Real-time Geometry, Albedo and Motion Reconstruction Using a Single RGBD Camera

KAIWEN GUO and FENG XU, Tsinghua University
TAO YU, Beihang University and Tsinghua University
XIAOYANG LIU, QIONGHAI DAI, and YEBIN LIU, Tsinghua University

This paper proposes a real-time method that uses a single-view RGBD input to simultaneously reconstruct a casual scene with a detailed geometry model, surface albedo, per-frame non-rigid motion and per-frame low-frequency lighting, without requiring any template or motion priors. The key observation is that accurate scene motion can be used to integrate temporal information to recover the precise appearance, whereas the intrinsic appearance can help to establish true correspondence in the temporal domain to recover motion. Based on this observation, we first propose a shading-based scheme to leverage appearance information for motion estimation. Then, using the reconstructed motion, a volumetric albedo fusing scheme is proposed to complete and refine the intrinsic appearance of the scene by incorporating information from multiple frames. Since the two schemes are iteratively applied during recording, the reconstructed appearance and motion become increasingly more accurate. In addition to the reconstruction results, our experiments also show that additional applications can be achieved, such as relighting, albedo editing and free-viewpoint rendering of a dynamic scene, since geometry, appearance and motion are all reconstructed by our technique.

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

Additional Key Words and Phrases: single-view, surface reconstruction, real-time, non-rigid, albedo

ACM Reference format:
Kaiwen Guo, Feng Xu, Tao Yu, Xiaoyang Liu, Qionghai Dai, and Yebin Liu. 2017. Real-time Geometry, Albedo and Motion Reconstruction Using a Single RGBD Camera. *ACM Trans. Graph.* XX, X, Article XX (March 2017), 13 pages.
DOI: XXXXXXX.YYYYYYY

Fig. 1. Our system can capture fast and natural motions, geometry, and surface albedo and simultaneously render them in new lighting environments in real time.

1 INTRODUCTION

Dynamic scene reconstruction involves capturing and reproducing various aspects of the real visual world, including static geometry, detailed motion, and intrinsic or observed appearance. Simultaneously reconstructing all of these aspects or even part of them enables important applications in computer vision and graphics. For example, reconstructed geometry and surface motion as well as observed appearance can be used for free-viewpoint video. Reconstructed kinematic motion can be transferred to new objects or used to generate new photo-realistic animations. The intrinsic appearance of a dynamic scene/object can be used in applications such as appearance editing and relighting. The real-time reconstruction of geometry, motion and appearance enables more realistic rendering for virtual reality scenarios, for example, Holoportation [7].

Although considerable efforts have been devoted to dynamic scene reconstruction, the problem remains challenging because of the extraordinarily large solution space, necessitating a carefully designed capture environment [5, 34], high-quality lighting equipment [3, 9] and many video cameras [15]. Several recent works have successfully eliminated various constraints on acquisition by using convenient capture equipment, such as a single Kinect [32] or binocular camera [37]. However, they require many scene priors to constrain the problem space, such as pre-scanned model templates [32, 37], finely embedded skeletons [37] and pre-captured lighting environments [37].

The work on DynamicFusion [23] represents a step forward in dynamic scene reconstruction. This work uses a single depth sensor to fuse scene geometry from multiple frames during non-rigid registration processing, thereby gradually obtaining a static model with fine geometry details and a motion sequence that matches the non-rigid motion in the scene. However, it has two major drawbacks. First, it does not recover the appearance and lighting information of a scene; thus, it does not yield a complete reconstruction of the visual information of the scene. Second, it uses only geometric information to estimate inter-frame motions based on the slow motion assumption,

This work was supported by the National key foundation for exploring scientific instrument No. 2013YQ140517 and NSFC (No. 61671268, 61522111 and 61531014).
Authors' addresses: K. Guo, X. Liu; email: {gkw11, liu-xy14}@mails.tsinghua.edu.cn; Q. Dai, Y. Liu (corresponding author); emails: {qhdai, liuyebin}@tsinghua.edu.cn; Department of Automation and TNList, Tsinghua University, Beijing, 100084, P.R. China; F. Xu (corresponding author); email: feng-xu@tsinghua.edu.cn; School of Software and TNList, Tsinghua University, Beijing, 100084, P.R. China; T. Yu; email: ytrock@buaa.edu.cn; School of Instrumental Science and Opto-electronics Engineering, Beihang University, Beijing, 100191, P.R. China.

DOI: XXXXXXX.YYYYYYY

which restricts its application to only slow and controlled motions in a scene. Recently, a concurrent work VolumeDeform [14] has been proposed that uses image features to improve motion estimation. However, even with sparse features, the applicability of this technique is still restricted to slow and controlled motions, and it also does not reconstruct appearance or lighting information.

In this paper, we propose a novel method in which the surface geometry, motion and intrinsic appearance (albedo) of a dynamic scene are jointly estimated. Our system utilizes both geometry and appearance information to estimate inter-frame motions; thus, it is able to handle casual motions. Our system requires only a single RGBD sensor; hence, it is very convenient to set up. Moreover, our technique requires no prior knowledge of the scene; consequently, it has no pre-computation step and is fully automatic. Finally, our system is performed in real time, and thus, it is applicable for online applications.

Similar to [41] in handling static scene reconstruction, our key observation is that, on the one hand, to achieve appearance reconstruction for a dynamic scene from a single-view input, information from multiple frames of a sequence needs to be integrated to achieve a complete reconstruction that is robust to noise. However, accurate integration strongly depends on accurate temporal correspondence among frames, and the geometry-based ICP method is far from achieving accurate correspondence. On the other hand, intrinsic appearance is an important cue for establishing correspondence because it does not change over time. Essentially, intrinsic appearance recovery and correspondence estimation are highly correlated in the context of dynamic scene reconstruction, and our key concept is to leverage one to jointly achieve the other in the tracking procedure.

We thus propose a unified optimization procedure to introduce albedo information (intrinsic appearance) into motion reconstruction, thereby enabling more accurate estimation of the correspondence between the reconstructed geometry and the input depth/color. Given the reconstructed correspondence, a gradually refined surface albedo is obtained by integrating information from multiple frames via the proposed volumetric albedo optimization procedure. With the iteratively optimized motion and appearance, the system achieves full visual reconstruction of a dynamic scene and has the ability to handle challenging motions, such as the stretching of cloth, object interactions, fast boxing and other casual motions.

In this paper, we present a novel method that can simultaneously fuse object geometry and surface albedo for a non-rigid scene in real time. The specific contributions of our technique include the following:

- Accurate and robust surface registration for non-rigid surface reconstruction in real time. By exploiting dense shading information, our system establishes temporally consistent geometric and photometric correspondences for casual motion reconstruction and consequently outperforms the previous methods proposed in [14, 23].
- Temporally coherent albedo estimation and fusion in real time. Our system decomposes the photometric information for each frame into albedo and low-frequency environmental lighting and then fuses multiple frames to refine the surface albedo based on temporally coherent correspondences, thereby enabling applications such as relighting and appearance editing.

2 RELATED WORK

The reconstruction of dynamic 3D scenes in the real world is a widely investigated topic in computer vision and graphics. With multi-view inputs and controlled environmental lighting, the dynamic geometry of a scene can be reconstructed via the shape-from-silhouette, multi-view stereo or photometric stereo technique [3, 20, 29, 34]. To better constrain the solution space, some methods utilize shape templates for captured objects. Thus, dynamic scene reconstruction becomes a shape-tracking problem [1, 2, 5]. Furthermore, because the motion of a human character always follows a skeleton structure, a pre-defined skeleton can be embedded into a surface template to realize character motion driven by joint rotations [21, 33, 44]. Note, however, that all these techniques require either a controlled environment or careful initialization, which significantly limits their applications.

In addition to accurate depth acquisition [25, 40], performance capture can also be achieved using consumer depth sensors. Ye and Yang [45] used a Gaussian mixture model in which an articulated deformation model was embedded. Wu et al. [37] further utilized pre-captured surface reflectance and environmental lighting to achieve detailed surface reconstruction. However, the purpose of these techniques is to reconstruct characters with skeleton structures, such as humans. It is difficult to extend these techniques to general objects.

In the literature, surface deformation techniques have been proposed in which the skeleton restriction is removed to achieve the reconstruction of general objects. Li et al. [17] and Zollhöfer et al. [48] reconstructed non-rigid motions using ICP-defined correspondence and local rigid motion priors. Guo et al. [10, 11] further used an L0-based motion prior to obtain improved results for articulated objects. Zhang et al. [46] trained a parametric pose model and used it in shape estimation for humans. Although these techniques achieve accurate dynamic reconstruction for a large variety of motions, they still require an initial geometry prior.

To further eliminate the dependency on geometry priors, some techniques attempt to directly reconstruct water-tight surfaces from a dynamic scene. Liao et al. [19] focused on handling continuous and predictable motions by stitching together partial surfaces obtained from multiple frames. Li et al. [18] achieved the accurate reconstructions of loose cloths but permitted only smooth pose changes among multi-view scans. However, non-rigid registration techniques, which generally assume multi-view scans [28] or involve additional motion priors [35], are also applicable to this task. Recently, based on single-view depth inputs, fine 3D models have been reconstructed without any motion priors by gradually fusing multi-frame geometries whose relative motions have been effectively removed [8, 23]. Using these techniques, it is possible to eliminate the static modeling step prior to non-rigid motion reconstruction. The concurrent work by [14] has added SIFT features to the ICP registration framework, thereby improving the accuracy of motion reconstruction. Note that neither of these recent techniques [14, 23] using a single depth sensors seeks correct correspondence to ensure high-quality appearance reconstruction. By contrast, in this paper, we demonstrate a technique for more accurate non-rigid motion reconstruction in real

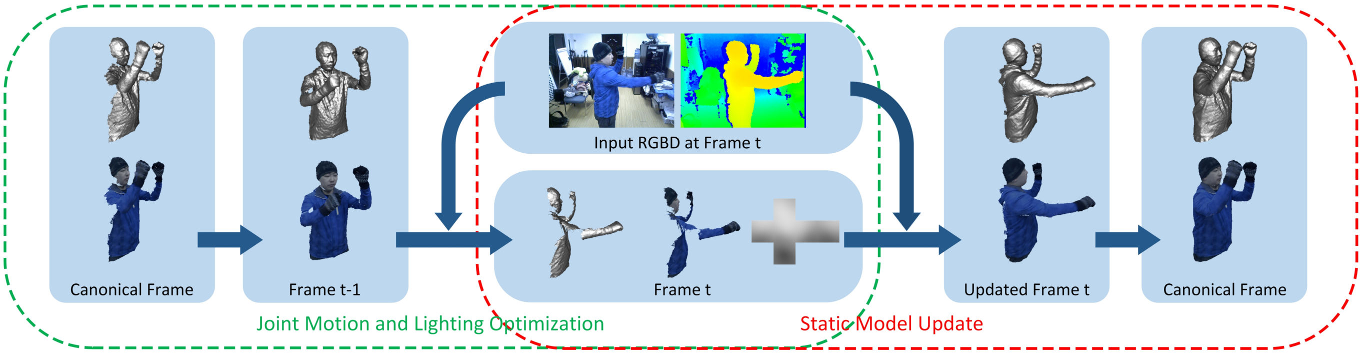


Fig. 2. Overview of our proposed pipeline. The green box represents the optimization of the motion field and the environmental lighting, and the red box represents the updating of the geometry and albedo of the canonical model.

time based on jointly solving for and leveraging the surface albedo, and we show that this accurate motion reconstruction technique enables competent texturing and relighting applications.

In addition to surface geometry and motion reconstruction, a rich body of work on appearance reconstruction also exists. The majority of these works focus on static objects. A comprehensive survey can be found in [6]. Moreover, some recent works have achieved accurate reconstruction with simple setups [41–43]. For dynamic scenes, Theobalt et al. [31] estimated a parametric BRDF model for a dynamic human body under calibrated lights. Li et al. [16] reconstructed motion, appearance and lighting using a multi-view camera array and an initial shape template. Imber et al. [13] also utilized multi-view input to reconstruct intrinsic textures without a smooth prior in the temporal domain. Taheri et al. [30] jointly optimized albedo and face pose by utilizing information from multiple video frames. However, all these appearance-related reconstruction techniques critically require pre-calibrated camera arrays, controlled lighting conditions or temporal information.

3 OVERVIEW

Our system runs in a frame-by-frame manner to process an input sequence. Fig. 2 illustrates the processing of one frame given previous frames. In our system, we represent the static model containing both the geometry and surface albedo of the captured object in a coordinate system defined in a frame called the canonical frame. The first step of our system is the *joint motion and lighting optimization* (Sec. 5.1), in which the non-rigid surface motion from the static model to the current captured frame is estimated along with the low-frequency environmental lighting of this frame. This is achieved by first warping the static model using the reconstructed motion from the previous frame, followed by solving a unified optimization problem to fit the current depth and color. Because both the lighting and motion are correlated with the observed appearance in the image, a unified framework facilitates convergence to the global optimum through jointly solving for lighting and motion, and the results are consequently more consistent with the observed depth and color. Compared with DynamicFusion, because shading information is considered, temporal coherence (the temporal correspondence between each pair of adjacent frames) is well preserved.

The second step is the *static model update* (Sec. 5.2). We warp the static model using the newly solved motion and update the warped static model with the current depth and color. As the low-frequency lighting is solved for, the albedo is updated by solving a volumetric optimization problem to fit the current observed image and the original albedo as accumulated from previous frames. This optimization makes the static model more complete because new parts of the object can be observed as it moves. Furthermore, because temporal consistency is preserved in the first step, this optimization correctly fuses multi-frame observations, thereby making the result more robust against noise and reconstruction errors. Finally, the updated static model is warped back to the canonical frame to be used in the processing of the next frame.

4 PRELIMINARIES

In this section, we present the symbols and other notation used in this paper. To reconstruct a dynamic scene, a consumer RGBD sensor, such as a Kinect or an Xtion sensor, is used to simultaneously record a depth sequence $\{\mathcal{D}^t\}$ and a color sequence $\{\mathcal{C}^t\}$ (t indicates the frame label). The two sequences are synchronized, and the sensor is pre-calibrated; thus, the per-pixel correspondence between $\mathcal{D}^t$ and $\mathcal{C}^t$ is obtained by default. The output of our system includes a fused static model $\mathcal{S}$ of the captured object, the per-frame low-frequency environmental lighting $\mathcal{L}^t$ and a per-frame motion field $\mathcal{W}^t$ that represents the non-rigid deformation between $\mathcal{S}$ and the real object in each frame t.

We define $\mathcal{S} = \{\mathcal{V}, \mathcal{A}\}$, where $\mathcal{V}$ and $\mathcal{A}$ represent the geometry and appearance, respectively, of the captured object. The geometry $\mathcal{V}$ is not represented as a set of surface points in 3D space but rather as a truncated signed distance function (TSDF) [4] in a canonical frame. We use the coordinate frame of the depth camera as the canonical frame. Specifically, the canonical space is voxelized and each voxel $\mathbf{x}$ is assigned a value indicating its signed distance to the closest surface to it in the scene. Signed distances with values larger than a threshold τ (set to 0.01m) are truncated to τ. Mathematically, $\mathcal{V} = \{[d(\mathbf{x}) \in \mathbb{R}, \omega(\mathbf{x}) \in \mathbb{R}^+]^\mathrm{T}\}$. Here, $d(\mathbf{x})$ encodes the signed distance value for voxel $\mathbf{x}$, whereas $\omega(\mathbf{x})$ indicates the confidence of that value; further detailed are provided in Sec. 5.2.1. Following [32, 38, 39], we assume the object surfaces to be predominantly Lambertian, and the appearance $\mathcal{A}$ is represented by a set of RGB albedo vectors $\mathbf{a}(\mathbf{x})$ for the voxels in the canonical frame. Note that the albedo represented on the voxels is used to interpolate the true albedo on the object surfaces; thus, only voxels on or near object surfaces have valid albedo values. The reason for using a volumetric data structure to represent the geometry is that compared with a mesh data structure, a regular volumetric structure is more suitable for parallel memory allocation and access on a GPU. It also enables more efficient geometric operations (e.g., normal calculations) on a GPU. These characteristics help to achieve real-time performance.

The environmental lighting $\mathcal{L}^t$ is represented by spherical harmonics (SH) [26]. For Lambertian surfaces, we use only the first nine SH basis functions, which correspond to the low-frequency lighting and provide a good approximation of Lambertian reflectance [12]. With this representation, the reflected irradiance B of a voxel $\mathbf{x}$ is expressed as follows:

$$B(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \sum_{m=1}^{b^2} \ell_m H_m(\mathbf{n_x}). \tag{1}$$

Here, $\{H_m\}$ represents the SH basis functions, $\mathbf{n_x}$ denotes the normal of voxel $\mathbf{x}$, and the $\{\ell_m\}$ are the SH coefficients that define the environmental lighting. $b = 3$ means that we consider only SH basis functions of up to the 2nd order.

We follow DynamicFusion [23] in representing the motion field $\mathcal{W}^t$ by a graph-based motion representation, which can effectively represent non-rigid deformation for a surface of any shape and can be applied to deform voxels. Specifically, $\mathcal{W}^t = \{[\mathbf{p}_j \in \mathbb{R}^3, \sigma_j \in \mathbb{R}^+, T_j \in \mathbf{SE}(3)]^{\mathrm{T}}\}$, where j denotes the index of the jth node in graph $\mathcal{G}$. $\mathbf{p}_j$ is the position of the jth node, and σ_j is a radius parameter related to the weight with which the jth node influences voxel $\mathbf{x}$. This weight is defined as $w_j(\mathbf{x}, \sigma_j) = exp(-\left\|\mathbf{x} - \mathbf{p}_j\right\|^2 / (2\sigma_j^2))$. σ_j is a predefined parameter. Finally, T_j is the 6D transformation (3D translation and 3D rotation) of the jth node. Details regarding how to generate the graph $\mathcal{G}$ from the geometry $\mathcal{V}$ and how to update $\mathcal{G}$ for an updated $\mathcal{V}$ can be found in [23].

5 METHOD

We now describe the overall method of our system. First, we propose a novel unified framework for estimating the motion field and low-frequency environmental lighting for each captured frame. In this framework, the static model $\mathcal{S}$ as well as the low-frequency environmental lighting $\mathcal{L}^{t-1}$ and the motion field $\mathcal{W}^{t-1}$ of the previous frame $t-1$ are used to estimate $\mathcal{L}^t$ and $\mathcal{W}^t$ based on the input depth $\mathcal{D}^t$ and the color C^t of the current frame t (Sec. 5.1). Through the integration of shading information, this step jointly achieves lighting estimation and more accurate motion reconstruction. Subsequently, $\mathcal{D}^t$ and C^t are used to update the geometry $\mathcal{V}$ and the appearance $\mathcal{A}$ of the static model $\mathcal{S}$ (Sec. 5.2). Benefiting from the coherent temporal correspondence, this fusion makes $\mathcal{S}$, which will subsequently be used in the processing of the next frame, more complete and more robust to input noise and reconstruction errors. Finally, we introduce our real-time solver (Sec. 5.3) and some details of the implementation (Sec. 5.4).

5.1 Joint Motion and Lighting Optimization

As previously mentioned, our system operates based on a tracking procedure. Therefore, for frame t, we have the current static model $\mathcal{S}$ in the canonical frame (obtained by fusing the information from previous frames), the low-frequency environmental lighting $\mathcal{L}^{t-1}$ and the motion field $\mathcal{W}^{t-1}$. Using the newly recorded color and depth in frame t, the algorithm presented in this section reconstructs $\mathcal{L}^t$ and $\mathcal{W}^t$ to fit the current shape and appearance of object. For this purpose, we propose a novel unified optimization framework based on a graph-based motion representation. The optimization is achieved by minimizing an energy function, which is formulated as follows:

$$E_{\text{total}}(\mathcal{W}^t, \mathcal{L}^t) = \omega_d E_{\text{depth}} + \omega_s E_{\text{shading}} + \omega_m E_{\text{mreg}} + \omega_l E_{\text{lreg}}, \tag{2}$$

where E_{depth} and E_{shading} are data terms that constrain the result to be consistent with the depth and color input, E_{mreg} regularizes the resolved motion to be as locally rigid as possible, and E_{lreg} is a temporal smoothing regularization that is applied to the environmental lighting because lighting typically does not change abruptly over time.

Prior to solving the energy function, we first extract a surface mesh $\mathcal{M}$ from the current static model $\mathcal{S}$ and interpolate the albedo values for all vertices. Additionally, the motion graph $\mathcal{G}$ is obtained via graph generation or graph updating [23].

E_{depth} represents a point-to-plane energy term [17], as follows:

$$E_{\text{depth}}(\mathcal{W}^t) = \sum_{(\mathbf{v}, \mathbf{u}^t) \in P} (\mathbf{n}_{\mathbf{u}^t}^{\mathrm{T}}(\mathbf{v}' - \mathbf{u}^t))^2, \tag{3}$$

where $\mathbf{v}$ is a vertex on $\mathcal{M}$ and $\mathbf{v}'$ is the transformed vertex defined by the formula $\mathbf{v}' = \sum_j w_j(\mathbf{v}, \sigma_j) T_j^t \mathbf{v}$. T_j^t is the transformation of the jth node in frame t, which will be solved for during the optimization process. $\mathbf{u}^t$ is a 3D point obtained by projecting a pixel in the depth frame $\mathcal{D}^t$ back into the 3D camera space, and $\mathbf{n}_{\mathbf{u}^t}^{\mathrm{T}}$ represents its normal. P contains all correspondence pairs.

To obtain the correspondence P, we first calculate $\hat{\mathcal{M}}$ by deforming $\mathcal{M}$ using the current estimated motion field $\hat{\mathcal{W}}$. Then, we render $\hat{\mathcal{M}}$ with respect to the depth camera to label the visible vertices as V_D^t. Without loss of generality, we can always treat the camera as fixed by regarding the camera motion as part of the global motion of the captured object. In this way, the rendering is achieved by directly projecting the deformed model $\hat{\mathcal{M}}$ using the intrinsic matrix K_d of the depth camera. Finally, for each $\mathbf{v}$ in V_D^t, if a valid $\mathbf{u}^t$ exists with the same image coordinates in $\mathcal{D}^t$, then the pair $(\mathbf{v}, \mathbf{u}^t)$ is treated as a correspondence and stored in P. This scheme, which is based on projective ICP [27], guarantees the real-time performance of our method.

E_{shading} is an SH shading term that is formulated as follows:

$$E_{\text{shading}}(\mathcal{W}^t) = \sum_{\mathbf{v} \in V_C^t} \left\|C^t(M_c(\mathbf{v}')) - B^t(\mathbf{v}')\right\|_2^2, \tag{4}$$

where V_C^t is the set of all visible vertices obtained by projecting $\hat{\mathcal{M}}$ using the projection matrix of the color camera, M_c; $C^t(M_c(\mathbf{v}'))$ is the projected color of vertex $\mathbf{v}'$ in the tth color frame; and $B^t(\mathbf{v}')$ is the SH shaded irradiance of this vertex, which has already been

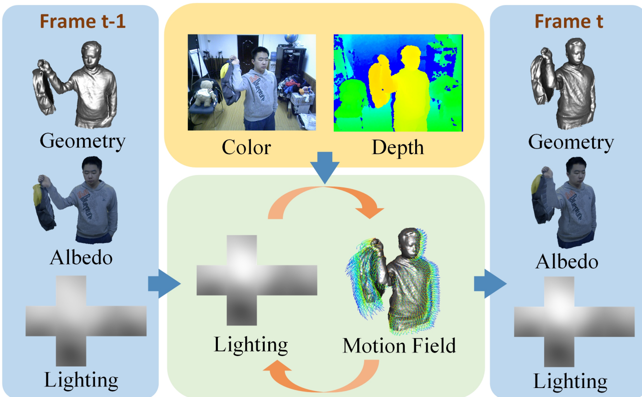


Fig. 3. Scheme of the joint motion and lighting optimization process. The first column shows the input geometry, albedo and low-frequency lighting from the previous frame. The last column shows the geometry and albedo after warping using the motion field $\mathcal{W}^t$ and the lighting $\mathcal{L}^t$ that have been determined based on the input color and depth of the current frame.

defined in Eqn. 1. By including this term, we ensure that our optimization can more accurately estimate the inter-frame motion because the appearance of an object never changes over time. Note that because we assume the object surfaces to be purely diffuse, low-frequency lighting represented by the first nine SH coefficients is sufficient to model the lighting and shading of a pixel using Eqn. 1. We believe that more sophisticated appearance models, such as SVBRDF [41], may yield better results and allow the handling of more general objects. However, it is not trivial to apply these models in such a system, where geometry, lighting and appearance are all treated as unknowns. Furthermore, real-time performance becomes more difficult to maintain.

E_{mreg} is an as-rigid-as-possible regularizer, as follows:

$$E_{\text{mreg}}(\mathcal{W}^t) = \sum_{j \in \mathcal{G}} \sum_{i \in N_j} \left\| T_j^t \mathbf{p}_j - T_i^t \mathbf{p}_j \right\|_2^2, \tag{5}$$

where N_j denotes the set of neighboring nodes of the jth node. This term avoids overfitting to the noisy depth input. Furthermore, because we use only a single-view input, some object regions will not be captured by the sensor. If these regions already exist in $\mathcal{S}$, then this regularization term is important for ensuring that they will move with the visible regions as rigidly as possible.

E_{lreg} stabilizes the lighting by constraining it to be similar to the lighting in the previous frame:

$$E_{\text{lreg}}(\mathcal{L}^t) = \sum_{i=1}^{b^2} \left\| \ell_i^{t-1} - \ell_i^t \right\|_2^2. \tag{6}$$

To minimize the energy in Eqn. 2, we propose a two-step iterative algorithm that alternately updates $\mathcal{W}^t$ and $\mathcal{L}^t$. The initial values of the unknowns $\hat{\mathcal{W}}$ and $\hat{\mathcal{L}}$ are set to $\mathcal{W}^{t-1}$ and $\mathcal{L}^{t-1}$, respectively. The first step of the iterative procedure is to solve a geometry alignment problem with known and fixed lighting. The second step is to solve a shading optimization problem to recover the low-frequency lighting for a given motion field. This joint optimization scheme is illustrated in Fig. 3.

5.1.1 Motion Estimation. In the first step of the two-step iterative procedure, we estimate a non-rigid deformation defined on $\mathcal{G}$, which deforms $\mathcal{M}$ to align with the input depth and color, for fixed environmental lighting $\{\ell_m\}$ and vertex albedos $\{\mathbf{a}_\mathbf{v}\}$. Because the lighting is fixed, $\omega_l = 0$ in Eqn. 2. We also set $\omega_d = 1$, $\omega_s = 0.0001$, and $\omega_m = 5$ in all of our experiments to balance the influence of these terms.

With these settings, the minimization of Eqn. 2 becomes an optimization on the motion parameters $X_\mathcal{G} = \{\xi_j\}_{j \in \mathcal{G}}$ (i.e., the 6D rotation and translation parameters for all nodes), which is a nonlinear least-squares problem. We solve this problem through Gauss-Newton iterations. In each iteration, the problem is linearized around the transformations from the previous iteration. Then, a linear problem is solved to obtain the updated transformations for the current iteration.

Specifically, we use the twist representation [22] to represent the 6D motion parameters of each node, denoted by ξ_j. We use an exponential map to convert ξ_j into an $\mathbf{SE}(3)$ transformation matrix, i.e., $T_j = e^{\hat{\xi}_j}$. In each Gauss-Newton step, we represent the update of the motion parameters of each node as a twist update (denoted by ζ_j) around the current estimated deformation and linearize the energy formulation around $\zeta_j = \mathbf{0}$. The linearization relies on a first-order Taylor expansion of the exponential map for the twist update, defined as follows:

$$e^{\hat{\zeta}_j} \approx \mathbf{I} + \hat{\zeta}_j = \begin{pmatrix} 1 & -\gamma_j & \beta_j & t_{jx} \\ \gamma_j & 1 & -\alpha_j & t_{jy} \\ -\beta_j & \alpha_j & 1 & t_{jz} \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{7}$$

Here, we define $\zeta_j = [\alpha_j, \beta_j, \gamma_j, t_{jx}, t_{jy}, t_{jz}]^\mathrm{T}$ to denote the 6 transformation parameters of node j. Given this linearization, we can express the position $\mathbf{v}'$ and the normal $\mathbf{n}_{\mathbf{v}'}$ as linear functions of the transformation parameters of all nodes, denoted by $Z_\mathcal{G} = [\zeta_j]_{j \in \mathcal{G}}^\mathrm{T}$. After further linearization of the color frame $C^t(M_c(\mathbf{v}'))$ via Taylor expansion in the image domain, we have the following fully linearized problem:

$$\mathbf{J}^\mathrm{T}\mathbf{J}\hat{\mathbf{x}} = \mathbf{J}^\mathrm{T}\mathbf{b}, \tag{8}$$

where $\hat{\mathbf{x}} = Z_\mathcal{G}$. After solving for $\hat{\mathbf{x}}$, we update $\{T_j\}$ for all nodes as follows:

$$T_j' = e^{\hat{\zeta}_j} T_j, \tag{9}$$

where T_j is the latest 4×4 transformation matrix for node j from the previous iteration and T_j' is the updated transformation matrix. Then, we perform the same expansion in the subsequent iterations. Details concerning this linearization can be found in [22, 39]. To achieve real-time performance of our overall system, we implement it using a highly efficient GPU-based scheme, which will be introduced in Sec. 5.3.

5.1.2 Lighting Estimation. In the second step of the two-step iterative procedure, we solve for the low-frequency environmental lighting $\mathcal{L}^t$ with a given motion field and albedo. The energy function to be optimized, expressed in Eqn. 2, has only two terms: E_{shading} and E_{lreg}. We set $\omega_d = 0$, $\omega_s = 1.0$ and $\omega_m = 0$. Note that we allow lighting changes in the scene; thus, ω_l is set to a relatively small number (because E_{lreg} is considerably smaller than E_{shading},

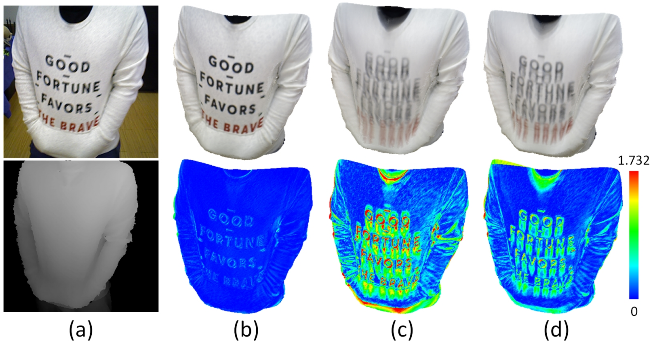


Fig. 4. Motion estimation with shading information: (a) input color and depth images, (b) the reconstructed result obtained using the shading term and its color-coded residual compared with the input color image, (c) the reconstructed result obtained without the shading term (DynamicFusion) and its corresponding residual, and (d) the reconstructed result obtained using feature matching information (VolumeDeform) and its corresponding residual. The residuals are calculated as the per-pixel Euclidean distances of the RGB values. Each color channel is normalized to [0,1].

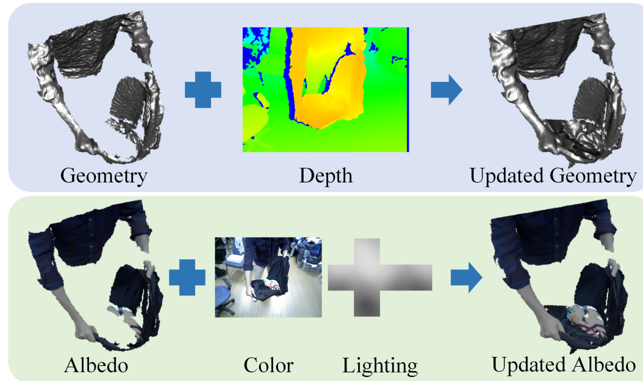


Fig. 5. Scheme of the static model update procedure. The first row shows the geometry update, and the second row shows the albedo update.

we use 10^5 for ω_l.). The minimization of Eqn. 2 thus becomes a linear least-squares problem. We perform parallel reduction on a GPU, download the coefficients of the system to a CPU, and solve the problem using SVD factorization.

As stated previously, our iterative optimization algorithm utilizes shading information to more accurately estimate scene motions. As shown in Fig. 4, when the captured actor is stretching the cloth downward, our method successfully reconstructs the motion, which cannot be recovered without the shading term because the traditional ICP algorithm does not provide the correct correspondences. We observe that in the result obtained without the shading term, the position of the cloth texture does not match the input image. An alternative solution is to use image feature matching to guide the motion estimation, as recently proposed by [14]. Although this approach considers color information, because the image features are generally sparse and strongly depend on the texture in the scene, it still achieves only limited improvement for this challenging motion, as shown in Fig. 4(d).

5.2 Static Model Update

In this subsection, we discuss how to update the static model $\mathcal{S}$ in the canonical frame using the input depth and color from the tth frame, the reconstructed low-frequency environmental lighting $\mathcal{L}^t$ and the motion field $\mathcal{W}^t$ (shown in Fig. 5). In the following, we introduce how to update $\mathcal{V}$ and $\mathcal{A}$.

5.2.1 Geometry Update. To update $\mathcal{V}$ in the canonical frame, following the method presented in [23], we warp each canonical voxel into the current frame, project it into the depth map, and update the TSDF value and weight. However, this scheme does not inherently handle voxel collisions (see *bag open and close* sequence for an example of voxel collisions); instead, it generates erroneous surfaces within the colliding voxels. The reason is that voxels near but outside one surface in the canonical frame will be warped close to the depth of the other surface when two surfaces come close to colliding in the live frame. In this case, the SDFs of the voxels are updated based on the depth, and an erroneous surface is generated in the canonical frame, as shown in Fig. 6(d).

To solve this problem, we borrow the idea of [7] to detect colliding voxels by checking whether a voxel is moving to the same position as another, non-neighboring voxel in the live frame. Specifically, we first voxelize the live frame into regular voxel cells. Then, we check whether more than one voxel in the canonical frame is moving to the same voxel cell in the live frame. If so, all these voxels are labeled as colliding voxels. In [7], colliding voxels with relatively small absolute SDF values are still updated in the live frame; however, when the two surfaces come into contact with each other, these voxels can still be updated based on the depths of surfaces that are far from them in the canonical frame, thus generating erroneous surfaces. To overcome this problem, in our method, we use a more strict approach in which SDF updating is prevented for all colliding voxels to achieve collision handling, as shown in Fig. 6(b, c).

5.2.2 Albedo Update. After the geometry update, we apply a novel optimization scheme to update the albedo values at voxels rather than vertices. This is because our basic geometry representation is a TSDF defined on voxels. Thus, it is more direct and efficient to update the albedo values on the voxels. The energy function is formulated as follows:

$$E_{\text{albedo}}(\mathcal{A}) = E^{v}_{\text{shading}} + w_{as} E_{\text{asreg}} + w_{at} E_{\text{atreg}}. \tag{10}$$

The shading term E^{v}_{shading} is similar to E_{shading} but is expressed on voxels:

$$E^{v}_{\text{shading}} = \sum_{\mathbf{x} \in V^t_M} \phi(\mathbf{n}(\mathbf{x}^t) - \mathbf{c}(\mathbf{x}^t)) \left\| C^t(M_c(\mathbf{x}^t)) - B^t(\mathbf{x}) \right\|_2^2. \tag{11}$$

This term constrains the voxel albedo to be consistent with the color frame under the estimated environmental lighting. Here, we define V^t_M as $\{\mathbf{x} | \left|\mathbf{psdf}^t(\mathbf{x})\right| < 0.5\tau\}$ to guarantee that only voxels that are close to visible surfaces in frame t are considered in the optimizations. This is done for two reasons. First, the motion of the voxels is interpolated from the motions of nodes on the visible surfaces; thus, the motions obtained for voxels that are close to these surfaces will be more accurate. Second, albedo is essentially defined

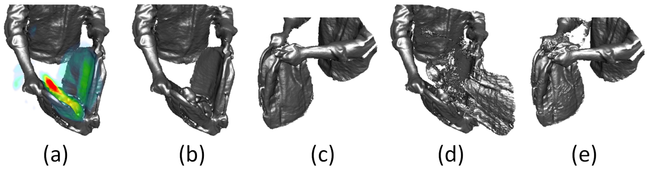


Fig. 6. Illustration of collision detection. (a) shows the detected colliding voxels in the canonical frame, where the color indicates the density of the colliding voxels. (b) and (c) show the reconstructed results obtained with collision detection in the canonical and live frames, respectively. (d) and (e) show the reconstructed results obtained without collision detection in the canonical and live frames, respectively. Note that erroneous surfaces are generated because of the incorrect TSDF updating of the colliding voxels.

on surfaces. It is not reasonable to update albedo values for voxels that are far from surfaces. Here, $\mathbf{x}^t$ represents voxel $\mathbf{x}$ deformed by $\mathcal{W}^t$; $\phi(x)$ is the Huber kernel, which is used to control the weights based on $\mathbf{n}(\mathbf{x}^t)$, the normal of voxel $\mathbf{x}^t$, and $\mathbf{c}(\mathbf{x}^t)$, the direction from $\mathbf{x}^t$ to the camera center. The idea is that when the normal of a voxel is quite different from the projection direction, it is likely that a small error in the voxel position or the camera parameters will cause a foreground voxel to be projected to a background pixel. To mitigate this effect, we reduce the weight in this situation.

The purpose of E_{atreg} is to maintain the temporal coherence of the albedo, and it is defined as follows:

$$E_{\text{atreg}} = \sum_{\mathbf{x} \in V_M^t \cap V_M^{t-1}} \left\| \mathbf{a}^{t-1}(\mathbf{x}) - \mathbf{a}^t(\mathbf{x}) \right\|_2^2 . \tag{12}$$

In practice, we set w_{at} to a relatively large value ($w_{at} = 100$, whereas $w_{as} = 1$) because the albedo of ordinary objects does not change over time. However, this is not a hard constraint. The reason is that images may contain noise, and the motion and lighting may not be perfectly estimated. Fusing the results from multiple frames will make the result more accurate and robust. As is demonstrated in the results section, with the accumulation of frames, the albedo becomes increasingly more feasible and stable.

We also follow [47] in including a spatial smoothing constraint on the albedo, as follows:

$$E_{\text{asreg}} = \sum_{\mathbf{x} \in V_M^t} \sum_{\mathbf{z} \in N(\mathbf{x}) \cap V_M^t} \phi(\Gamma(\mathbf{x}^t) - \Gamma(\mathbf{z}^t)) \left\| \mathbf{a}^t(\mathbf{x}) - \mathbf{a}^t(\mathbf{z}) \right\|_2^2 . \tag{13}$$

Here, $\Gamma(\mathbf{x}^t) = C^t(M_c(\mathbf{x}^t))/I^t(M_c(\mathbf{x}^t))$ denotes the chromaticity of the pixel corresponding to voxel $\mathbf{x}$, and $I^t(M_c(\mathbf{x}^t))$ represents its illumination attribute [40].

Because E_{albedo} is a quadratic energy function when $\mathbf{x}^t$ and $\{\ell_m\}$ are fixed, $\mathbf{a}^t(\mathbf{x})$ is calculated by solving a linear system.

5.3 Efficient Gauss-Newton Solver

Here, we introduce how to implement a highly efficient Gauss-Newton solver on a GPU for solving Eqn. 8. Similar to the previous real-time solution [48], we use the preconditioned conjugate gradient (PCG) method. To save memory when storing $\mathbf{J}$ and $\mathbf{J}^{\mathrm{T}}$, Zollhöfer et al. [48] implemented two sparse matrix-vector multiplication (SpMV) kernels to calculate $\mathbf{Jz}$ and $\mathbf{J}^{\mathrm{T}}\mathbf{z}$ on the fly. Here, $\mathbf{z}$ represents an arbitrary vector with the corresponding number of dimensions. The drawback of this implementation is that calculating $\mathbf{J}$ in each PCG iteration incurs considerably more repeated computations and accesses to global memory. This is not tolerable in our method for two reasons. First, our shading term involves much more computation for calculating $\mathbf{Jz}$ and $\mathbf{J}^{\mathrm{T}}\mathbf{z}$ and requires more access to global memory. Second, the difference in the number of floating-point operations between the depth data term and the shading term causes workload imbalance and thread divergence for the warps in each CUDA block, which dramatically decreases the performance.

To overcome these drawbacks, we return to explicitly calculating $\mathbf{J}^{\mathrm{T}}\mathbf{J}$, following the method used in [24]. Specifically, we treat $\mathbf{J}^{\mathrm{T}}\mathbf{J} \in \mathbb{R}^{6N \times 6N}$ as an $N \times N$ matrix with block elements $B_{ij} \in \mathbb{R}^{6\times 6}$. As mentioned in [24], the benefit of this block structure is that all B_{ij}s can be constructed in parallel. In particular, B_{ij} is related to nodes i and j and is nonzero if nodes i and j together contribute to at least one constraint in Eqn. 2 and can be calculated independently from other blocks. Otherwise, B_{ij} is a zero block. In each ICP iteration, the indices of the nonzero blocks of B_{ij} do not change; thus, we pre-calculate the nonzero pattern via fast GPU radix sort and use it to construct $\mathbf{J}^{\mathrm{T}}\mathbf{J}$ for all inner PCG iterations. In our implementation, we calculate each nonzero block in one CUDA block using warp-level reduction. Since modern GPU architectures include many register resources, we exploit the on-chip registers and use warp shuffle instructions to accelerate block reduction for each $\mathbf{B}_{ij}$. This provides a fast solution for communication within one warp and also increases GPU occupancy. For the core solver, we use the kernel-merged PCG solver used in a previous work [36]. Again, we implement the solver at the warp level and tune it for maximum efficiency. This PCG solver is also used to calculate the solution to Eqn. 10.

5.4 Implementation Details

System Pipeline. Our pipeline consists of three components: the front end, a solvers-and-integration module and a rendering component. The front end fetches synchronized depth-color pairs, executes bilateral filtering on the depth maps, generates vertex and normal maps from depth maps, and removes mismatched pixels at the boundaries of the depth and color maps. The solvers-and-integration module first executes the rigid projective ICP processing and then optimizes both the motion field and the lighting coefficients. After updating the camera pose, this module also non-rigidly fuses the depth data into the canonical volume and solves the optimization problem for updating the albedo volume. The rendering component executes the marching cube algorithm on the TSDF volume and then extracts a triangle surface for rendering. These three components run asynchronously and exchange data via semaphores and buffers. Most of the technical aspects of these components are implemented on a GPU. The linear and non-linear solvers are implemented using the CUDA API, which provides more convenient intrinsics for warp-level optimization. The TSDF and albedo integration and the rendering techniques are implemented using GLSL and the OpenGL API. We constrain the maximum processing time of each component to guarantee a 30 Hz output frequency, matching the output frequency of the sensor.

Initialization. Here, we discuss how to initialize the first input frame, for which $\mathcal{S}$, $\mathcal{W}^{t-1}$ and $\mathcal{L}^{t-1}$ are unknown. By assuming $\{T_j\}$ in $\mathcal{W}^0$ to be the identity matrix, we can directly use the depth frame $\mathcal{D}^0$ to estimate the TSDF values of $\mathcal{V}^0$ using the method described in Sec. 5.2.1. Because $\mathcal{L}^0$ is still unknown, we then extract the object surface from $\mathcal{V}^0$ and estimate $\mathcal{L}^0$ with a fixed uniform albedo [47] using the method presented in Sec. 5.1.2. The chosen albedo value also determines the scale of the albedo in Eqn. 1, which contains a scale ambiguity. Subsequently, we use the estimated $\mathcal{L}^0$ and $\mathcal{V}^0$ to estimate the initial albedo $\mathcal{A}^0$ via the method introduced in Sec. 5.2.2. Once $\mathcal{W}^0$, $\mathcal{L}^0$ and the static model $\mathcal{S}$ have been reconstructed, the subsequent frames can be processed using our tracking procedure as introduced previously.

Back Reconstruction for Free-viewpoint Video Rendering. In the real-time processing chain described above, the reconstructed geometry is accumulated frame by frame; thus, the reconstructed surface may not be complete, particularly for early frames. Once the entire sequence has been processed, we can use the final reconstructed static model $\mathcal{S}^F$ to back reconstruct the entire sequence to obtain an as-complete-as-possible reconstruction for all frames. This is achieved by deforming $\mathcal{S}^F$ using the calculated warping motion for each frame. Note that in the early frames, only some of the nodes (observed nodes) exist, and no motions are associated with the missing nodes in these frames. We use dual-quaternion blending to propagate the motions of the observed nodes to the missing nodes based on the Euclidean distances between nodes. Because the missing nodes generally lie in invisible regions, which we assume do not exhibit independent motions but instead merely follow the motions in the visible regions, our dual-quaternion blending does not generate noticeable artifacts in the final results of our experiments. Fig. 17 shows the rendering of a free-viewpoint video from a novel viewpoint opposite to the captured view.

6 RESULTS

In this section, we demonstrate the effectiveness of the two key components of our method, namely, the *joint motion and lighting optimization*, which reconstructs more accurate surface motions, and the *static model update*, which achieves the decomposition of lighting and albedo and gradually improves the albedo quality by fusing information from multiple frames. For motion reconstruction, we compare our results with those of the recent methods DynamicFusion [23] and VolumeDeform [14]. We present the results obtained on various motion sequences, followed by several applications, including relighting, free-viewpoint video and albedo editing. Finally, we discuss the limitations of our technique.

Performance. Our system is fully implemented on a single NVIDIA GeForce GTX TITAN X graphics processing unit using both the OpenGL API and the NVIDIA CUDA API. The host configuration is a 3.2 GHz 4-core Xeon E3-1230 with 16 GB of memory. The pipeline runs at 32 ms per frame, of which 23 ms is devoted to the joint motion-lighting optimization and albedo update, 5 ms is used for TSDF integration, and 4 ms is allocated for all other operations. For all of our sequences, the processing time for one frame is less than 32 ms. Therefore, we lock the frame rate of our system to 30 Hz, matching the sensor's frame rate. The pipeline also includes the preprocessing of each depth and color image, including the bilateral filtering and the calculation of image gradients. For the joint motion and lighting optimization, we alternate twice between shading-based registration and the calculation of the lighting coefficients. The shading-based registration involves 3 ICP iterations. The working volume is set to 1.2 m^3 for the *backpack* and *walking ted* sequences and to 1.0 m^3 for the rest. For all cases, the resolutions of the TSDF and the albedo volume are both 2 mm. The node sampling radius is 25 mm. The number of nodes can be up to 3K for half body motion.

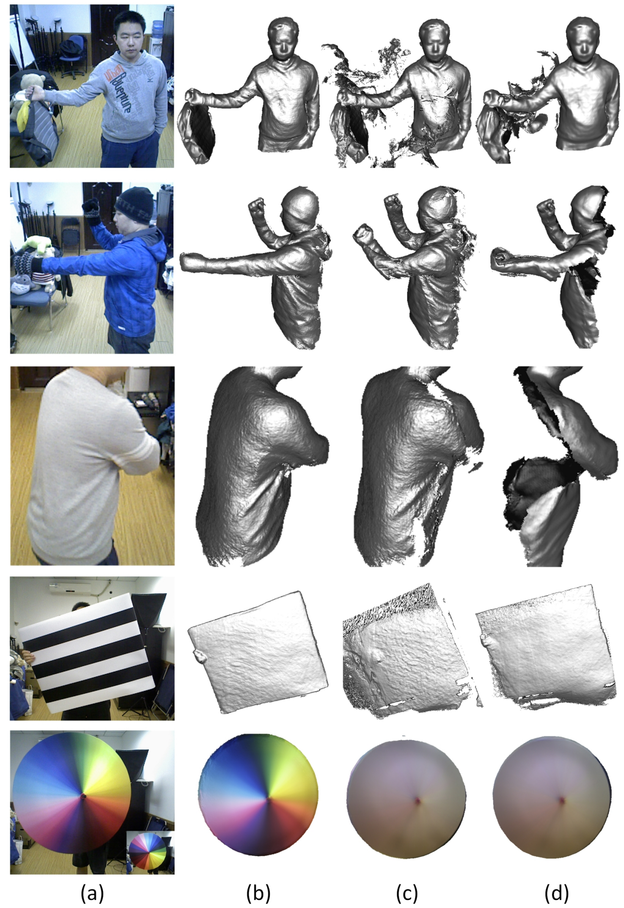


Fig. 7. Comparisons with DynamicFusion and VolumeDeform. (a) Input color images; (b-d) the reconstructed results of our method, DynamicFusion and VolumeDeform, respectively. The first image in the last row contains a sub-image that shows the texture of the umbrella. The results presented in the first 3 rows of (d) were provided by the first author of VolumeDeform. The other results for DynamicFusion and VolumeDeform were produced using our implementations based on these methods.

6.1 Evaluation

Because we reconstruct the motion, albedo and low-frequency lighting of a scene, we evaluate our method with respect to all three of these aspects.

Motion. Our method considers shading information during motion reconstruction and detects colliding voxels to handle collisions;

thus, more accurate and robust reconstructions are achieved compared with previous state-of-the-art methods, including DynamicFusion and VolumeDeform. Note that our reconstructed results, shown in the following figures and the accompanying videos, were obtained by relighting the reconstructed geometry and albedo using the estimated lighting in the corresponding frames.

Fig. 7 compares our method with the other two methods on various sequences. The top row shows a case of surface collision. As shown, the previously developed methods generate erroneous surfaces, whereas our method achieves the correct reconstruction. The second and third rows show cases of casual boxing motion and fast loop closure, respectively. Because our algorithm uses the shading information, we succeed in reconstructing the motion, whereas the other methods generate noticeable artifacts. The fourth row shows a case of tangential motion, in which a board is facing the camera and moving upward. DynamicFusion fails because the geometry does not provide sufficient information to indicate the tangential motion, and VolumeDeform also fails because no features suitable for matching can be extracted from the simply textured board. By contrast, our method reconstructs the motion correctly because it benefits from the shading information.

The last row in Fig. 7 compares the motion reconstructions for a rotating umbrella with spatially varying colors. Because the rotation cannot be detected from pure geometry, we rendered the reconstructed appearance to display in the figure. For DynamicFusion and VolumeDeform, because these methods do not reconstruct texture or albedo, we directly fused the captured colors to generate textured results. We can clearly observe that the texture of the umbrella is heavily blurred in the results of DynamicFusion and VolumeDeform, whereas our result is consistent with the input. The results presented in Fig. 4 confirm that to achieve plausible texture or albedo rendering, the reconstructed motion needs to be accurate. Our method represents a substantial improvement over the state-of-the-art methods in this respect, thereby enabling surface texturing and relighting applications.

We also numerically compare the three methods in terms of geometry based on the results of rendering a pre-reconstructed 3D motion to obtain ground-truth depth and color frames as input. Two frames of the sequence are shown in Fig. 8. Compared with the ground truth, our method demonstrates considerably better geometry reconstruction compared with both DynamicFusion and VolumeDeform. Both of the latter methods fail to reconstruct this dynamic scene because of complex object collisions. The average geometry errors for the entire sequence are also presented in Tab. 1.

	Our Method	DynamicFusion	VolumeDeform
error (mm)	1.036	109.567	84.245

Table 1. Average numerical errors on the *bag open and close* sequence. We calculate the errors only for the visible surface regions.

Albedo. After the *joint motion and lighting optimization*, our method further factors out the estimated environmental lighting and reconstructs the surface albedo via the albedo update scheme. Fig. 9 shows the estimated albedo and lighting for an input color image. We can clearly observe that the shading component, which leads to spatial variations in the color of the pillow in the color image, has been factored out in our estimated albedo map.

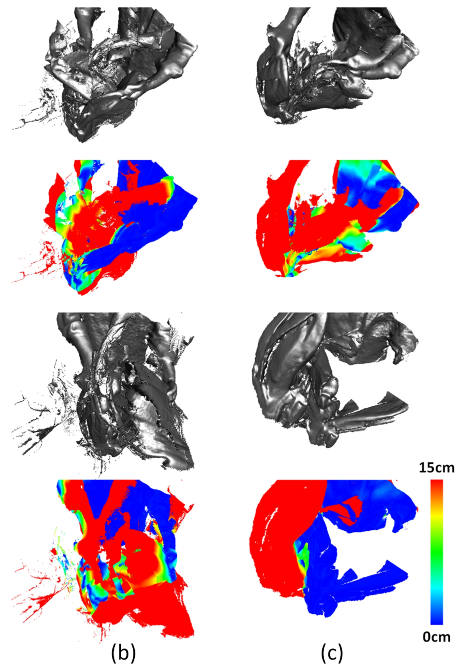


Fig. 8. Quantitative comparison of our results with the ground truth for the *bag open and close* sequence. (a), (b) and (c) present the results of our method, DynamicFusion and VolumeDeform, respectively. The 1st and 2nd rows present the geometry and error maps, respectively, for frame 191, and the 3rd and 4th rows present the geometry and error maps, respectively, for frame 277. The results for DynamicFusion and VolumeDeform were produced using our implementations based on these methods.

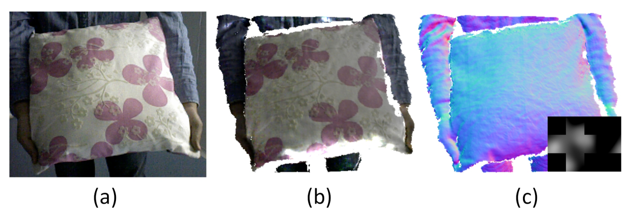


Fig. 9. Example of albedo reconstruction: (a) input image, (b) reconstructed albedo, and (c) reconstructed surface normal and lighting.

Furthermore, benefiting from the correctly reconstructed motions, we can align multiple frames and use their integrated information to fuse the surface albedo. As an increasing number of frames are observed and incorporated, our fused albedo becomes increasingly more robust to noise and errors. Fig. 10 summarizes this process. As shown in this figure, with frame accumulation, not only does the geometry gradually become more complete but the quality of the albedo also improves.

An alternative solution for albedo fusion is to directly blend the reconstructed per-frame albedos. The blending weights could be defined similarly to the weights used in geometry fusion. This alternative solution does not require an optimization step; however, it

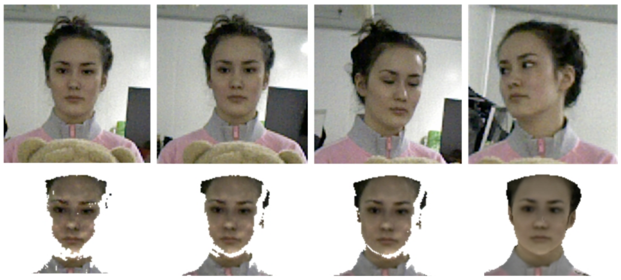
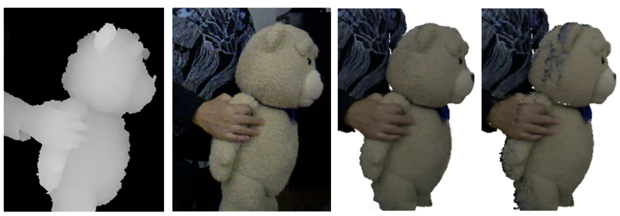
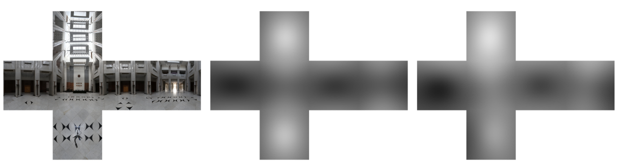


Fig. 10. Process of albedo fusion. Top row: selected sequential frames in a sequence. Bottom row: the corresponding static models $\mathcal{S}$ in the canonical frame.



(a) (b) (c) (d)

Fig. 11. Validation of our albedo fusion scheme: (a) depth input, (b) color input, (c) the result of our scheme, and (d) the result of direct albedo averaging.



(a) (b) (c)

Fig. 12. Low-frequency lighting reconstruction: (a) ground-truth lighting, (b) low-frequency component of the ground-truth lighting, and (c) our reconstructed low-frequency lighting.

does not generate results that are as good as our solution. Fig. 11 presents a comparison of the two solutions. Because our algorithm assigns lower confidences to albedos with surface normals orthogonal to the camera direction and considers the spatial consistency of the albedo, it generates visually pleasing albedo fusion results. Conversely, albedo blending incorporates the texture of the background into the foreground object and generates a result with spatially inconsistent albedo.

Lighting. The low-frequency environmental lighting is a byproduct of our method that is used in both motion estimation and albedo extraction. To test the accuracy of this component, we compare our result with the ground truth for a scene with spatially varying lighting conditions (shown in Fig. 12). Note that we reconstruct the low-frequency environmental lighting using only the first nine SH coefficients. From this comparison, we observe that our result generally matches the ground truth. The corresponding motion sequence from this experiment is shown in the accompanying video.

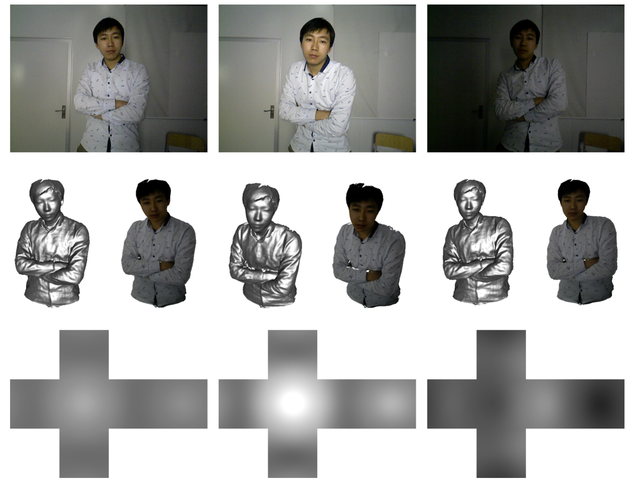


Fig. 13. Reconstructed low-frequency lighting, albedo and geometry for a scene with time-varying lighting. Top row: input images. Middle row: reconstructed geometry and albedo. Bottom row: estimated low-frequency lighting.

Benefiting from the lighting estimation, our method can be applied to scenes with time-varying lighting. As shown in Fig. 13, our method successfully factors out the lighting in the scene and estimates both the albedo and geometry of the moving object. Note that the reconstructed albedo is both spatially and temporally consistent in the results. The sequence result is shown in our accompanying video.

6.2 Results

We present the results of running our algorithm on sequences with various casual motions, including body motion, cloth motion, object manipulation and loop closure. Some intermediate frames are shown in Fig. 14. We also present sequence results in the accompanying video. We also ran our system on sequences of publicly available data from [14], and the corresponding results are also shown in the accompanying video. Although we may observe holes in the geometry or artifacts in the rendered images obtained through the reconstruction procedure, as an increasing number of frames are considered, these artifacts diminish by virtue of the integration of additional information. This beneficial behavior can be attributed to the two key components of our method: the *joint motion and lighting optimization*, which achieves the accurate estimation of temporal correspondence, and the *volumetric optimization scheme for albedo updating*, which fuses photometric information from multiple frames to generate temporally coherent albedo model. We also demonstrate our live system in the accompanying video. In addition to real-time reconstruction, our system also supports real-time relighting and interactions for rotating, scaling and translating the reconstructed scene.

6.3 Applications

Because our algorithm reconstructs $\mathcal{S}^F$ and per-frame motions, we can relight a motion sequence using a given set of environmental

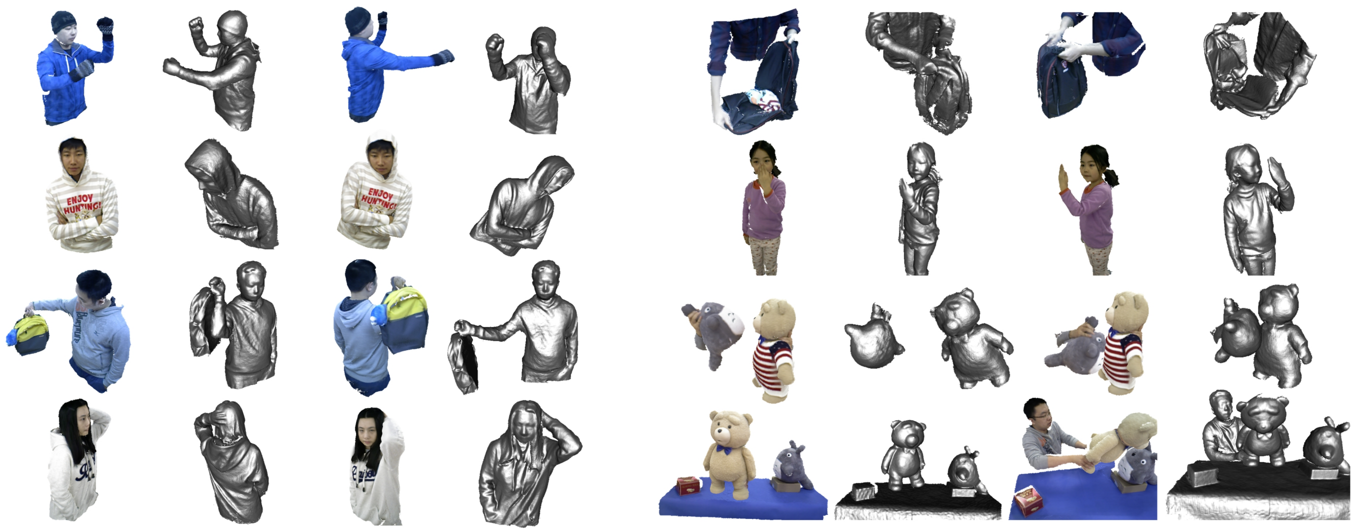


Fig. 14. Images of the results of our method, which are shown as rendered appearances and shaded geometries.

Fig. 15. Selected frames in 3 relighted sequences.

Fig. 16. Relighting a scene with edited albedo under 4 different lighting conditions.

Fig. 17. Comparison of the results of back reconstruction with partially reconstructed results for the *backpack* sequence. (a) and (b) show comparisons of back-reconstructed results with our partially fused results at frame 160 and frame 490, respectively. The left and middle figures in each group show the partially reconstructed models produced by our algorithm in the first pass rendered from the original camera view and a virtual view, respectively. The right figure in each group shows the back-reconstructed model produced from the final motion field generated in the first pass as rendered from the same virtual view.

lighting coefficients, perform appearance editing by changing the albedo of object surfaces, and back reconstruct a sequence using a more complete geometry model, thereby enabling high-quality free-viewpoint video rendering. The results of such applications are presented in the accompanying video and in Fig. 15, Fig. 16 and Fig. 17. These results demonstrate that the simultaneous reconstruction of geometry, albedo and motion effectively enables important applications in computer vision and graphics. Note that these applications cannot be achieved using either DynamicFusion or VolumeDeform because these methods do not reconstruct texture, albedo or complex motions as our method does. Furthermore, since the use of shading information in the joint optimization helps to determine more accurate motions and thus to generate better geometry and appearance models for a dynamic scene, our system can also perform real-time 3D self-portraits. This application is demonstrated in the *Applications* section of the primary accompanying video.

6.4 Limitations

Our system is based on the Lambertian assumption, and thus, it works well for casual lighting and diffuse materials but not for high specularity; see the artifacts in Fig. 18(a). Our system uses shading information to better reconstruct motions, which is not applicable

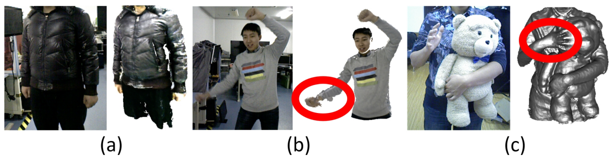


Fig. 18. Failure cases of our system: (a-c) input and reconstructed results for cases of highlights, motion blur and object separation, respectively.

for objects with a purely uniform appearance. Our system is also restricted by the capabilities of the sensor. For instance, the color frames captured by the Xtion are still of low quality, which limits the quality of our reconstructed albedo. Additionally, because the utilized sensor runs at 30 FPS and with a 20-30 ms exposure time, our system produces artifacts or even fails when tracking fast motions because of the large inter-frame differences and severe motion blur (as shown in Fig. 18(b)). Moreover, although the collision detection approach is suitable for open-to-closed motions, the system still cannot handle topological changes such as object separation and closed-to-open motions (as shown in Fig. 18(c)).

7 CONCLUSION

In this paper, we have presented a novel real-time algorithm for simultaneously reconstructing the geometry, albedo and motion of a dynamic scene using a single RGBD camera. The proposed method achieves temporal coherence and high-quality fusion of both geometry and albedo by treating surface motion registration and intrinsic surface estimation as optimization problems. With our approach, we are able to conveniently produce competent results on casual dynamic scenes using a single RGBD camera, without the need for a controlled environmental setup, multi-view camera input or highly skilled manual operations. Moreover, our method can be extended by including sparse feature terms, e.g., a SIFT feature term, to provide global anchors and to potentially achieve better results for motions such as loop-closing motions. We believe that our method represents a step toward enabling the use of consumer depth cameras for a wider range of graphics and video applications, such as 3D modeling, free-viewpoint video, video editing, motion capture and animation.

ACKNOWLEDGMENTS

We gratefully acknowledge Matthias Innmann for running his algorithm on our sequences *backpack*, *umbrella*, *robust loop closure*, *Ted and Totoro* and *boxing*.

REFERENCES

[1] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. 2008. Markerless garment capture. In *ACM Transactions on Graphics (TOG)*, Vol. 27. ACM, 99.
[2] Cedric Cagniart, Edmond Boyer, and Slobodan Ilic. 2010. Free-form mesh tracking: a patch-based approach. In *CVPR*. IEEE, 1339–1346.
[3] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. 2015. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 69.
[4] Brian Curless and Marc Levoy. 1996. A Volumetric Method for Building Complex Models from Range Images. In *SIGGRAPH*. ACM, New York, NY, USA, 303–312. DOI: https://doi.org/10.1145/237170.237269
[5] Edilson De Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. 2008. Performance capture from sparse multi-view video. *ACM Transactions on Graphics (TOG)* 27, 3 (2008), 98.
[6] Julie Dorsey, Holly Rushmeier, and François Sillion. 2010. *Digital modeling of material appearance.* Morgan Kaufmann.
[7] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, and others. 2016. Fusion4D: real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 114.
[8] Mingsong Dou, Jonathan Taylor, Henry Fuchs, Andrew Fitzgibbon, and Shahram Izadi. 2015. 3D Scanning Deformable Objects with a Single RGBD Sensor. In *CVPR*. 493–501.
[9] Per Einarsson, Charles-Felix Chabert, Andrew Jones, Wan-Chun Ma, Bruce Lamond, Tim Hawkins, Mark T Bolas, Sebastian Sylwan, and Paul E Debevec. 2006. Relighting Human Locomotion with Flowed Reflectance Fields. *Rendering techniques* 2006 (2006), 17th.
[10] Kaiwen Guo, Feng Xu, Yangang Wang, Yebin Liu, and Qionghai Dai. 2015. Robust Non-Rigid Motion Tracking and Surface Reconstruction Using L0 Regularization. In *ICCV*. 3083–3091.
[11] Kaiwen Guo, Feng Xu, Yangang Wang, Yebin Liu, and Qionghai Dai. 2017. Robust Non-rigid Motion Tracking and Surface Reconstruction Using L0 Regularization. *IEEE transactions on visualization and computer graphics* (2017).
[12] Samuel W Hasinoff, Anat Levin, Philip R Goode, and William T Freeman. 2011. Diffuse reflectance imaging with astronomical applications. In *ICCV*. IEEE, 185–192.
[13] James Imber, Jean-Yves Guillemaut, and Adrian Hilton. 2014. Intrinsic textures for relightable free-viewpoint video. In *ECCV*. Springer, 392–407.
[14] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. 2016. VolumeDeform: Real-time Volumetric Non-rigid Reconstruction. In *ECCV*.
[15] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. 2015. Panoptic Studio: A Massively Multiview System for Social Motion Capture. In *ICCV*. 3334–3342.
[16] Guannan Li, Chenglei Wu, Carsten Stoll, Yebin Liu, Kiran Varanasi, Qionghai Dai, and Christian Theobalt. 2013. Capturing relightable human performances under general uncontrolled illumination. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 275–284.
[17] Hao Li, Bart Adams, Leonidas J Guibas, and Mark Pauly. 2009. Robust single-view geometry and motion reconstruction. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 175.
[18] Hao Li, Etienne Vouga, Anton Gudym, Linjie Luo, Jonathan T Barron, and Gleb Gusev. 2013. 3D self-portraits. *ACM Trans. Graph.* 32, 6 (2013), 187.
[19] Miao Liao, Qing Zhang, Huamin Wang, Ruigang Yang, and Minglun Gong. 2009. Modeling deformable objects from a single depth camera. In *ICCV*.
[20] Yebin Liu, Qionghai Dai, and Wenli Xu. 2010. A point-cloud-based multiview stereo algorithm for free-viewpoint video. *IEEE transactions on visualization and computer graphics* 16, 3 (2010), 407–418.
[21] Yebin Liu, Juergen Gall, Carsten Stoll, Qionghai Dai, Hans-Peter Seidel, and Christian Theobalt. 2013. Markerless motion capture of multiple characters using multiview image segmentation. *IEEE transactions on pattern analysis and machine intelligence* 35, 11 (2013), 2720–2735.
[22] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. 1994. *A mathematical introduction to robotic manipulation.* CRC press.
[23] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. 2015. DynamicFusion: Reconstruction and Tracking of Non-Rigid Scenes in Real-Time. In *CVPR*.
[24] Björn Nutti, Åsa Kronander, Mattias Nilsing, Kristofer Maad, Cristina Svensson, and Hao Li. 2014. Depth Sensor-Based Realtime Tumor Tracking for Accurate Radiation Therapy.. In *Eurographics (Short Papers)*. Citeseer, 1–4.
[25] Roy Or-El, Guy Rosman, Aaron Wetzler, Ron Kimmel, and Alfred M Bruckstein. 2015. Rgbd-fusion: Real-time high precision depth recovery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5407–5416.
[26] Ravi Ramamoorthi and Pat Hanrahan. 2001. A signal-processing framework for inverse rendering. In *SIGGRAPH*. ACM, 117–128.
[27] Szymon Rusinkiewicz and Marc Levoy. 2001. Efficient variants of the ICP algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. IEEE, 145–152.
[28] Andrei Sharf, Dan A Alcantara, Thomas Lewiner, Chen Greif, Alla Sheffer, Nina Amenta, and Daniel Cohen-Or. 2008. Space-time surface reconstruction using incompressible flow. In *ACM Transactions on Graphics (TOG)*, Vol. 27. ACM, 110.
[29] Jonathan Starck and Adrian Hilton. 2007. Surface capture for performance-based animation. *Computer Graphics and Applications* 27, 3 (2007), 21–31.
[30] Sima Taheri, Aswin C Sankaranarayanan, and Rama Chellappa. 2013. Joint albedo estimation and pose tracking from video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35, 7 (2013), 1674–1689.
[31] Christian Theobalt, Naveed Ahmed, Hendrik Lensch, Marcus Magnor, and Hans-Peter Seidel. 2007. Seeing people in different light-joint shape, motion, and

reflectance capture. *Visualization and Computer Graphics, IEEE Transactions on* 13, 4 (2007), 663–674.

[32] Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. 2015. Real-time expression transfer for facial reenactment. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 183.

[33] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. 2008. Articulated mesh animation from multi-view silhouettes. In *ACM Transactions on Graphics (TOG)*, Vol. 27. ACM, 97.

[34] Daniel Vlasic, Pieter Peers, Ilya Baran, Paul Debevec, Jovan Popović, Szymon Rusinkiewicz, and Wojciech Matusik. 2009. Dynamic shape capture using multi-view photometric stereo. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 174.

[35] Michael Wand, Bart Adams, Maksim Ovsjanikov, Alexander Berner, Martin Bokeloh, Philipp Jenke, Leonidas Guibas, Hans-Peter Seidel, and Andreas Schilling. 2009. Efficient reconstruction of nonrigid shape and motion from real-time 3D scanner data. *ACM Transactions on Graphics (TOG)* 28, 2 (2009), 15.

[36] Daniel Weber, Jan Bender, Markus Schnoes, Andre Stork, and Dieter W. Fellner. 2013. Efficient GPU Data Structures and Methods to Solve Sparse Linear Systems in Dynamics Applications. *Computer Graphics Forum* (2013). DOI:https://doi.org/10.1111/j.1467-8659.2012.03227.x

[37] Chenglei Wu, Carsten Stoll, Levi Valgaerts, and Christian Theobalt. 2013. On-set performance capture of multiple actors with a stereo camera. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 161.

[38] Chenglei Wu, Kiran Varanasi, Yebin Liu, Hans-Peter Seidel, and Christian Theobalt. 2011. Shading-based dynamic shape refinement from multi-view video under general illumination. In *ICCV*. IEEE, 1108–1115.

[39] Chenglei Wu, Kiran Varanasi, and Christian Theobalt. 2012. Full body performance capture under uncontrolled and varying illumination: A shading-based approach. In *ECCV*. Springer, 757–770.

[40] Chenglei Wu, Michael Zollhöfer, Matthias Nießner, Marc Stamminger, Shahram Izadi, and Christian Theobalt. 2014. Real-time shading-based refinement for consumer depth cameras. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2014)* 33 (2014), 3.

[41] H. Wu, Z. Wang, and K. Zhou. 2015. Simultaneous Localization and Appearance Estimation with a Consumer RGB-D Camera. *IEEE Transactions on Visualization and Computer Graphics* PP, 99 (2015), 1–1. DOI:https://doi.org/10.1109/TVCG.2015.2498617

[42] Hongzhi Wu and Kun Zhou. 2015. AppFusion: Interactive Appearance Acquisition Using a Kinect Sensor. In *Computer Graphics Forum*, Vol. 34. 289–298.

[43] Zhe Wu, Sai-Kit Yeung, and Ping Tan. 2016. Towards Building an RGBD-M Scanner. *arXiv preprint arXiv:1603.03875* (2016).

[44] Genzhi Ye, Yebin Liu, Nils Hasler, Xiangyang Ji, Qionghai Dai, and Christian Theobalt. 2012. Performance capture of interacting characters with handheld kinects. In *ECCV*.

[45] Mao Ye and Ruigang Yang. 2014. Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera. In *CVPR*. IEEE, 2353–2360.

[46] Qing Zhang, Bo Fu, Mao Ye, and Ruigang Yang. 2014. Quality Dynamic Human Body Modeling Using a Single Low-cost Depth Camera. In *CVPR*. IEEE, 676–683.

[47] Michael Zollhöfer, Angela Dai, Matthias Innmann, Chenglei Wu, Marc Stamminger, Christian Theobalt, and Matthias Nießner. 2015. Shading-based Refinement on Volumetric Signed Distance Functions. *ACM Trans. Graph.* 34, 4, Article 96 (July 2015), 14 pages. DOI:https://doi.org/10.1145/2766887

[48] Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, and others. 2014. Real-time Non-rigid Reconstruction using an RGB-D Camera. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 156.

Received November 2016; revised February 2017; final version March 2017; accepted March 2017