Robust shape estimation for 3D deformable object manipulation

Tao Han, Xuan Zhao, Peigen Sun, and Jia Pan

Existing shape estimation methods for deformable object manipulation suffer from the drawbacks of being off-line, model dependent, noise-sensitive or occlusion-sensitive, and thus are not appropriate for manipulation tasks requiring high precision. In this paper, we present a real-time shape estimation approach for autonomous robotic manipulation of 3D deformable objects. Our method fulfills all the requirements necessary for the high-quality deformable object manipulation in terms of being real-time, model-free and robust to noise and occlusion. These advantages are accomplished using a joint tracking and reconstruction framework, in which we track the object deformation by aligning a reference shape model with the stream input from the RGB-D camera, and simultaneously upgrade the reference shape model according to the newly captured RGB-D data. We have evaluated the quality and robustness of our real-time shape estimation pipeline on a set of deformable manipulation tasks implemented on physical robots.

1. Introduction

Autonomous manipulation of deformable objects is an important and challenging topic in robotics, and recently it attracts much interest due to its potential applications in robot-assisted surgery [11, 16, 21, 27] and service robots, including garments folding [6, 13, 26], ironing [12], and robot-assisted dressing [7]. Despite the difference in their technical details employed for specific tasks, most existing systems for deformable object manipulation can be described using the same shape control framework as shown in the top row of Figure 1. Given a desired shape of the target object, the robot system iteratively estimates the object's shape state according to either the sensor
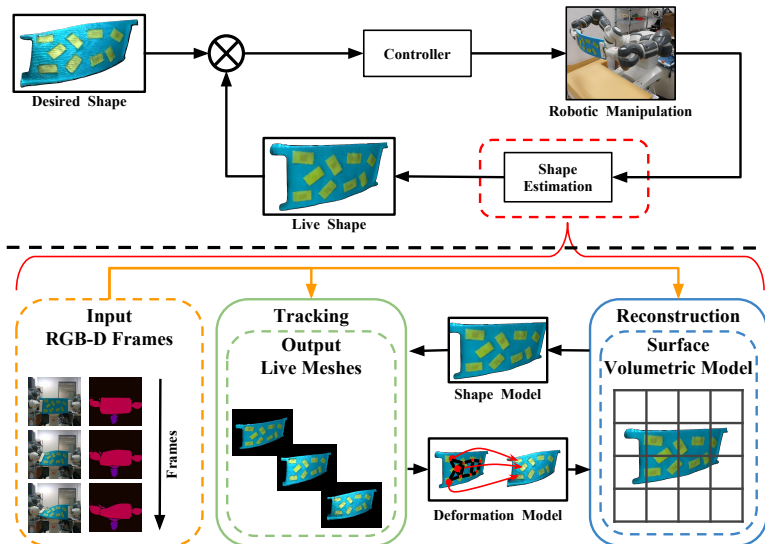
Figure 1. Overview of our shape estimation pipeline for deformable object manipulation. **Top row** is the entire control loop for deformable object manipulation. In this work, we focus on solving the shape estimation problem, as highlighted in the red-dashed box. An ideal shape estimation approach should fulfill the requirements of being real-time, model-free and robust to noise and occlusion. **Bottom row** illustrates the pipeline of our shape estimation system. Our system takes the frame sequence captured by a RGB-D camera as input. It is composed of two parallel threads, namely, *tracking* and *reconstruction*. These two threads estimate and update the deformation model and the shape model iteratively when new RGB-D frames are streaming into the system. Please refer to Section 2.3 for more details about our pipeline.

measurements or the simulation results, and applies the difference between the object's current shape state and the desired shape as an error signal to generate a control output to reduce or eliminate the error by deforming the object appropriately. The entire process repeats until the shape state converges to the desired value.

In this work, we focus on the shape estimation problem arising from the aforementioned control loop (as shown in the bottom row of Figure 1). In particular, to represent the shape state of a deforming object, two models are needed: a *shape model* encoding the geometry and texture of the

deformable object's surface, and a *deformation model* describing the deformation kinematics or dynamics of the object surface. When provided with sufficient prior knowledge about these two models, modern physically-based simulators such as [1–3] can provide a long-horizon prediction about the shape state of the underlying deformable object. As a result, recent work such as [7, 21–23] embedded a physically-based engine into their pipeline and designed or learned a manipulation control according to the shape feedbacks provided by the simulator. The resulting model-based control could be robust to noise and occlusion, if the simulator has been carefully calibrated to be consistent with the real-world physics, which unfortunately is difficult in practice. In particular, the quality of the simulation is extremely sensitive to the model parameters, and this is considered as one main bottleneck of the model-based control for tasks involving deformable objects. In addition, running a physically-based simulation is time-consuming and thus infeasible for estimating fast or large deformation in real time.

For designing real-time and model-free manipulation system, recent work [8, 10, 15, 17] used the *shape servoing* technique deriving from the traditional *visual servoing* framework [9]. Instead of representing the object's shape via dense mesh structure as in the model-based method, the shape servoing method approximates the shape state using sparse key features extracted from image data. Because the feature descriptor is usually low-dimensional, the shape servoing method can learn the control policy between the shape feature and the manipulator motion directly in a data-driven manner. Such online policy learning framework makes the shape servoing method independent from an explicit deformation model to achieve shape control. However, existing methods in this direction still suffer from several drawbacks:

- **Low-resolution shape modeling**: Using sparse features as the shape feedback will omit some geometric details of the object. In other words, even when the feature representation of the object's current configuration perfectly matches the target feature vector, there is no guarantee that the object's shape can completely fit into its target shape. This limitation can be problematic for manipulation tasks which require high-precision goal reaching. As a result, a rich representation for the object deformation state is desirable.

- **Noise-sensitive feature extraction**: Most existing shape servoing approaches extract shape features from a single frame of image. The extracted vector can be unreliable for closed-loop control due to the

noises in the feedback image that are ubiquitous in real robotic systems. As a result, we need a sophisticated method to obtain robust features from a sequence of feedback inputs.

- **Occlusion-sensitive feature correspondences**: Existing shape servoing methods rely on a feature descriptor to determine inter-frame feature correspondences. However, many deformable objects lack visually significant feature points, and thus they must have additional markers mounted on the surface to provide reliable feedbacks. Such marker-based workaround leads to inconvenience for practical applications. Moreover, most shape servoing systems assume that the full state of the object's surface can be observed all the time during the task. As a result, when some feature points or markers become invisible to the visual sensor due to occlusion, these systems may fail to capture enough feature vectors for servoing control.

The aforementioned problems in both model-based method and shape servoing method motivated us to propose a novel shape estimation method in this paper. Our method satisfies the requirements of being real-time, model-free and robust to noise and occlusion, and thus can be easily embedded into current robotic systems for autonomous manipulation of general 3D soft objects. In our work, we further divide the shape estimation problem into two subproblems, namely *tracking* and *reconstruction* (as shown in the bottom row of Figure 1). In the tracking phase, we estimate an inter-frame deformation model through non-rigid registration between a reference shape model and the depth images provided by an RGB-D camera. In the reconstruction phase, we integrate multiple RGB-D images into the reference shape model according to the estimated deformation model. One key contribution of our work is that our simultaneous tracking and reconstruction framework can capture the surface model of a deforming object, while gradually complete and refine its details based on new RGB-D measurements. Because the generated surface model is of high precision and is also robust to single-frame noise and occlusion, it serves as an excellent feedback signal for shape control.

2. Overview

In this section, we first present the mathematical notation to define the shape and deformation models employed in our work. Then we outline the pipeline of our simultaneous shape tracking and reconstruction framework.

$\mathcal{F}^0, \mathcal{F}^t$	reference frame, live frame
$\mathcal{M}^0, \mathcal{M}^t$	mesh model defined in the corresponding frame space
$\mathbf{x}^0, \mathbf{x}^t$	voxel defined in the corresponding frame space
$\mathbf{v}_m^0, \mathbf{n}_m^0$	the m-th vertex element and normal element of mesh $\mathcal{M}^0$
$\mathbf{v}_m^t, \mathbf{t}_m^t$	the m-th vertex element and normal element of mesh $\mathcal{M}^t$
$\mathbb{V}$	reference volume defined in the space of $\mathcal{F}^0$
$\mathcal{D}(\cdot)$	TSDF component of the voxel in $\mathbb{V}$
$\mathcal{C}(\cdot)$	color component of the voxel in $\mathbb{V}$
$\Omega(\cdot)$	weight component of the voxel in $\mathbb{V}$
$d(\cdot)$	TSDF component of the voxel contributed by $\mathcal{F}^t$
$c(\cdot)$	color component of the voxel contributed by $\mathcal{F}^t$
$\omega(\cdot)$	weight component of the voxel contributed by $\mathcal{F}^t$
$\mathcal{G}$	deformation model of the target soft object
$\mathbf{T}_{\text{rigid}}$	rigid component separated from $\mathcal{G}$
$\mathcal{G}_{\text{graph}}$	non-rigid component of $\mathcal{G}$ represented as graph
$\mathbf{g}_i$	the i-th node of the graph $\mathcal{G}_{\text{graph}}$ defined in the space of $\mathcal{F}^0$
$\mathbf{T}_i$	local deformation defined in $\mathbf{g}_i$
σ_i	effective radius of $\mathbf{T}_i$
$\mathcal{N}_i$	neighbor set of $\mathbf{g}_i$
$\mathcal{W}(\cdot; \mathcal{G})$	deformation function maps $\mathcal{M}^0$ to $\mathcal{M}^t$, parameterized by $\mathcal{G}$
$C^t(\cdot)$	color map of $\mathcal{F}^t$
$D^t(\cdot)$	depth map of $\mathcal{F}^t$
$V^t(\cdot)$	vertex map extracted from $D^t(\cdot)$
$N^t(\cdot)$	normal map extracted from $D^t(\cdot)$

Table 1. Nomenclature

2.1. Shape representation

The main objective of deformable object manipulation is to deform the object's surface from an initial shape into a desired shape. The inner state of the target object is actually ignored for shape control in most current methods. Therefore in this work we are only interested in how to use an appropriate shape representation to model the surface state. To generate high-quality shape model containing rich geometry and texture details, one commonly used solution, similar to the model-based method, is to represent the surface based on the mesh structure extracted from the RGB-D image data [20]. While such mesh-based representation is suitable to serve as the reference model for shape tracking, its graph structure makes it hard to be
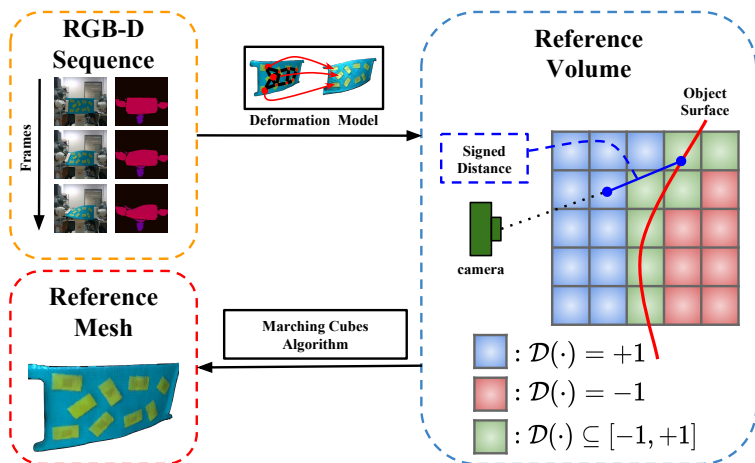
Figure 2. Illustration of the reference volume $\mathbb{V}$ in 2D case. The signed distance value of each voxel is defined as the distance from its center to its closest point on the surface (shown as the solid blue, positive when the voxel is on the "front" side of the surface and negative when it is on the "back" side of the surface). We further truncate the signed distance with a threshold to make its value belong to the range $[-1, +1]$. A reference mesh can be extracted from the reference volume $\mathbb{V}$ by locating the zero-level of the TSDF value.

fused with multiple image frames for shape reconstruction. Instead, encoding the surface geometry and texture into a 3D volumetric grid structure is more feasible since the shape reconstruction progress can be implemented efficiently via parallel operation on the grids.

To combine the advantages of both representations, we follow previous work [5, 18, 19] by projecting multiple image frames data back into the space of a reference frame $\mathcal{F}^0$ (which is usually set as the initial frame) according to the estimated inter-frame deformations and then integrating these frames into a *reference mesh* $\mathcal{M}^0$. For efficient image integration, the reference mesh $\mathcal{M}^0$ is maintained via a discrete truncated signed distance function (TSDF) volume (as illustrated in Figure 2), which we denote as the *reference volume* $\mathbb{V}$. In this reference volume $\mathbb{V}$, the surface geometry is voxelized as $\{\mathcal{D}(\mathbf{x}^0), \Omega(\mathbf{x}^0)\}$, where $\mathcal{D}(\mathbf{x}^0) \subseteq [-1, +1]$ encodes the truncated signed distance value for each voxel $\mathbf{x}^0$, and $\Omega(\mathbf{x}^0) \subseteq [0, 1]$ is the associated weight. The content of each voxel will be updated independently for image integration. To obtain a high-quality mesh with texture, we also maintain

the RGB information $\mathcal{C}(\mathbf{x}^0) \subseteq [0, 255]^3$ in each voxel. As a summary, our reference volume can be represented as $\mathbb{V}(\mathbf{x}^0) = \{\mathcal{D}(\mathbf{x}^0), \mathcal{C}(\mathbf{x}^0), \Omega(\mathbf{x}^0)\}$.

From the reference volume $\mathbb{V}$, we extract the reference mesh $\mathcal{M}^0$ with Marching Cubes algorithm [14]. The reference mesh $\mathcal{M}^0$ can be further deformed according to the estimated inter-frame deformation model to obtain the *live mesh* $\mathcal{M}^t$, which indicates the object shape in the live frame $\mathcal{F}^t$. For the convenience of discussion, we define the mesh vertices and corresponding normals as $\{\mathbf{v}_m, \mathbf{n}_m\}_{m=1}^M$ in this work, where M is the number of vertices in the mesh.

2.2. Deformation representation

To achieve simultaneous shape tracking and reconstruction, we need a model to formulate the deformation from the reference frame $\mathcal{F}^0$ to the live frame $\mathcal{F}^t$. In consideration of our model-free requirement, skeleton-based kinematic models for articulated objects are infeasible to represent the deformation of general soft objects. One possible solution is to model the deformation based on Eulerian (grid-based) method or Lagrangian (particle-based) method used in fluid simulation [4]. While both methods can provide high-quality representation for general deformation based on their dense structures, such high-dimensional models are not feasible for real-time estimation.

As a trade-off between complexity and precision, we employ the sparse deformation graph model [25] with reduced dimensions for real-time implementation. In this method, the graph nodes are uniformly sampled from the mesh model to have a layout roughly conform to the object's shape. Then the whole deformation is divided into a set of local transformations, which are then assigned to the graph nodes one-to-one. The graph nodes have overlapping domain of influence with their neighboring nodes in local transformations. Thus for any given point in the nearby space of the graph nodes, a smooth deformation function can be computed via interpolation of the local transformations in the point's nearest graph nodes.

Moreover, to avoid over-fitting of the deformation graph model during estimation, a regularization constraint is also needed. In our work, we regularize the deformation graph via the widely used *as-rigid-as-possible* (ARAP) constraint [24], which penalizes inconsistent local transformations between neighboring graph nodes. Such penalty function is usually be represented geometrically as the graph edges.

Except the graph model, we further divide the global rigid component from the total deformation and model it separately. Overall, our deformation model can be represented as $\mathcal{G} = \mathbf{T}_{\text{rigid}} \cup \mathcal{G}_{\text{graph}}$. Here $\mathbf{T}_{\text{rigid}} \in \text{SE}(3)$ defines

the separated global rigid transformation. $\mathcal{G}_{\text{graph}}$ denotes the non-rigid component of $\mathcal{G}$ represented in the graph model. We further parameterize the graph model as $\mathcal{G}_{\text{graph}} = \{\mathbf{T}_i, \mathbf{g}_i, \sigma_i, \mathcal{N}_i\}_{i=1}^{K}$. $\mathbf{T}_i \in \text{SE}(3)$ indicates the local transformation in the i-th node. $\mathbf{g}_i$ represents the position of the i-th node in the reference frame $\mathcal{F}^0$. σ_i defines the effective radius of $\mathbf{T}_i$. The neighbor set $\mathcal{N}_i$ contains the indices of those nodes which are connected with the i-th node by graph edges. These nodes $\{\mathbf{g}_j | j \in \mathcal{N}_i\}$ are considered as the closest neighbors of the i-th node. Note that in our method, $\{\mathbf{g}_i, \sigma_i, \mathcal{N}_i\}_{i=1}^{K}$ remains constant during estimation, and thus our deformation model can be fully parameterized as $\mathcal{G} = \{\mathbf{T}_{\text{rigid}}\} \cup \{\mathbf{T}_i\}_{i=1}^{K}$.

To deform the reference mesh $\mathcal{M}^0$ according to the above model $\mathcal{G}$, we first assign each mesh vertex $\mathbf{v}_m^0 \in \mathcal{M}^0$ to its k-nearest nodes $\mathcal{S}_m \subseteq \{1, \ldots, K\}$ on the graph model of $\mathcal{G}$ based on a set of skinning weights $\{\omega_{m,k} : k \in \mathcal{S}_m\} \subseteq [0, 1]$. The skinning weight is calculated as $\omega_{m,k} = \frac{1}{Z} \exp(-\|\mathbf{v}_m^0 - \mathbf{g}_k\|^2 / 2\sigma_k^2)$, where Z is a normalization factor ensuring $\sum_k \omega_{m,k} = 1$. Then we calculate the deformed vertex $\mathbf{v}_m^t$ in the live frame $\mathcal{F}^t$ based on the following blending function:

$$(1) \qquad \mathbf{v}_m^t = \mathcal{W}(\mathbf{v}_m^0; \mathcal{G}) = \mathbf{T}_{\text{rigid}} \sum_{k \in \mathcal{S}_m} \omega_{m,k} \mathbf{T}_k \mathbf{v}_m^0.$$

Similarly, the corresponding normal $\mathbf{n}_m^0$ of the vertex $\mathbf{v}_m^0$ can be deformed using the following blending function:

$$(2) \qquad \mathbf{n}_m^t = \mathcal{W}(\mathbf{n}_m^0; \mathcal{G}) = \mathbf{T}_{\text{rigid}} \sum_{k \in \mathcal{S}_m} \omega_{m,k} \mathbf{T}_k \mathbf{n}_m^0.$$

2.3. Our system

As demonstrated in Figure 1, our system takes the image stream provided by an RGB-D camera as input. It is composed of two parallel threads: tracking and reconstruction. The tracking thread takes charge of the real-time estimation of the deformation model $\mathcal{G}$. It aligns the live RGB-D frame $\mathcal{F}^t$ with the reference mesh model $\mathcal{M}^0$ for geometric consistency. To capture the surface geometry for alignment, the tracking thread extracts dense features from the received depth map D^t of the live frame $\mathcal{F}^t$, including a vertex map V^t and a corresponding normal map N^t. At the core of this thread is a highly-efficient GPU solver which optimizes the deformation model $\mathcal{G}$ for frame-to-model alignment under the regularization constraint. We implement the optimization solver based on a kernel merged *preconditioned conjugate gradient* (PCG) algorithm using CUDA.

Given the estimation of the deformation model $\mathcal{G}$, the reconstruction thread computes the voxel-to-pixel correspondences between the reference volume $\mathbb{V}$ and the live frame $\mathcal{F}^t$ and updates the content in each voxel of $\mathbb{V}$ accordingly. After the previous volume fusion operation, the reconstruction thread extracts a new reference mesh $\mathcal{M}^0$ from the reference volume $\mathbb{V}$ and obtains the associated live mesh $\mathcal{M}^t$ based on the estimated deformation model $\mathcal{M}^t = \mathcal{W}(\mathcal{M}^0; \mathcal{G})$.

2.3.1. Tracking. The objective of the tracking thread is to provide accurate estimation of the deformation model to assist the reconstruction thread. In the tracking step, we optimize the deformation model to obtain the best frame-to-model alignment between the live frame $\mathcal{F}^t$ and the reference mesh model $\mathcal{M}^0$.

To estimate the deformation model $\mathcal{G}$, we formulate the following energy function $E(\mathcal{G})$:

$$(3) \qquad E(\mathcal{G}) = \lambda_{\text{data}} E_{\text{data}}(\mathcal{G}) + \lambda_{\text{reg}} E_{\text{reg}}(\mathcal{G}),$$

where $E_{\text{data}}(\mathcal{G})$ is the data term which penalizes the misalignment between the reference mesh model $\mathcal{M}^0$ and the dense features extracted from the live image frame $\mathcal{F}^t$. $E_{\text{reg}}(\mathcal{G})$ is the regularization term which penalizes the inconsistent local transformations between neighboring nodes in the graph model of $\mathcal{G}$. λ_{data} and λ_{reg} denote the associated weights of these two terms. **Data Term** To measure the misalignment between the reference mesh model $\mathcal{M}^0$ and the live frame $\mathcal{F}^t$, we first calculate a vertex map V^t and a normal map N^t from the depth map D^t to represent the geometric features of the live frame $\mathcal{F}^t$. Then we deform the reference mesh vertices $\mathbf{v}^0$ and normals $\mathbf{n}^0$ according to the estimation of the deformation model $\mathcal{G}$ to obtain their prediction represented in the live frame $\mathcal{F}^t$. In particular, the predicated vertices are $\tilde{\mathbf{v}}^t(\mathcal{G}) = \mathcal{W}(\mathbf{v}^0; \mathcal{G})$ and the predicated normals are $\tilde{\mathbf{n}}^t(\mathcal{G}) = \mathcal{W}(\mathbf{n}^0; \mathcal{G})$. Finally, we quantify the misalignment between the predicted vertices $\tilde{\mathbf{v}}^t$ and the geometric features $\{V^t, N^t\}$ based on the point-to-plane error function widely used in the Iterative Closest Point (ICP) algorithm. As a result, we represent the geometric data term as

$$(4) \qquad E_{\text{geo}}(\mathcal{G}) = \sum_m \|\mathbf{n}_d^\top (\tilde{\mathbf{v}}_m^t(\mathcal{G}) - \mathbf{v}_d)\|_2^2,$$

where $\{\mathbf{v}_d, \mathbf{n}_d\}$ denote the 3D-to-2D projected correspondences of the predicted vertex $\tilde{\mathbf{v}}_m^t$ in the feature map $\{V^t, N^t\}$.

Regularization Term The deformation graph model can easily become over-fitting if it is not well regularized during estimation. To solve this problem, one widely used template-free method for general soft objects is to introduce the ARAP constraint. In our work, we encode the neighboring relationship of the ARAP constraint into the deformation graph neighbor set $\{\mathcal{N}_i\}_{i=1}^K$. Based on the neighbor set, we define the regularization term as

$$(5) \qquad E_{\mathrm{reg}} = \sum_{i=1}^{K} \sum_{j \in \mathcal{N}_i} \|\mathbf{T}_i \mathbf{g}_i^t - \mathbf{T}_j \mathbf{g}_i^t\|_2^2.$$

2.3.2. Reconstruction. The reconstruction thread takes the live frame $\mathcal{F}^t$ and the newly estimated deformation model $\mathcal{G}$ as input. It updates the surface geometry and texture by integrating multiple image frames incrementally into the reference volume $\mathbb{V}$. Because the surface geometry and texture are encoded in the volumetric structure, we refer this procedure as volume fusion.

We implement the volume fusion operation based on a non-rigid projective fusion approach [19]. In this approach, we first scan the voxels of $\mathbb{V}$ and get their positions in the reference frame space, which are denoted as $\mathbf{x}^0$. Then we calculate the corresponding deformed positions $\mathbf{x}^t = \mathcal{W}^{0,t}(\mathbf{x}^0; \mathbb{G})$ in the live frame based on the blending function in Equation 1. The deformed voxels $\mathbf{x}^t$ are projected into the live frame image map to find their corresponding pixels $\mathbf{u}^t$. Thus we determine the voxel-to-pixel correspondence $\{\mathbf{x}^0 \leftrightarrow \mathbf{u}^t\}$ between the reference volume $\mathbb{V}$ and the live frame image $\mathcal{F}^t$. For each voxel $\mathbf{x}^0$, we calculate its new TSDF component $d(\mathbf{x}^0)$ and color component $c(\mathbf{x}^0)$ contributed by the live frame $\mathcal{F}^t$ as

$$(6) \qquad d(\mathbf{x}^0) = \max\left(\min\left(\frac{D^t(\mathbf{u}^t) - \lfloor \mathbf{x}^t \rfloor_z}{\tau}, 1\right), -1\right),$$

and

$$(7) \qquad c(\mathbf{x}^0) = C^t(\mathbf{u}^t),$$

respectively. Here $D^t(\cdot)$ and $C^t(\cdot)$ denote the depth image and color image of the live frame $\mathcal{F}^t$. $\lfloor \mathbf{x} \rfloor_z$ represents the position of point $\mathbf{x}$ along Z-axis. τ is the truncated threshold of TSDF value. Besides, we assign a weight $\omega(\mathbf{x}^0)$ to the new components. Finally, we update the reference volume $\mathbb{V}$ as

$$(8) \qquad \mathcal{D}(\mathbf{x}^0) \leftarrow \frac{\mathcal{D}(\mathbf{x}^0)\Omega(\mathbf{x}^0) + d(\mathbf{x}^0)\omega(\mathbf{x}^0)}{\Omega(\mathbf{x}^0) + \omega(\mathbf{x}^0)},$$

(9) $$\mathcal{C}(\mathbf{x}^0) \leftarrow c(\mathbf{x}^0),$$

(10) $$\Omega(\mathbf{x}^0) \leftarrow \min\left(\Omega(\mathbf{x}^0) + \omega(\mathbf{x}^0), \Omega_{\max}\right),$$

where $\Omega_{\max}$ is the upper threshold of the weight. In Equation 9, we do not update the color component via integration as in Equation 8. The main reason here is that our current work does not model and track the light environment and material albedo. As a result, setting the color component $\mathcal{C}(\mathbf{x}^0)$ directly as its new value $c(\mathbf{x}^0)$ in live frame can obtain better response than data fusion.

3. Experiments and results

We implement our shape estimation pipeline on a desktop PC with Intel Core i7 3.4GHz CPU, 32GB of RAM and an NVIDIA GeForce GTX 1080 GPU. To setup the working environment of typical deformable object manipulation tasks, we employ one dual-arm robot (ABB YuMi, with seven degrees-of-freedom in each arm) to perform demonstrations with different materials. Besides, we take the RGB-D data provided by an Intel RealSense SR300 camera as input. The entire experimental setup is shown in the bottom left corner of Figure 3.

Because we encode the reference surface model $\mathcal{M}^0$ explicitly into the volumetric structure $\mathbb{V}$, the real-time performance of our pipeline largely depends on the parameters of the volumetric model $\mathbb{V}$, including the volume's dimension in voxels, the truncated threshold of TSDF value τ, the weight of the newly captured TSDF component ω, and the upper-bound weight of reference TSDF component $\Omega_{\max}$. In our experiment, we pre-defined a $0.7\,\text{m}^3$ cubic space as the reference volume $\mathbb{V}$ and discretized it based on 512^3 voxels. Thus the actual resolution of the volumetric model is 731^3 voxels per cubic meters. Besides, we set τ, $\omega(\cdot)$ and $\Omega_{\max}$ as constants, in particular, $\tau = 0.01m$, $\omega(\cdot) = 1$, and $\Omega_{\max} = 32$. We measured the runtime cost of each main computational components in our pipeline during our experiment, including $3\,\text{ms}$ for preprocessing (e.g., depth image filtering, vertex map and normal map extracting), $35\,\text{ms}$ for deformation tracking, and $10\,\text{ms}$ for volume fusion. On average, our pipeline runs at $50\,\text{ms}$ per frame to track and reconstruct the surface of all deforming targets employed in our experiment, which satisfies the real-time requirement of most robotic applications.

As previously mentioned, one main advantage of our method is that it is robust to occlusion. Such robustness is crucial for applications involving
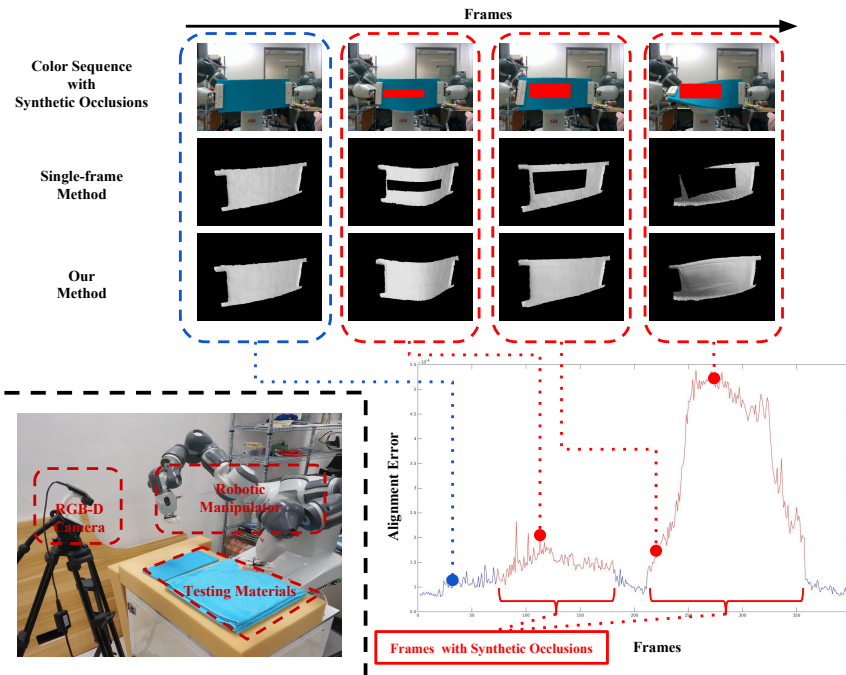
Figure 3. **Bottom left corner** illustrates our experiment setup for a typical deformable object manipulation tasks using a dual arm robot. **Top part** shows the results of the plastic sheet bending task containing synthetic occlusions. The synthetic occlusion masks are highlighted as the red-solid boxes in the first row. **Bottom part** shows the alignment errors between the generated mesh models of our method and the raw image sequence without synthetic occlusions. Because our method approximates the deformation behavior of the target object based on the ARAP constraint, it can provides shape estimation even for the unobserved parts of the surface.

human-machine cooperation, where the human body may occlude the target object from the camera. To demonstrate the advantage of our method, we test it with a plastic sheet bending task. In this task, we let the robotic arm deform the target sheet in front of an RGB-D camera. During the task, we introduce some synthetic occlusions into the captured RGB-D stream and make the corresponding surface areas become invisible to our system. The synthetic occlusion masks are illustrated as the red-solid boxes in the top row of Figure 3. To compared with most previous work [8, 10, 15, 17] which employed single-frame data for shape estimation, we show the corresponding
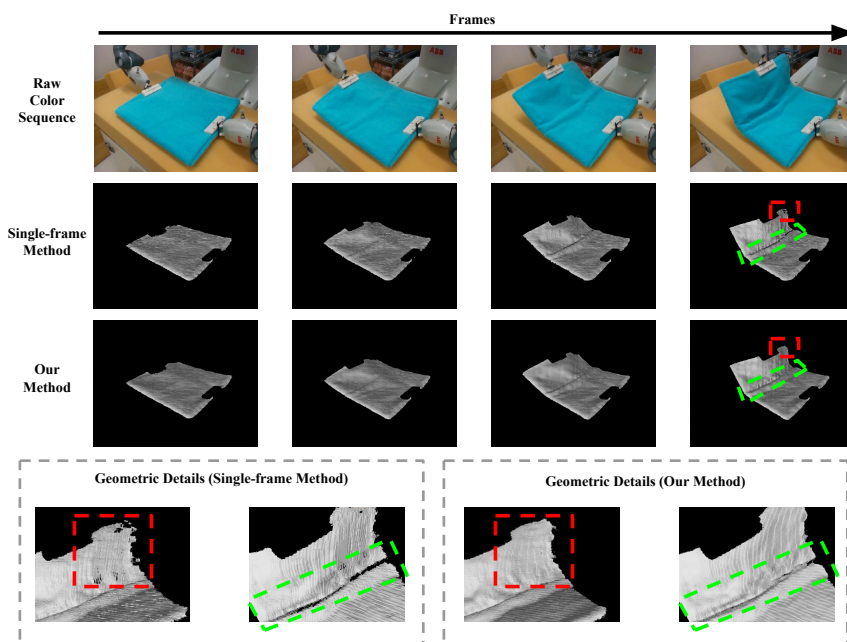
Figure 4. Shape estimation results in our simulated towel folding task. **Top row** shows the raw color sequence captured by a RGB-D camera. **Middle row** shows the live mesh model reconstructed from the single-frame depth data. **Bottom row** shows the live mesh model generated by our method. We zoom in on the highlighted areas in the last frame to demonstrate their geometric details.

surface mesh model extracted from each frame in the second row of Figure 3. Note since we encoded all surface information captured by the single-frame image into such mesh model, it indicates the input data adopted by the aforemetioned work. In our experiment, we extracted such mesh model by projecting the recently captured singe-frame data into a new TSDF volume, and then locating the zero-level surface based on the Marching Cubes algorithm [14] without consideration of the previously observed data. We refer to such method as the *single-frame method.* The mesh models generated by our method are illustrated in the third row of Figure 3. As we can observe, the single-frame method cannot capture the geometry of the occluded part of the object. Our method, on the other hand, approximates the deformation behavior of the unobserved part based on the ARAP constraint and generates a complete shape estimation accordingly.

Because the occlusions in the aforementioned experiment are synthetic, we further evaluated the accuracy of our method, especially for the estimate of the occluded surface part, by measuring the non-rigid alignment error between the reconstructed mesh model and the raw image data without synthetic occlusions based on the data term in Equation 4. We plot the alignment errors measured at different time instants in the bottom right corner of Figure 3. Consider the occluded surface part is totally driven by the ARAP constraint in our method, there is no doubt a gap between our deformation graph model and the object's real deformation behavior. When the occluded part undergoes small deformation as in the second column of Figure 3, the ARAP constraint holds well and the alignment error is quite close to the case without occlusion. The mentioned gap will become obvious when the occluded part undergoes large deformation as in the fourth column of Figure 3. However, even in the latter case, the ARAP constraint can still contribute to a compelling mesh reconstruction. Moreover, its independence of object prior knowledge is essential for our model-free implementation.

To demonstrate the robustness of our method to sensor data noise, we design another folding towel task for testing. Again, we compare the reconstruction results of two different methods in Figure 4, including the single-frame method and our method. Because the RGB-D camera cannot provide stable depth measurement for the wrinkle areas (as highlighted in the green rectangle) and the nearly parallel areas (as highlighted in the red rectangle) on the folded towel, the single-frame reconstructions omit some important geometric details for the shape feedback. This problem exists in most current shape servoing methods. Instead, our method updates the geometry of the surface model via efficient image data fusion, and is capable of providing continuous and smooth mesh reconstruction.

4. Conclusion, limitations and future work

We present a novel shape estimation method to provide reliable shape feedback for the deformable object manipulation problem. A series of experiments are conducted to show the advantages of our method in terms of being real-time, model-free and robust to noise and occlusion. All these features make our method promising to be embedded into current robotic manipulation systems for challenging applications.

Our method still has some limitations. First, our method relies on high-precision deformation estimation for consistent and accurate shape reconstruction. In other words, when the estimation step fails in some cases, the drift error will be accumulated into the reconstruction result and cannot be

corrected. One possible solution to this problem is to add a module into the pipeline which does not rely on the deformation estimates provided by the tracking thread for drift correction.

Second, our deformation model, especially the ARAP regularization term, cannot always hold when the target object undergoes large deformations or complex topological changes. The reasons for such limitation are two-fold: On one hand, to stay within our computational budget for real-time application, we approximated the ARAP regularization term by penalizing inconsistent local deformations close in Euclidean space rather than on the mesh manifold, which unfortunately introduces a gap between the employed deformation model and the real-world physics. On the other hand, the proposed system lacks the ability for perceiving and inferring the surface topology. Thus even if the ARAP constraint is formulated strictly according to the distance on the mesh manifold, it is still difficult for the system to track and reconstruct the deforming surface undergoes fast or complex topological changes. Due to these reasons, we only present experiments based on deforming objects with simple topology and geometry. Note improving the system's robustness to handle topological changes is still challenging for all model-free methods in related fields, and we sincerely believe that a topological segmentation front-end is essential for solving such a problem.

Besides resolving above limitations, for our future work, we will also present a complete shape control pipeline embedded with our shape estimation method.

References

[1] *Bullet physics engine*, available: `www.bulletphysics.org`.

[2] *Havok physics engine*, `www.havok.com`.

[3] *PhysX physics engine*, `www.geforce.com/hardware/technology/physx`.

[4] R. Bridson, Fluid Simulation for Computer Graphics, CRC Press (2015).

[5] Y.-P. Cao, T. Ju, J. Xu, and S.-M. Hu, *Extracting sharp features from RGB-D images*, in: Computer Graphics Forum, Vol. 36, 138–152, Wiley Online Library (2017).

[6] A. Doumanoglou, J. Stria, G. Peleka, I. Mariolis, V. Petrik, A. Kargakos, L. Wagner, V. Hlaváč, T.-K. Kim, and S. Malassiotis, *Folding clothes autonomously: A complete pipeline*, IEEE Transactions on Robotics **32** (2016), no. 6, 1461–1478.

[7] Z. Erickson, A. Clegg, W. Yu, G. Turk, C. K. Liu, and C. C. Kemp, *What does the person feel? learning to infer applied forces during robot-assisted dressing*, in: IEEE International Conference on Robotics and Automation, 6058–6065, IEEE (2017).

[8] Z. Hu, P. Sun, and J. Pan, *Three-dimensional deformable object manipulation using fast online Gaussian process regression*, IEEE Robotics and Automation Letters **3** (2018), no. 2, 979–986.

[9] S. Hutchinson, G. D. Hager, and P. I. Corke, *A tutorial on visual servo control*, IEEE Transactions on Robotics and Automation **12** (1996), no. 5, 651–670.

[10] B. Jia, Z. Hu, J. Pan, and D. Manocha, *Manipulating highly deformable materials using a visual feedback dictionary*, preprint (2017), `arXiv:1710.06947`.

[11] I. Leizea, A. Mendizabal, H. Alvarez, I. Aguinaga, D. Borro, and E. Sanchez, *Real-time visual tracking of deformable objects in robot-assisted surgery*, IEEE Computer Graphics and Applications **37** (2017), no. 1, 56–68.

[12] Y. Li, X. Hu, D. Xu, Y. Yue, E. Grinspun, and P. K. Allen, *Multi-sensor surface analysis for robotic ironing*, in: IEEE International Conference on Robotics and Automation, 5670–5676 (2016).

[13] J. Liu, S. Xin, Z. Gao, K. Xu, C. Tu, and B. Chen, *Caging loops in shape embedding space: Theory and computation*, preprint (2018), `arXiv:1807.11661`.

[14] W. E. Lorensen and H. E. Cline, *Marching cubes: A high resolution 3D surface construction algorithm*, in: ACM SIGGRAPH Computer Graphics, Vol. 21, 163–169 (1987).

[15] D. Navarro-Alarcon and Y.-H. Liu, *Fourier-based shape servoing: a new feedback method to actively deform soft objects into desired 2-D image contours*, IEEE Transactions on Robotics **34** (2018), no. 1, 272–279.

[16] D. Navarro-Alarcón, Y.-H. Liu, J. G. Romero, and P. Li, *Model-free visually servoed deformation control of elastic objects by robot manipulators*, IEEE Transactions on Robotics **29** (2013), no. 6, 1457–1468.

[17] D. Navarro-Alarcon, H. M. Yip, Z. Wang, Y.-H. Liu, F. Zhong, T. Zhang, and P. Li, *Automatic 3-D manipulation of soft objects by robotic arms with an adaptive deformation model*, IEEE Transactions on Robotics **32** (2016), no. 2, 429–441.

[18] R. A. Newcombe, D. Fox, and S. M. Seitz, *Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time*, in: IEEE Conference on Computer Vision and Pattern Recognition, 343–352 (2015).

[19] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, *Kinect-Fusion: Real-time dense surface mapping and tracking*, in: IEEE International Symposium on Mixed and Augmented Reality, 127–136, IEEE (2011).

[20] X. Pan, Y. Zhou, F. Li, and C. Zhang, *Superpixels of RGB-D images for indoor scenes based on weighted geodesic driven metric*, IEEE transactions on visualization and computer graphics **23** (2017), no. 10, 2342–2356.

[21] S. Patil and R. Alterovitz, *Toward automated tissue retraction in robot-assisted surgery*, in: IEEE International Conference on Robotics and Automation, 2088–2094 (2010).

[22] J. Schulman, A. Lee, J. Ho, and P. Abbeel, *Tracking deformable objects with point clouds*, in: IEEE International Conference on Robotics and Automation, 1130–1137, IEEE (2013).

[23] U. Sinha, B. Li, and G. Sankaranarayanan, *Modeling and control of tissue compression and temperature for automation in robot-assisted surgery*, in: IEEE International Conference on Engineering in Medicine and Biology Society, 366–370, IEEE (2014).

[24] O. Sorkine and M. Alexa, *As-rigid-as-possible surface modeling*, in: Symposium on Geometry processing, Vol. 4, 109–116 (2007).

[25] R. W. Sumner, J. Schmid, and M. Pauly, *Embedded deformation for shape manipulation*, ACM Transactions on Graphics **26** (2007), no. 3, 80.

[26] D. Tanaka, S. Arnold, and K. Yamazaki, *EMD Net: An encode–manipulate–decode network for cloth manipulation*, IEEE Robotics and Automation Letters **3** (2018), no. 3, 1771–1778.

[27] M. C. Yip and D. B. Camarillo, *Model-less feedback control of continuum manipulators in constrained environments*, IEEE Transactions on Robotics **30** (2014), no. 4, 880–889.

DEPARTMENT OF MECHANICAL AND BIOMEDICAL ENGINEERING
CITY UNIVERSITY OF HONG KONG
TAT CHEE AVENUE, KOWLOON, HONG KONG SAR
E-mail address: tao.han@my.cityu.edu.hk
E-mail address: xuan.zhao@my.cityu.edu.hk
E-mail address: pgsunbacon@gmail.com
E-mail address: jiapan@cityu.edu.hk