text{ARAP}}(\mathsf{V}, \mathbf{R}\mathsf{V}^0 + \mathbf{t}, \mathcal{R}) \tag{8}$$

The distance measure is itself a minimization problem over n rotations, and is typically solved via an alternating block coordinate descent strategy. Again, we prefer to lift the inner minimization parameters into a block $\mathcal{R} = \{\mathbf{R}_1, ..., \mathbf{R}_n\}$, and solve for them simultaneously with all others in order to enjoy the superlinear convergence of the Gauss-Newton method. Our implementation parameterizes $\mathcal{R}$ by Euler angles, so the energy is more correctly written $\mathcal{E}_{\text{reg}}(\mathsf{V}, \mathcal{R}(\Phi))$ where Φ is a vector of $3n$ angle parameters. Notice that the global transformation parameters are not strictly necessary here, because $D(\mathsf{V}, \hat{\mathsf{V}}) = D(\mathsf{V}, \mathbf{R}\hat{\mathsf{V}} + \mathbf{t})$, but their inclusion will improve our initial estimates, and ensures the Euler angles are always parameterizing near-identity rotations, avoiding gimbal lock.

5.2 Energy minimization: Gauss-Newton core solver

To summarize the above, we wish to minimize, at every timestep, the sum

$$\begin{aligned}\mathcal{E}(\mathsf{V}^t, \mathsf{U}, \mathsf{W}, \Phi, \mathbf{R}, \mathbf{t}) &= \mathcal{E}_{\text{point}}(\mathsf{V}^t, \mathsf{U}, \mathsf{W}) \\ &+ \mathcal{E}_{\text{plane}}(\mathsf{V}^t, \mathsf{U}, \mathsf{W}) + \mathcal{E}_{\text{color}}(\mathsf{V}^t) + \mathcal{E}_{\text{reg}}(\mathsf{V}^t, \mathcal{R}(\Phi), \mathbf{R}, \mathbf{t}).\end{aligned} \tag{9}$$

The main computational tool will be a Gauss-Newton solver. The primary requirement for such a solver is that the energy function be in the form of a sum of squared *residuals*, that is that if $\mathbf{x}$ is the vector of unknown parameters, we have

$$E(\mathbf{x}) = \sum_i f_i(\mathbf{x})^2 = \|\mathbf{f}(\mathbf{x})\|^2.$$

This form is ensured by the various lifting transformations described above, noting that terms of the form $\psi(\|\mathbf{e}\|)$, which expand to include $w^2\|\mathbf{e}\|^2 = w^2 e_1^2 + ...$, are stacked into $\mathbf{x}$ as $w\mathbf{e}$.

At solver iteration k, a Gauss-Newton iteration step updates a parameter vector $\mathbf{x}^k$ as

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \mathbf{h} \quad \text{with} \quad J^\top J\mathbf{h} = J^\top \mathbf{f} \tag{10}$$

where J is the Jacobian of $\mathbf{f}$ evaluated at $\mathbf{x}^k$, with $(i,j)^{\text{th}}$ entry $J_{ij} = \frac{\partial f_i}{\partial x_j}$.

To compute $\mathbf{h}$, we have to solve a linear system, which we do iteratively using a preconditioned conjugate gradient (PCG) solver, the essential computational unit of which is repeated multiplication of the Jacobian by a vector. The key to real-time performance is thus to implement routines for the computation of $J\mathbf{h}$ (and $J^\top\mathbf{h}$) as efficiently as possible. This is enabled by exploiting the particular sparsity structure of J, illustrated in Fig. 5. To make this structure as sparse as possible, we implement a hybrid optimizer: the global parameters $\mathbf{R}$ and $\mathbf{t}$ are estimated in an initial ICP step, and W is updated in an outer loop. This means the core solver is optimizing the $8n$ variables $\mathbf{x} = (\mathsf{V}, \mathsf{U}, \Phi)$, giving the structure in Fig. 5.

5.2.1 GPU implementation of Jacobian multiplication

We never compute the Jacobian explicitly, but compute its entries on-the-fly in the routines for $J\mathbf{h}$ and $J^\top\mathbf{h}$. This strategy, common in CPU-based optimizers (e.g., see [Wilamowski and Yu 2010], or the JacobMult parameter to MATLAB's `lsqnonlin`), has less computational overhead on the GPU [White et al. 2005], but does not appear to be widely employed in the vision, graphics, and learning literature, where many implementations appear to explicitly store an (unstructured) Jacobian.

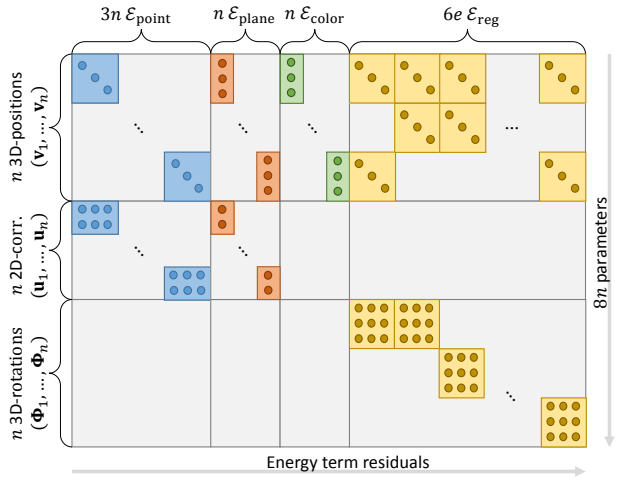


Figure 5: *Block structure of the* $(\mathsf{V}, \mathsf{U}, \Phi)$ *Jacobian (transposed). The Jacobian is never stored explicitly; instead the products* $J\mathbf{h}$ *and* $J^\top\mathbf{h}$ *are computed on demand on the GPU.*

To multiply J and $J^\top$ with a vector $\mathbf{h}$, we use two computational kernels. The first one multiplies a given row i of J with a given vector $\mathbf{h}$. According to the row i, the kernel determines which energy term is used, which columns are not equal to zero, computes the non-zero entries and immediately multiplies them with the appropriate entries of $\mathbf{h}$ and sums them up.

The second kernel does the same for $J^\top$. Yet, in this kernel each row corresponds to a parameter, so the kernel has to determine the non-zero entries in the i^{th} *column* of J and compute the scalar product. By this, we can well exploit the sparsity of J, and the GPU vector capabilities. Computing $J\mathbf{h}$ or $J^\top\mathbf{h}$ requires only one kernel call with $3n$ and $3n + 2m$ threads, respectively.

As a further optimization, both kernels interpret the $\mathbf{h}$ as a vector of 3D floats. Furthermore, rows of J and $J^\top$ are merged to 3D floats. This allows us to map multiple operations to float3 vector arithmetic.

5.2.2 Preconditioned Conjugate Gradient on the GPU

As well as implementing Jacobian multiplication on the GPU, we also implemented the PCG solver itself. In its loop, we evaluate the matrix-vector product $J^\top(J\mathbf{h})$, which we can do very efficiently as shown above, but also a number of other terms. A naive implementation would require 12 kernel calls in the inner loop, where the kernel switches dominate computation. So instead we use a variant of the fast PCG-solver described by Weber et al. [2013]. This solver reduces the number of kernel calls in the inner loop to three. However, since in our case the system matrix is $J^T J$, we end up with four kernel calls. We use block diagonal preconditioning. The inverses of the diagonal blocks of $J^\top J$ are precomputed in the initialization stage of PCG, using a similar kernel to the Jacobian multipliers.

5.2.3 Energy minimization: Coarse-to-fine outer loop

The previous sections describe the energy minimization for the model vertices at a single resolution. For both speed and accuracy, a multi-scale optimizer is used. In an outer loop, the solver iterates over the mesh hierarchy from coarse to fine. In its inner loop, it optimizes for the optimal deformation on the current resolution as

described in Sec. 5.2. At the end of the inner loop, a prolongation step transfers the solution on the current hierarchy level to the next finer one as described by Sumner et al. [2007], using the weights from Li et al. [2009]. The weights are precomputed at the time of creation of the initial template model. We found that it was necessary to apply prolongation not just to the model vertices V but also to the rotation parameters Φ. The latter is performed by estimating rotations at the new scale using the closed-form Kabsch algorithm to solve (7) before starting Gauss-Newton. In conjunction with this hierarchical solving strategy, a number of other continuation strategies are used to improve speed and/or convergence. Each run of the Gauss Newton solver is limited to $5-8$ iterations. On the finest hierarchy level, we apply an exponential average in order to reduce temporal flickering. Note that this only affects visualization, but not the optimization procedure.

The parameters which affect the location of energy minima are the energy weights λ_{point}, λ_{plane}, λ_{reg} and the robust kernel width τ, and their setting is discussed below. Typically $\lambda_{\text{point}} = 0.2, \lambda_{\text{plane}} = 0.8$ can be kept fixed, and the value of λ_{reg} is chosen to coarsely reflect the amount of deformation in a given sequence. These settings are for the finest scale of the hierarchy. To improve convergence at the coarser scales, λ_{reg} is increased by a factor of 20, and τ by 10, so that only gross outliers are rejected. Note that these parameters affect only the *rate* of convergence, not the location of the energy minimum, which is affected by rather fewer parameters (see next paragraph). This can certainly mean that with different settings, the model may or may not converge completely in one frame if the object has undergone fast motion, but it will typically converge in a number of frames, particularly if the object slows down (see Fig. 8)

5.3 Initialization: Correspondence finding

As mentioned above, initialization of V^t simply takes the value from the previous frame, and initialization of Φ is to the parameters of the identity rotation. The parameters U represent, for each model vertex, the image location of the closest point to the vertex, and given that $\mathbf{d}(\mathbf{u})$ may be quite non-smooth, a more careful initialization is warranted. This is achieved by a simple local search in a window around the previous frame's estimate transformed by the global transformation $\mathbf{R}, \mathbf{t}$:

$$\mathbf{u}_i^t = \underset{\mathbf{u}\in Q_i}{\operatorname{argmin}} \|\mathbf{R}(\mathbf{v}_i^{t-1}) + \mathbf{t} - \mathbf{d}^t(\mathbf{u})\|^2$$

where the window Q_i is $\pi^t(\mathbf{R}(\mathbf{v}_i^{t-1}) + \mathbf{t}) + [-24, 24] \times [-24, 24]$. For speed, this is computed in two stages: first checking only every third pixel in Q_i, then checking all pixels in a 5×5 window around the sub-sampled answer. The GPU implementation uses an efficient parallel block reduction in shared memory.

At this stage, we can also update the visibility flags $\{\eta_i\}_{i=1}^n$ to discard correspondences to data points that lie close to the boundaries to the background. We also threshold the orientation difference between data point (provided by finite differencing) and model vertex normals, and impose a maximal distance threshold between associated point pairs.

5.4 Detail Integration

The result of energy minimization is a mesh at the second-finest resolution which matches the data, but does not feature fine-scale details, such as folds or wrinkles, that may be present in the current depth data. Because such transient surface details cannot be built into the initial template model, we prolong the fitted result to the finest hierarchy level and add the missing detail by computing in the least-square-sense optimal per-vertex scalar displacements d_i along the vertex normal.

Since the mesh is already very close to the measured data, the residual detail displacements can be assumed to be small. Therefore, the following algorithm can be used that fulfills our speed and plausibility requirements. We assume a thin shell deformation model whose minimal energy deformation state is found by minimizing the stretching and bending energies expressed as the differences in the first and second fundamental forms [Botsch and Sorkine 2008]. The thin shell deformation energy is simplified by replacing the change of first and second fundamental forms by changes of first and second order partial derivatives of the 3D displacement function $\mathbf{r}$ on the surface. This deformation energy is minimized by variational calculus, and linearization yields the following Euler Lagrange equations [Botsch and Sorkine 2008]:

$$\begin{aligned} -\lambda_s \Delta\mathbf{r} + \lambda_b \Delta^2 \mathbf{r} &= 0 \\ \text{with } \Delta\mathbf{r} &= \text{div}\nabla\mathbf{r} = \mathbf{r}_{uu} + \mathbf{r}_{vv} \\ \Delta^2\mathbf{r} &= \mathbf{r}_{uuuu} + 2\mathbf{r}_{uuvv} + \mathbf{r}_{vvvv} \end{aligned} \tag{11}$$

Here, $\mathbf{r}_u$, $\mathbf{r}_{uu}$, and $\mathbf{r}_{uuuu}$ are the first, second, and fourth partial derivatives of $\mathbf{r}$ w.r.t the surface parameterization of the template mesh; v-directions are defined analogously. λ_s and λ_b define stretching and bending resistance, respectively. To obtain the target displacements for each vertex, we find the closest intersection point in the input data by raymarching in normal direction. The resulting intersection point is further refined using a simple bisection approach. We incorporate the resulting target displacements as soft-constraints into the optimization problem. In our case, $\mathbf{r}$ is the residual displacement field on the mesh, and can be found by minimizing Eq. 11 under the soft constraints using the fast GPU-based preconditioned conjugate gradient solver from Sec. 5.2.1. As initial guess for the iterative solve, we use the computed displacements for the previous frame to warm start the optimizer leading to fast convergence. Given the noise in the input data, we employ a temporal averaging scheme, similar to Li et al. [Li et al. 2009], based on exponential weighting to compute the final displacements. This nicely removes noise, while still being responsive to changes in transient surface detail.

6 Results

Now that we have described our system in detail, in this section we present a variety of results from live capture, ground truth experiments, and comparisons to existing work.

6.1 Live Non-rigid Capture

Our system is fully implemented on the GPU using CUDA. Results of live scene captures for our test scenes are shown in Figures 1 and 6 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 non-rigid reconstruction. Further, these sequences are tracked over long time periods comprising several minutes.

We captured a variety of diverse non-rigidly moving and deforming objects. The table in Fig. 6 shows the number of vertices in the different hierarchy levels, and indicates whether a tetrahedralized version of the mesh has been used to incorporate weak volume constraints. In the FACE sequence, we show how our system can generate compelling reconstructions of faces. Our results convey subtle expressions including detailed skin deformations and wrinkles. Our system also models large deformations captured during actions such as talking, frowning, and smiling, and demonstrates the benefits of modeling the fine facial deformations. This is in contrast to existing real-time methods based on parametric morphable models [Li et al. 2013b; Weise et al. 2011; Weise et al. 2009b], which often fail to convey facial details.

However, our system is also able to reconstruct many other types of scenes beyond faces. In the HAND and UPPER BODY sequence, we show two challenging sequences which exhibit large amounts of occlusions when the user either places the hand in front of his/her body or significantly bends the fingers. Despite these occlusions our system is able to track non-rigid motions, although extreme poses and rapid motions can cause errors. In the TEDDY and BALL sequence, we finally show how our method can generalize to non-human tracking and reconstruction.

6.2 Performance

We measured performance of our entire non-rigid tracking pipeline including run-time overhead on an Intel Core i7 3.4GHz CPU, 16GB of RAM, and a single NVIDIA GeForce GTX780. The average timing (see also Fig. 6) among all test scenes is 33.1ms (i.e., 30.2fps) with 4.6ms for preprocessing (computing derivatives of normals, depth, and color data) (14% of the overall pipeline), 2.93ms (8.9%) for rigid ICP pose estimation (on average 4 iterations), 21.3ms (64%) for the non-rigid fitting on the coarse and medium mesh level (2×5 Gauss-Newton iterations, each with 10 PCG iterations), and 3.36ms (10%) for fine detail integration (10 iteration steps). On top of this, the timings of our stereo matcher are 17ms, or alternatively 26ms with variational refinement enabled. Note that in our current implementation we run the depth estimation on a separate second GPU, which allows for a complete runtime of 33ms (30fps) for our full pipeline by introducing a delay of one frame.

6.3 Applications

This type of non-rigid capture enables many compelling applications as shown in Fig. 7. In the RE-TARGET sequence we demonstrate a real-time, motion re-targeting scenario, where the user controls two avatars by transferring detailed non-rigid motions and expressions. Real-time avatar re-targeting can lead to new scenarios for gaming or video conferencing, where more detailed shape and motion can be reconstructed resulting in more expressive user experiences. We use a simple re-targeting method by manually specifying a sparse set of per-vertex correspondences between our reconstructed template and the new target mesh. We use these correspondences to drive the animation using mesh skinning. Although this simple approach produces compelling results, more advanced re-targeting techniques such as [Sumner and Popović 2004] could easily be applied. Our live system can also be used for performance and motion capture in home and semi-professional movie and animation production. In the RE-TEXTURE sequence (Fig. 7) we demonstrate how the estimation of detailed deformations enables convincing augmented reality applications such as pasting digital content onto non-rigidly deforming physical objects. Application scenarios include virtual clothing and makeup. Our deformation regularization prevents geometric drift which keeps texturing locally stable.

6.4 Evaluation

Fitting and Regularization Error Fig. 8 shows the fitting error with respect to the iteration count over several frames of the examples FACE, HAND, and BALL. The spikes in the graph coincide with the arrival of new input frames. To examine convergence, we perform 8 Gauss-Newton iteration steps per frame on a single hierarchy level, with 20 PCG iterations in the inner loop. It can be seen that the registration converges quickly and in most of the cases, convergence is reached in less than 5 Gauss-Newton iterations.

The energy of the ARAP regularizer for example scenes is provided in Fig. 9. This allows us to localize the regions undergoing locally non-rigid deformations in real-time. Interesting areas of future work include leveraging the ARAP residuals to either: 1) refine the template model in these regions (akin to the method of Li et al. [2009]) in order to adapt the deformation model to the seen deformations, or 2) use this residual error to localize user interactions with objects. For example, in the ball sequence we clearly identify where the user is touching and pressing the ball. This leads to the possibility of making such physical objects interactive and enables new types of user experiences, e.g., for gaming.

6.5 Comparisons

In Fig. 10, we compare results obtained with our system with ground truth data. To this end, we used data from [Valgaerts et al. 2012]. We generate synthetic depth maps by rendering the mesh from a single view. Our method is applied using eight Gauss-Newton iteration steps with 20 PCG iterations in the inner loop. The figure compares renderings of the original mesh and our reconstruction, as well as plots of the deviation for three frames of the animation, where red corresponds to a fitting error of 3mm. This shows that our algorithm can match the facial expression and fine scale details exhibited in this sequence. The method of Valgaerts et al. is an offline technique, with has a runtime of about 9 minutes per frame on the CPU. Our results show qualitatively similar results but with a system that is about 4 orders of magnitude faster, and with the ability to track a variety of general objects beyond faces.

Fig. 11 shows a comparison with the results of Li et al. [2009]. Both sequences were generated from the same input data. In both cases the reconstructed mesh has 70k vertices. Whereas Li's method requires more than one minute per frame on the CPU[1], our novel GPU pipeline runs at almost 30Hz and is thus more than three orders of magnitude faster.

6.6 Other Reconstruction Scenarios

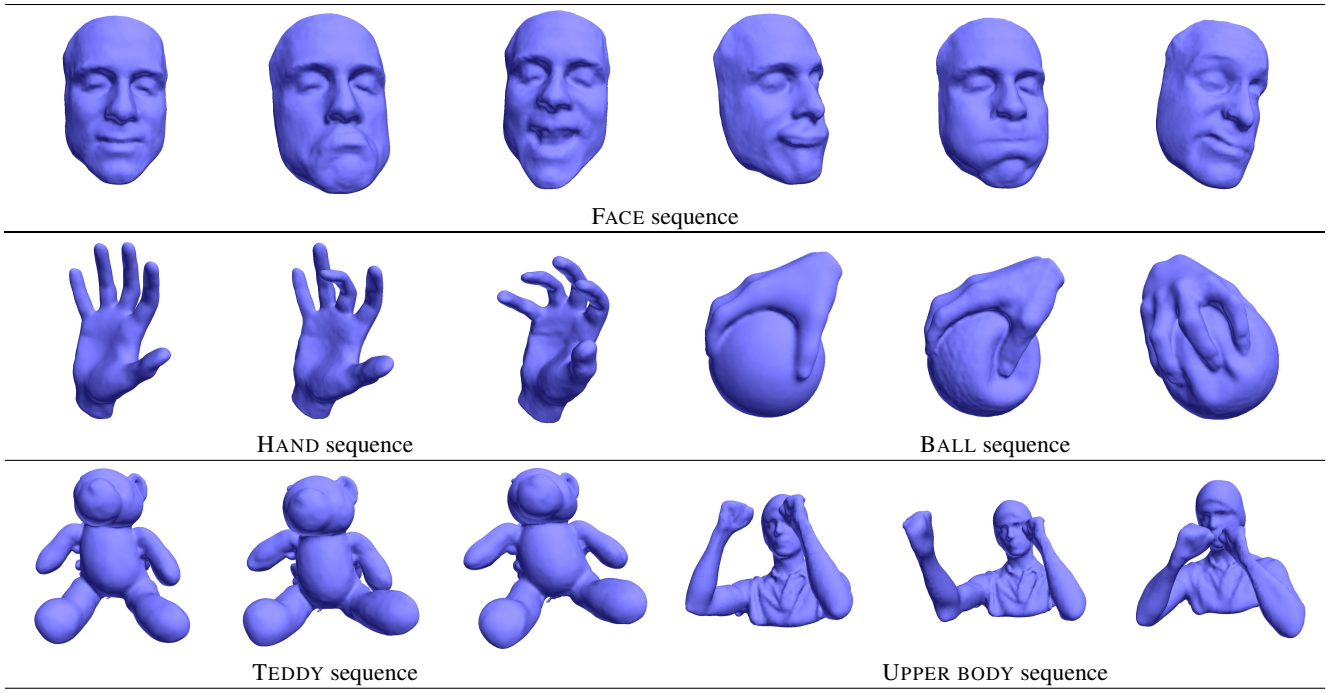
Our technique can also be applied to multi-view setups. In Fig. 12, we show reconstructions obtained with our method from the data sets SAMBA and SQUAT from [Vlasic et al. 2008]. The figure also shows a reconstruction of the GHOST data set [Vlasic et al. 2009], which demonstrates our ability to deal with motions that cannot be parameterized by a skeleton. For multiple views, our reconstruction pipeline has to perform more work in the preprocessing stage of the pipeline, processing the data from each camera separately. We assign each vertex to the best suited camera, based on visibility and orientation, after which we can perform surface fitting within our presented fitting pipeline, which is independent of the number of cameras.

Finally, we also illustrate results of our method using a regular Kinect camera (see also Fig. 12). Note that while our method produces far higher quality results with our stereo setup (for close ranges), the Kinect results could still be used in interactive applications, where quality is perhaps a secondary requirement, or where larger distance reconstruction is desired. This provides the exciting possibility of building both new multi-camera systems for performance capture using our method, as well as the possibility to use consumer depth cameras for certain interactive scenarios.

7 Limitations

Even though we demonstrated one of the first methods for real-time non-rigid reconstruction from a single view, this problem is still ill-posed. In particular, the restriction to a single view leads to large occlusions resulting in missed correspondences. At each time step, typically less than half of the tracked object is visible [Li et al. 2009],

[1]Timings by Li et al. [2009]; expected to run faster on current CPUs.



	Prepro-cess	Rigid Fit	Non-Rig. Fit	Lin. Fit	Misc	Sum	#vert. coarse	#vert. medium	#vert. fine	tetr. mesh	#frames
FACE	4.65	3.22	20,9	2.46	1.16	32.4	1.2k	2.5k	20k	no	1490
HAND	4.60	2.62	20,6	2.70	0.79	31.3	0.6k	2.5k	20k	yes	587
BALL	4.66	3.12	19,5	4.20	1.16	32.6	1.2k	2.5k	40k	no	813
TEDDY	4.58	2.84	19,0	4.69	0.80	31.9	1.0k	2.5k	40k	no	599
BODY	4.64	2.85	26.3	2.73	0.80	37.3	1.1k	2.5k	20k	yes	1500
Avg.	4.62	2.93	21.3	3.36	0.94	33.1					

Figure 6: *A number of different deformable objects and the corresponding timings during a live session.*

and the behavior of unobserved regions has to be inferred through regularization constraints. Offline methods tackle this problem by using sophisticated correspondence finding mechanisms coupled with a slow relaxation of the model rigidity during the optimization process in order to avoid local minima in the energy landscape. Given the tight real-time constraint (33ms/frame) of our approach, we rely on temporal coherence of the RGB-D input stream making the processing at 30Hz a necessity. If the frame rate is too low, or frame-to-frame motion is too large, our method might lose tracking. Similar problems may be caused by occluded regions, sparse/noisy input data, or a violation of the topological prior. Offline methods, e.g., [Li et al. 2008; Li et al. 2009; Beeler et al. 2011], fail in similar cases as ours; however, they are more stable due to a larger time budget that allows for more elaborate strategies, such as global optimization, re-meshing of the deformation template, or anchor frames. In the following, we address specific failure cases in more detail and give ideas on how to improve in these situations.

Topological Changes Most template-based methods, no matter if they are online or offline, share our inability to robustly and efficiently handle topological changes. Only a few methods handle such situations [Wand et al. 2009], but at computation times that are far from real-time performance. A semantically incorrect prior counteracts the actual deformation (e.g., opening the mouth) which inevitably leads to surface sliding. While one could imagine the template to be modified at runtime, it would cause severe optimization instabilities and add significant computational complexity, which is (currently) infeasible in real-time. Similar problems occur if object parts not represented in the template are revealed during surface tracking (e.g., teeth). In scenarios where the topological assumption is satisfied (e.g., hand, boxer, teddy, ball), our method allows for robust tracking without surface sliding (see Fig. 7).

Sparse Input and Occlusions Sparse input data and occlusions are inherent problems of a single-view camera setup. This causes missing correspondences, and thus increases the importance of regularization constraints. In these regions, there is no guarantee that deformations conform to the real-world, since we do not consider material or statistical shape priors. Methods focusing on a single domain, such as faces [Weise et al. 2011; Li et al. 2013b], are more robust towards occlusions since they have less degrees of freedom; however, they are less general and require a significant amount of training data. If our method misses large deformations due to occlusions, the temporal coherence assumption is violated once these regions become visible. This might lead to tracking instabilities or slow convergence. Given the tight real-time constraint, we can only afford searching correspondences on each level of the hierarchical solver independently. In theory, we would always like to consider alignment at the highest resolution, even when processing lower hierarchy levels; however, this comes at additional costs. Another problem of our vertex-to-input correspondence search is the possibility of undersampling the input depth data, which might lead to

Figure 7: *Applications for live non-rigid capture. Left: detailed facial expressions are re-targeted. Right: spatio-temporal coherent re-texturing of template meshes for the same input sequence.*

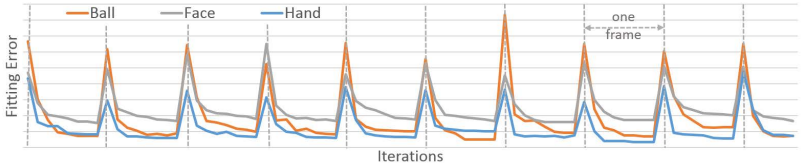


Figure 8: *Convergence of the non-rigid deformation. The spikes correspond to new frames. Note convergence "through" the new-frame spike on the last frame of "Face".*

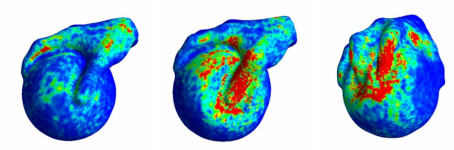


Figure 9: *Energy of the ARAP regularizer at each vertex for the* BALL *example.*

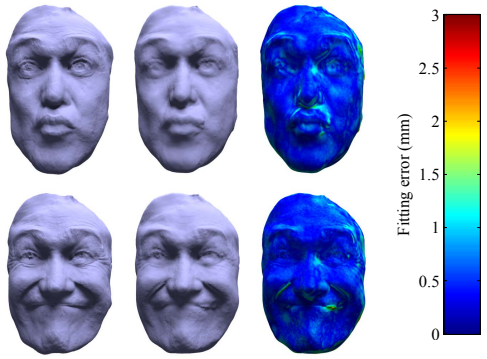


Figure 10: *Ground truth comparison. Left: detailed input data. Middle: reconstruction from synthetic depth maps. Right: fitting error.*

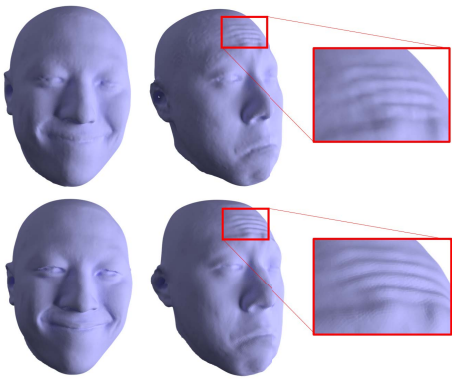


Figure 11: *Comparison of our real-time reconstruction (top row) with offline reconstructions [Li et al. 2009] while only using depth data (i.e., no color term).*

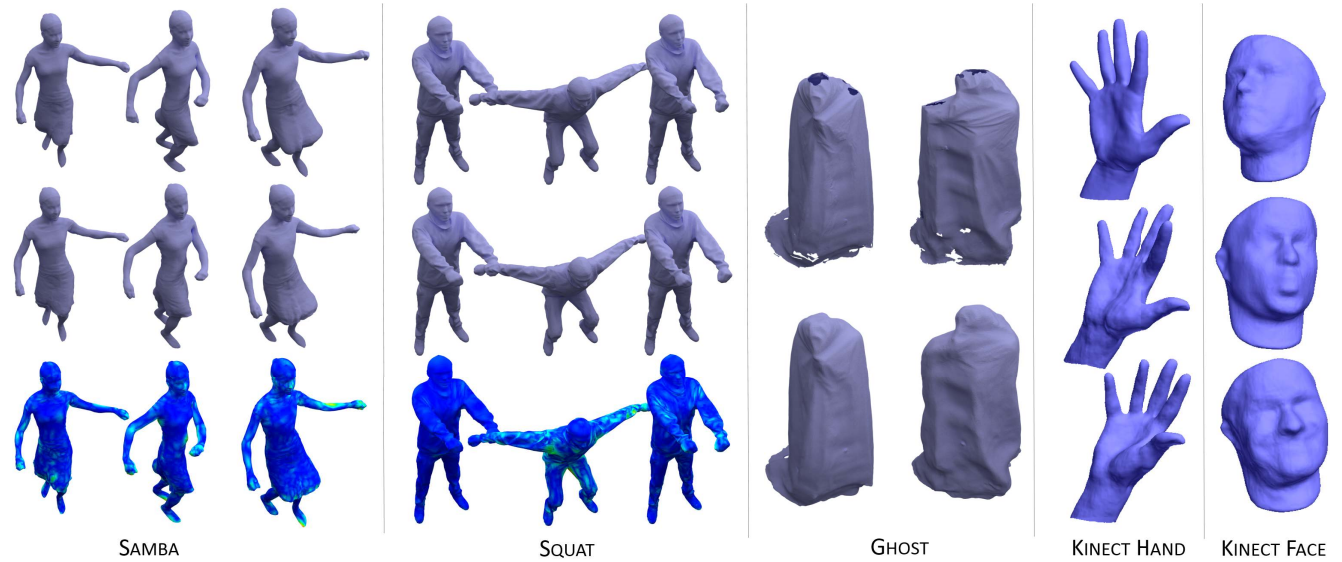


Figure 12: SAMBA, SQUAT*: Reconstructions of synthetic multi-view input (8 depth cameras). Input (top row), our reconstruction (middle row), error (bottom row) with red=30mm. The right column of* SQUAT *shows the result at the end of the animation after four squats.* GHOST*: also reconstructed from synthetic multi-view input (8 cameras). Top row: input, bottom row: reconstruction.* KINECT HAND & FACE*: Three frames of a sequences reconstruction using a single Kinect sensor.*

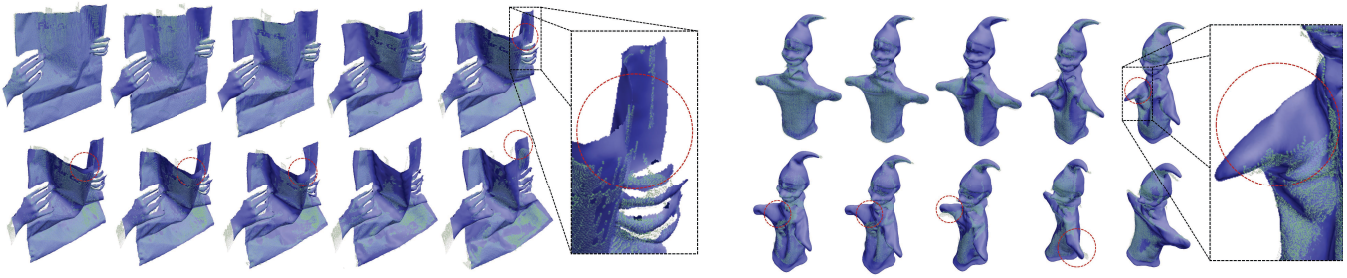


Figure 13: *Limitations. Left: tracking instability due to sparse input leading to a slight misalignment of the paper. Right: tracking of the puppet's arm fails due to large and fast motion. However, our method recovers at the end of the sequence. Note, that our approach is less stable in these sequences, compared to the ones shown in Fig. 6, since no color data is used.*

misalignments; see Fig. 13 (left). Ideally, one would prefer explaining all input data instead. Again, this is currently infeasible due to computational limitations.

Fast and Large Deformation A typical strategy to deal with fast and large deformations, is to incrementally relax the model rigidity in order to avoid local minima in the energy landscape. Therefore, offline approaches spend significant effort on slowly relaxing regularization constraints using many iterations. In our real-time scenario, we can only handle a limited amount of frame-to-frame deformation. In order to process reasonably fast motion, we enforce high temporal coherence leveraging our 30Hz input RGB-D stream. If the temporal coherence assumption is violated, tracking might fail; e.g., see Fig. 13 (right). However, note that our method can recover in most cases. In the future, we also expect RGB-D cameras to have higher frame rates, thus making faster motion possible.

8 Conclusions

In this paper, we have introduced what we believe to be the first 'general purpose' non-rigid reconstruction system that provides *real-time* performance, several orders of magnitude faster than general methods, without using a specific 'baked in' kinematic or shape model. Instead our system allows users to acquire a template online, and use this for *live* non-rigid reconstructions. Our system is simple to use and self-contained, with a single lightweight stereo camera setup, moving closer to commodity or consumer use. Additionally, in terms of reconstructed geometric detail, our method also narrows the gap between offline and online methods. Our system attempts to hit a 'sweet spot' between methods that can reconstruct general scenes, and techniques that rely on a stronger shape prior (e.g., a blendshape face model or body or hand skeleton), which are beginning to demonstrate real-time performance. As shown in the results section, the simplicity of our method and its real-time performance does not significantly compromise the overall reconstruction quality. Our work brings us a step closer to high-quality real-time performance capture systems for consumer scenarios including gaming, home and semi-professional movie production, and human-computer interaction.

Acknowledgements

We would like to thank Angela Dai for the video voice over, Frank Bauer for help with renderings, and Hao Li for comparison data. This research was mainly conducted at Microsoft Research Cambridge, and co-funded by the German Research Foundation (DFG), grant GRK-1773 Heterogeneous Image Systems, and the ERC Starting Grant 335545 CapReal.

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 TOG (Proc. SIGGRAPH) 30*, 4, 75.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3D faces. In *Proc. SIGGRAPH*, 187–194.

BLEYER, M., RHEMANN, C., AND ROTHER, C. 2011. Patchmatch stereo: Stereo matching with slanted support windows. In *Proc. BMVC*, vol. 11, 1–11.

BOJSEN-HANSEN, M., LI, H., AND WOJTAN, C. 2012. Tracking surfaces with evolving topology. *ACM Trans. Graph. 31*, 4, 53.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Trans. Vis. Comp. Graph 14*, 1, 213–230.

BRADLEY, D., POPA, T., SHEFFER, A., HEIDRICH, W., AND BOUBEKEUR, T. 2008. Markerless garment capture. *ACM TOG (Proc. SIGGRAPH) 27*, 3, 99.

BROWN, B. J., AND RUSINKIEWICZ, S. 2007. Global non-rigid alignment of 3D scans. *ACM TOG 26*, 3, 21–30.

CAGNIART, C., BOYER, E., AND ILIC, S. 2010. Free-form mesh tracking: a patch-based approach. In *Proc. CVPR*.

CAO, C., WENG, Y., LIN, S., AND ZHOU, K. 2013. 3D shape regression for real-time facial animation. *ACM TOG 32*, 4, 41.

CHEN, J., IZADI, S., AND FITZGIBBON, A. 2012. Kinêtre: animating the world with the human body. In *Proc. UIST*, 435–444.

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*, 99–106.

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*, 1746–1753.

GARRIDO, P., VALGAERT, L., WU, C., AND THEOBALT, C. 2013. Reconstructing detailed dynamic face geometry from monocular video. *ACM TOG (Proc. SIGGRAPH Asia) 32*, 6, 158.

HELTEN, T., BAAK, A., BHARAJ, G., MULLER, M., SEIDEL, H.-P., AND THEOBALT, C. 2013. Personalization and evaluation of a real-time depth-based full body tracker. In *Proc. 3DV*, 279–286.

HERNÁNDEZ, C., VOGIATZIS, G., BROSTOW, G. J., STENGER, B., AND CIPOLLA, R. 2007. Non-rigid photometric stereo with colored lights. In *Proc. ICCV*, 1–8.

IZADI, S., KIM, D., HILLIGES, O., MOLYNEAUX, D., NEWCOMBE, R., KOHLI, P., SHOTTON, J., HODGES, S., FREEMAN, D., DAVISON, A., AND FITZGIBBON, A. 2011. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proc. UIST*, 559–568.

KOLB, A., BARTH, E., KOCH, R., AND LARSEN, R. 2009. Time-of-flight sensors in computer graphics. In *Proc. Eurographics State-of-the-art Reports*, 119–134.

LI, H., SUMNER, R. W., AND PAULY, M. 2008. Global correspondence optimization for non-rigid registration of depth scans. In *Proc. SGP*, Eurographics Association, 1421–1430.

LI, H., ADAMS, B., GUIBAS, L. J., AND PAULY, M. 2009. Robust single-view geometry and motion reconstruction. *ACM TOG 28*, 5, 175.

LI, H., VOUGA, E., GUDYM, A., LUO, L., BARRON, J. T., AND GUSEV, G. 2013. 3D self-portraits. *ACM TOG 32*, 6, 187.

LI, H., YU, J., YE, Y., AND BREGLER, C. 2013. Realtime facial animation with on-the-fly correctives. *ACM Transactions on Graphics 32*, 4 (July).

LIAO, M., ZHANG, Q., WANG, H., YANG, R., AND GONG, M. 2009. Modeling deformable objects from a single depth camera. In *Proc. ICCV*, 167–174.

MITRA, N. J., FLÖRY, S., OVSJANIKOV, M., GELFAND, N., GUIBAS, L. J., AND POTTMANN, H. 2007. Dynamic geometry registration. In *Proc. SGP*, 173–182.

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.

NIESSNER, M., ZOLLHÖFER, M., IZADI, S., AND STAMMINGER, M. 2013. Real-time 3D reconstruction at scale using voxel hashing. *ACM TOG 32*, 6, 169.

OIKONOMIDIS, I., KYRIAZIS, N., AND ARGYROS, A. A. 2011. Efficient model-based 3D tracking of hand articulations using Kinect. In *Proc. BMVC*, 1–11.

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*, 83–88.

SORKINE, O., AND ALEXA, M. 2007. As-rigid-as-possible surface modeling. In *Proc. SGP*, 109–116.

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

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. In *ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, SIGGRAPH '04, 399–405.

SUMNER, R. W., SCHMID, J., AND PAULY, M. 2007. Embedded deformation for shape manipulation. *ACM TOG 26*, 3, 80.

TAYLOR, J., SHOTTON, J., SHARP, T., AND FITZGIBBON, A. 2012. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *Proc. CVPR*, 103–110.

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 31*, 2, 12.

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.

TONG, J., ZHOU, J., LIU, L., PAN, Z., AND YAN, H. 2012. Scanning 3D full human bodies using Kinects. *TVCG 18*, 4, 643–650.

VALGAERTS, L., WU, C., BRUHN, A., SEIDEL, H.-P., AND THEOBALT, C. 2012. Lightweight binocular facial performance capture under uncontrolled lighting. *ACM TOG (Proc. SIGGRAPH Asia) 31*, 6 (November), 187.

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 28*, 15.

WASCHBÜSCH, M., WÜRMLIN, S., COTTING, D., SADLO, F., AND GROSS, M. 2005. Scalable 3D video of dynamic scenes. In *Proc. Pacific Graphics*, 629–638.

WEBER, D., BENDER, J., SCHNOES, M., STORK, A., AND FELLNER, D. 2013. Efficient gpu data structures and methods to solve sparse linear systems in dynamics applications. *Computer Graphics Forum 32*, 1, 16–26.

WEI, X., ZHANG, P., AND CHAI, J. 2012. Accurate realtime full-body motion capture using a single depth camera. *ACM TOG 31*, 6 (Nov.), 188.

WEISE, T., WISMER, T., LEIBE, B., , AND GOOL, L. V. 2009. In-hand scanning with online loop closure. In *IEEE International Workshop on 3-D Digital Imaging and Modeling*.

WEISE, T., LI, H., GOOL, L. V., AND PAULY, M. 2009. Face/off: Live facial puppetry. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer animation (Proc. SCA'09)*, Eurographics Association, ETH Zurich.

WEISE, T., BOUAZIZ, S., LI, H., AND PAULY, M. 2011. Realtime performance-based facial animation. *ACM TOG 30*, 4, 77.

WEISS, A., HIRSHBERG, D., AND BLACK, M. J. 2011. Home 3D body scans from noisy image and range data. In *Proc. ICCV*, 1951–1958.

WHITE, B. S., MCKEE, S. A., DE SUPINSKI, B. R., MILLER, B., QUINLAN, D., AND SCHULZ, M. 2005. Improving the computational intensity of unstructured mesh applications. In *Proc. ACM Intl. Conf. on Supercomputing*, 341–350.

WILAMOWSKI, B. M., AND YU, H. 2010. Improved computation for levenberg-marquardt training. *IEEE Trans. Neural Networks 21*, 6, 930–937.

WU, C., STOLL, C., VALGAERTS, L., AND THEOBALT, C. 2013. On-set performance capture of multiple actors with a stereo camera. *ACM TOG 32*, 6, 161.

YE, G., LIU, Y., HASLER, N., JI, X., DAI, Q., AND THEOBALT, C. 2012. Performance capture of interacting characters with handheld kinects. In *Proc. ECCV*. Springer, 828–841.

ZENG, M., ZHENG, J., CHENG, X., AND LIU, X. 2013. Template-less quasi-rigid shape modeling with implicit loop-closure. In *Proc. CVPR*, 145–152.