Scene-level Tracking and Reconstruction without Object Priors

Haonan Chang and Abdeslam Boularias[1]

***Abstract*— We present the first real-time system capable of tracking and reconstructing, individually, every visible object in a given scene, without any form of prior on the rigidness of the objects, texture existence, or object category. In contrast with previous methods such as Co-Fusion and MaskFusion that first segment the scene into individual objects and then process each object independently, the proposed method dynamically segments the non-rigid scene as part of the tracking and reconstruction process. When new measurements indicate topology change, reconstructed models are updated in real-time to reflect that change. Our proposed system can provide the live geometry and deformation of all visible objects in a novel scene in real-time, which makes it possible to be integrated seamlessly into numerous existing robotics applications that rely on object models for grasping and manipulation. The capabilities of the proposed system are demonstrated in challenging scenes that contain multiple rigid and non-rigid objects. Supplementary material, including video, can be found at `https://github.com/changhaonan/STAR-no-prior`.**

I. INTRODUCTION

Robots are increasingly deployed in unstructured environments such as households, warehouses, and workshops. The large variety of object types, shapes, and textures that are encountered in such environments makes it virtually impossible for robots to always rely on prior models of the objects for grasping, dexterous manipulation, and motion planning. Moreover, visual demonstrations for teaching robots new tasks in real-world setups are often performed in complex scenes that contain a number of unknown and novel objects. The objects are often non-rigid and partially occluded during the demonstrations. Consequently, tracking and 3D reconstruction of previously unseen objects is an important component of any intelligent robotic system.

Thanks to the recent availability of low-price depth-sensing cameras, tracking and 3D reconstruction of objects became two popular topics in robot vision [1].

These two problems are however interrelated; reconstructed 3D object models facilitate subsequent tracking of the object, while object tracking is often essential for the reconstruction of its 3D model, which is highly challenging in the absence of object priors, as it is often the case. Therefore, these two problems need to be solved jointly in a unified framework.

The Scene-level Tracking and Reconstruction (**STAR**) problem brings the above challenge to a scene scale: Given a sequence of RGB-D images of an entire scene, can we reconstruct the geometry model and track the pose and deformation of each moving object in the scene simultaneously?

[1]Both authors are with the Department of Computer Science of Rutgers University, NJ, USA. This work is supported by NSF awards 1734492, 1846043, and 2132972.

One approach to solving this problem is to first segment the scene images into point clouds of individual objects and then treat them separately during tracking and reconstruction [2]–[4]. This approach however heavily relies on the existence of a pre-trained neural network (such as *MaskRCNN*) as a prior for segmentation, which limits its applicability in settings with entirely new types of objects.

Another approach is to reverse the order of segmentation and tracking, reconstruction. First, reconstruct the non-rigid scene geometry in its entirety, using all the RGB-D images of the scene. Then, segment the reconstructed 3D scene model into individual objects. A major challenge in the first step lies in handling topological changes that affect the feasibility of the second step. Topological changes occur for example when separate objects, or parts of an object, become physically connected and vice versa.

Topological changes in tracking and reconstruction of non-rigid objects are handled differently depending on the type of the geometry model that is adopted by the method, such as *TSDF* (Truncated Signed Distance Field) and *Surfels* (surface elements). For TSDF-based methods, some methods reset the entire model whenever a major topological change is detected [5], [6]. Other methods use the status of spatial compression to determine areas where topology changes and replace those areas [7], [8]. These methods however either rely on complex and ultra-precise sensors [5], [6], or depend on accurate texture-based registration [7], [8], which makes them difficult to use in general robotics applications. A recent work based on a *killing-field* approximation reduces the dependency on texture while handling large topological change, but its use of an SDF-based geometry makes object segmentation very difficult and non-trivial [9], [10]. Surfel-based models are more flexible for handling topology changes in non-rigid reconstruction [11], [12]. Instead of resetting the entire model whenever a major topology change is detected and thus losing important information, surfel-based methods can locally re-initialize the geometry around affected areas. This can be achieved because, unlike TSDF models, surfel models do not have any internal constraints, which makes the removal and appending of point clouds relatively easy. Furthermore, a surfel representation can be beneficial in geometric separation leading to object segmentation.

In this paper, we propose a novel system that can solve the scene-level tracking and reconstruction problem without any object or category prior, or any assumption regarding the rigidness or texture of the objects. The contributions of this work can be summarized as follows: **(1) A new surfel-based method for non-rigid scene reconstruction that can handle topology change.** Our proposed non-rigid geometry

reconstruction pipeline improves the multi-view SurfelWarp technique [12] by introducing a local re-initialization strategy, which is used in our approach to detect and handle topology changes. Compared to existing solutions to this problem, ours does not rely on texture-based registration or a category-dependent refiner. A comparison between our proposed pipeline and multi-view SurfelWarp [12] clearly illustrates the advantages of our approach. **(2) The first real-time scene-level, tracking and reconstruction solution that does not use any object prior.** Existing solutions such as MaskFusion require objects to be rigid and rely on pre-trained neural networks as priors for segmenting scenes into objects. To our best knowledge, our system is the first that solves this problem and returns an individual model for each moving object in the scene without any prior information about the objects, their rigidness, or texture. The proposed system is tested over a set of RGB-D images of challenging scenes, with various types of objects.

TABLE I: Properties of different real-time, dense-SLAM scene reconstruction systems.

Method	Category free	Dynamic scene	Non-rigid objects	Topology change	Segmentation	Texture free
SLAM++	✗	✗	✗	✗	✗	✗
DynamicFusion [13]	✓	✓	✓	✗	✗	✓
Volume Deform	✓	✓	✓	✗	✗	✗
SurfelWarp [12]	✓	✓	✓	✗	✗	✓
Fusion4D [5]	✓	✓	✓	✓	✗	✗
Motion2Fusion [6]	✓	✓	✓	✓	✗	✗
Functon4D [14]	✗	✓	✓	✓	✗	✓
TCAFusion [8]	✓	✓	✓	✓	✗	✗
Co-fusion [2]	✗	✓	✗	✓	✓	✓
MaskFusion [3]	✗	✓	✗	✓	✓	✓
RigidFusion [4]	✗	✓	✗	✓	✓	✓
Ours	✓	✓	✓	✓	✓	✓

II. RELATED WORKS

We discuss in the following some of the recent related techniques. Table I shows a taxonomy of these techniques.

Simultaneous tracking and reconstruction. Simultaneous tracking and reconstruction is an important problem in robotic manipulation, since manipulation planning and learning algorithms often require geometric models of the objects and their poses [2]. Unlike earlier works centered on single object tracking and reconstruction, more recent techniques focus on dealing with multiple objects simultaneously. Examples of such techniques include Co-fusion [2], MaskFusion [3] and RigidFusion [4]. These techniques require priors for segmenting the given scene initially into multiple objects and then tracking and reconstructing each object separately. Object priors are not always available, and initial segmentation errors can lead to significant tracking and reconstruction errors later. Moreover, these techniques assume that the objects are rigid.

Dynamic Scene Reconstruction. Dynamic scene reconstruction typically uses a static geometric model and a deformation field to describe a deformable object or a dynamic scene. Existing techniques require solving joint optimization problems, which are computationally expensive [15], [16]. DynamicFusion [13] is a parallel GPU-based solution that solves this problem more efficiently and that can be considered as the first online non-rigid reconstruction system. However, DynamicFusion cannot handle topological changes in measurements. More recent techniques such as Fusion4D [5], Motion2Fusion [6], and others [7], [8], try to solve this problem with texture-based, learning-based registration or global re-initialization. Some of these works achieved excellent reconstruction results. But to the best of our knowledge, all existing topology-aware non-rigid reconstruction methods either rely on texture-rich measurements or topology-prior for registration, or on category-level priors (such as a human or a pre-trained network) for model refinement. In contrast with existing solutions, we exploit an intriguing property of surfels that makes them easy to remove or append locally, to perform local re-initialization. Our proposed method makes no assumption on the existence of texture or category-level prior to the target scene.

Surfel-based Reconstruction. Surfels (surface elements) are first proposed by Pfister et al. [17] as rendering primitives. A surfel is a zero-dimensional n-tuple that can approximate a local surface. Keller et al. [11] defined a surfel as a tuple of a 3D position, a normal, and a radius and first used surfels for real-time reconstruction. Then SurfelWarp [12] was proposed to extend the use of surfels to non-rigid scene reconstruction. Using surfels for geometry representation in non-rigid reconstruction has the advantage of faster processing and smaller memory usage [12]. However, current surfel-based methods do not handle large topological change (e.g., surface splitting) without global re-initialization. We found in this work that surfels are efficient for local re-initialization because they can be structured as a simple unordered list of points. Compared to TSDF-based models, removing failure parts and appending new geometry locally is much easier for surfel-based models. Meanwhile, this property also makes it easy to split a model into models for different objects, which is another key requirement for scene-level segmentation.

III. PROBLEM FORMULATION AND BACKGROUND

We consider the problem of simultaneous tracking and reconstruction of all the objects that are present in a given dynamic scene, using as inputs a sequence of RGB-D images taken from K different camera poses. The objects and their number are completely unknown *a priori*. The objects can also be non-rigid. A depth map in a given RGB-D image is denoted as $d : \Omega \to \mathbb{R}$ where Ω is the set of pixels in the image, and $d(u)$ is the depth of pixel $u = (x, y) \in \Omega$. Similarly, we denote the RGB image as $c : \Omega \to [0,1]^3$.

The output of the proposed system at each time-step t is a set containing a *Surfel-based* geometry S_i^t for each individual object i in the scene, and a corresponding deformation graph G_i^t. Surfel-based geometry S_i^t is a set of surfels s_j. A surfel s_j is defined as $s_j = (v_j, n_j, c_j, r_i)$, where v_j, n_j, c_j, r_i are respectively the 3D coordinates, normal, color and radius of surfel $s_j \in S_i^t$. Different from previous methods [13], [12], which keep a canonical geometry model and a live geometry model, we only keep the latest geometry model, because we only care about the current geometry of the scene

and objects. Deformation graph G_i^t is defined by a set of nodes $\{g_i^t\}$ that correspond to 3D points belonging to the same topology (i.e., same object). Each node in the graph is connected to its nearest-neighbors. Nodes and edges in G_i^t are added or dropped dynamically as topology changes are detected. Deformation graph G_i^t of object i is accompanied with a *warp field* W_i^t. A warp field is defined as $W = \{[p_j \in \mathbb{R}^3, \delta_j \in \mathbb{R}^+, T_j \in SE(3)]\}$, wherein j is the node index in the accompanying graph, p_j is the 3D point that corresponds to node j, δ_j is a the node's radius of influence, and T_j is the 6-D transformation defined on node g_j. Here T_j is represented by a *dual quaternion* q_j for smooth interpolation [18]. Warp field W is used to describe the deformation between two consecutive time steps. For each surfel $s = (v, n, c, r) \in S$, we compute its 6-D transformation $\bar{W}(s)$ based on warp field W by applying the formula

$$\bar{W}(s) = normalize\Big(\sum_{k \in N(s)} \omega_k(v, p_k) q_k\Big) \tag{1}$$

Here, $N(s)$ denotes the set of nodes that are the neighbors of the surfel s (see Sec. IV-B and Fig. 2 for details). $\omega(s)$ is an interpolation parameter, defined as $\omega(s) = \exp\big(\|v - p_k\|_2^2 / (2\delta_k^2)\big)$, v is 3-D position of surfel s. The local transformation $\bar{W}(s)$ is then used to describe the deformation of surfel s as follows,

$$\begin{aligned} \bar{W}^t(s) v &= v_{warp}, \\ rotation\big(\bar{W}(s)\big) n &= n_{warp}, \end{aligned} \tag{2}$$

wherein v, n are vertex and normal of s before deformation, and v_{warp}, n_{warp} are those after warping.

IV. PROPOSED APPROACH

A. Overview

An overview of the proposed system is shown in Fig.1. This pipeline is divided into four main components: (1) measurement fusion (shown in blue), (2) non-rigid alignment (yellow), (3) geometry and deformation graph update (pink), and (4) topology separation (grey). The green box refers to the entire scene representation model $(S^{t-1}, G^{t-1}, W^{t-1}, d_h^{t-1})$ at time-step $t-1$ as input to the system, and as its output at time-step t. At each time-step, RGB-D measurements of the scene from different cameras are fused into a single surfel-based geometry M, defined as a set of surfels (Sec. III). Current model $(S^{t-1}, G^{t-1}, W^{t-1}, d_h^{t-1})$ is aligned with in-coming fused measurement M through non-rigid alignment (Sec. IV-E), which results in a new scene geometry S_{align}. Then, a registration between S_{align} and M is performed (Sec. IV-F). The parts of S_{match} and M_{match} that match together are fused. The unmatched parts of the geometry from the existing model (i.e., outliers) are removed, and the unregistered measurement (newly observed parts of the scene) is appended to the model. In parallel to the updates of the geometry model, deformation graph G^{t-1} is updated by removing nodes that are out of track and expanding the graph with nodes corresponding to newly observed points. Finally, historical maximum distance d_h^{t-1} is updated. In addition to the updated scene model, the system returns a set of local models (S_i^t, G_i^t), one for each detected object after applying a topology-based segmentation.

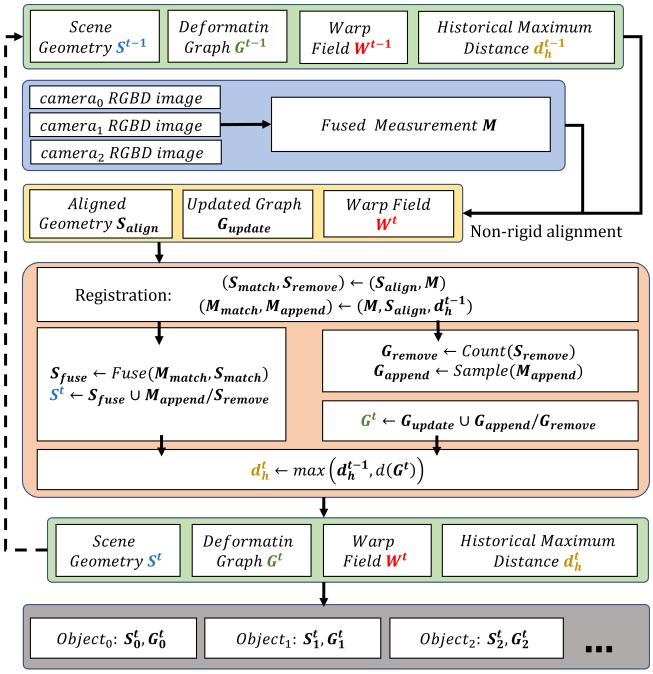


Fig. 1: Overview of the proposed pipeline

B. Deformation model representation

The proposed method uses a surfel-based model instead of the TSDF model which is the main-stream model used for non-rigid scene reconstruction. We found from our investigation that a surfel-based representation is better than TSDF when facing frequent topology changes. TSDF is efficient at maintaining local topology because each surface is determined by all its nearby voxels, which also implies however that local topology is hard to change. When a local geometry is lost, due to occlusion, the entire TSDF model needs to be reset to fix the local problem. In contrast, a surfel-based representation has no internal constraints over topology. Therefore, failures can be fixed locally.

On the other hand, surfel representations have the disadvantage of not carrying any topology information. A typical surfel-based model cannot distinguish between two surfels belonging to different objects. Thus, we propose a hybrid model that combines a surfel-based geometry S with a topology-aware deformation graph G. As shown in Fig. 2, for each surfel $s_i \in S$, we assign a node $g_j = \text{support}(s_i, G)$ from G as its support node. Support node of s_i is defined as the nearest point in G to s_i when it joined the geometry S, using the Euclidean distance. Each surfel is assigned to a unique support node, and one support node typically supports numerous surfels. Graph G is dynamically decomposed into several topologies (i.e., connected components), one per object. Surfel s_i is assigned to the same topology as its supporting node g_j. When g_j is removed, all surfels supported by g_j are also removed from S. Deformation graph

G is the skeleton of the 3D model, and surfel set S is its skin. The topology of the surfel set S is represented by its skeleton G.

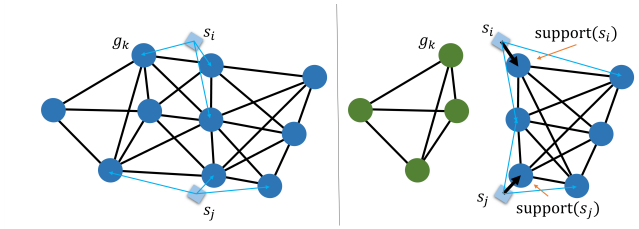
Deformation graph G is dynamic, its nodes and edges change over time. Each node in G is connected to its k-nearest neighboring nodes in G, using the **historical maximum distance** d_h as a metric, which we define here as:

$$\forall g_i, g_j \in G, d_h(g_i, g_j) = max_{t \in T}\{d^t(g_i, g_j)\}, \qquad (3)$$

wherein T is the number of time-steps (or frames) in the sequence of images, and $d^t(g_i, g_j)$ is the Euclidean distance between nodes g_i and g_j at that time step t. The historical maximum distance proposed here is better than the Euclidean distance in terms of determining topology. Two nodes g_i and g_j belonging to different objects can be close to each other for arbitrarily long periods, which happens for example when one of the objects is resting on the other. But if the two nodes have been observed to be far away from each other at any moment in the past frames, then they should not be neighbors in G.

Given deformation graph G, we define the set $N(s)$ of neighbors of surfel $s \in S$ as the k'-nearest neighbors of s among the k-nearest neighbors of g, the support node of s. In other terms, $N(s) = k'\mathrm{NN}\big(s, k\mathrm{NN}(g, G)\big)$, with $g = \mathrm{support}(s, G)$ and $k' \leq k$. Unlike the k-nearest neighbors of g selected using the historical maximum distance, the k'-nearest neighbors of s are obtained using the Euclidean distance at the latest time step. Because all neighbors of s are selected among the neighbors of its support node in G, they necessarily belong to the same connected component in G, i.e., same topology.

In summary, we use S (surfel-based geometry), G (deformation graph), W (warp field), d_h (historical maximum distance) to describe the deformable model of the entire scene.



(a) Traditional embedded deformation graph (b) Topology-aware deformation graph

Fig. 2: Proposed topology-aware deformation graph. In traditional embedded deformation graphs, neighbors of a node (such as g_k) are decided by their current Euclidean distances, and neighbors of a surfel (such as s_i and s_j) are also defined by their Euclidean distance to the nodes in the graph. In our proposed deformation graph, neighbors of a node are determined by a historical maximum distance d_t. Also, neighbors of a surfel can only be selected from the same sub-graph as its support node (support(s_i) and support(s_j)).

C. Initialization

S, G, W, d_h are initialized at the first frame. The geometry S is initialized with the measurement M from the first frame. The nodes of deformation graph G are uniformly sampled from those initializing surfels spatially with the same algorithm as DynamicFusion [13]. The initializing neighbor set $N(g)$ of every node g in G is their kNN set by Euclidean distance at the first frame. The historical maximum distance d_h is initialized by the Euclidean distance at the starting frame. The warp-field W associated with the deformation graph G is initialized to an identity transformation.

D. Measurement

The measurement acquisition system in our pipeline is similar to that of Function4D [14], a sparse multi-camera system consisting of three RealSense-415 RGB-D cameras. Multi-view measurement fusion is achieved through a TSDF-based fusion pipeline similar to the one used in [5], [6]. It is worth noting that we use TSDF here instead of the surfel-based representation because surfel models are more sensitive to distortion and calibration error across different cameras, and correcting for these errors is hard and computationally expensive. This problem with surfels was addressed in [15] by projecting the point cloud to local planes, but this solution is still computationally inefficient and relies on a multi-camera system with 106 cameras. Alternatively, a TSDF model can dramatically decrease measurement noise and achieve a good smoothness between measurements from different sources with a low computational cost.

Since we rely on the measurement to detect areas where topology changes (Sec. IV-F), we only keep the best measurement. According to [19], the noise level of depth measurement at a point is proportional to the angle between the local normal and the camera view direction. Thus, we discard measurements that have a viewing angle larger than 70°.

After each measurement fusion, we use the Raycast algorithm described in KinectFusion [20] to transform the TSDF into a measurement surfel model M for further processing.

E. Non-rigid alignment

We show here how to compute the non-rigid warp field by solving a massive optimization problem on a GPU. For this step, we principally follow the method of DynamicFusion [13] and SurfelWarp [12], and adapt them to our surfel-based model and multi-view setting. The key idea of non-rigid warp field estimation is to align the geometry model S^{t-1} to the current measurement M. This can be formulated as the following optimization problem,

$$\min_W E_{total}(W) \text{ with } E_{total}(W) = E_{depth}(W) + \lambda E_{reg}(W). \quad (4)$$

Here, E_{depth} is used to align measurement M and geometry S. E_{reg} is used to regulate the deformation within each topology favoring rigidity. λ is a balancing parameter. Parameter λ is typically small because strong regularization constraints prevent topology from separation. Furthermore, E_{depth} is defined as follows,

$$E_{depth}(W) = \sum_{i=1}^{K} \sum_{(s^{t-1}, s_M) \in P_i} \left(n_M^T (v^{t-1} - v_M)\right)^2. \qquad (5)$$

Here, K is the number of cameras, P_i is a set of pairs of measured depths s_M from each camera pose i and their corresponding rendered models S^{t-1}. The normals of s_M and s^{t-1} are denoted by n_M and n^{t-1}, while v^{t-1} and v_M are the vectors containing the 3D vertices of s^{t-1} and s_M.

Rendered models are obtained by rendering the global geometry S^{t-1} under different camera views. Each rendered image has the same size as its corresponding measurement depth image. Thus, for each pixel $u=(u_x,u_y)$ on the rendered geometry map, we search within a small neighborhood $(u_x \pm \sigma, u_y \pm \sigma)$ for the best correspondence in terms of position and normal difference. If their position distance or normal difference are above thresholds $\gamma_{distance}$ and γ_{normal}, this pair will be discarded from the cost computation in Equation 5.

As for the regulation term E_{reg}, it is defined as follows,

$$E_{reg}(W) = \sum_{g_j \in G} \sum_{g_i \in N(g_j)} \left\| T_j p_j - T_i p_i \right\|_2^2. \tag{6}$$

This regulation term is used to constrain deformations within each topology to be as rigid as possible. The topology structure is described by $N(g_j)$, the set of neighbors of node g_j in the deformation graph. Neighbors are not defined as the nearest ones by Euclidean distance, but by a historical maximum distance d_h (See Sec. IV-B).

This optimization problem is a nonlinear least-squares problem. Thus, we solve it with the Gaussian-Newton algorithm. Every step here is implemented with CUDA on single GPU efficiently, and the solution process runs on real-time.

The output of this step is the warp-field estimate in time step t, W^t. The aligned geometry $S_{align} = W^t S^{t-1}$. This deformation is defined in Equations 1 and 2. We also update the node position of the deformation graph G_{update} with the following formulas:

$$\begin{aligned} g_{update} &\leftarrow \bar{W}^t(g^{t-1}) g^{t-1} \\ \bar{W}^t(g^{t-1}) &= normalize\Big(\sum_{k \in N(g^{t-1})} \omega_k(g^{t-1}, p_k) q_k \Big) \end{aligned} \tag{7}$$

Here ω_k has the same definition as Equation 1. $N(g^{t-1})$ is the neighbor set of g^{t-1} in graph G^{t-1}.

F. Geometry and deformation graph update

There are four steps in this section: registration, fusion, appending, and removal.

In a nutshell, after the non-rigid alignment (See Sec. IV-E), the geometry from the last time step S^{t-1} is warped to S_{align}, close to the current measurement M. However, there still exists a discrepancy between the aligned geometry S_{align} and the measurement surfel M. There could be multiple reasons for this: measurement noises, newly observed surfaces, topology changes, etc. Thus, we perform registration between S_{align} and M. A match between s_{align} and s_M means the geometry is being observed in the current measurement. We need to fuse s_M into s_{align} to average the measurement noise. Meanwhile, if a given s_M has no match in S_{align}, this s_M can be a newly observed surface or a noise. If we verify this s_M as newly observed geometry, we should append it to S^t. And when a given s_{align} finds no correspondence to M, it is likely to be an outlier or just currently invisible. If it is an outlier, then it should be removed from geometry S^t.

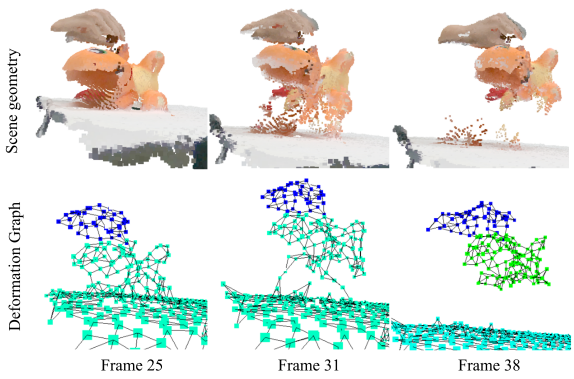


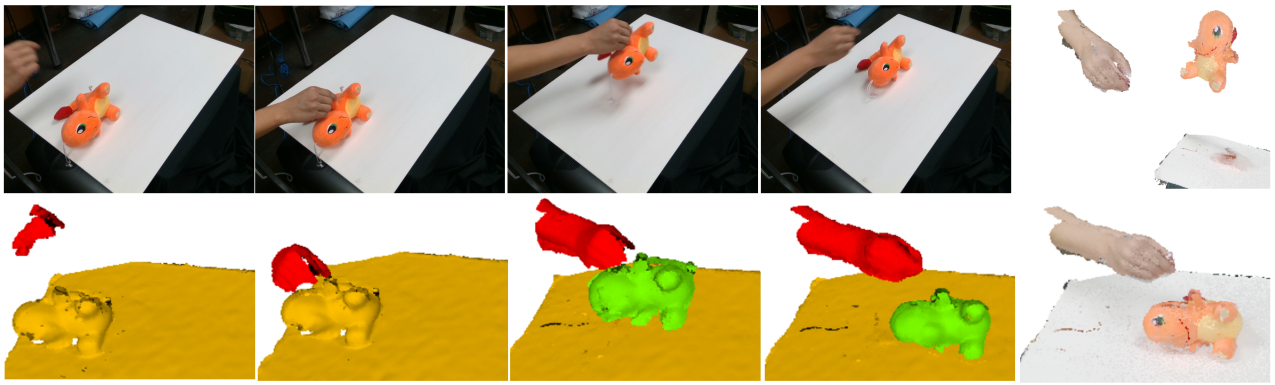
Fig. 3: Change of the deformation graph when topological change is detected. At the beginning, the non-rigid plush toy belongs to the same topology as the table. The toy is then grasped and lifted. After the non-rigid alignment step, the upper part geometry of the toy is aligned to the current measurement. However, the remaining part of the toy still belongs to the same topology as the table, staying still at its original location. Since this part is out of track, its surfels are removed from the geometry. When enough surfels are removed due to their inconsistency with the free space, the removal of their support nodes is also triggered. After enough nodes in-between the deformation graphs of the toy and the table are deleted, the table and the toy's topologies are separated.

Registration. The geometry registration pipeline is similar to SurfelWarp [12], but in a multi-view version. We first render the aligned geometry S_{align} and the fused measurement M to each camera pose k to generate an index map $\mathcal{I}_k$ and a depth map D_k. Each surfel will be projected to the camera's coordinates with corresponding camera intrinsic I_k and camera pose T^k_{camera}. To avoid nearby surfels being projected to the same pixel in the index map. The index map is super-sampled by a scale of 4×4. Then by comparing the index map $\mathcal{I}_k$ with depth measurement D_k, we can get the correspondence between geometry surfel S_{align} and measurement surfel M. Given a pixel position u on the depth measurement, we search a 4×4 area on the corresponding part of the index map. Among these points, we select the correspondence with the following criteria: (1) Ignore surfels whose distance to the depth vertex is larger than $\gamma_{distance}$. (2) Ignore surfels whose normal is far away from depth normal. i.e. $dot(n_{surfel}, n_{depth}) < \gamma_{normal}$. (3) If there are multiple such surfels, choose the one that is nearest to the depth vertex.

Fusion. If such a correspondence between s_{align} and s_M is found, we will use the following formula to fuse them: $v_{fuse} \leftarrow \frac{c_{align} v_{align} + c_M v_M}{(c_{align} + c_M)}, n_{fuse} \leftarrow \frac{c_{align} n_{align} + c_M n_M}{(c_{align} + c_M)}, r_{fuse} \leftarrow \frac{c_{align} r_{align} + c_M r_M}{(c_{align} + c_M)}, t_{fuse} \leftarrow t_M, c_{fuse} \leftarrow c_{align} + c_M$. Here, v, n, r, t, c are vertex, normal, radius, timestamp and confidence of a surfel. The update is done in the above order sequentially.

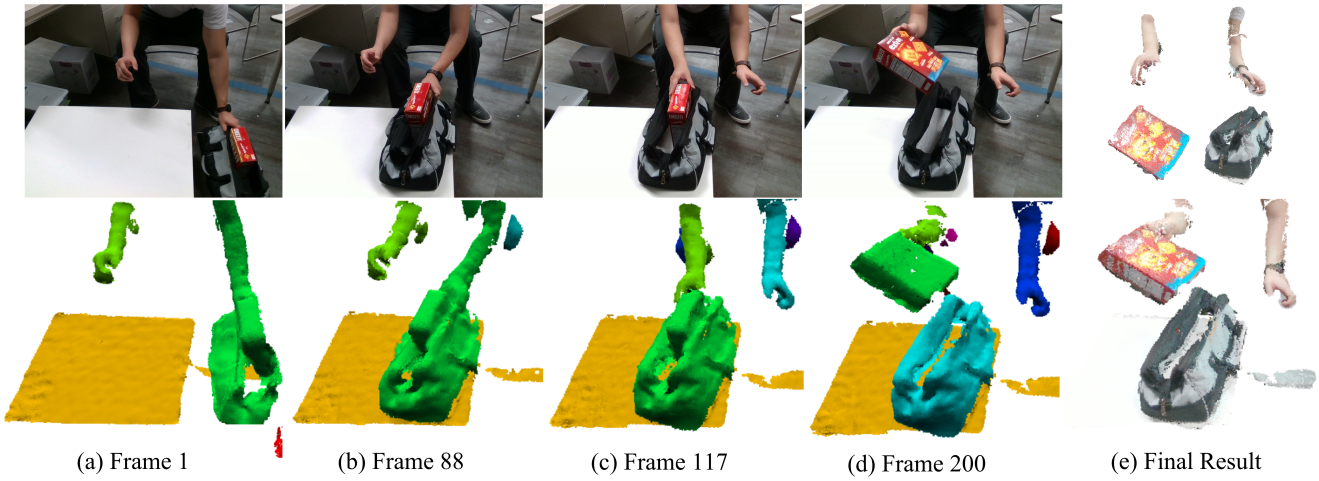
Append. If no correspondence is found for a measurement surfel s_M, we will mark it as an appending candidate. Candidates under the following three cases will not be appended:

If s_M is far away from existing geometry, it is likely to be a measurement noise. Furthermore, the multi-view camera



(a) Frame 10 (b) Frame 30 (c) Frame 60 (d) Frame 80 (e) Final Model

(a) We lift a fire dragon plush toy from the table and release it. The first row is the RGB image from camera 0, the second row is the corresponding segmentation result at that frame. The top right shows the separated models for each object. The bottom right is the scene geometry at the final frame.



(a) Frame 1 (b) Frame 88 (c) Frame 117 (d) Frame 200 (e) Final Result

(b) We use one hand to lift a non-rigid bag and put it on the table. Then, we pick a cracker box up from the bag with another hand.

Fig. 4: Experiments on two scenarios with a multitude of topological changes. Whenever a topology is discovered to be separate from others, it will be identified as an individual object. No object prior is used during the entire process. While such topological changes can relatively easily be detected by other techniques that make rigid-body assumptions, the non-rigidity of objects (i.e. the plush toy, hand, bag) makes the simultaneous reconstruction, tracking, and topology change handling extremely challenging.

setting brings a problem of different distortion models. In our practice, we find that there exists a shift in depth among different cameras even towards the same area. In case we create multiple surfaces for the same geometry, candidates too close to existing surfaces are thrown away. Finally, candidates supported by a *compressed* node should not be appended. A node $g_i \in |G|$ can be defined as compressed if the following holds,

$$\exists g_j \in |G|, d_h^t(g_i, g_j) > \gamma_{upper}, d^t(g_i, g_j) < \gamma_{lower}, \quad (8)$$

d_h^t is the historical maximum distance, and d^t is the Euclidean distance in the current frame. A node becomes compressed when there exist some other nodes that are far away in history but currently near each other. This often happens when separated topologies approach each other (i.e., a hand approaching the table). To avoid having different topologies wrongly connected together by appending nodes, the appending process is not performed near these areas.

In summary, the following criteria are imposed: (1) Discard surfels s whose distance to nearest deformation node is larger than γ_{nn}. (2) Discard surfels s whose support (i.e. closest) node $support(s)$ is in compression status. (3) Discard surfels s whose maximal distance to nearby depth surfels along the camera view direction is smaller than γ_{inlier}. All remaining candidates are appended to the latest geometry S^t. After new surfels are appended to the geometry, we collect surfels with shortest distances to the existing deformation node larger than r_{sample}. These surfels do not have a valid support node. Thus, we perform a spatially uniform sampling from these surfels, and sampled nodes are appended to the deformation graph G^t. Since new nodes are appended to the deformation graph, the historical maximum distance d_h also needs to be updated. If g_k belongs to the newly appended nodes, we initialize their d_h by the following formula,

$$\forall g_j \in G^t, d_h^t(g_k, g_j) = max\{d_h^t(g_i, g_j) - d^t(g_i, g_k), d^t(g_j, g_k)\}.$$

wherein g_i is the nearest neighbor node of g_k. According to the triangle inequality, $d_h^t(g_k, g_j)$ is a lower bound for the

historical maximal distance between k and j.

Removal. Surfels s_{align} that are not registered are marked as removal candidates. There are three cases where a surfel s should be removed: it is unstable for a long time (low confidence and not updated in a long time), overlapped with nearby surfels, or inconsistent with measurement M.

A removal counter for each node is maintained to count how many of its supporting surfels have been removed. If this counter passes a threshold γ_{remove}, the removal of the corresponding node is triggered. Because it is very likely that there exists a topological change and tracking failure nearby this node, we perform a local re-initialization by removing this node and its supporting surfels.

After the update of geometry S^t and deformation graph G^t is finished, we update historical maximum distance d_h^t between every two graph node with the follows:

$$\forall g_i, g_j \in \{G^t \cap G^{t-1}\}, d_h^t \leftarrow max(d_h^{t-1}, d^t(g_i, g_j)),$$

wherein d^t is the Euclidean distance in G^t.

G. Topological separation

The final step of our system is the topological separation. This step separates the geometry S and deformation graph G into a set $\{(S_i, G_i)\}$, wherein $\{G_i\}$ are the connected components of G, which have been already separated based on the maximum historical distance d_h. These components are obtained by applying the flood-fill algorithm on G.

V. EXPERIMENTS

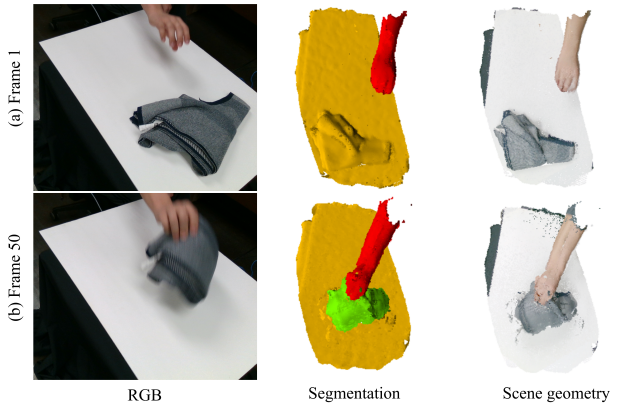


Fig. 5: We pick a scarf from the table. The RGB images show how both our arm and the scarf are dramatically deformed. Despite that, our system successfully isolated each topology, tracked and reconstructed each object in the scene.

For non-rigid reconstruction problems, it is hard to provide quantitative results with real scenes [12], [13], because it is extremely challenging and expensive to obtain ground truth for deformable models. Thus, we conduct a set of qualitative experiments over a wide range of real-world scenarios that involve non-rigid object manipulation, which are closely related to robotic manipulation scenarios. The experiments aim to assess the functionality of the proposed system and the validity of our novel topological change handling strategy.

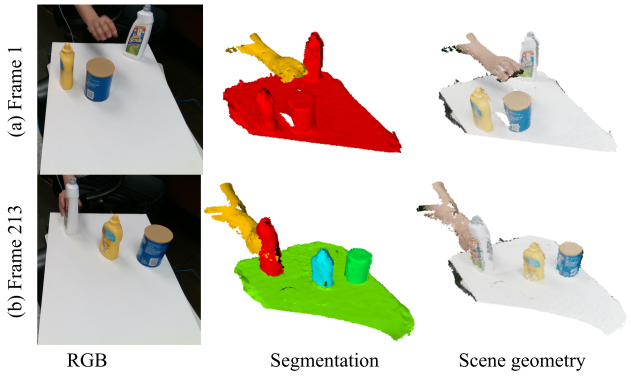


Fig. 6: An object re-arrangement example. Three objects are moved to different positions in a 9-seconds video. This experiment demonstrates the capability of our system in dealing with multiple moving objects, and its potential use in robot re-arrangement tasks.

Fig. 4a and Fig. 4b show the process of topological change handling. For example, in Fig. 4a, four distinct topological changes are happening during this experiment; (1) the hand grasps the non-rigid plush toy, (2) the toy separates from the table, (3) the hand releases the toy, (4) and the toy rests on the table again. The Fig. 4b is even more complicated than Fig. 4a. Our system detected and handled all of those topological changes correctly.

Fig. 5 and Fig. 6 demonstrate the stability of our proposed system under different challenging settings. In Fig. 5, our proposed framework successfully handles a scene with large deformation, picking up a scarf. Fig. 6 illustrates the capability of our system in processing multiple moving objects (5 objects in this scene, including the table), which demonstrates a potential usage in robot re-arrangement tasks.

Fig. 7 compares our reconstruction result with SurfelWarp [12] with/without global re-initialization, from which we can observe how global re-initialization loses too much information and no re-initialization cannot handle the topological change. This shows the superiority of the proposed local re-initialization strategy over the global re-initialization strategy. To the best of our knowledge, this is the first real-time non-rigid reconstruction system using local re-initialization as the topology handling strategy.

Performance. The proposed system can run at 40 fps. On average, the measurement fusion takes $4ms$, non-rigid alignment takes $10ms$, geometry and graph updates take $8ms$, and topology segmentation takes $3ms$. This performance is obtained by testing the system on a desktop machine with a GeForce RTX 3090 and an AMD-Ryzen 9 5900X, on the scene shown in Fig. 4a, which contains $13k$ surfels and 900 deformation nodes.

Limitation and Discussion. Although our system is able to solve the scene-level tracking and reconstruction problem without any prior, it still has many limitations. First, our current system relies on only depth measurement for registration and global registration is lacking here. If there is a fast and large deformation or motion in the scene, our system is not able to track it. How to combine global registration with our current system is a future direction of our work. Our

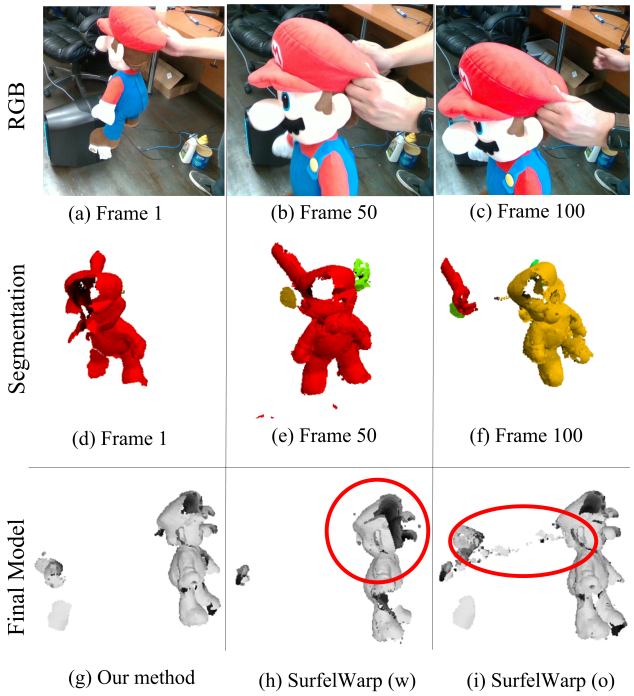


Fig. 7: We move a Mario plush toy from one hand to another. The top row is the RGB measurement from camera 0. The middle row is the corresponding segmentation result. The bottom row shows a comparison between the final model of our proposed method, with SurfelWarp (w) [12] with global re-initialization, and SurfelWarp (o) without re-initialization. In the third row, we can observe that the global re-initialization in the sparse multi-camera settings is losing too much information. There is a big hole in Mario's face. Without re-initialization, we observe that the hand and the toy are mistakenly connected. Our proposed method correctly separates the hand and the toy while maintaining most of the information.

proposed pipeline relies heavily on measurements for surfel appending and removal. Although we are using a multi-view setting, we often lose the measurement of partially visible scenes. The reason is that our multi-view camera system is sparse (we only use 3 cameras in our experiment). For those areas missing measurement, we are unable to make any update or refinement towards their geometry. This is the reason why the edges of our reconstructed model are sometimes not smooth. Furthermore, if a local geometry is out of vision but deforms largely, likely, we have already lost track of it when it comes back to our view. This loop-closure problem is an ongoing topic in all works related to non-rigid scene reconstruction.

VI. CONCLUSION

In this paper, we presented the first real-time solution for the STAR (Scene-level Tracking and Reconstruction) problem with no object prior, regardless of the rigidness or texture existence in the scene. Instead of segmenting the scene and reconstructing each object individually, the operation order is reversed by reconstructing the non-rigid scene first and then performing topology-based segmentation. A novel surfel-based local re-initialization strategy is introduced to deal with frequent topological changes in the scene while maintaining most of the global geometry. The proposed method can be integrated seamlessly into several robotics applications, such as learning manipulation skills from visual demonstrations.

REFERENCES

[1] Z. Sui, H. Chang, N. Xu, and O. C. Jenkins, "Geofusion: Geometric consistency informed scene estimation in dense clutter," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5913–5920, 2020.

[2] M. Runz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4471–4478, 2017.

[3] M. Runz, M. Buffier, and L. Agapito, "MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects," *Proceedings of the 2018 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2018*, pp. 10–20, 2019.

[4] Y.-S. Wong, C. Li, M. Nießner, N. J. Mitra, and A. Research, "RigidFusion: RGB-D Scene Reconstruction with Rigidly-moving Objects," Tech. Rep. 2, 2021.

[5] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. O. Escolano, C. Rhemann, D. Kim, J. Taylor, P. Kohli, V. Tankovich, and S. Izadi, "Fusion4D: Real-time performance capture of challenging scenes," in *ACM Transactions on Graphics*, vol. 35, no. 4. Association for Computing Machinery, jul 2016.

[6] M. Dou, P. Davidson, S. R. Fanello, S. Khamis, A. Kow-Dle, C. Rhemann, V. Tankovich, S. Izadi, and A. Kowdle, "Motion2Fusion: Real-time Volumetric Performance Capture. 1, 1, Article 246," Tech. Rep., 2017. [Online]. Available: https://doi.org/10.475/1234

[7] K. Zampogiannis, C. Fermuller, and Y. Aloimonos, "Topology-Aware Non-Rigid Point Cloud Registration," nov 2018. [Online]. Available: http://arxiv.org/abs/1811.07014http://dx.doi.org/10.1109/TPAMI.2019.2940655

[8] C. Li and X. Guo, "Topology-Change-Aware Volumetric Fusion for Dynamic Scene Reconstruction," jul 2020. [Online]. Available: http://arxiv.org/abs/2007.06853

[9] M. Slavcheva, M. Baust, and S. Ilic, "Variational Level Set Evolution for Non-rigid 3D Reconstruction from a Single Depth Camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, feb 2020.

[10] ——, "Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2646–2655.

[11] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3D reconstruction in dynamic scenes using point-based fusion," in *Proceedings - 2013 International Conference on 3D Vision, 3DV 2013*, 2013, pp. 1–8.

[12] W. Gao and R. Tedrake, "SurfelWarp: Efficient Non-Volumetric Single View Dynamic Reconstruction," apr 2019. [Online]. Available: http://arxiv.org/abs/1904.13073

[13] R. A. Newcombe, D. Fox, and S. M. Seitz, "DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time," Tech. Rep.

[14] T. Yu, Z. Zheng, K. Guo, P. Liu, Q. Dai, and Y. Liu, "Function4D."

[15] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, H. Hoppe, A. Kirk, S. Sullivan, and D. Calabrese, "High-Quality Streamable Free-Viewpoint Video."

[16] M. Dou, J. Taylor, H. Fuchs, A. Fitzgibbon, and S. Izadi, "3d scanning deformable objects with a single rgbd sensor," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 493–501.

[17] H. Pfister, M. Zwicker, J. Van Baar, and M. Gross, "Surfels: Surface elements as rendering primitives," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 335–342.

[18] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan, "Skinning with dual quaternions," *Proceedings - I3D 2007, ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 39–46, 2007.

[19] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "ElasticFusion: Dense SLAM without a pose graph," *Robotics: Science and Systems*, vol. 11, 2015.

[20] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison *et al.*, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 559–568.