ArticulatedFusion: Real-Time Reconstruction of Motion, Geometry and Segmentation Using a Single Depth Camera

Chao Li, Zheheng Zhao, and Xiaohu Guo(✉)

Department of Computer Science, The University of Texas at Dallas, Richardson, USA
{Chao.Li2,Zheheng.Zhao,xguo}@utdallas.edu

Abstract. This paper proposes a real-time dynamic scene reconstruction method capable of reproducing the motion, geometry, and segmentation simultaneously given live depth stream from a single RGB-D camera. Our approach fuses geometry frame by frame and uses a segmentation-enhanced node graph structure to drive the deformation of geometry in registration step. A two-level node motion optimization is proposed. The optimization space of node motions and the range of physically-plausible deformations are largely reduced by taking advantage of the articulated motion prior, which is solved by an efficient node graph segmentation method. Compared to previous fusion-based dynamic scene reconstruction methods, our experiments show robust and improved reconstruction results for tangential and occluded motions.

Keywords: Fusion · Articulated · Motion · Segmentation

1 Introduction

Dynamic scene reconstruction is a very important topic for digital world building. It includes capturing and reproducing geometry, appearance, motion, and skeleton, which enables more realistic rendering for VR/AR scenarios like Holoportation [5]. An example is that the reconstructed geometry can be directly used for a virtual scene, and the articulated motion can be retargeted to new models to generate new animations, making scene production more efficient.

Although many efforts have been devoted to this research field, the problem remains challenging due to extraordinarily large solution space but real-time rendering requirements for VR/AR applications. Recently, volumetric depth fusion

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01237-3_20) contains supplementary material, which is available to authorized users.

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11212, pp. 324–340, 2018.
https://doi.org/10.1007/978-3-030-01237-3_20

methods for dynamic scene reconstruction, such as DynamicFusion [17], VolumeDeform [10], Fusion4D [5] and albedo based fusion [8] open a new gate for people in this field. This type of method enables quality improvements over temporal reconstruction models in terms of both accuracy and completeness of the surface geometry. Among all these works, fusion methods by a single depth camera [10,17] are more promising for popularization, because of their low cost and easy setup. However, this group of methods still faces some challenging issues, like high occlusion from the single view, limited computational resource to achieve real-time performance, and no geometry/skeleton prior knowledge, and thus are restricted to limited motions. DoubleFusion [30] can reconstruct both the inner body and outer surface for faster motions by adding body template as prior knowledge. Later, KillingFusion [21] and SobolevFusion [22] is proposed to reconstruct dynamic scenes with topology changes and fast inter-frame motions.

DynamicFusion is the pioneering work acheiving template-less non-rigid reconstruction in real time from single depth camera. However, its robustness can be significantly improved by utilizing skeleton prior, as been shown in work of BodyFusion [29]. In this paper, we propose to add articulated motion prior into the depth fusion system. Our method contributes to this field by pushing the limitation from skeleton-prior-based methods to skeleton-less ones. The motions of many objects in our world including human motion follows articulated structures. Thus, articulated motions can be represented by skeleton/cluster-based motion and can be extracted from non-rigid motion as a prior. Our self-adaption segmentation inherits the rigid feature of traditional skeleton structure while does not require any pre-defined skeleton. The segmentation constrains all nodes labeled to the same segment having transformation as close as possible and can reduce the solution space of the optimization problem. Therefore, the self-adapted segmentation can result in better reconstruction results.

Our method iteratively optimizes the motion field of a node graph and its segmentation, which helps each other to get a better reconstruction performance. Integrating the articulated motion prior into the reconstruction framework assists in the non-rigid surface registration and geometry fusion, while surface registration results improve the quality of segmentation and its reconstructed motion. Although the advantages of such unification is obvious, in practice, designing a real-time algorithm to take advantage of both merits of these two aspects is still an unstudied problem, especially on how to segment a node graph based on its motion trajectory in real-time. We have carefully designed our ArticulatedFusion system, to achieve simultaneous reconstruction of motion, geometry, and segmentation in real-time, given a single depth video input. The contributions in this paper are as follows:

1. We present ArticulatedFusion, a system that involves registration, segmentation, and fusion, and enables real-time reconstruction of motion, geometry, and segmentation for dynamic scenes of human and non-human subjects.
2. A two-level registration method which can narrow down the optimization solution space, and result in better reconstructed motions in many challenging cases, with the help of node graph segmentation.

3. A novel real-time segmentation method to solve the clustering of a set of deformed nodes based on their motion trajectories by merging and swapping operations.

2 Related Work

The most popular dynamic 3D scene reconstruction method is to use a predefined model or skeleton as prior knowledge. Most of these methods focus on the reconstruction of human body parts such as face [3,14], hands [24,25], and body [20,27]. Other techniques are proposed to reconstruct general objects by using a pre-scanned geometry [13,32] as a template instead of predefined models.

To further eliminate the dependency on geometry priors, some template-less methods were proposed to utilize more advanced structure to merge and store geometry information across the motion sequence. Wand et al. [28] proposed an algorithm to align and merge pairs of adjacent frames in a hierarchical fashion to gradually build the template shape. Recently, fine 3D models have been reconstructed without any shape priors by gradually fusing multi-frame depth images from a single view depth camera [5,10,17,18]. Innmann et al. [10] proposed to add SIFT features to the ICP registration framework, thereby improving the accuracy of motion reconstruction.

Our method is partly inspired by Pekelny and Gotsman's method [19], However their method requires the user to manually segment a range scan in advance, whereas we automatically solve for the segmentation in real-time. Chang and Zwicker's method [4] is also lack of real data of human motions and takes much time to reconstruct for each frame. Tzionas and Gall's recent work [26] introduces an algorithm to build rigged models of articulated objects from depth data of a single camera. But it requires to pre-scan the target object as the geometry prior knowledge.

Guo et al. [6] proposes an L_0 regularizer to constrain local non-rigid deformation only on joints of articulated motion, which reduces the solution space and yields a physically plausible and robust deformation. However, our method is designed to achieve real-time performance while their method requires around 60s for the L_0 optimization of each frame [7]. Ours directly solves the segmentation of human body in the proposed energy function while theirs implicitly involves the articulated motion property in an L_0 regularizer. Their method also needs a pre-scaned shape as a template. Yu et al.'s method [29] is the one most related to our work, but it requires the skeleton information of the first frame as initialization while our method does not need any prior information. Our method can estimate the segmentation of dynamic scene during the reconstruction process. Therefore, it also works for non-human objects where a predefined skeleton is not available, as illustrated in Figs. 6 and 8. There is also a rich body of work proposed on articulated decomposition of animated mesh sequences [11,12]. Both of these methods can only work on animated sequences with fixed mesh connectivity, and cannot meet our real-time reconstruction requirement.

3 Overview

Figure 1 illustrates the pipeline of processing one frame given the geometry, motion and segmentation reconstructed from earlier frames. [8,10,17], our system runs in a frame-by-frame manner. Two main data structures are used in our system. The geometry is represented in a volume with the Truncated Signed Distance Function (TSDF), while the segmentation and motions are defined in an embedded graph of controlling nodes similar to DynamicFusion [17].

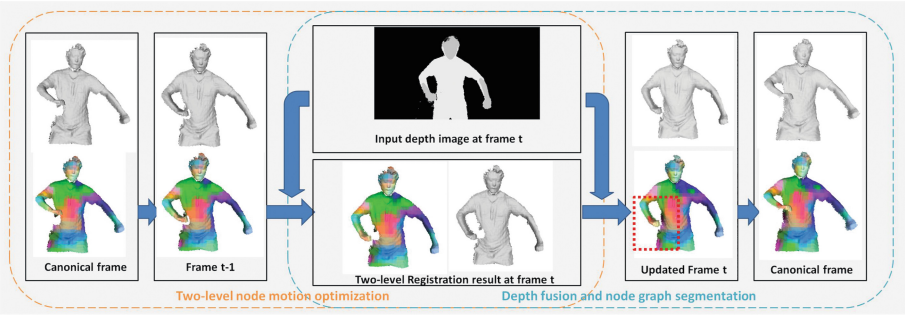


Fig. 1. Overview of our pipeline. The orange box represents our two-level node motion optimization, and the blue box represents fusion of depth and node graph segmentation. (Color figure online)

The first frame is selected as the *canonical frame*. The first step of our system is *two-level node motion optimization* (Sect. 4.2). In this step, motions of controlling nodes from the canonical frame to the current frame are estimated. This is achieved by first warping a mesh using reconstructed motion and segmentation from earlier frames, and followed by solving a two-level optimization problem to fit this mesh with the current depth image. The mesh is extracted from the TSDF volume by marching cube algorithm [15]. The first level of our node motion optimization is run on each segmented cluster, thus can reduce the solution space and make optimization converging faster. The second level of optimization is run on each individual node, so it can keep track of the high-frequency details of the target object. The depth is fused into the TSDF volume to obtain a new integrated geometry (Sect. 4.3). The final step is *node graph segmentation*, in which nodes are segmented by our novel clustering method to minimize the error between the articulated cluster deformation of nodes and their non-rigid deformation (Sect. 4.4). This segmentation makes the node motion estimation of next frame to perform better than employing non-rigid estimation only.

4 Method

4.1 Preliminaries and Initialization

Only a single depth camera is used to capture the depth information in our system. The input to our pipeline is a depth image sequence $\{\mathcal{D}^t\}$. The output of our pipeline includes a fused geometry $\mathcal{V}$ of the target object, the embedded graph segmentation $\mathcal{C}$, and the two-level warping field $\{\mathcal{W}^t\}$, where $\mathcal{W}^t$ represents the non-rigid node motion from the canonical frame to each live frame t. The TSDF volume and level-two warping field in our system is the same as those described in DynamicFusion [17].

For the first frame, we directly integrate the depth information into the canonical TSDF volume, extract a triangular mesh $\mathcal{M}$ from the canonical volume using the marching cube algorithm, uniformly sample deformation nodes on the mesh and construct a node graph to describe the non-rigid deformation. To search for nearest-neighboring nodes, we also create a dense k-NN field in the canonical volume. Because our segmentation method is based on the motion trajectory from canonical frame to a live frame, we cannot get a segmentation result for the first frame. Therefore, we employ the non-rigid registration method of DynamicFusion [17] to align the mesh to the second frame.

4.2 Registration

As mentioned above, the first step of our system is to fit the canonical mesh $\mathcal{M}$ to the depth image $\mathcal{D}^t$ of live frame t. We have the current mesh $\mathcal{M}$ (obtained by fusing the depth from earlier frames), the segmentation $\mathcal{C}$, and the motion field $\mathcal{W}^{t-1}$. Using the newly captured depth in frame t, the algorithm presented in this section estimates $\mathcal{W}^t$ to fit $\mathcal{M}$ with $\mathcal{D}^t$. For this purpose, we propose a two-level optimization framework based on Linear Blend Skinning (LBS) model and node graph motion representation. The optimization is solved by minimizing the following energy function first in LBS model and then in node graph model:

$$E_{total}(\mathcal{W}^t) = \omega_f E_{fit} + \omega_r E_{reg}, \tag{1}$$

where E_{fit} is the data term to minimize the fitting error between deformed vertex and its corresponding point on depth image, and E_{reg} regularizes the motion to be locally as rigid as possible. ω_f and ω_r are controlling weights to balance the influence of two energy terms. In all of our experiments, we set $\omega_f = 1.0$ and $\omega_r = 10.0$.

Before solving the energy function, we build the two-level deformation model based on the node graph and its segmentation by defining the following skinning weight for each vertex $\mathbf{v}_i$ on mesh $\mathcal{M}$:

$$\mathbf{w}_i^{(l)} = \begin{cases} \frac{1}{\Lambda}\sum_{j=1}^{k} \lambda_{i,j}\mathbf{g}_j & l = 1, \\ \frac{1}{\Lambda}\sum_{j=1}^{k} \lambda_{i,j}\mathbf{h}_j & l = 2, \end{cases} \tag{2}$$

where l denotes the level, and $\lambda_{i,j}$ is the weight describing the influence of the j-th node $\mathbf{x}_j$ on vertex $\mathbf{v}_i$ and is defined as $\lambda_{i,j} = exp\left(-\|\mathbf{v}_i - \mathbf{x}_j\|_2^2 / (2\sigma_j)^2\right)$.

Λ is a normalization coefficient, the summation of all spatial weights $\lambda_{i,j}$ for the same i. Here, σ_j is the given influence radius of controlling node $\mathbf{x}_j$. When level $l = 1$, $\mathbf{g}_j = (g_{j,1}, g_{j,2}, ..., g_{j,m})$ is the binding of controlling node $\mathbf{x}_j$ to m clusters. Because each node only belongs to one cluster, only one element of $\mathbf{g}_j$ is 1 and all other elements are 0. $\mathbf{w}_i^{(1)} = \left(w_{i,1}^{(1)}, w_{i,2}^{(1)}, ..., w_{i,m}^{(1)}\right)$ includes the skinning weights of vertex $\mathbf{v}_i$ w.r.t. m clusters. When level $l = 2$, $\mathbf{h}_j = (h_{j,1}, h_{j,2}, ..., h_{j,k})$ is the binding of $\mathbf{v}_i$'s neighboring node $\mathbf{x}_j$ to itself. Thus only $h_{j,j}$ is 1 and all other elements are 0. $\mathbf{w}_i^{(2)} = \left(w_{i,1}^{(2)}, w_{i,2}^{(2)}, ..., w_{i,k}^{(2)}\right)$ includes the skinning weight of vertex $\mathbf{v}_i$ w.r.t. its k-NN controlling nodes.
The fitting term E_{fit} represents the point-to-plane energy, as follows:

$$E_{fit}(\mathcal{W}^t) = \sum_i \left(\mathbf{n}_{\mathbf{u}_i^t}^\top \left(\hat{\mathbf{v}}_i - \mathbf{u}_i^t\right)\right)^2, \tag{3}$$

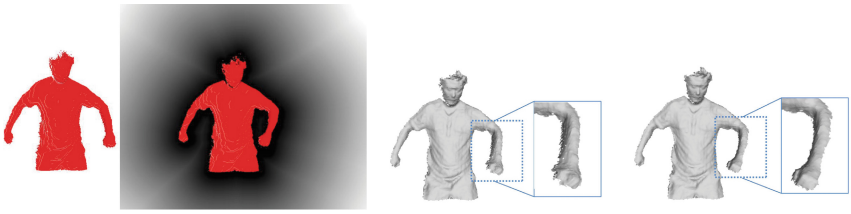
where $\hat{\mathbf{v}}_i$ is the transformed vertex defined by the formula:

$$\hat{\mathbf{v}}_i = \sum_j w_{i,j}^{(l)} \left(\mathbf{R}_j^t \mathbf{v}_i + \mathbf{t}_j^t\right). \tag{4}$$

Here $\mathbf{v}_i$ is a vertex on $\mathcal{M}$, and $\{\mathbf{R}_j^t, \mathbf{t}_j^t\}$ are the unknown rotation and translation of either the j-th cluster (level $l = 1$) or the j-th node (level $l = 2$), which will be solved during the optimization process. $\mathbf{u}_i^t$ is the corresponding 3D point on depth frame D^t for $\mathbf{v}_i$, and $\mathbf{n}_{\mathbf{u}_i^t}$ represents its normal. To obtain the pair of such correspondences, we render the deformed mesh $\mathcal{M}$ with the current warping field to exclude occluded vertices and project visible vertices onto the screen space of D^t. Then we look up the corresponding pixel with the same coordinates. For vertices lying on the silhouette of projected 2D image, we employ Tagliasacchi et al.'s method [24] – using 2D Distance Transform (DT) to locate the corresponding pixel and back-projecting it to 3D camera space. This correspondence search mechanism can guarantee better convergence when meeting large deformations in the direction perpendicular to the screen space (tangential motions) between two adjacent frames. Figure 2 shows a comparison of results with and without distance transform correspondences. Figure 2(a) are point clouds from two adjacent frames. The subfigure on the right illustrates the computed distance transform based on depth image contour. Figure 2(b) represents the tracking reconstruction result without using distance transform correspondences for silhouette points while Fig. 2(c) represents the result with distance transform correspondences search which is converged better than the one in Fig. 2(b).
The regularity term E_{reg} is an as-rigid-as-possible constraint:

$$E_{reg}(\mathcal{W}^t) = \sum_{j_1} \sum_{j_2 \in \mathcal{N}(j_1)} \alpha^{(l)}(\mathbf{g}_{j_1}, \mathbf{g}_{j_2}) \cdot \|\mathbf{R}_{j_1}^t \mathbf{x}_{j_2} + \mathbf{t}_{j_1}^t - \mathbf{R}_{j_2}^t \mathbf{x}_{j_2} - \mathbf{t}_{j_2}^t\|^2, \tag{5}$$

where $\mathcal{N}(j_1)$ denotes the set of neighboring nodes of the j_1-th node. $\alpha^{(l)}(\mathbf{g}_{j_1}, \mathbf{g}_{j_2})$ is a clustering-awareness weight. In level $l = 1$, $\alpha^{(1)}(\mathbf{g}_{j_1}, \mathbf{g}_{j_2}) = 1$ when the j_1-th node and the j_2-th node belong to the same cluster, and $\alpha^{(1)}(\mathbf{g}_{j_1}, \mathbf{g}_{j_2}) = 0$



(a) DT of the 2nd frame (b) Without DT (c) With DT

Fig. 2. Tracking results comparison from one frame to its next frame without and with Distance Transform (DT) correspondences.

otherwise. In level $l = 2$, $\alpha^{(2)}(\mathbf{g}_{j_1}, \mathbf{g}_{j_2})$ is always equal to 1. This regularization term is important to ensure that all vertices will move with the visible regions as rigidly as possible if some object regions are occluded due to our single-camera capture environment.

The minimization of Eq. (1) is a nonlinear problem. In level $l = 1$, we solve the transformations of each cluster, while in level $l = 2$, we solve the transformations of each node. Both levels are solved through Gauss-Newton iterations. In each iteration, the problem is linearized around the transformations from the previous iteration: $\mathbf{J}^\top \mathbf{J}\hat{\mathbf{x}} = \mathbf{J}^\top \mathbf{f}$, where $\mathbf{J}$ is the Jacobian of function $\mathbf{f}(\hat{x})$ from the energy decomposition: $E_{total}(\hat{x}) = \mathbf{f}(\hat{x})^\top \mathbf{f}(\hat{x})$. Then, a linear system is solved to obtain the updated transformations of $\hat{\mathbf{x}}$ for the current iteration with the twist representation [16] to represent the 6D motion parameters of each cluster or node. In order to meet the real-time requirement, we use the same method as in Fusion4D [5]: $\mathbf{J}^\top \mathbf{J}$ and $\mathbf{J}^\top \mathbf{f}$ is constructed on GPU, and then Preconditioned Conjugate Gradient (PCG) method is employed to solve the transformations.

4.3 Depth Fusion

After solving for the deformation of each node, we integrate the depth information into the TSDF volume of canonical frame and uniformly sample the newly added surface to update the nodes [17]. However, this integration method may result in issues due to voxel collision: if several voxels are warped to the same position in the live frame, then the TSDF of all these voxels will be updated. To resolve this ambiguity, we modify the method presented in Fusion4D [5] to a stricter strategy. If two or more voxels in the canonical frame are warped to the same position, we reject their TSDF integration. This method avoids the generation of erroneous surfaces due to voxel collisions.

4.4 Segmentation

The optimal articulated clustering of node graph $\mathcal{C} = \{C_n\}$ can be solved based on the motion trajectory from the canonical frame to live frame t. We assume

that each cluster is associated with a rigid transformation $\{\mathbf{R}_n^t, \mathbf{t}_n^t\}$. The following energy function measures the error between rigidly transformed node positions to their non-rigidly warped positions in live frame t:

$$E_{seg} = \sum_{n=1}^{m} \sum_{\mathbf{x} \in C_n} \|\mathbf{R}_n^t \mathbf{x} + \mathbf{t}_n^t - \mathbf{y}^t\|^2, \tag{6}$$

where t is the index of the live frame, n is the index of clusters, m is the total number of clusters, $\mathbf{x}$ is position of a node in the canonical frame and $\mathbf{y}^t$ is its corresponding node position after being warped into frame t. $\mathbf{x}$ and $\mathbf{y}^t$ have one-to-one correspondence because $\mathbf{y}^t$ are all deformed from the canonical frame.

The minimization of Eq. (6) implicitly includes the information of the motion trajectory – nodes with similar motions will be merged into the same cluster. By using our following method, the unknown clustering $\{C_n\}$ and per-cluster transformation $\{\mathbf{R}_n^t, \mathbf{t}_n^t\}$ can be solved simultaneously and efficiently. Although they are correlated, we find that $\{\mathbf{R}_n^t, \mathbf{t}_n^t\}$ has a closed-form solution for fixed clustering in Eq. (6) [9,23]. In this paper, we employ the merging and swapping idea as proposed by Cai et al. [1,2] to solve for $\{C_n\}$ and $\{\mathbf{R}_n^t, \mathbf{t}_n^t\}$ simultaneously.

Now we formulate the optimal clustering by minimizing the energy of Eq. (6) while keeping their rigid transformation $\{\mathbf{R}_n^t, \mathbf{t}_n^t\}$ fixed:

$$\{C_n\}_{n=1}^m = \min_{C_n} \sum_{n=1}^{m} \sum_{\mathbf{x} \in C_n} \|\mathbf{R}_n^t \mathbf{x} + \mathbf{t}_n^t - \mathbf{y}^t\|^2. \tag{7}$$

For each cluster C_n, we define its centroid in the canonical frame as $\mathbf{c}_n$:

$$\mathbf{c}_n = \frac{\sum_{\mathbf{x} \in C_n} \mathbf{x}}{\sum_{\mathbf{x} \in C_n} 1}, \tag{8}$$

and so is its corresponding vertex centroid $\mathbf{c}_n^t$ in live frame t. Then Eq. (7) can be rewritten by applying the closed-form solution of $\{\mathbf{R}_n^t, \mathbf{t}_n^t\}$:

$$\{C_n\}_{n=1}^m = \min_{C_n} \sum_{n=1}^{m} E^*(C_n), \tag{9}$$

where:

$$E^*(C_n) = \sum_{\mathbf{x} \in C_n} [(\mathbf{x} - \mathbf{c}_n)^\top (\mathbf{x} - \mathbf{c}_n) + (\mathbf{y}^t - \mathbf{c}_n^t)^\top (\mathbf{y}^t - \mathbf{c}_n^t)] - 2 \sum_{q=1}^{3} \sigma_{nq}^t, \tag{10}$$

and σ_{nq}^t is the singular value of cross covariance matrix $\mathbf{A}^t(C_n)$:

$$\mathbf{A}^t(C_n) = \sum_{\mathbf{x} \in C_n} (\mathbf{x} - \mathbf{c}_n)(\mathbf{y}^t - \mathbf{c}_n^t)^\top. \tag{11}$$

Equation (9) can be solved in two stages: initial clustering by merging operations, and clustering optimization by swapping operations.

Initial Clustering by Merging Operations: Inspired by the surface simplification idea of Cai et al. [2], we define merging operations to partition the nodes

of the canonical frame into m clusters as initialization. It will result in a good initial clustering for the next stage of swapping-based optimization.

In the first step of the merging operation, each node is treated as an individual cluster, which forms potential merging pairs with its neighboring clusters. When a pair of clusters is merged to a new cluster, a merge cost is calculated and associated with this merge operation. For a merging operation $(C_i, C_j) \rightarrow C_k$, the merging cost is defined as: $E^*(C_k) - E^*(C_i) - E^*(C_j)$. Figure 3 shows the concept of such an operation.

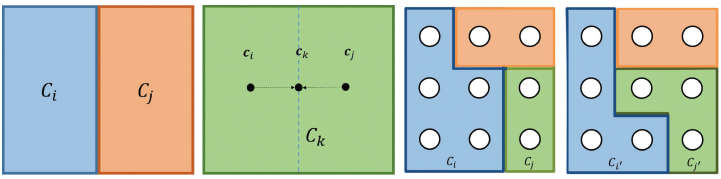


Fig. 3. Merging and swapping operation for a pair of clusters. C_i and C_j is merged to C_k. (a) Before merging. (b)After merging, the centroid of new cluster $\mathbf{c}_k$ is different from both $\mathbf{c}_i$ and $\mathbf{c}_j$. (c) The center node $\mathbf{x}_l$ is swapped from C_i to C_j. Clustering before swapping: region *Blue* is C_i, and region *Green* is C_j. *Circle* represents nodes in clusters. (d) Clustering after swapping: region *Blue* is $C_{i'}$, and region *Green* is $C_{j'}$. After the swapping operation, the belonging of node $\mathbf{x}_l$ is changed from $C_{i'}$ to $C_{j'}$. (Color figure online)

A heap is maintained to store all possible merging operations in the current clustering, paired with the corresponding costs as the key value. Next, the least-cost merging is performed. Each time after the least-cost pair is selected from the heap, only a local update is needed to maintain the validity of the merging heap: the remaining pairs of the two merged clusters in the heap are deleted, and the potential merging between the new cluster and its direct neighbors are inserted. This step is iteratively performed until the number of clusters reaches m. As shown in Supplementary Material, the merging cost can be computed with $O(1)$ complexity, which is independent of the number of nodes in each cluster.

Clustering Optimization by Swapping Operations: Only greedily merging the least-cost pair of clusters as initialization cannot guarantee the optimal solution for Eq. (9). The second stage of swapping operations can continue to optimize it based on the above initialization. In the greedy merging process, each time a pair of clusters is merged, nodes from both clusters are bound to reside in the same new cluster. Those nodes cannot freely decide where to go, so a swapping operation is necessary to relax the binding between nodes and clusters from the above initialization.

The swapping operation is defined as swapping a boundary node from its belonged cluster C_i to swapping-available clusters. A boundary node $\mathbf{x}_l$ is the node which resides in C_i and has at least a neighboring node $\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_l)$ that

does not belong to C_i. We denote the set of clusters that $\mathcal{N}(\mathbf{x}_l)$ reside in as swapping-available clusters $NC_{\mathbf{x}_l}$. Whether swapping $\mathbf{x}_l$ from C_i to $C_j \in NC_{\mathbf{x}_l}$ is determined by the sign of energy change after the swapping operation. We call this energy change as swapping cost.

If the swapping cost is less than 0, it means this swapping can decrease the energy of our objective function Eq. (6). Otherwise, the current clustering is best suitable for the tested node, and there is no further operation needed. If there is more than one cluster in $NC_{\mathbf{x}_l}$ that can optimize the clustering, we select the one that leads to the largest decrease of energy. To be more precise, as shown in the Supplementary Material, the swapping cost can be efficiently computed with $O(1)$ complexity, which is independent of the number of nodes in each cluster. Figure 3(c) and (d) illustrates a typical swapping operation by swapping the center node $\mathbf{x}_l$ from C_i to C_j which results in new clusters $C_{i'}$ and $C_{j'}$.

In order to achieve real-time reconstruction, we need to accelerate the segmentation step. We only employ the merging operation after registering the mesh of canonical frame with the second frame. For the segmentation step of later frames, we initialize the clustering with the previous result and then perform swapping based on such initialization. For newly added nodes after depth fusion, their cluster belongings are determined by their closest existing neighbor nodes. Because of such initialization, the maintenance of heap structure is no longer needed. We can use GPU to compute the cross covariance matrix $\mathbf{A}^t(C_n)$ and the energy $E^*(C_n)$ in parallel according to Eqs. (10) and (11).

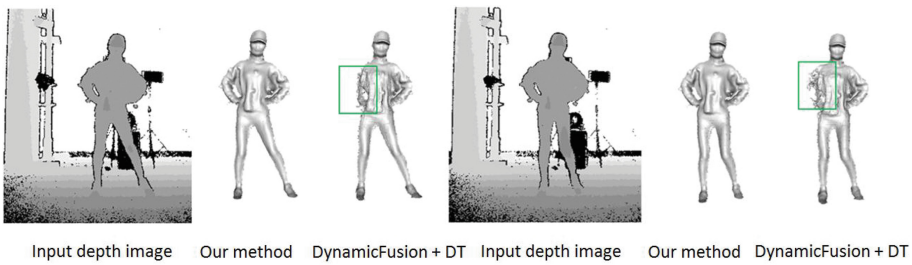


Fig. 4. Segmentation improves the reconstruction result for fast inter-frame motion in direction parallel to the screen. In each group from left to right: input depth image, the reconstructed result of our method, and the result of DynamicFusion with only DT.

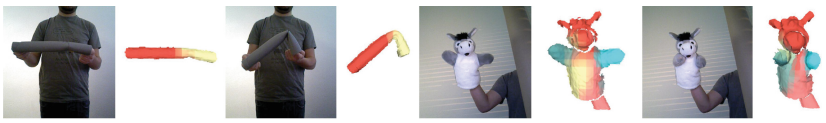
Figure 4 shows a comparison example between our method and DynamicFusion with DT in the registration step. Although both cases employ the DT-based correspondences computing, the reconstruction result of our method is much better because the introduction of segmentation.

The number of clusters can be given as a constant, or can be estimated dynamically by adding an energy threshold in the merging step. When the increased energy after one merging operation is bigger than the threshold, the

merging step stops. This mechanism can automatically determine the number of clusters. Considering real-time performance, we can break any cluster with error higher than a given threshold into two new clusters and adjust the boundaries of new clusters in the swapping step. Cluster breaking can be achieved by merging all original nodes into two new clusters. Because only a small number of nodes in that cluster needs to be re-merged, the real-time performance can still hold. Due to the space limit of the paper, details about dynamic clustering such as how the number of clusters influences the results, and the comparison of reconstruction results can be found in our Supplementary Material.

(a) Body turning (b) Boxing (c) Rolling arms

Fig. 5. Selected human motion reconstruction results by our system. From left to right for each motion: input depth, reconstructed geometry, segmentation.



(a) Bending cloth pipe (b) Playing "donkey" hand puppet

Fig. 6. Selected non-human reconstruction results by our system. (a) shows our reconstructed results of bending a cloth pipe at the 1/4 location; (b) shows our results of playing a "donkey" hand puppet.

5 Results

In this section, we describe the performance of our system and details of its implementation, followed by qualitatively comparisons with state-of-the-art methods and evaluations. We captured more than 10 sequences with persons performing natural body motions like "Boxing","Dancing", "Body turning","Rolling arms", and "Crossing arms", etc. We have also experimented our algorithm on an existing dataset for articulated model reconstruction [26].

Figure 5 shows some of our reconstruction results for motions "Body turning","Boxing", and "Rolling arms". Our ArticulatedFusion system enables simultaneous geometry, motion, and segmentation reconstruction. As shown in

Fig. 5(c), the human body is segmented by deformation clustering so hands, arms and head are segmented out because of their articulated motion property.

Figure 6 shows that our system can also reconstruct geometry, motion, and segmentation for non-human motion sequences without any prior skeleton information or template. It automatically learns the segmentation from control nodes clustering. As shown in the 2nd and 4th columns of Fig. 6(a) and (b), faithful segmentation can be automatically generated during the reconstruction process with motions and fine geometry.

5.1 Performance

Our system is fully implemented on a single NVIDIA GeForce GTX 1080 graphics processing unit using both the OpenGL API and the NVIDIA CUDA API. The pipeline runs at 34–40 ms per frame on average. The time breaking of main steps is as follows (Table 1): the preprocessing of the depth information (including bilateral filtering and calculation of the depth normals) requires 1 ms; the rendering of the results requires 1 ms. For two-level node motion optimization, we run 5 and 2 iterations respectively. In each iteration, to solve the linear equation, we run 10 iterations of PCG. The voxel resolution is 5 mm. For each vertex, 8 nearest nodes is used as its control node. The number of segments ranges from 6 to 40. In all examples, we capture the depth stream using a Kinect v2 with 512×424 depth image resolution.

Table 1. Average computation time per frame for several motions (ms). Column "Init" is the time to initialize and update node graph. Column "DT" is the time to calculate distance transform. Columns "Level 1" and "Level 2" represent the time to solve level-1 and level-2 registration. Column "TSDF" represents the time to perform TSDF integration. Column "Seg" is the time of segmetation.

	# of Segs	# of Nodes	Init (ms)	DT (ms)	Level 1 (ms)	Level 2 (ms)	TSDF (ms)	Seg (ms)	Total (ms)
Boxing	20	1442	2.7	4.9	8.3	13.9	4.7	2.5	37.0
Rolling arms	20	1914	3.4	4.6	8.5	15.0	4.9	2.7	39.1
Crossing arms	12	1130	2.5	4.6	7.1	13.4	5.1	1.9	34.6
Dancing	30	1569	3.0	4.7	9.0	14.4	5.2	3.0	39.3
Body turning	20	2002	3.5	4.7	8.6	14.5	4.8	2.8	38.9

5.2 Comparisons and Evaluations

We compare our ArticulatedFusion with two state-of-the-art methods DynamicFusion [17] and VolumeDeform [10]. Figure 7 shows visual comparisons on motion "Dancing". We can see both DynamicFusion and VolumeDeform fail in the left

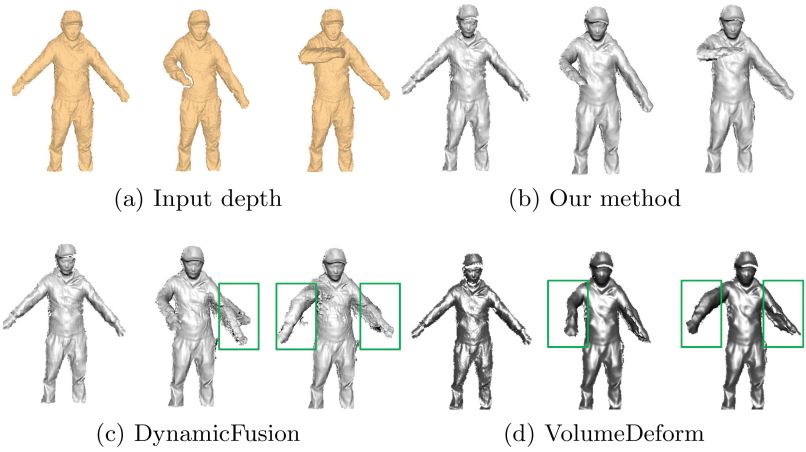


Fig. 7. Visual comparisons of the results between: (b) our method, (c) DynamicFusion [17], and (d) VolumeDeform [10], with input depth images shown in (a).

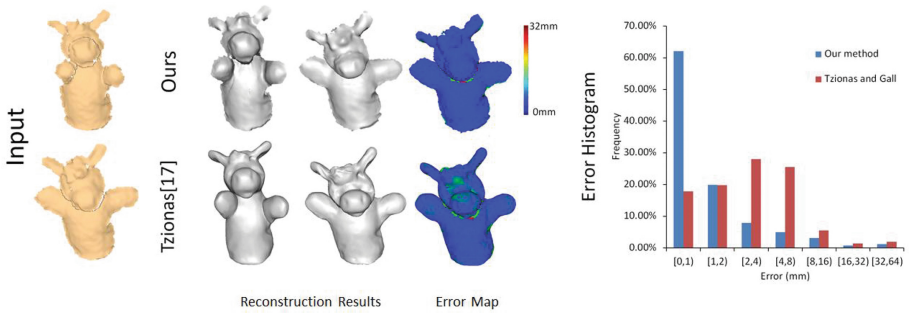


Fig. 8. Non-human object reconstruction comparison on "donkey" hand puppet.

and right arms region. Our method generates more faithful results for motions in tangential direction or motions having large occlusions.

To further quantitatively evaluate our reconstructed segmentation and motion, we compare our results with the other state-of-the-art methods by using the Vicon-captured groundtruth data from BodyFusion [29]. In Fig. 9, it is noted that our reconstruction error is comparative to BodyFusion (slightly higher though), but our method is more general and can be applied to dynamic scenes where Kinect-based skeleton is not available, such as non-human-body motions (Figs. 6, 8 and 10(b)) and human-body motions without initial skeleton information (Fig. 10(a)). In Fig. 10(a), the skeleton of the person on the back cannot be provided by Kinect because of high occlusion in the body. It is noted that the highlighted head and leg part is well reconstructed with the help of our segmentation, while they are not correctly tracked by DynamicFusion.

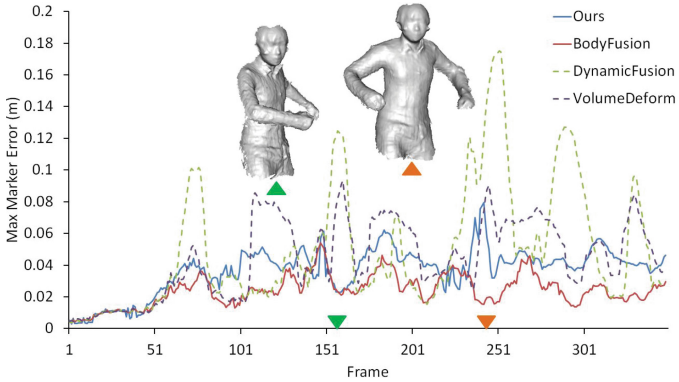


Fig. 9. Quantitative comparison: max marker errors of our method, BodyFusion, DynamicFusion and VolumeDeform for a motion sequence.

We compare our method with two other reconstruction methods that can reconstruct non-human objects. Figure 8 shows a detailed comparison of the near-articulated example "donkey" hand puppet with the template-based reconstruction result in Tzionas and Gall's work [26]. The first column of Fig. 8 shows two input depth images. From both the error map and the error histogram, we can find our method has better error distribution than theirs. In order to have a fair comparison in error histogram, we only count visible vertices in both cases. Because of the introduction of segmentaion in the registration step, our method is more robust for fast motion. Figure 10(b) shows another example of non-human object reconstruction. In VolumeDeform [10], their reconstruction fails when skipping 4 or more frames before next frame. But our method can still get a good result, while every petal of the sunflower is clustered as one segment.

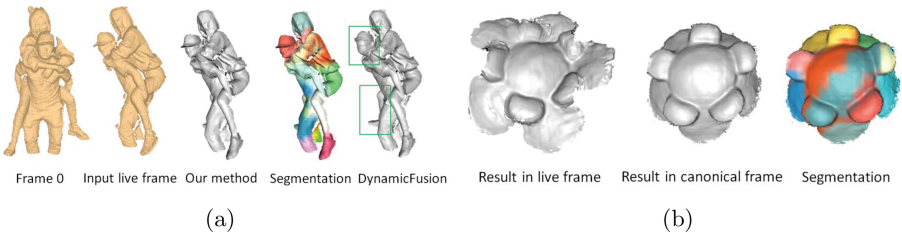


Fig. 10. (a) Reconstruction result comparison of our method and DynamicFusion [17]. (b) Reconstruction result of the failure case in VolumeDeform [10] (shown in their Fig. 9) for 5x speed input (skipping 5 frames).

6 Conclusion and Future Work

In this paper, we have seen that our two-level node optimization equipped efficient node graph segmentation enables better reconstructions for tangential and occluded motions, for non-rigid human and non-human motions captured with a single depth camera. We believe that our system represents a step forward towards a wider adoption of depth cameras in real-time applications, and opens the door for leveraging the high-level semantic information in reconstruction, e.g. differentiating dynamic and static scenes as shown in MixedFusion [31].

Our system still has limitations in the reconstruction of very fast motions because of the blurred depth and our reliance on ICP-based local correspondence matching. Topological change of surfaces is also difficult to handle. In the future we would like to consider the integration of color information [8,10] for further improvement on the motion optimization, and extracting a consistent tree-based skeleton structure from our segmentation.

Acknowledgement. We would like to thank the reviewers for their valuable comments. We are grateful to Matthias Innmann for the help on comparison results of VolumeDeform, Tao Yu for providing their Vicon-based ground-truth marker data in BodyFusion, and Dimitrios Tzionas for providing their data. This work was partially supported by National Science Foundation under grant number IIS-1149737. Chao would like to thank the support provided by Hua Guo during the preparation for this paper.

References

1. Cai, Y., Guo, X.: Anisotropic superpixel generation based on Mahalanobis distance. Comput. Graph. Forum **35**(7), 199–207 (2016)
2. Cai, Y., Guo, X., Liu, Y., Wang, W., Mao, W., Zhong, Z.: Surface approximation via asymptotic optimal geometric partition. IEEE Trans. Vis. Comput. Graph. **23**(12), 2613–2626 (2017)
3. Cao, C., Weng, Y., Lin, S., Zhou, K.: 3D shape regression for real-time facial animation. ACM Trans. Graph. **32**(4), 41 (2013)
4. Chang, W., Zwicker, M.: Global registration of dynamic range scans for articulated model reconstruction. ACM Trans. Graph. (TOG) **30**(3), 26 (2011)
5. Dou, M., et al.: Fusion4D: real-time performance capture of challenging scenes. ACM Trans. Graph. **35**(4), 114 (2016)
6. Guo, K., Xu, F., Wang, Y., Liu, Y., Dai, Q.: Robust non-rigid motion tracking and surface reconstruction using L_0 regularization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3083–3091 (2015)
7. Guo, K., Xu, F., Wang, Y., Liu, Y., Dai, Q.: Robust non-rigid motion tracking and surface reconstruction using L_0 regularization. IEEE Trans. Vis. Comput. Graph. (2017)
8. Guo, K., Xu, F., Yu, T., Liu, X., Dai, Q., Liu, Y.: Real-time geometry, albedo, and motion reconstruction using a single RGB-D camera. ACM Trans. Graph. **36**(3), 32 (2017)
9. Horn, B.K.P.: Closed-form solution of absolute orientation using unit quaternions. J. Opt. Soc. Am. A **4**(4), 629–642 (1987)

10. Innmann, M., Zollhöfer, M., Nießner, M., Theobalt, C., Stamminger, M.: VolumeDeform: real-time volumetric non-rigid reconstruction. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9912, pp. 362–379. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46484-8_22
11. James, D.L., Twigg, C.D.: Skinning mesh animations. ACM Trans. Graph. **24**(3), 399–407 (2005)
12. Le, B.H., Deng, Z.: Smooth skinning decomposition with rigid bones. ACM Trans. Graph. **31**(6), 199 (2012)
13. Li, H., Adams, B., Guibas, L.J., Pauly, M.: Robust single-view geometry and motion reconstruction. ACM Trans. Graph. (TOG) **28**(5), 175 (2009)
14. Li, H., Yu, J., Ye, Y., Bregler, C.: Realtime facial animation with on-the-fly correctives. ACM Trans. Graph. **32**(4), 42-1 (2013)
15. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3D surface construction algorithm. In: ACM siggraph computer graphics, vol. 21, pp. 163–169. ACM (1987)
16. Murray, R.M., Li, Z., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC Press, Boca Raton (1994)
17. Newcombe, R.A., Fox, D., Seitz, S.M.: DynamicFusion: reconstruction and tracking of non-rigid scenes in real-time. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 343–352 (2015)
18. Newcombe, R.A., et al.: KinectFusion: real-time dense surface mapping and tracking. In: 10th IEEE international symposium on Mixed and Augmented Reality, pp. 127–136 (2011)
19. Pekelny, Y., Gotsman, C.: Articulated object reconstruction and markerless motion capture from depth video. Comput. Graph. Forum **27**(2), 399–408 (2008)
20. Pons-Moll, G., Baak, A., Helten, T., Müller, M., Seidel, H.P., Rosenhahn, B.: Multisensor-fusion for 3D full-body human motion capture. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 663–670 (2010)
21. Slavcheva, M., Baust, M., Cremers, D., Ilic, S.: KillingFusion: non-rigid 3D reconstruction without correspondences. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
22. Slavcheva, M., Baust, M., Ilic, S.: SobolevFusion: 3D reconstruction of scenes undergoing free non-rigid motion. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
23. Sorkine, O.: Least-squares rigid motion using SVD. Technical notes (2017)
24. Tagliasacchi, A., Schröder, M., Tkach, A., Bouaziz, S., Botsch, M., Pauly, M.: Robust articulated-ICP for real-time hand tracking. Comput. Graph. Forum **34**(5), 101–114 (2015)
25. Tkach, A., Pauly, M., Tagliasacchi, A.: Sphere-meshes for real-time hand modeling and tracking. ACM Trans. Graph. **35**(6), 222 (2016)
26. Tzionas, D., Gall, J.: Reconstructing articulated rigged models from RGB-D videos. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9915, pp. 620–633. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49409-8_53. http://files.is.tue.mpg.de/dtzionas/Skeleton-Reconstruction
27. Vlasic, D., Baran, I., Matusik, W., Popović, J.: Articulated mesh animation from multi-view silhouettes. ACM Trans. Graph. **27**(3), 97 (2008)
28. Wand, M., et al.: Efficient reconstruction of nonrigid shape and motion from real-time 3D scanner data. ACM Trans. Graph. **28**(2), 15 (2009)
29. Yu, T., et al.: Bodyfusion: real-time capture of human motion and surface geometry using a single depth camera. In: The IEEE International Conference on Computer Vision (ICCV). IEEE, October 2017

30. Yu, T., et al.: Doublefusion: real-time capture of human performances with inner body shapes from a single depth sensor. In: The IEEE International Conference on Computer Vision and Pattern Recognition(CVPR). IEEE, June 2018
31. Zhang, H., Xu, F.: MixedFusion: real-time reconstruction of an indoor scene with dynamic objects. IEEE Trans. Vis. Comput. Graph. (2017)
32. Zollhöfer, M., et al.: Real-time non-rigid reconstruction using an RGB-D camera. ACM Trans. Graph. **33**(4), 156 (2014)