SurfelWarp: Efficient Non-Volumetric Single View Dynamic Reconstruction

Wei Gao
Massachusetts Institute of Technology
Cambridge, Massachussets
weigao@mit.edu

Russ Tedrake
Massachusetts Institute of Technology
Cambridge, Massachussets
russt@mit.edu

***Abstract*—We contribute a dense SLAM system that takes a live stream of depth images as input and reconstructs non-rigid deforming scenes in real time, without templates or prior models. In contrast to existing approaches, we do not maintain any volumetric data structures, such as truncated signed distance function (TSDF) fields or deformation fields, which are performance and memory intensive. Our system works with a flat point (surfel) based representation of geometry, which can be directly acquired from commodity depth sensors. Standard graphics pipelines and general purpose GPU (GPGPU) computing are leveraged for all central operations: i.e., nearest neighbor maintenance, non-rigid deformation field estimation and fusion of depth measurements. Our pipeline inherently avoids expensive volumetric operations such as marching cubes, volumetric fusion and dense deformation field update, leading to significantly improved performance. Furthermore, the explicit and flexible surfel based geometry representation enables efficient tackling of topology changes and tracking failures, which makes our reconstructions consistent with updated depth observations. Our system allows robots maintain a scene description with non-rigidly deformed objects that potentially enables interactions with dynamic working environments. The video demo and source code are available on our project page.**

I. Introduction

The wide availability of commodity depth cameras provides robots with powerful but low-cost 3D sensing capabilities. However, measurements from depth sensors are noisy and incomplete, and they often contain numerous outliers. To address this issue, KinectFusion [13] and many related approaches [22, 21, 3, 10] estimate the camera pose and fuse depth frames online for a complete, smoothed 3D geometry scanning. In contrast to rigid scenes where the motion is encoded by a single 6 degree of freedom (DOF) camera pose, researchers also focus on the tracking and reconstruction of dynamic scenes [14, 16, 6, 4], which is more challenging but critically important to enable robots to interact with non-static working environments.

The problem of non-rigid reconstruction has been widely studied in recent years. To deal with the extraordinarily large deformation space, researchers have exploited carefully designed capture environments [20], well-controlled lighting conditions [2], and the use of many cameras for multi-view observations [8]. Some approaches also rely on motion priors such as templates [11] or embedded skeletons [17]. These approaches can achieve high quality and visually pleasing tracking and reconstruction results. However, special purpose capturing environments are not easy to setup and calibrate, and pre-scanned templates or skeletons require problem-specific initial alignments. These restrictions limit the applicability of these methods to typical robot tasks.

The work of DynamicFusion [14] represents a substantial step forward, which uses only a depth camera to acquire a live stream of depth images and fuse multiple frames by online non-rigid registration techniques. Later works extend this pipeline for improved robustness and capabilities: Innmann et al. [7] used sparse SIFT features and fine-scale volumetric deformation field to improve the non-rigid motion estimation; Guo et al. [6] exploited RGB frames to estimate the surface albedo and low-frequency lighting; Slavcheva et al. [18] proposed to directly align TSDF fields for motion estimation; Dou et al. [4] and Dou et al. [5] extended the framework to multi-view setups, used combined canonical and live TSDF volumes for better robustness, and achieved the performance capture of challenging scenes.

Despite diversities among these contributions, they all rely on volumetric data structures as the underlying representation of geometry (TSDF fields) and motion (nearest neighbor field for [14, 6, 4, 5] and deformation field for [18, 7]). As pointed out by Keller et al. [10], there is a quality and efficiency trade-off between volumetric and surfel based reconstructions: volumetric methods generate a smooth triangle mesh, but they are performance and memory intensive; surfel based methods are more efficient, but they need post-processing if mesh model is required. Compared with rigid SLAM, the computational burden is much more prominent for non-rigid scenes, which indicates surfel based representation is a promising alternative for online dynamic reconstructions.

In this paper, we propose a novel surfel based approach for real-time reconstruction of dynamic scenes, which requires no prior models or templates. As we will present, standard graphics pipelines and GPGPU computing can be leveraged for efficient implementation of all central operations: i.e., nearest-neighbor maintenance, correspondence association, non-rigid deformation estimation and fusion of depth measurements. The elimination of volumetric data structures avoids expensive volumetric operations such as marching cubes, volumetric fusion and dense deformation field updates, which leads to significantly improved performance. Moreover, the explicit surfel representation enables direct recovery from tracking failures and topology changes, which further improves the robustness of our pipeline.

II. Related Works

A. Dynamic Scene Reconstruction

Dynamic scene reconstruction typically involves the estimation of both the geometry and deformation field. Compared with high-quality offline reconstruction algorithms such as Collet et al. [2], online reconstruction methods are more suitable for robotic applications. Zollhöfer et al. [23] proposed to first perform a live static scanning, yielding a template for later online tracking. Newcombe et al. [14] combined the motion and geometry estimation into a unified framework, achieved the online reconstruction of dynamic scenes from a single depth camera. Many later works [7, 6, 4, 5, 18] improve the pipeline of DynamicFusion [14] with additional capabilities, as reviewed in Sec. I.

Compared with these contributions, the key distinction of our work is a highly efficient, non-volumetric pipeline for the non-rigid fusion of depth observations and the maintenance of deformation field. As we will demonstrate in the following text, the elimination of volumetric operations leads to better performance, less memory usage, and improved robustness.

B. Non-Rigid Tracking

Non-rigid tracking approaches typically used predefined geometry models, instead of estimating it online. For instance, Li et al. [11] and Cagniart et al. [1] exploited pre-scanned meshes as shape templates, and estimated deformation fields that align templates with incoming observations. It is also observed that many captured objects, for instance human bodies, hands and robots, contain articulate structures. Thus, incorporating skeletons into geometry templates can reduce the non-rigid deformation to joint space, results in significantly improved performance and robustness [17, 19].

Although non-rigid tracking pipelines with geometry templates or prior motion models have demonstrated impressive performance, they usually require either controlled capturing environments or careful initial alignments. These restrictions limit their applicability.

C. Surfel-Based 3D Reconstruction

In Pfister et al. [15], a surfel is formally defined as a zero-dimensional n-tuple with shape and shade attributes that locally approximate an object surface. In Keller et al. [10], a surfel is associated with a 3D position, a normal orientation and a radius. Whelan et al. [22] also associates surfels with albedo information. The surfel-based online rigid SLAM system was first introduced in Keller et al. [10]. Whelan et al. [22] further improved this technique by surface albedo reconstruction, light-source estimation and online loop closure.

Conceptually, our contribution can be regarded as an extension of Keller et al. [10] which enables the tracking and reconstruction of dynamic scenes. To achieve this goal, we redesign the pipeline of Keller et al. [10] for data fusion and deformation estimation of dynamic objects.

III. Preliminaries

In this section, we present symbols, definitions and other notations used in this paper. A consumer depth camera such as Kinect or Xtion is used to record a depth image sequence $\{\mathcal{D}^t\}$, where t is the frame label. A vertex map $\mathcal{V}$ is an image where each pixel records a 3-dimensional (3D) position; similarly, a normal map $\mathcal{N}$ is an image where each pixel records a normal orientation.

We follow DynamicFusion [14] to represent the deformation field $\mathcal{W}$ as a node graph $\mathcal{G}$ [11], which is sparse but efficient compared to the volumetric deformation field [7]. More specifically, $\mathcal{W} = \{[p_j \in R^3, \sigma_j \in R^+, T_j \in SE(3)]\}$, where j is the node index, p_j is the position of the jth node, σ_j is a radius parameter, and T_j is the 6 DOF transformation defined on jth node. T_j is represented by DualQuaternion [9] $\hat{q}_j$ for better interpolation. For a 3D point x, the deformation by $\mathcal{W}$ at x can be interpolated as

$$\mathcal{W}(x) = \text{normalized}(\Sigma_{k \in N(x)} w_k(x) \hat{q}_k) \tag{1}$$

where $N(x)$ is the nearest neighbor set of x, and the weight $w_k(x)$ can be computed as $w_k(x) = \exp(-||x - p_k||_2^2 / (2\sigma_k^2))$.

A surfel s is a tuple composed of position $v \in R^3$, normal $n \in R^3$, radius $r \in R^+$, confidence $c \in R$, the initialization time $t_{\text{init}} \in N$ and the most recent observed time $t_{\text{observed}} \in N$. Upper case S is used to denote the array of surfels and $S[i]$ is the ith surfel in this array. A surfel can be deformed by the deformation field $\mathcal{W}$ by Equ. 1,

$$\begin{aligned} v_{\text{live}} &= \mathcal{W}(v_{\text{ref}}) v_{\text{ref}} \\ n_{\text{live}} &= \text{rotation}(\mathcal{W}(v_{\text{ref}})) n_{\text{ref}} \end{aligned} \tag{2}$$

where v_{live} and n_{live} are the deformed vertex position and normal, v_{ref} and n_{ref} are the vertex position and normal before deformation. Other attributes, such as radius, time and confidence, remain the same after deformation.

IV. Overview

As shown in Fig. 1, our system runs in a frame-by-frame manner to process an input depth stream. Upon receiving a new depth image, we first solve a deformation field that aligns the reference geometry with the current depth observation. This is performed by first initializing the deformation field from previous frame, followed by a iterative closest point (ICP) based optimization similar to Newcombe et al. [14]. Once the deformation field has been updated, we perform data fusion to accumulate current depth observations into a global geometry model.

Unlike previous approaches, we adopt an efficient surfel-based representation of geometry and deformation field throughout our approach. Our system maintains two arrays of surfels, one in the reference frame S_{ref} and the other in the live frame S_{live}. The warp field $\mathcal{W}$ is defined in the reference frame, i.e., nodes p_j in the warp field $\mathcal{W}$ are sub-sampled from S_{ref} and $\mathcal{W}$ warps $S_{\text{ref}}[i]$ to $S_{\text{live}}[i]$. The nearest neighbor

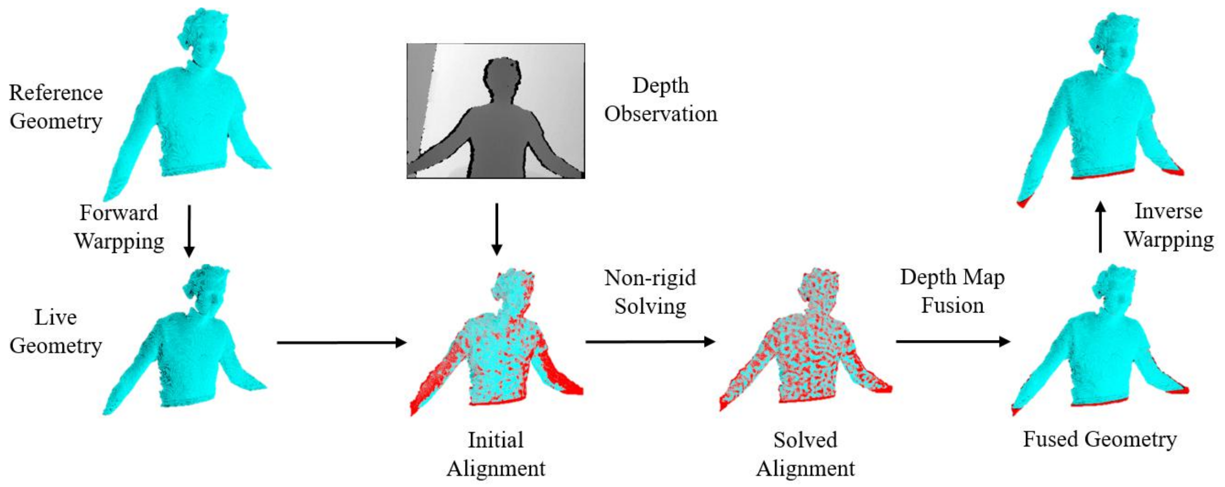

Fig. 1. An overview of our pipeline. Vertices from the depth image are in red, while vertices from the accumulated global model are in blue. Upon receiving a depth image, the system first solves a deformation field that aligns the reference geometry with the depth observation. Then the depth observation is fused with the global model, and previously unseen depth surfels are appended to the global model to complete the geometry model.

set $N(s_i)$ and associated weights $\{w(s_i)\}$ are maintained and shared for $S_{\text{ref}}[i]$ and $S_{\text{live}}[i]$. An important observation is: given appropriately maintained nearest neighbor sets and weights, we can define *inverse warping* as

$$\begin{aligned} v_{\text{ref}} &= \mathcal{W}(v_{\text{ref}})^{-1} v_{\text{live}} \\ n_{\text{ref}} &= \text{rotation}(\mathcal{W}(v_{\text{ref}}))^{-1} n_{\text{live}} \end{aligned} \tag{3}$$

where the SE(3) deformation $\mathcal{W}(v_{\text{ref}})$ can be computed by querying $N(s_{\text{ref}})$ and $\{w(s_{\text{ref}})\}$. Thus, unlike previous volumetric approaches that perform the geometry update in the reference frame, our pipeline updates geometry models in the live frame and warps them back to the reference frame.

$S_{\text{ref}}[i]$ shares radius r, confidence c, time stamps t_{init} and t_{observed} with $S_{\text{live}}[i]$, as they are invariant during warping.

V. Depth Image Processing

Let $\mathbf{u} = (x, y)^T \in R^2$ denote the pixel coordinate. At each pixel $\mathbf{u}$, we will compute a surfel s_{depth} defined in Sec. III. We first transform the depth value $\mathcal{D}(\mathbf{u})$ to the position $\mathcal{V}(\mathbf{u})$ of the surfel s_{depth} by $\mathcal{V}(\mathbf{u}) = \mathcal{D}(\mathbf{u})K^{-1}(\mathbf{u}^T, 1)^T$, where K is the intrinsic matrix of the depth camera. Then the normal at this pixel $\mathcal{N}(\mathbf{u})$ is estimated from the vertex map $\mathcal{V}$ using central difference. The confidence of this surfel s_{depth} is computed as $c = \exp(-\gamma^2/(2\phi^2))$, where γ is the normalized radial distance of the current depth measurement from the camera center, and $\phi = 0.6$ in accordance with Keller et al. [10]. The time stamps t_{init} and t_{observed} are initialized to be current frame t. Similar to Whelan et al. [22], the radius r of the surfel s_{depth} is computed as

$$r = \frac{\sqrt{2}d}{f|n_z|} \tag{4}$$

where $d = \mathcal{D}(\mathbf{u})$ is the depth value, f is the focal length of the camera, n_z is the z component of the normal expressed in camera frame. To prevent arbitrarily large surfels from oblique views, we follow Keller et al. [10] to clamp radii for grazing observations exceeding 75^o.

VI. Depth Map Fusion and Warp Field Update

Assuming an already solved deformation field (the solving method will be described in Sec. VII), our depth map fusion and warp field update pipeline first performs data fusion in the live frame, then warps the live surfels back to the reference frame, after which the warp field is updated base on newly observed reference surfels, as shown in Fig. 1.

A. *Fusion at the Live Frame*

After solving the deformation field, we first perform a forward warp on reference surfel array S_{ref} to obtain the live surfel array S_{live} that is aligned with the depth observation. Then, methods similar to Keller et al. [10] are used to fuse the depth map with live surfels. Specifically, we render the live surfel array S_{live} into a *Index Map* $\mathcal{I}$: given camera intrinsic K and the estimated camera pose T_{camera}, each live surfel $S_{\text{live}}[k]$ is projected into the image plane of current camera view, where the respective point index k is stored. Surfels are rendered as unsized points in index map $\mathcal{I}$, i.e., each surfel can be projected to at most one pixel. The index map is super-sampled over the depth map to avoid nearby surfels projecting to the same pixel. We use a 4×4 super-sampled index map in our implementation.

For each pixel $\mathbf{u}$ in the depth map $\mathcal{D}$, we identify at most one live surfel s_{live} that is in correspondence with the depth observation at $\mathbf{u}$ by a 4×4 window search centered at $\mathbf{u}$ on index map $\mathcal{I}$ (assuming suitable coordinate transform from $\mathcal{D}$ to $\mathcal{I}$). The criterion to find the corresponded surfel are:

- Discard surfels whose distance to the depth vertex are larger than δ_{distance}.
- Discard surfels that the normal is not aligned with the depth normal, i.e., $\text{dot}(n_{\text{depth}}, n_{\text{surfel}}) < \delta_{\text{normal}}$.

- For remaining surfels, select the one with highest confidence.
- If multiple such surfels exist, select the one which is closest to s_{depth}.

If a corresponded surfel s_{live} is found, the depth surfel s_{depth} at $\mathbf{u}$ is fused into s_{live} by:

$$\begin{aligned} c_{\text{live\_new}} &= c_{\text{live\_old}} + c_{\text{depth}} \\ t_{\text{observed}} &= t_{\text{now}} \\ v_{\text{live\_new}} &= (c_{\text{live\_old}} v_{\text{live\_old}} + c_{\text{depth}} v_{\text{depth}}) / c_{\text{live\_new}} \end{aligned} \tag{5}$$

$$\begin{aligned} n_{\text{live\_new}} &= (c_{\text{live\_old}} n_{\text{live\_old}} + c_{\text{depth}} n_{\text{depth}}) / c_{\text{live\_new}} \\ r_{\text{live\_new}} &= (c_{\text{live\_old}} r_{\text{live\_old}} + c_{\text{depth}} r_{\text{depth}}) / c_{\text{live\_new}} \end{aligned}$$

where v, n, c, t and r are vertex, normal, confidence, recent observed stamp and radius associated with surfels; the depth surfel s_{depth} is computed as described in Sec. V.

B. *Live Surfel Skinning*

In Sec. VI-A, for any depth pixel $\mathbf{u}$, if no corresponding live surfel s_{live} is found, then this depth surfel s_{depth} would potentially be appended to the live surfel array S_{live} to complete the geometry (The appending pipeline will be described in section. VI-D). As mentioned in Sec. IV, it is required to compute the nearest neighbor set $N(s_{\text{depth}})$ and weights for this surfel to warp s_{depth} back to the reference frame. However, the node graph $\mathcal{G}$ is defined in the reference frame, computing the nearest neighbor set $N(s_{\text{depth}})$ in the reference frame before knowing the warp-back position of s_{depth} is not feasible.

One way to resolve this paradox is to first warp the node graph $\mathcal{G}$ to the live frame $\hat{\mathcal{G}}$, then perform skinning at the live frame. However, at open-to-close topology changes, surfel s_{depth} might be skinned to inconsistent nodes: in Fig. 6 the surfels on the hand might be skinned to nodes on the face when they come close. To solve this problem, we propose the following pipeline:

- Compute an initial nearest neighbor set $N(s_{\text{depth}})$ for s_{depth} using node graph $\hat{\mathcal{G}}$ at the live frame. Let $node_{\text{0\_live}} \in N(s_{\text{depth}})$ denote the node which is closest to s_{depth} at the live frame.
- For any $node_{\text{i\_live}} \in N(s_{\text{depth}}), i \neq 0$, if

$$1 - \epsilon < \frac{|node_{\text{i\_live}} - node_{\text{0\_live}}|}{|node_{\text{i\_ref}} - node_{\text{0\_ref}}|} < 1 + \epsilon \tag{6}$$

  then keep $node_{\text{i\_live}}$ in $N(s_{\text{depth}})$, else remove this node.

In Equ. 6, the threshold ϵ measures the extent of intrinsic deformation. The above criteria ensure the deformation of s_{depth} is controlled by a consistent set of nodes, i.e., nodes in $N(s_{\text{depth}})$ are close in the reference frame and have smooth SE(3) deformation values.

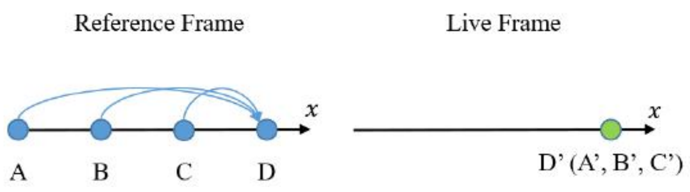

Fig. 2. A one dimensional illustration of the compressive warp field. A compressive warp field on segment AD maps point A, B, C, and D in the reference frame to the same position. From the observation at D' in the live frame, it is impossible to infer the position at the reference frame.

C. *Resolving Compressive Warp Field*

Another problem that arises from open-to-close topology changes is the compressive warp field, or colliding voxels in volumetric approaches [6, 4]. A one dimensional illustration is presented in Fig. 2. Suppose a compressive warp field on segment AD maps points A, B, C and D at the reference frame to the same live frame position, it is impossible to infer the reference frame position from the live observation at D'. Thus, directly performing data fusion at open-to-close topology changes will generate erroneous surfaces, at shown in Fig. 6.

Our solution is to bound the sensitivity of inverse warping at the live frame. Using the one dimensional illustration in Fig. 2, we discard live surfels at which

$$|\frac{\mathrm{d}x_{\text{ref}}}{\mathrm{d}x_{\text{live}}}| > 1 + \epsilon \tag{7}$$

where the threshold ϵ is defined in Equ. 6. In 3D case, the gradient in Equ. 7 becomes a 3×3 strain tensor and is computed by finite difference. The reference frame position is computed by inverse warping defined in Sec. IV and skinning method in Sec. VI-B.

D. *Surfel Appending and Warp Field Update*

We use methods similar to Keller et al. [10] for appending depth surfels s_{depth} to S_{live} that do not correspond with any model surfels, with two additional criteria:

- Discard surfels that the sum of nearest neighbor weights is less than δ_{nn}. These surfels are far away from nodes and are likely to be outliers.
- Discard surfels believed to be compressively mapped according to Sec. VI-C.

Existing surfels s_{live} will also be discarded if:

- The surfel remains unstable for a long time, i.e., its confidence c is less than a confidence threshold δ_{stable} for a period $t_{\text{low\_confid}}$ after its initialization time t_{init}.
- It is very similar to its neighbors on the index map $\mathcal{I}$.

The formal criterion and detailed methods can be found in Keller et al. [10]. After fully updating S_{live}, an inverse warping defined in Sec. IV is performed to update S_{ref} from S_{live}.

The warp field extending pipeline introduced in DynamicFusion [14] is performed on the appended reference frame surfels $S_{\text{ref\_append}}$, to update the node graph $\mathcal{G}$ and the warp field $\mathcal{W}$.

Intuitively, the extent which the new geometry $S_{\text{ref\_append}}$ is *covered* by the current node graph $\mathcal{G}$ is computed. Then the set of uncovered vertices are sub-sampled and appended to the node graph $\mathcal{G}$. Finally, the edges in the node graph are recomputed. The formal description and detailed methods can be found in Sec. 3.4 of DynamicFusion [14].

The nearest neighbors and weights of surfels are also updated base on the new node graph $\mathcal{G}$. For each surfel s_{ref} in S_{ref}, its nearest neighbor set $N(s_{\text{ref}})$ is compared with appended nodes and updated if required. The number of appended nodes is usually no more than 10, thus a brute-force search over appended nodes is sufficient.

E. Initialization

Upon receiving the first depth image $\mathcal{D}^0$, we first compute the depth surfels according to Sec. V. Then, surfels corresponded to valid depth pixels are collected into both S_{ref} and S_{live}, using GPU selection. The nodes in the warp field $\mathcal{W}$ are initialized by sub-sampling S_{ref}. The edges in node graph $\mathcal{G}$ are computed according to Newcombe et al. [14], and the SE(3) deformation values of nodes are initialized to identity. After the initialization of the warp field $\mathcal{W}$, the skinning of surfels is performed: the nearest neighbors and weights of each surfel are computed base on their reference frame positions and nodes in $\mathcal{W}$.

F. Model Reinitialization

As mentioned in Dou et al. [4], it is impossible to interpret all motions from a single reference, and incorrect to assume the non-rigid tracker never fails. Thus, Dou et al. [4] used two TSDF volumes, one for the reference frame and another for the live frame, and reset the reference volume by the live volume to handle large motions, tracking failures and topology changes. However, maintaining an additional TSDF volume and the associated nearest neighbor field as in [4] inevitably increases computation cost and memory usage.

Our pipeline naturally supports similar ideas in a more compact, efficient way: inferring erroneous surfels from depth observations is more explicit and easier than incorrect TSDF values. We discard any live surfel s_{live} that should be visible from current camera view but do not have corresponded depth observations. The correspondence is identified by projecting the live surfel s_{live} into the depth image and performing a window search. After cleaning S_{live}, we reset S_{ref} to S_{live} and initialize the warp field on it.

In our implementation, the model reinitialization is invoked by the misalignment energy defined in Equ. 8 and the number of appended surfels. When large misalignment energy values and lots of appended surfels are observed for several continuous frames, it is likely that the non-rigid tracker is struggling with incoming depth observations.

G. Complexity

In this subsection, we will analyze the complexity of all the operations that are introduced in this section.

Let $|V|$ denote the size of an array V, $|\mathcal{W}|$ denote the number of nodes in the warp field, and S_{append} denote the array of appended surfel candidates. The data fusion (Sec. VI-A), inverse warping and nearest neighbor array updating (Sec. VI-D) are in the complexity of $O(|S_{\text{live}}|)$. The live surfel skinning (Sec. VI-B) and compression resolving (Sec. VI-C) are in the complexity of $O(|S_{\text{append}}|)$. An $O(|S_{\text{live}}| + |S_{\text{append}}|)$ GPU selection is performed to compact the surfel array. It is also noted that all these operations can be trivially GPU parallelized.

In the initialization (Sec. VI-E) and the model reinitialization (Sec. VI-F), the number of nodes $|\mathcal{W}|$ in surfel skinning is large enough to benefit from structured search algorithms like Muja and Lowe [12]. We choose to build the search index on CPU and perform approximate nearest neighbor query on GPU, where the complexity is approximately $O(|S_{\text{ref}}|\log(|\mathcal{W}|))$. Although divergent executions in the GPU parallel querying may weaken the performance, the (re)initialization is typically efficient (i.e., 3-5 ms) in our pipeline because of the compact sized S_{ref}.

From our observation, S_{live} contains about 3×10^5 surfels. S_{append} typically has no more than 5000 surfels, i.e., less than $\frac{1}{20}$ of valid depth pixels. Thus, sophisticated operations such as live surfel skinning and compression resolving are performed only on a small fraction of pixels, which further ensures the efficiency of our pipeline.

As a comparison, in volumetric approaches such as [14, 6, 4], the data fusion, forward voxel warping, marching cubes, update volumetric nearest-neighbor field and collision resolving are all in the complexity of $O(\text{number of voxels})$, which is usually one or two orders larger than $|S_{\text{live}}|$. Moreover, in the initialization frame, there is typically 300-1000 nodes in the warp field $\mathcal{W}$, which incurs a substantial computational burden to estimate the skinning of each voxel. From the analysis, our method is much more efficient than volumetric approaches in terms of performance and memory usage.

VII. Non-Rigid Warp Field Estimation

The SE(3) deformations $T_j \in \mathcal{W}$ are estimated given a new depth observation $\mathcal{D}$, the previous deformation field $\mathcal{W}^{\text{prev}}$, and geometry models S_{ref} and S_{live} (S_{live} is deformed from S_{ref} according to $\mathcal{W}^{\text{prev}}$). To perform this estimation, we first predict the visibility of live surfels S_{live}, then follow DynamicFusion [14] to cast the deformation estimation into an optimization problem.

The visibility prediction is similar to the approach of Keller et al. [10]: we render the live surfels S_{live} as overlapping, disk-shaped *surface splats* that are spanned by the position v_{live}, normal n_{live} and radius r_{live} of the live surfel s_{live}. Although advanced rendering techniques such as Zwicker et al. [24] are available, we simply render opaque splats for optimal performance.

Different from rigid SLAM pipelines such as [10, 22], we also render an index map $\mathcal{I}$ at the same resolution of depth images for efficient querying of nearest neighbors and weights. Moreover, to improve the robustness of the warp field estimation to outliers in S_{live}, we only render stable surfels, i.e., surfels with confidence c larger than δ_{stable} defined

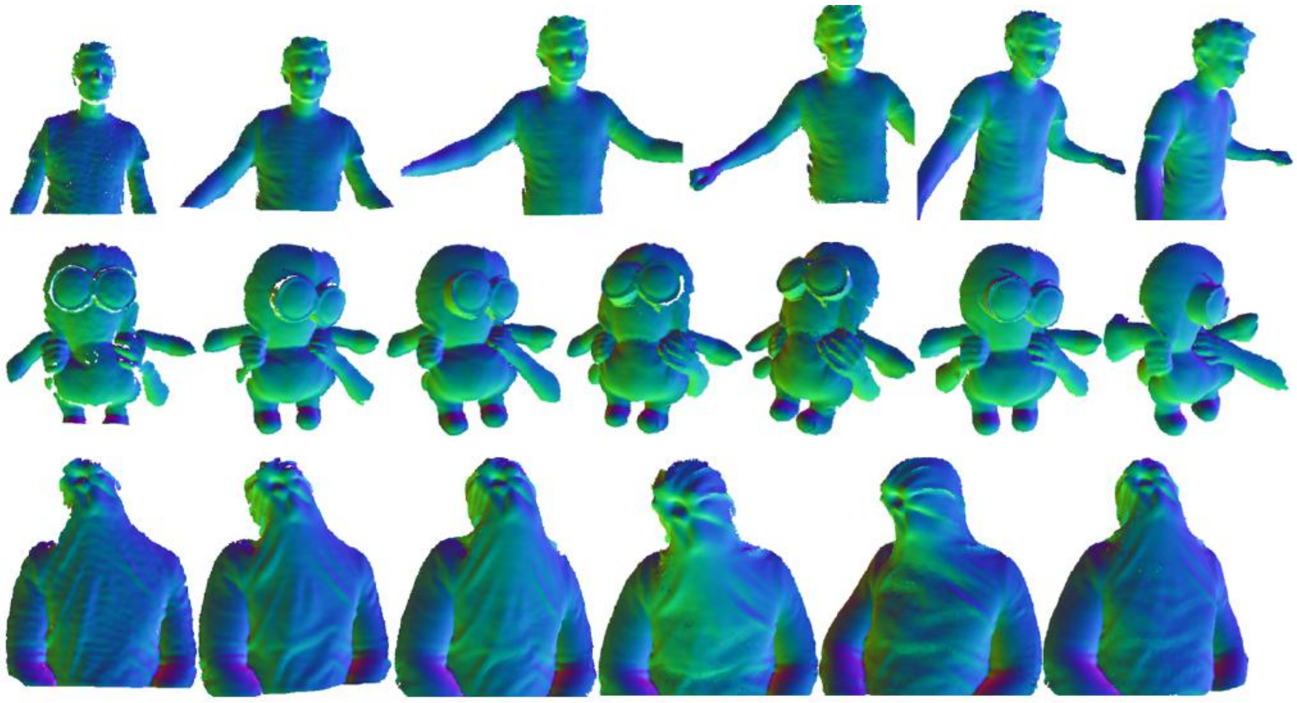

Fig. 3. Dynamic scenes reconstructed by our algorithm. Sequences are from left to right. From top to bottom: upperbody, minion and hoodie. Readers are recommended to watch the accompanying video demo on our project page for more results.

in Sec. VI-D. For the first few frames or frames just after model reinitialization, we also render surfels that are observed recently, i.e., surfels such that $t_{\text{now}} - t_{\text{observed}} \leq \delta_{\text{recent}}$, where t_{now} is current time and t_{observed} is the last observed time of this surfel.

Following the approach of DynamicFusion [14], we solve the following optimization problem to estimate SE(3) deformations $T_j \in \mathcal{W}$:

$$E_{\text{total}}(\mathcal{W}) = E_{\text{depth}} + \lambda E_{\text{regulation}} \tag{8}$$

where E_{depth} is the data term that constrains the deformation to be consistent with the depth input $\mathcal{D}$, $E_{\text{regulation}}$ regularizes the motion to be as rigid as possible, λ is a constant value to balance between two energy terms.

E_{depth} is a point to plane energy term [14]:

$$E_{\text{depth}}(\mathcal{W}) = \Sigma_{(s_{\text{model}}, s_{\text{depth}}) \in P} (n_{\text{depth}}^T (v_{\text{model}} - v_{\text{depth}}))^2 \tag{9}$$

where P is a set of corresponded model and depth surfel pairs, n and v are the normal and the vertex associated with the surfel. The method to find P given the depth observation $\mathcal{D}$ and the rendered live geometry can be found in Guo et al. [6].

$E_{\text{regulation}}$ is an as-rigid-as-possible regularizer:

$$E_{\text{regulation}}(\mathcal{W}) = \Sigma_{j \in \mathcal{G}} \Sigma_{i \in N_j} ||T_j p_j - T_i p_j||_2^2 \tag{10}$$

where N_j is the set of neighboring nodes of the jth node in the node graph. This term ensures the non-visible parts of the geometry model will move with visible regions as rigidly as possible. Also, this term avoids arbitrarily deformed geometry to fit the noisy depth inputs.

The optimization problem Equ. 8 is a nonlinear least squares problem and solved by Gaussian-Newton algorithm implemented on GPU. Before the non-rigid estimation, we first perform a rigid alignment using the method in KinectFusion [13].

VIII. Implementation Details

Our pipeline consists of three major components: the depth image processor, the non-rigid solver and the geometry update module. The depth image processor fetches depth images and computes depth surfels as described in Sec. V. We follow Dou et al. [4] to implement the non-rigid solver: first construct the matrix $J^T J$, then solve the linear system $J^T J x = J^T e$, where J and e are the Jacobian and residual for least square terms in Equ. 8. We constrain the maximum number of Gauss-Newton iterations to be 10, while the solver typically converges in 3 or 4 iterations. For the geometry update module, the pipeline parallelizes operations in Sec. VI over either the array of surfels S_{live} or depth surfels from the depth image processor.

On a single Titan Xp GPU, the time to process one depth image is usually about 2 ms, and the geometry update typically takes 2-3 ms per frame. Thus, the overall performance is primarily determined by the non-rigid solver, which varies a lot with different dynamic scenes. Currently, our implementation is single threaded, and the performance can be further improved by the thread-level parallelism used in Guo et al. [6] or the stream mechanism used in Dou et al. [5].

Table. 1. Parameters for Reconstruction Results

Name	Value	Description
σ	2.5 cm	Node sampling distance from [14]
λ	5	Regulation constant in Equ. 8
δ_{distance}	1 mm	Distance threshold in Sec. VI-A
δ_{normal}	0.85	Dot-product threshold in Sec. VI-A
ϵ	0.2	Intrinsic deformation bound in Sec. VI-B
δ_{stable}	10	Stable confidence threshold in Sec. VI-D
$t_{\text{low\_confid}}$	30	Maximum unstable period in Sec. VI-D
δ_{recent}	2	Recent observed threshold in Sec. VII

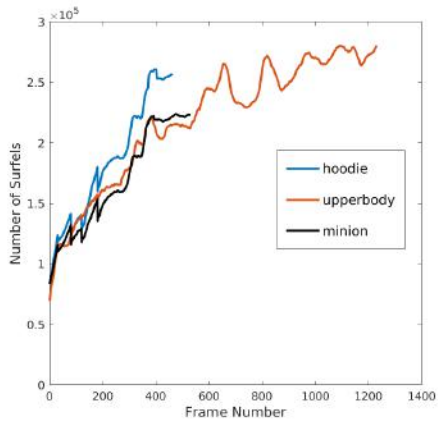

Fig. 4. The number of surfels plotted over depth frame for different reconstructions. It is noted that each depth frame observes about 10^5 valid surfels. Our technique keeps a compact representation of the scene geometry.

IX. Results

In this section, we present a variety of dynamic scenes reconstructed by our technique. To demonstrate the improved performance and robustness, we compare our approach with our implementation of DynamicFusion [14], which uses the same non-rigid tracker but volumetric representations of geometry and nearest neighbor field. Parameters used for all reconstructions presented in this section are summarized in Table. 1. The depth streams recorded by Innmann et al. [7] and Slavcheva et al. [18] are used in the presented results. The video demo and source code are available on our project page

Fig. 3 shows the reconstruction of "upperbody", "minion" and "hoodie" datasets by our techniques. The live geometries are rendered as normal maps. Readers are recommended to watch the accompanying video for more reconstruction results.

Fig. 4 shows the number of surfels remains roughly constant after objects are fully observed. It is noted that each depth frame provides about 10^5 valid surfels in these examples. This demonstrates new surfels are not continuously added and the global model is refined but kept compact.

Fig. 5 shows the performance comparison between our method and DynamicFusion [14] (our implementation) on "upperbody" dataset. The hardware platform for this comparison is a Nvidia Titan Xp GPU. Note that both methods share the same implementation of the non-rigid tracker. The process time result is the average of over 1200 frames. Due to the non-volumetric pipeline, our method represents the geometry and motion more compactly and demonstrates a significant speedup.

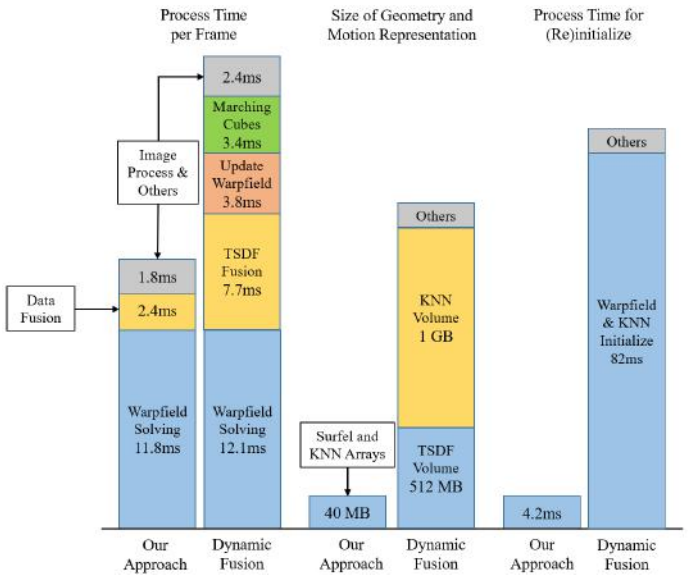

Fig. 5. Performance comparison between our approach and DynamicFusion [14] (our implementation) on "upperbody" dataset, both methods share the same non-rigid tracker. Benefiting from the non-volumetric pipeline, our approach represents the geometry and motion more compactly and achieves a significant speedup.

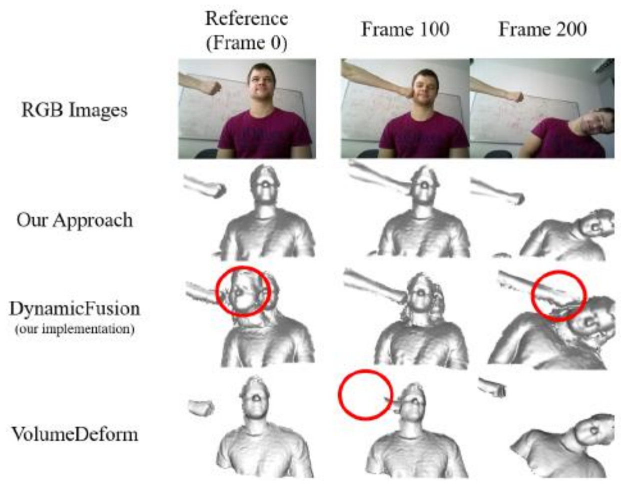

Fig. 6. Comparison of our approach with DynamicFusion [14] and VolumeDeform [7]. This sequence contains a open-to-close topology change. Our approach correctly resolves the compressive deformation field and yields a faithful reconstruction; DynamicFusion [14] cannot resolve compressive deformation and generates erroneous surfaces; the solution of VolumeDeform [7] almost disables the update of the geometry model and results in incomplete surfaces.

Fig. 6 compares our method with DynamicFusion [14] and VolumeDeform [7] on the "boxing" dataset. This sequence contains a open-to-close topology change. The result of VolumeDeform [7] is from the original author. From the figure, DynamicFusion [14] cannot correctly resolve compressive deformation and yields erroneous surfaces. VolumeDeform [7] almost disables the update of the geometry model and re-

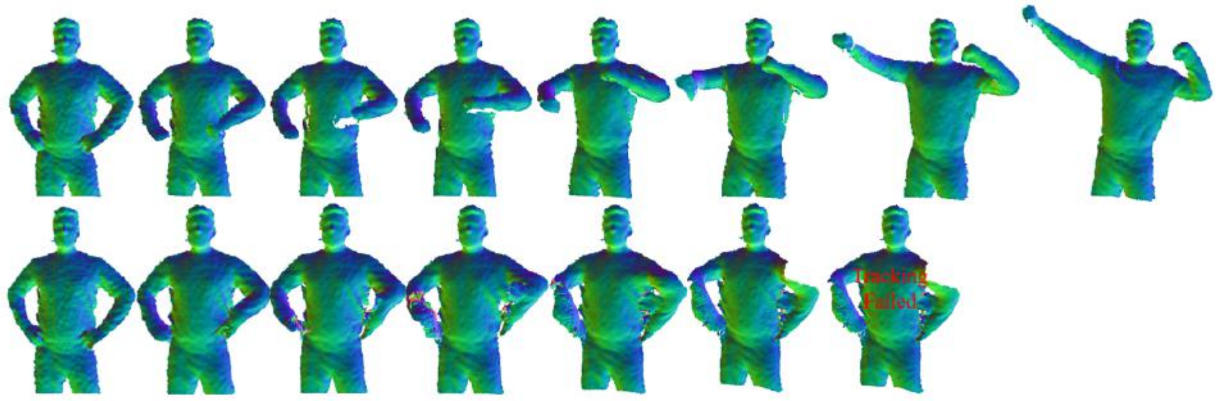

Fig. 7. Demonstration of using model reinitialization mechanism to handle close-to-open topology changes and fast motion. Top: the reconstruction by our approach. Bottom: the reconstruction by DynamicFusion [14] (our implementation).

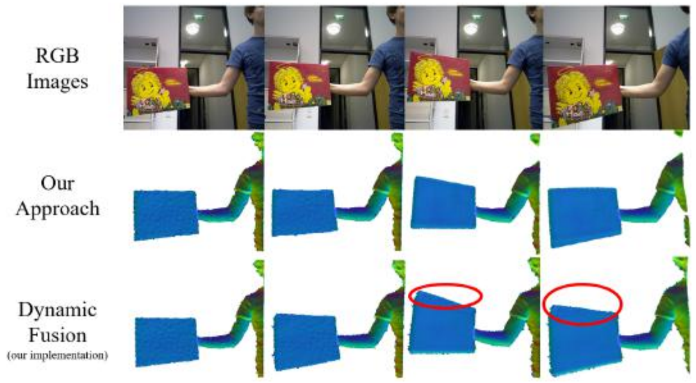

Fig. 8. Demonstration of using model reinitialization mechanism to clean incorrect surfaces. This sequence contains sequential motions that cannot be correctly solved by depth-only tracking. However, the model reinitialization removes incorrect surfels base on depth observations and enables our approach to correctly capture the geometry.

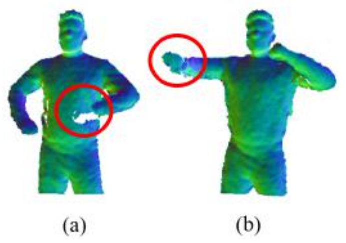

Fig. 9. Limitations of our approach: (a) incomplete geometry under topology changes; (b) tracking failures at tangential and fast motion.

sults in incomplete surfaces. Our method correctly resolves the compressive deformation field and generates a faithful reconstruction of this scene.

Fig. 7 and Fig. 8 demonstrate that the model reinitialization mechanism enables our technique to handle tracking failures and close-to-open topology changes. In Fig. 8, the sequence contains tangential motion that cannot be solved by the depth-only non-rigid tracker. In Fig. 7, motion is fast (performed in approximately 2 seconds) and contains a close-to-open topology change. Our approach successfully recoveries from these failures. In comparison, DynamicFusion [14] cannot correctly reconstruct the motion and geometry. For sequence in Fig. 8, the model reinitialization is invoked periodically alongside the criteria in Sec. VI-F.

X. Limitation and Future Work

Model reinitialization significantly improves the robustness of our pipeline. However, cleaning of erroneous surfels may lead to incomplete geometry, as shown in Fig. 9 (a). Currently, the non-rigid tracker in our pipeline uses the same energy terms as DynamicFusion [14], which struggles with tangential or fast motions, as shown in Fig. 9 (b). We plan to address these limitations by extending our non-rigid tracker with energy terms in Dou et al. [4], using adaptive node-graph refinement similar to Li et al. [11] and reconstructing the surface albedo as in Guo et al. [6].

XI. Conclusion

This paper presents a dense SLAM system that reconstructs both the motion and geometry from a single depth camera. Unlike previous approaches, our method is purely point-based, i.e., without resorting to volumetric data structures such as TSDF or nearest neighbor fields, which makes our system highly efficient. We demonstrate all central operations, such as the nearest neighbor maintenance, warp field estimation and data fusion, can be parallelized by leveraging standard graphics pipeline and GPGPU computing. The explicit, flexible surfel array representation of geometry also enables efficient recovery from topology changes and tracking failures, which further improves the robustness of our pipeline. Experimental comparison with existing methods demonstrates the effectiveness of our method.

Acknowledgments

The authors would like to acknowledge the support from NSF Award IIS1427050 and Amazon Research Award. The views expressed in this paper are those of the authors themselves and are not endorsed by the funding agencies.

References

[1] Cedric Cagniart, Edmond Boyer, and Slobodan Ilic. Free-form mesh tracking: a patch-based approach. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1339–1346. IEEE, 2010.

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

[3] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)*, 36(3):24, 2017.

[4] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)*, 35(4):114, 2016.

[5] Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. Motion2fusion: real-time volumetric performance capture. *ACM Transactions on Graphics (TOG)*, 36(6):246, 2017.

[6] Kaiwen Guo, Feng Xu, Tao Yu, Xiaoyang Liu, Qionghai Dai, and Yebin Liu. Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera. *ACM Transactions on Graphics (TOG)*, 36(3):32, 2017.

[7] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision*, pages 362–379. Springer, 2016.

[8] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3334–3342, 2015.

[9] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)*, 27(4):105, 2008.

[10] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *3DTV-Conference, 2013 International Conference on*, pages 1–8. IEEE, 2013.

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

[12] Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2227–2240, 2014.

[13] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

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

[15] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342. ACM Press/Addison-Wesley Publishing Co., 2000.

[16] Martin Rünz and Lourdes Agapito. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 4471–4478. IEEE, 2017.

[17] Tanner Schmidt, Richard A Newcombe, and Dieter Fox. Dart: Dense articulated real-time tracking. In *Robotics: Science and Systems*, 2014.

[18] Miroslava Slavcheva, Maximilian Baust, Daniel Cremers, and Slobodan Ilic. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, page 7, 2017.

[19] Andrea Tagliasacchi, Matthias Schröder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. Robust articulated-icp for real-time hand tracking. In *Computer Graphics Forum*, volume 34, pages 101–114. Wiley Online Library, 2015.

[20] Daniel Vlasic, Pieter Peers, Ilya Baran, Paul Debevec, Jovan Popović, Szymon Rusinkiewicz, and Wojciech Matusik. Dynamic shape capture using multi-view photometric stereo. *ACM Transactions on Graphics (TOG)*, 28(5):174, 2009.

[21] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice Fallon, John J Leonard, and John McDonald. Real-time large-scale dense rgb-d slam with volumetric fusion. *The International Journal of Robotics Research*, 34(4-5):598–626, 2015.

[22] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, 35(14): 1697–1716, 2016.

[23] Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Chris-

tian Theobalt, et al. Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (TOG)*, 33(4):156, 2014.

[24] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378. ACM, 2001.