frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 23 December 2020
doi: 10.3389/frobt.2020.600584

Combining Self-Organizing and Graph Neural Networks for Modeling Deformable Objects in Robotic Manipulation

*Angel J. Valencia and Pierre Payeur**

School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON, Canada

Modeling deformable objects is an important preliminary step for performing robotic manipulation tasks with more autonomy and dexterity. Currently, generalization capabilities in unstructured environments using analytical approaches are limited, mainly due to the lack of adaptation to changes in the object shape and properties. Therefore, this paper proposes the design and implementation of a data-driven approach, which combines machine learning techniques on graphs to estimate and predict the state and transition dynamics of deformable objects with initially undefined shape and material characteristics. The learned object model is trained using RGB-D sensor data and evaluated in terms of its ability to estimate the current state of the object shape, in addition to predicting future states with the goal to plan and support the manipulation actions of a robotic hand.

Keywords: deformable objects, dynamic shape modeling, manipulation, robotics, shape, sensing

OPEN ACCESS

Edited by:
David Navarro-Alarcon, Hong Kong Polytechnic University, Hong Kong

Reviewed by:
Jihong Zhu, Delft University of Technology, Netherlands
Juan Antonio Corrales Ramon, Sigma Clermont, France

***Correspondence:**
Pierre Payeur
ppayeur@uottawa.ca

Specialty section:
This article was submitted to Robotic Control Systems, a section of the journal Frontiers in Robotics and AI

Received: *30 August 2020*
Accepted: *23 November 2020*
Published: *23 December 2020*

Citation:
Valencia AJ and Payeur P (2020) Combining Self-Organizing and Graph Neural Networks for Modeling Deformable Objects in Robotic Manipulation. Front. Robot. AI 7:600584. doi: 10.3389/frobt.2020.600584

1. INTRODUCTION

In the context of robotic manipulation, object models are used to provide feedback signals that a robot can control when performing a specific task. For deformable objects, the object pose is not a sufficient state representation (Khalil et al., 2010) to guarantee even low-level manipulation tasks (e.g., pick-and-place), as manipulation actions produce changes in the object shape. Likewise, high-level manipulation tasks (e.g., making a bed or cleaning surfaces) involve knowledge of future behaviors to develop hierarchical plans. Therefore, an object model that integrates shape representation and prediction is required in order to perform a variety of tasks with deformable objects.

Early attempts to estimate the object shape in robotic manipulation mainly adopted an analytical approach, which is commonly adjusted in simulation (Nadon et al., 2018). This comes with some drawbacks in real robotic environments, as simulators are currently not sophisticated enough to provide realistic models of non-rigid objects (Billard and Kragic, 2019), and the support for sensor measurements and hardware in simulators is very limited. Furthermore, certain assumptions about objects are often made (e.g., homogeneous composition or isotropic materials). On the contrary, it is rarely possible to determine these conditions in advance for every new object encountered in the environment. This lack of a general-purpose methodology to estimate the object shape makes it difficult to develop more autonomous and dexterous robotic manipulation systems capable to handle deformable objects (Sanchez et al., 2018).

In this paper, we present a data-driven approach to estimate and predict the state of initially unknown deformable objects without the dependency on simulators or predefined material parameters. The contributions of this work can be summarized as follows: First, we develop a method for shape estimation using Self-Organizing Neural Networks (SONNs). Second, we design and implement an original method for shape prediction using Graph Neural Networks (GNNs) that leverages the initial SONN-based model. Third, we test the combination of the shape estimation and prediction methods as a learned model of deformable objects in real robotic environments. This paper represents a significant extension to previous work (Valencia et al., 2019) that corroborates the learned model across different types of deformable objects with experimental evaluations.

2. RELATED WORK

Various methods that explore analytical modeling approaches for non-rigid objects in robotic environments are inspired by physics-based models, extensively studied in computer graphics (Nealen et al., 2006). These include continuous mesh models such as Euler-Bernoulli (EB) (Fugl et al., 2012), linear Finite Element Method (FEM) (Lang et al., 2002; Frank et al., 2014; Jia et al., 2014; Petit et al., 2015; Duenser et al., 2018) and non-linear FEM (Leizea et al., 2017; Sengupta et al., 2020). Also, discrete mesh models such as linear Mass-Spring Systems (MSS) (Leizea et al., 2014) and non-linear MSS (Zaidi et al., 2017) are considered. Additionally, discrete particle models such as Position Based Dynamics (PBD) (Güler et al., 2015) have been introduced. In these methods, a crucial step is to determine the material parameters of a deformable object (e.g., Young's modulus and Poisson's ratio). This is typically done via specific sensor measurements or assuming prior material information. More generally, these parameters are obtained by simultaneously tracking the shape while applying optimization techniques in the model.

Alternatively, data-driven approaches leverage sensor data to approximate the behavior of deformable objects typically using learning-based models. These include Single-layer Perceptron (SLP) (Cretu et al., 2012; Tawbe and Cretu, 2017). Other methods combine analytical and data-driven approaches in different parts of the modeling pipeline. For example, a Gaussian Process Regression (GPR) is used to estimate the deformability parameter of a PBD model (Caccamo et al., 2016). An Evolutionary Algorithm (EA) is proposed to search for the parameter space of an MSS model (Arriola-Rios and Wyatt, 2017). In these methods, an important aspect for a correct modeling is the information extracted from the sensor measurements. For RGB-D data, these correspond to properties of the shape (e.g., surfaces or feature points) and typically provide a structured representation suitable for the type of deformation model used. As such, B-spline snakes (Arriola-Rios and Wyatt, 2017) can be used to create a mesh-like representation. On the other hand, optical flow (Güler et al., 2015) and neural gas (Cretu et al., 2012) are used to create a particle-like representation.

Recent learning-based models such as Graph Neural Networks (GNN) have demonstrated the ability to act as a physics engine (Battaglia et al., 2016; Mrowca et al., 2018). Although there is little exploration of training such models using only sensor measurements. The most advanced attempt to model deformable objects beyond simulation is presented in Li et al. (2019a), where a real robotic gripper performs a shape control task on a deformable object. However, the models are initially trained entirely in simulation. Conversely, while aiming at exploiting real shape measurements for the modeling and prediction stages, this paper expands on the work of Cretu et al. (2012), as we aim to contribute a general-purpose methodology for modeling deformable objects in real robotic environments. In this way, we extend the latter by exploring recent learning-based models with physical reasoning capabilities (Battaglia et al., 2016) using RGB-D sensor measurements.

3. METHODOLOGY

In this section, the proposed data-driven approach to model deformable objects is introduced (**Figure 1**). The main components of the learned object model are the shape estimation and prediction methods.

3.1. Shape Estimation

A Self-Organizing Neural Network (SONN) based model is proposed to estimate the object state from the sensor measurements. This model is called Batch Continual Growing Neural Gas (BC-GNG) and is an extension of the continual formulation of the Growing Neural Gas (C-GNG) algorithm (Orts-Escolano et al., 2015). C-GNG is extended by implementing a batch training procedure that enables to update the model parameters while avoiding an individual iteration on each sample during the execution of the algorithm. This approach provides benefits such as computational efficiency and faster convergence. First, the core principles of GNG models are described and then the technical details of our proposal are explained.

Growing Neural Gas: A GNG model (Fritzke, 1995) produces a graph representation $G = (O, R)$ from a data distribution P of size N. Where, $O = \{\mathbf{o}_i\}_{i=1:\mathcal{N}_O}$ is the set of nodes with $\mathcal{N}_O$ cardinality, and $R = \{\mathbf{r}_k, u_k, v_k\}_{k=1:\mathcal{N}_R}$ is the set of edges with $\mathcal{N}_R$ cardinality, which connects an unordered pair of nodes u_k and v_k. Also, each node has an associated feature vector $\mathbf{o}_i = \{\boldsymbol{x}_i, e_i\}$, which contains the position and spatial error, respectively. Likewise, each edge has an associated feature vector $\mathbf{e}_k = \{a_k\}$, which contains the connection age. The position is a direct measure of the spatial location of a node with respect to the sample, while the spatial error and connection age serve as measures for the addition and removal processes of nodes and edges from the graph.

The GNG model receives as input distribution, P, the current frame of the point cloud data associated to the object and produces the graph, G, as an estimation of the object shape. The model is trained following the execution of Algorithm 1. First, the graph is initialized by creating two nodes with position set to random values and spatial error set to zero. In addition, an

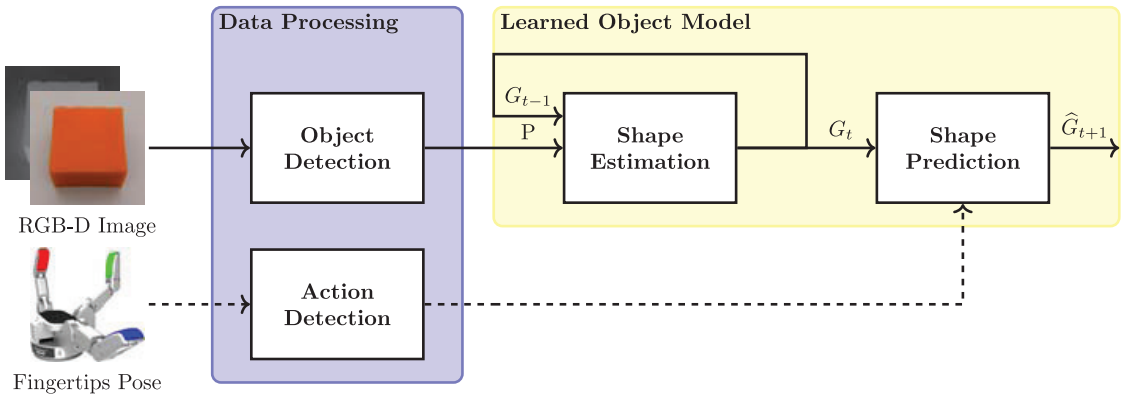

FIGURE 1 | Components of the proposed data-driven approach for deformation modeling. The framework takes as input RGB-D sensor images and robotic hand's fingertips pose. (Data Processing) object and robot manipulation actions are detected. (Learned Object Model) the processed data is combined for the estimation and prediction of the object shape.

edge connecting these nodes is created with age set to zero. After initialization, an individual sample, ξ, is randomly drawn from the distribution, and then the ADAPTATION and GROWING phases are run. During the former, nodes and edges features are sequentially updated, while during the latter and after receiving a certain number of samples, λ, new nodes and edges are added to the graph. These phases follow the original algorithm proposed by Fritzke (1995). In this work, the algorithm is executed until the quantization error (QE) reaches certain limit, which gives more flexibility to control the representation, as during the GROWING phase, nodes are dynamically created in an attempt to best fit the samples available in the input data, but does not require setting a fixed number of nodes. The quantization error is evaluated over the distribution P and computes the average difference between the closest node position (i.e., the smallest Euclidean distance) $\boldsymbol{x}_{s_1}$ and the associated sample ξ.

$$\mathrm{QE} = \frac{1}{N} \sum_{\xi \in P} \left\| \boldsymbol{x}_{s_1} - \xi \right\| \tag{1}$$

Algorithm 1:Steps of computation in GNG

Input: P
Output: G
1: $G \leftarrow$ init_graph(P)
2: **while** QE $>$ $\mathrm{QE}_{\max}$ **do**
3: **for all** n $\in N$ **do**
4: $\xi \sim P$
5: ADAPTATION(G, ξ)
6: **if** (n **mod** λ) $= 0$ **then**
7: GROWING(G)
8: **end if**
9: **end for**
10: **end while**

Outlier Regularization: One problem that limits the use of GNG in problems with time constraints, such as tracking the shape of a deformable object as it evolves, relates to the requirement to retrain the model for every new input distribution collected by sensors. A continual formulation of the Growing Neural Gas (C-GNG) (Orts-Escolano et al., 2015) implements a technique that leverages the knowledge already learned during previous executions. Specifically, the graph from the previous data frame G_{t-1} is used to initialize the graph in the current data frame G. This provides a significant practical improvement but makes its formulation more sensitive. For example, outliers can affect the graph by creating nodes that do not adapt to the input distribution. The presence of these dead nodes represents a serious issue for the estimation of the object shape, especially when it is meant to vary over time. Therefore, we propose to regularize the influence of the outliers during the procedure that updates the position feature of each node. During the ADAPTATION phase, the nodes position are updated (Equation 2) for those that are found as closest $\boldsymbol{x}_{s_1}$ or topological neighbors $\boldsymbol{x}_n$ to the sample ξ. The parameters, ϵ_{s_1}, and, ϵ_n, correspond to the learning rates that control the influence of the adjustment of each contribution to the position feature.

$$\begin{aligned} \boldsymbol{x}_{s_1} &\leftarrow \boldsymbol{x}_{s_1} + \epsilon_{s_1}(\xi - \boldsymbol{x}_{s_1}) \\ \boldsymbol{x}_n &\leftarrow \boldsymbol{x}_n + \epsilon_n(\xi - \boldsymbol{x}_n) \end{aligned} \tag{2}$$

We introduce a new term, w_{s_1}, that modifies the learning rate of the closest node position (Equation 3). In this way, those pairs of nodes and samples for which distances are large are penalized due to the possibility of being outliers, whereas those with small distances remain unchanged.

$$\boldsymbol{x}_{s_1} \leftarrow \boldsymbol{x}_{s_1} + w_{s_1}\epsilon_{s_1}\left(\xi - \boldsymbol{x}_{s_1}\right) \tag{3}$$

This regularization term (Equation 4) evaluates a 1D Gaussian kernel function with mean equal to the difference between the Euclidean distance $\|\boldsymbol{x}_{s_1} - \xi\|$ and maximum quantization error

QE_{max}. And, standard deviation proportional to the maximum quantization error QE_{max}.

$$w_{s_1} = \begin{cases} 1, & \mu < 0 \\ K(\mu, \sigma) = e^{-\frac{\mu^2}{2\sigma^2}}, & \text{others} \end{cases} \tag{4}$$

Batch Training: We also introduce a new procedure to update the features of the nodes and edges in batches, which unifies the contributions of a node with respect to its role among the samples. First, the node position is updated by combining the contributions when the node is found as closest and as topological neighbor. Similarly, the age of the edges connecting the closest node with its neighbors is updated by accumulating the times in which the node is found as closest. More specifically, the Euclidean distances between all the samples and nodes position are computed, also finding the two closest nodes at once. With this information, the input distribution can be represented as a set $P = \{P_i\}_{i=1:\mathcal{N}_O}$, where P_i is the batch data associated with each node found as closest, with size N_i. In this way, the contribution of each node as closest is reformulated (Equation 5) as the average of the distances paired with that particular closest node.

$$\boldsymbol{x}_i^{s_1} = \frac{1}{N_i} \sum_{\xi \in P_i} (\xi - \boldsymbol{x}_i) \tag{5}$$

Also, the age of the neighbor edges is reformulated as an increment of the batch data size N_i. Since nodes are likely to be connected with more than one edge in the graph, the contribution of each node as neighbor requires an additional consideration. Initially, all the distances between the node and the samples associated due to the connections with all its neighbor nodes are collected, then the average of the collected distances is computed (Equation 6) similarly as in the previous step.

$$\boldsymbol{x}_i^{n} = \sum_{j} \frac{1}{D_i N_j} \sum_{\xi \in P_j} (\xi - \boldsymbol{x}_i) \tag{6}$$

Where, P_j is the batch data of each neighbor of the closest node and D_i is the number of edges of the closest node. Thus, the contributions of each node as closest and as neighbor are included in a single expression to update the position feature (Equation 7), thus replacing the two-step update process with only an ADAPTATION phase in the online training, as detailed in Algorithm 2. By computing the Euclidean distance for all the samples at once, this procedure is also highly parallelizable as nodes can be updated independently.

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \epsilon_{s_1} \boldsymbol{x}_i^{s_1} + \epsilon_n \boldsymbol{x}_i^{n} \tag{7}$$

Algorithm 2:Steps of computation in BC-GNG

Input: P, G_{t-1}
Output: G_t
1: **if** $t = 1$ **then**
2: $G \leftarrow \text{GNG}(P)$
3: **else**
4: $G \leftarrow G_{t-1}$
5: **end if**
6: **while** $\text{QE} > \text{QE}_{max}$ **do**
7: $\text{ADAPTATION}(G, P)$
8: **end while**

3.2. Shape Prediction

As described in section 1, shape estimation alone does not provide sufficient information to perform high-level manipulation tasks. Therefore, a prediction phase must be incorporated in order to characterize the future states of a deformable object. With the objective to support the requirements of path planning and dynamic interaction of a robotic hand with a deformable object, Graph Neural Network (GNN) based models are also adapted in our framework to predict the future object state using the information of the current object state and the manipulation actions of the robotic hand. Specifically, we use the Interaction Network (IN) framework (Battaglia et al., 2016) along with its extension known as PropNet (Li et al., 2019b) for supervised learning on graph structures. Unlike standard GNNs, the IN is specifically designed to learn the dynamics of physical interactive systems. This model is characterized by being able to make predictions for future states of the system, and also to extract latent physical properties.

3.2.1. Object-Action Representation

A new representation is created to jointly capture the object shape and the manipulation actions. This is defined as a directed graph $G = \langle O, R \rangle$. In which, $O = \{\mathbf{o}_i\}_{i=1:\mathcal{N}_O}$ is the set of nodes with $\mathcal{N}_O$ cardinality, and associated feature vector $\mathbf{o}_i = \{\boldsymbol{x}_i, \boldsymbol{v}_i\}$, which contains the object-action state defined as position and velocity. Also, $R = \{\mathbf{r}_k, v_k, u_k\}_{k=1:\mathcal{N}_R}$ is the set of edges with $\mathcal{N}_R$ cardinality, which due to the graph directionality connects an ordered pair of nodes, defined as sender node u_k and receiver node v_k.

The object state is the shape estimation G produced by the BC-GNG model (section 3.1) and the manipulation actions are included as contact points, which are captured from the fingertips pose of the robotic hand. This means that new nodes are added to the graph with their feature corresponding to the position components of the fingertips pose. Also, edges are created when physical interactions are detected between the fingertips and the object, thus assigning action nodes for the fingertips as senders, and object nodes as receivers in the directed graph. The edge direction adds a causality property, indicating that action nodes produce the displacement of the object nodes and not the opposite. Furthermore, the velocity feature is computed by differentiating the signal obtained by the position feature of the object-action nodes.

3.2.2. Interaction Networks

An IN model is trained to learn the transition dynamics of the object state. It takes the object-action graph at a certain time step G_t and outputs a prediction of the nodes position of the graph for the next time step $\widehat{G}_{t+1}$. The model is updated following the execution of Algorithm 3, which uses the evaluation of aggregation and update functions (Gilmer et al., 2017) to perform computations with the graph features. The update function, ϕ_R, is responsible to perform per-edge updates. This function evaluates the collected features of the edge along with the sender and receiver nodes, and thus computes the edge effect. Similarly, the update function, ϕ_O, is responsible to perform per-node updates. This function evaluates the collected features of the node along with those produced by the update function, and thus computes the node effect. Since the update function produces a variable number of effects associated with each node, these are reduced using an aggregation function, $\rho_{R\to O}$, in order to produce a single effect.

The update functions are implemented as Multi-layer Perceptron (MLP) modules while the aggregation function is a summation. The mean squared error (MSE) of the predicted and observed nodes velocity (Equation 8) is used as the loss function to train the models. This statistical metric computes the average of the squared errors between the predicted velocities, $\hat{\boldsymbol{v}}_{i,t+1}$, and the observed velocities, $\boldsymbol{v}_{i,t+1}$.

$$\text{MSE} = \frac{1}{\mathcal{N}_O}\sum_{i=1}^{\mathcal{N}_O}\left(\hat{\boldsymbol{v}}_{i,t+1} - \boldsymbol{v}_{i,t+1}\right)^2 \tag{8}$$

Algorithm 3:Steps of computation in IN

Input: G_t
Output: $\widehat{G}_{t+1}$

$\mathbf{e}_{k,t} \leftarrow \phi_R\left(\mathbf{r}_{k,t}, \mathbf{o}_{u_{k,t}}, \mathbf{o}_{v_{k,t}}\right)_{k=1:\mathcal{N}_R}$
$\mathbf{e}_{i,t} \leftarrow \rho_{R\to O}(\mathbf{e}_{k,t})_{i=1:\mathcal{N}_O}$
$\hat{\mathbf{o}}_{i,t+1} \leftarrow \phi_O(\mathbf{e}_{i,t}, \mathbf{o}_{i,t})_{i=1:\mathcal{N}_O}$

3.2.3. Propagation Networks

A limitation of the IN occurs for systems that require long and fast propagation effects, since its formulation only considers local pairwise interactions during each time step. Therefore, several iterations of the algorithm are needed in order to propagate the information on the graph, and thus reach remote nodes. As an extension to IN, the PropNet (Li et al., 2019b) formulation (Algorithm 4) proposes the inclusion of a multi-step propagation phase, which consists of computing the edge and node effects using an additional iterative process, where l corresponds to the current propagation step parameter, and is set to a value within the range of $1 \leq l \leq L$. Also, the update functions, ϕ_R^{enc}, ϕ_O^{enc}, are used to encode the input edge and node features, respectively. While the function, ϕ_O^{dec}, is used to decode the output node feature. In this way, these functions learn a latent representation of the graph features, which are also part of the model during training.

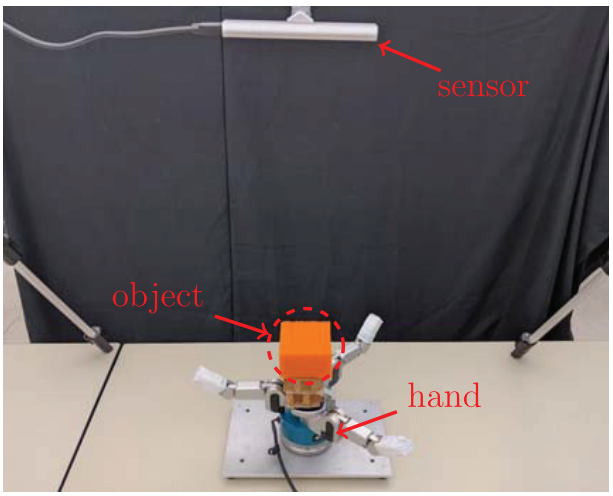

FIGURE 2 | Configuration of the real robotic environment. The location of the RGB-D sensor, deformable object and three-fingered robotic hand are marked in red.

Algorithm 4:Steps of computation in PropNet

Input: G_t
Output: $\widehat{G}_{t+1}$

$\mathbf{o}_{i,t}^{\text{enc}} \leftarrow \phi_O^{\text{enc}}\left(\mathbf{o}_{i,t}\right)_{i=1:\mathcal{N}_O}$
$\mathbf{r}_{k,t}^{\text{enc}} \leftarrow \phi_R^{\text{enc}}\left(\mathbf{r}_{k,t}, \mathbf{o}_{u_{k,t}}, \mathbf{o}_{v_{k,t}}\right)_{k=1:\mathcal{N}_R}$
$\mathbf{h}_{i,t}^0 \leftarrow \mathbf{0}$
for all $l \in L$ **do**
  $\mathbf{e}_{k,t}^l \leftarrow \phi_E(\mathbf{r}_{k,t}^{\text{enc}}, \mathbf{h}_{u_k,t}^{l-1}, \mathbf{h}_{v_k,t}^{l-1})_{k=1:\mathcal{N}_R}$
  $\mathbf{e}_{i,t}^l \leftarrow \rho_{R\to O}(\mathbf{e}_{k,t}^l)_{i=1:\mathcal{N}_O}$
  $\mathbf{h}_i^l \leftarrow \phi_O(\mathbf{o}_{i,t}^{\text{enc}}, \mathbf{e}_{i,t}^l, \mathbf{h}_{i,t}^{l-1})_{i=1:\mathcal{N}_O}$
end for
$\hat{\mathbf{o}}_{i,t+1} \leftarrow \phi_O^{\text{dec}}(\mathbf{h}_{i,t}^L)_{i=1:\mathcal{N}_O}$

4. EXPERIMENTAL EVALUATION

4.1. Experimental Setup

The configuration of the real robotic environment is shown in **Figure 2**, which consists of a Barrett BH8-280 robotic hand[1] resting on a flat table, an Intel RealSense SR305 RGB-D sensor[2] mounted overhead on a tripod, and a deformable object placed on the palm of the robotic hand. The complete set of deformable objects used to construct the datasets is shown in **Figure 3**.

All the sensors and hardware components used in the robotic manipulation setup are operated through ROS (Quigley et al., 2009). The data preparation, signal and image processing steps are implemented using SciPy (Virtanen et al., 2020) and OpenCV (Bradski, 2000) libraries. The models are implemented in the Deep Graph Library (Wang et al., 2019) using PyTorch (Paszke et al., 2019) as backend.

[1]https://advanced.barrett.com/barretthand
[2]https://www.intelrealsense.com/depth-camera-sr305

FIGURE 3 | Deformable objects used to construct the dataset. (Ball) plastic type found in toy stores. (Sponges) foam type used as cleaning utensils. (Towel) textile type found in warehouses. (Toy) stuffed type found in pillows or stuffed animals.

4.2. Sensor Measurements and Data Processing

The RGB-D sensor data is processed in a ROS node to detect the object and generate the point cloud data. Also, another ROS node is used to estimate the robotic hand's fingertips pose to generate the manipulation action information.

4.2.1. Object Detection

Classical image segmentation techniques are applied to both aligned color and depth images for the detection of the deformable objects. The color image is transformed to the HSV color space, and then a histogram backprojection technique is applied to obtain a binary mask. Then, the mask is filtered by applying a convolution with threshold operation to obtain a cleaner result. Moreover, the depth image is cropped by volume, truncating the spatial values based on available information about the object position relative to the camera. Thus, the resulting color and depth masks are combined and applied to the depth image to obtain the object of interest. The segmented image is then deprojected to convert the 2D pixels to 3D point clouds. For the small sponge object, this process transforms the RGB-D sensor images from 640×480 to approximate $80\times80\times3$.

4.2.2. Fingertips Pose Estimation

The data captured on the fingertips correspond to the pose (position and orientation) of each tip. To facilitate the accurate estimation of the pose, a set of AR markers are placed on each tip. The design is based on the ARTags fiducial marker system, and generated according to the following parameters: size of 1.8 cm, margin of 1-bit, and pattern of 25-bit 5×5 array. The latter controls the number of tags that can be created based on the marker dictionary. Given the physical dimensions of the robotic hand, this design enables to precisely fit each marker on the tip. In turn, the markers are visible enough to be detected in the images captured by the RGB-D sensor. The fingertips pose corresponds to that estimated by the markers. The pose enables to define the contact points, which is determined by a contact region with spherical shape, centered on the marker and with a radius of 2.3 cm. The latter is measured considering the tip size relative to the marker location.

4.3. Shape and Motion Estimation With GNG-Based Models

These experiments are run on a computer with $1\times$ Intel Core i5-7300U @ 2.60 GHz, 16 GB RAM, and GNU/Linux operating system. The parameters of the GNG models are shared as much as possible in order to consistently compare the performance of the different variations. For GNG, an age of 35, learning rate of 0.1 and 0.005, error decay of 0.5 and 0.9995 are used. For C-GNG, an age of 2,000, learning rate of 0.1 and 0.005. And for BC-GNG, an age of 2,000, learning rate of 0.4 and 0.01 are used. The sigma value of the regularization term used in C-GNG and BC-GNG corresponds to 0.6 for the towel and 4 for the rest of the objects.

The fingers trajectory are generated to perform a squeeze-like manipulation with each object. The base joints range is limited to $(-90^\circ,90^\circ)$, whereas the spread joint is limited to $(-45^\circ,45^\circ)$. Each trajectory is generated taking as final configuration a random joint position within the available moving range for each robotic finger, and using a linear interpolation with 50 points beginning from a predefined rest position of the hand. The trajectories are designed in this manner to produce brief rest periods at the end of each point with the intention of preventing slippage or sliding movements of the object, and thus mainly capturing information associated with the deformation. A dataset is created which consists of a file with 800 samples, using a sampling rate of 30 Hz. Each file stores the data generated in synchronization with the execution of the fingers trajectory, which takes approximately 27 s to complete. Results for a subset of the data frames that progressively reflects various deformation levels using the small sponge as an example of deformable object are shown in **Figure 4**. We refer the reader to the **Supplementary Material** for additional results with the other deformables objects considered, as per **Figure 3**.

As mentioned in section 2, the properties extracted from the object shape are the basis for any learned model. This means that motion changes should closely capture the dynamics of the deformation. A motion analysis can be used to determine whether the produced shape estimation is consistent with the deformation and reflects the current state of the deformable object. The latter is formalized by also considering the requirements of real robotic environments.

4.3.1. Real-Time Execution

We evaluate the performance of different variations of GNG for real-time shape estimation using point clouds as input data. The runtime of each model is recorded per data frame and the average over the entire manipulation trajectory is computed, as shown in a bar plot in **Figure 5**. The models are evaluated using three levels of quantization error, which are selected to provide an insight of the precision costs associated to the representation.

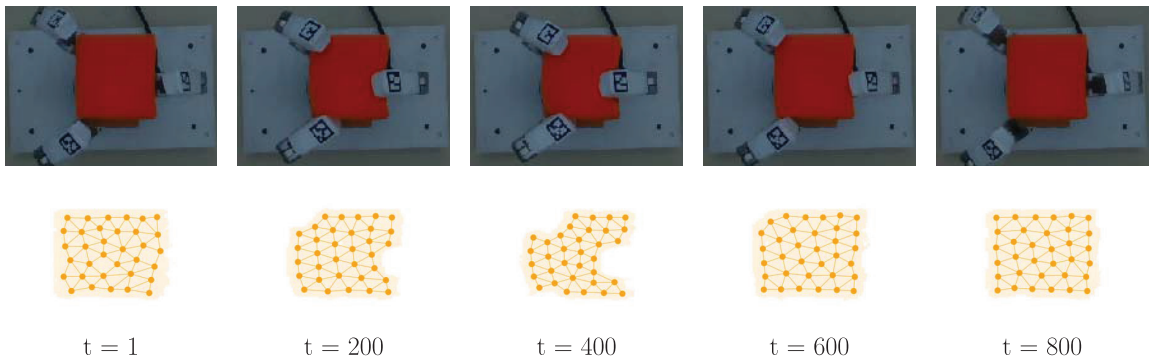

FIGURE 4 | Qualitative results of the shape estimation method. **(Top)** color image sequence of the scene with the small sponge at different deformation levels. **(Bottom)** point cloud and graph representation of the object shape obtained by the BC-GNG model.

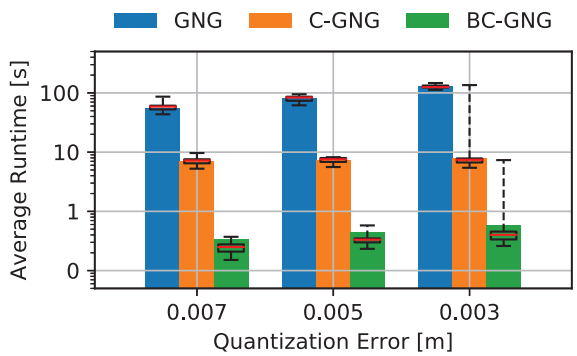

FIGURE 5 | Average of runtime sequence. GNG models formation computing time with respect to varying tolerance on model accuracy during the manipulation of the small sponge object.

For GNG, the runtime takes an average of 80.7 s when a quantization error tolerance of QE = 0.005 is imposed, and grows linearly if more precision (lower quantization error) on the shape representation is required. On the other hand, C-GNG runtime is several orders of magnitude faster mainly due to the reuse of the previous graphs over iterations. Although, this formulation is a great improvement, its runtime is not yet suited for real-time applications, at least for low-power CPUs and embedded systems. It takes an average of 7.4 s at each data frame but reveals less sensitive to the tolerance set on the model precision. Finally, the proposed BC-GNG variation that involves batch training considerably speeds up the execution. In this case, the algorithm needs an average of 0.4 s to construct the same graph with only a slight variation in computing time when the desired model accuracy is varied. In certain cases, sudden increase in time is observed when more accuracy is required, as shown in QE = 0.003. This occurs in data frames with high variations, since graphs with a fixed number of nodes cannot always adapt to such levels of accuracy. Therefore, early stopping mechanisms are required to avoid unnecessary iterations.

4.3.2. Temporal Smoothing

We evaluate the performance of different variations of GNG to generate stable displacements of the nodes that encode the object shape. The path followed by each individual node is measured relative to the centroid coordinate system to mitigate the influence of rigid motions. These local displacements estimate the actual deformation motion of the object shape. The 3-dimensional temporal evolution of local nodes for the small sponge object is shown in **Figure 6** for a subset of nodes (first 6 out of 34 nodes) extracted from the graph forming the shape model and over the 800 frames that correspond to a manipulation operation.

The continual models (C-GNG and BC-GNG) clearly produce more stable signals. An interesting property of BC-GNG is the low-pass filter effect that is observed in the signals. This behavior occurs due to the characteristic of the algorithm that uses the average of the nodes position during the update process. Therefore, the node displacements obtained by BC-GNG are much smoother, and desirable to estimate with confidence the motion and deformation quantities of a non-rigid object as its shape is not dominated by noise associated with the individual dynamics of nodes forming the graph-based representation.

4.3.3. Region Correspondence

Finally, we evaluate the performance of different variations of GNG to produce node displacements (**Figure 6**) that can be used as features of the object motion. The region correspondence of nodes position is non-existent in GNG due to the stochastic nature of the algorithm, which causes that new nodes are not created around the same location. For C-GNG, the displacements exhibit a localized motion of nodes position that preserve certain regions of the shape. However, there still exists some interference between nodes which causes unrecoverable positions and affects the region correspondence of the representation, more noticeable when large deformations occur. For BC-GNG, these displacements reflect a more localized motion of nodes position, even further to that observed in C-GNG. Interference between nodes is not causing strong deviations in their displacements, hence better preserving their correspondence throughout the manipulation task.

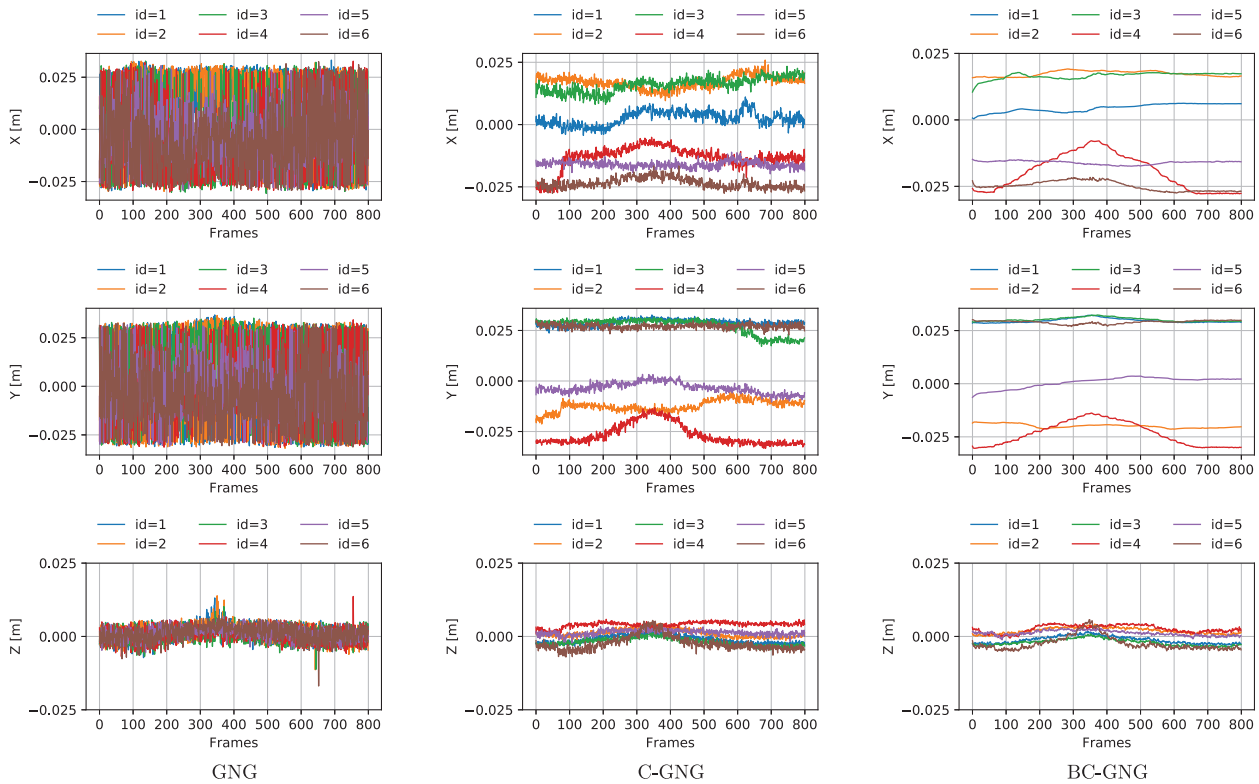

FIGURE 6 | Local displacements of nodes encoding the object shape. Three-dimensional coordinates of nodes obtained by the GNG models during the manipulation of the small sponge object.

4.4. Deformation Dynamics Prediction With GNN-Based Models

These experiments are run on a cloud instance with 1× Intel Xeon Processor @ 2.3GHz, 1× NVIDIA Tesla K80 GPU with 2,496 CUDA cores, 12 GB RAM GDDR5 VRAM, and GNU/Linux operating system. The training procedure of the GNN models consists of 20 iterations, and using a batch size of 1. The MLP modules are trained using the Adam optimizer (Kingma and Ba, 2015) with learning rate of 0.001 and momentums of 0.9 and 0.9999. A learning rate scheduler with factor of 0.8 and patience 3 is used.

The architecture design of the GNN models follows the configuration presented in Li et al. (2019a). This configuration is shared among models in order to consistently compare their performances, hence the main difference is the propagation step parameter L, which is 1 for IN and 2 for PropNet. In this way, the encoder functions ϕ_R^{enc}, ϕ_O^{enc} are 2-layer MLPs with hidden sizes of 200 and 300, with output size of 200. The update functions ϕ_R, ϕ_O are 1-layer MLPs with hidden size of 200, and output size of 200. And, the decoder function ϕ_O^{dec} is 2-layer MLP with hidden size of 200 and output size of 3, the latter corresponds to the components of the predicted velocity. All MLP modules use the rectified linear unit (ReLU) as the activation function. A dataset of graphs per object is created and consists of 20 files, each associated to a different fingers trajectory. This produces 16,000 samples in total. The dataset is divided into 80% for training, 10% for validation and 10% for test, which is equivalent to 16 trajectories (12,800 randomly shuffled samples) and 2×2 trajectories (2×1,600 samples) respectively. In addition, the dataset is normalized between 0 and 1 due to the varied scales of the position and velocity features.

The GNN models are primarily analyzed in two situations: first evaluating the performance of the predictions for the object deformation in single-step time sequences, and then evaluating the ability to generalize over multi-step time sequences. In order to enable a more direct interpretation of the results, the Root Mean Square Error (RMSE) of the predicted and observed nodes position is used as a metric. Thus, the nodes position of the next frame $\hat{\boldsymbol{x}}_{i,t+1}$ are calculated via explicit integration of the equation of motion (Equation 9), which uses the predicted velocities of the next frame $\hat{\boldsymbol{v}}_{i,t+1}$ and time per frame Δt to update the current position of each node.

$$\hat{\boldsymbol{x}}_{i,t+1} = \boldsymbol{x}_{i,t} + \Delta t \cdot \hat{\boldsymbol{v}}_{i,t+1} \tag{9}$$

Single-Step Predictions: The nodes position are predicted from the most recent observed data at each frame (t + 1). The GNN models obtain a relatively low and consistent error (**Table 1**) of the nodes position throughout the entire range of acquired data frames over the object manipulation duration. These results confirm a stable

prediction capability, one step ahead, with the GNN models, as shown on the left of **Figure 7**.

Multi-Step Predictions: The nodes position are predicted for every frame but with updates from observed data fed into the model at different frames (t > 1), which involves a longer-term prediction before new data is made available to the GNN models. The error produced by the models remains relatively low (**Table 1**) over a short range of frames (t + 5), but progressively degrades as the number of frames further increases (t + 50). As a consequence, at some point the models become unable to predict with confidence the nodes position, as shown on the right of **Figure 7**. The errors from previous iterations cumulate and the prediction diverges, causing the deformable object prediction to enter an unrecoverable state.

5. DISCUSSION

5.1. Shape Estimation Quality

All variants of GNG studied in this research produce a graph sequence that estimates the object shape. However, regardless of maintaining consistency during model training (i.e., shared parameters and stopping criterion), the proposed BC-GNG model performs better in terms of computing time and motion estimation, as demonstrated experimentally in section 4.3. Consider the data frames (**Figure 4**) where the largest deformation occurs (around $t = 400$). The areas on the shape where the object is compressed more (e.g., around the center and vertices) show a higher and more natural accumulation of nodes. Also, the estimation obtained when no interaction occurs between the fingers and the small sponge (around $t = 800$) produces a more symmetric node density that better resembles the object topology. These characteristics are also observed in the other deformable objects considered in these experiments.

We also observed that BC-GNG still exhibits some difficulty to recover the initial node position for elastic objects. Unlike C-GNG, such variations do not manifest as abrupt changes in the signal due to the smoother characteristic of the displacements. This behavior is more desirable since abrupt changes are directly associated with large deformations, which on the contrary do not correspond to the reality of what the object is experiencing. In particular, local displacements of large volumetric objects are more affected. These might be related to occlusions causing correspondence problems by further reducing the amount of points reported by the sensor when the object is manipulated.

TABLE 1 | Prediction error of nodes position.

GNN models	Frame steps		
	t + 1	t + 5	t + 50
IN	0.08 ± 0.02	9.52 ± 7.14	46.28 ± 30.53
PropNet	0.08 ± 0.02	9.53 ± 7.14	53.66 ± 37.08

RMSE (10^{-5}) values in meters obtained by the GNN models at different time steps during the manipulation of the small sponge.

5.2. Shape Prediction Reliability

The main advantage of combining a GNN predictive model (IN and PropNet) with a self-organizing model (BC-GNG) is the fact that the training data generated by the latter are dynamic graphs with efficient size. As noted in Li et al. (2019a), training GNNs with large static graphs may overload memory capacity and delay convergence. Furthermore, such models do not perform well in dynamical settings due to unnecessary interactions associated to a fully connected graph topology. The proposed combination of models contributes to overcome these important constraints, which can be detrimental to successful robotic manipulation of deformable objects. Thus, the GNN models trained in combination with the BC-GNG graphs effectively capture the immediate changes of the object shape when evaluated in single-step, or short-term, time sequences and demonstrate potential to produce robust and visually plausible predictions of the deformation dynamics. On the other hand, while their performance tends to degrade over longer term predictions, anticipating an object's shape deformation a few steps ahead is representative of what human beings can realistically achieve, and generally proves sufficient for robotic manipulation supported by modern RGB-D sensors that can now capture point clouds in real-time. Given that the modeling and prediction framework is meant to be part of the robotic

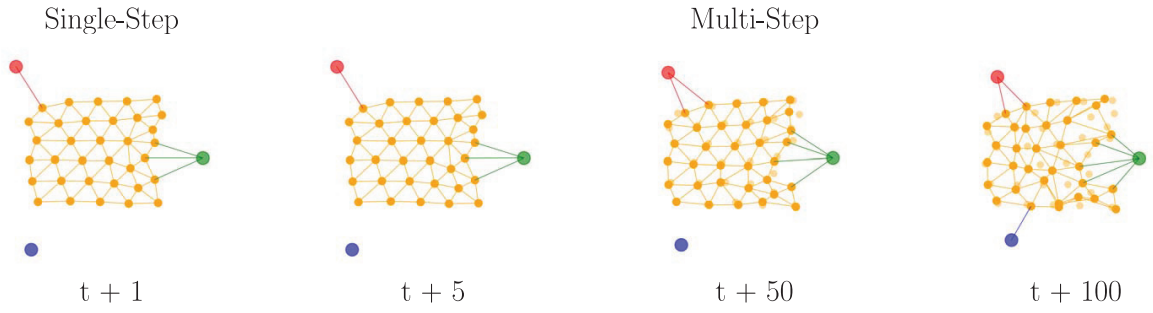

FIGURE 7 | Qualitative results of the shape prediction method. Graph sequences predicted by the GNN model (PropNet) over different time horizons for the small sponge with three contact points (nodes in red-green-blue). The predicted shape corresponds to the nodes position produced by the model. The observed nodes position are displayed in shaded color.

hand control loop, new RGB-D data is made available to update the deformable object representation, and provide an updated prediction, at the same frame rate as the robot controller. As a result, long-term prediction is not of essence in this type of application. According to the configuration used, we also notice that the performance of the GNN models are very similar. Although, the latter could be affected by the fact that PropNet shows faster convergence in training than IN due to the multi-step propagation phase.

6. CONCLUSION

This paper presents a first attempt at using graph models to learn the dynamics of deformable objects entirely from RGB-D sensor measurements. The proposed BC-GNG formulation improves the performance over C-GNG by producing graphs with better node stability, correspondence in regions with shape variations and lower computational cost. These properties enable to combine other graph models such as GNNs to predict the deformation dynamics of non-rigid objects.

By combining the relational structure of self-organizing and graph neural networks, the proposed approach successfully captures the object shape and predicts the deformation dynamics when evaluated over single-step or short-term time sequences. In comparison to analytical models, execution time is faster and information on the shape and physical properties of the object does not need to be known or approximated a priori. Therefore, the proposed combination of graph models and their adaptation demonstrate strong potential for characterizing deformable objects' shape and dynamics, as required to support advanced dexterous robotic manipulation.

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

Both authors contributed to the overall conception of the methodology, experimentation, analysis and manuscript writing.

FUNDING

The authors wish to acknowledged financial support to this research from the Natural Sciences and Engineering Research Council of Canada (NSERC) under research grant #RGPIN-2015-05328, the Canada Foundation for Innovation (CFI), and CALDO-SENESCYT scholars program.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/frobt.2020.600584/full#supplementary-material

REFERENCES

Arriola-Rios, V. E., and Wyatt, J. L. (2017). A multimodal model of object deformation under robotic pushing. *IEEE Trans. Cogn. Dev. Syst*. 9, 153–169. doi: 10.1109/TCDS.2017.2664058

Battaglia, P., Pascanu, R., Lai, M., Jimenez Rezende, D., and Kavukcuoglu, K. (2016). "Interaction networks for learning about objects, relations and physics," in *Advances in Neural Information Processing Systems 29*, eds D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Barcelona: Curran Associates, Inc.), 4502–4510.

Billard, A., and Kragic, D. (2019). Trends and challenges in robot manipulation. *Science* 364:eaat8414. doi: 10.1126/science.aat8414

Bradski, G. (2000). The OpenCV Library. *Dr Dobbs J. Softw.Tools* 25, 120–125. Available online at: https://github.com/opencv/opencv/wiki/CiteOpenCV

Caccamo, S., Güler, P., Kjellström, H., and Kragic, D. (2016). "Active perception and modeling of deformable surfaces using Gaussian processes and position-based dynamics," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)* (Cancun: IEEE), 530–537. doi: 10.1109/HUMANOIDS.2016.7803326

Cretu, A., Payeur, P., and Petriu, E. M. (2012). Soft object deformation monitoring and learning for model-based robotic hand manipulation. *IEEE Trans. Syst. Man Cybernet. B* 42, 740–753. doi: 10.1109/TSMCB.2011.2176115

Duenser, S., Bern, J. M., Poranne, R., and Coros, S. (2018). "Interactive robotic manipulation of elastic objects," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Madrid: IEEE), 3476–3481. doi: 10.1109/IROS.2018.8594291

Frank, B., Stachniss, C., Schmedding, R., Teschner, M., and Burgard, W. (2014). Learning object deformation models for robot motion planning. *Robot. Auton. Syst*. 62, 1153–1174. doi: 10.1016/j.robot.2014.04.005

Fritzke, B. (1995). "A growing neural gas network learns topologies," in *Proceedings of the 7th International Conference on Neural Information Processing Systems (NIPS'94)* (Cambridge, MA: MIT Press), 625–632.

Fugl, A. R., Jordt, A., Petersen, H. G., Willatzen, M., and Koch, R. (2012). "Simultaneous estimation of material properties and pose for deformable objects from depth and color images," in *Pattern Recognition*, eds A. Pinz, T. Pock, H. Bischof, and F. Leberl (Berlin; Heidelberg: Springer), 165–174. doi: 10.1007/978-3-642-32717-9_17

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). "Neural message passing for quantum chemistry," in *International Conference on Machine Learning* (Sydney, NSW: PMLR), 1263–1272.

Güler, P., Pauwels, K., Pieropan, A., Kjellström, H., and Kragic, D. (2015). "Estimating the deformability of elastic materials using optical flow and position-based dynamics," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)* (Seoul: IEEE), 965–971. doi: 10.1109/HUMANOIDS.2015.7363486

Jia, Y.-B., Guo, F., and Lin, H. (2014). Grasping deformable planar objects: squeeze, stick/slip analysis, and energy-based optimalities. *Int. J. Robot. Res*. 33, 866–897. doi: 10.1177/0278364913512170

Khalil, F. F., Curtis, P., and Payeur, P. (2010). "Visual monitoring of surface deformations on objects manipulated with a robotic hand," in *2010 IEEE International Workshop on Robotic and Sensors Environments* (Phoenix, AZ: IEEE), 1–6. doi: 10.1109/ROSE.2010.5675327

Kingma, D. P., and Ba, J. (2015). "Adam: a method for stochastic optimization," in *3rd International Conference for Learning Representations* (San Diego, CA).

Lang, J., Pai, D. K., and Woodham, R. J. (2002). Acquisition of elastic models for interactive simulation. *Int. J. Robot. Res*. 21, 713–733. doi: 10.1177/027836402761412458

Leizea, I., Álvarez, H., Aguinaga, I., and Borro, D. (2014). "Real-time deformation, registration and tracking of solids based on physical simulation," in *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (Munich), 165–170. doi: 10.1109/ISMAR.2014.6948423

Leizea, I., Mendizabal, A., Alvarez, H., Aguinaga, I., Borro, D., and Sanchez, E. (2017). Real-time visual tracking of deformable objects in robot-assisted surgery. *IEEE Comput. Graph. Appl.* 37, 56–68. doi: 10.1109/MCG.2015.96

Li, Y., Wu, J., Tedrake, R., Tenenbaum, J. B., and Torralba, A. (2019a). "Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids," in *International Conference on Learning Representations* (New Orleans, LA).

Li, Y., Wu, J., Zhu, J.-Y., Tenenbaum, J. B., Torralba, A., and Tedrake, R. (2019b). "Propagation networks for model-based control under partial observation," in *2019 International Conference on Robotics and Automation (ICRA)* (Montreal, QC: IEEE), 1205–1211. doi: 10.1109/ICRA.2019.8793509

Mrowca, D., Zhuang, C., Wang, E., Haber, N., Fei-Fei, L. F., Tenenbaum, J., et al. (2018). "Flexible neural representation for physics prediction," in *Advances in Neural Information Processing Systems 31* (Montreal, QC: Curran Associates, Inc.), 8799–8810.

Nadon, F., Valencia, A. J., and Payeur, P. (2018). Multi-modal sensing and robotic manipulation of non-rigid objects: a survey. *Robotics* 7:74. doi: 10.3390/robotics7040074

Nealen, A., Müller, M., Keiser, R., Boxerman, E., and Carlson, M. (2006). Physically based deformable models in computer graphics. *Comput. Graph. Forum* 25, 809–836. doi: 10.1111/j.1467-8659.2006.01000.x

Orts-Escolano, S., Garcia-Rodriguez, J., Morell, V., Cazorla, M., Saval, M., and Azorin, J. (2015). "Processing point cloud sequences with Growing Neural Gas," in *2015 International Joint Conference on Neural Networks (IJCNN)* (Killarney: IEEE), 1–8. doi: 10.1109/IJCNN.2015.7280709

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). "PyTorch: an imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, ed H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett (Vancouver, BC: Curran Associates, Inc.), 8024–8035.

Petit, A., Lippiello, V., and Siciliano, B. (2015). "Real-time tracking of 3D elastic objects with an RGB-D sensor," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Hamburg: IEEE), 3914–3921. doi: 10.1109/IROS.2015.7353928

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., et al. (2009). "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software* (Kobe), 5.

Sanchez, J., Corrales, J.-A., Bouzgarrou, B.-C., and Mezouar, Y. (2018). Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey. *Int. J. Robot. Res.* 37, 688–716. doi: 10.1177/0278364918779698

Sengupta, A., Lagneau, R., Krupa, A., Marchand, E., and Marchal, M. (2020). "Simultaneous tracking and elasticity parameter estimation of deformable objects," in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (Paris: IEEE), 10038–10044. doi: 10.1109/ICRA40945.2020.9196770

Tawbe, B., and Cretu, A.-M. (2017). Acquisition and neural network prediction of 3D deformable object shape using a kinect and a force-torque sensor. *Sensors* 17:1083. doi: 10.3390/s17051083

Valencia, A. J., Nadon, F., and Payeur, P. (2019). "Toward real-time 3D shape tracking of deformable objects for robotic manipulation and shape control," in *2019 IEEE Sensors* (Montreal, QC: IEEE), 1–4. doi: 10.1109/SENSORS43011.2019.8956623

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods* 17, 261–272. doi: 10.1038/s41592-019-0686-2

Wang, M., Yu, L., Zheng, D., Gan, Q., Gai, Y., Ye, Z., et al. (2019). "Deep graph library: towards efficient and scalable deep learning on graphs," in *ICLR Workshop on Representation Learning on Graphs and Manifolds* (New Orleans, LA).

Zaidi, L., Corrales, J. A., Bouzgarrou, B. C., Mezouar, Y., and Sabourin, L. (2017). Model-based strategy for grasping 3D deformable objects using a multi-fingered robotic hand. *Robot. Auton. Syst.* 95, 196–206. doi: 10.1016/j.robot.2017.06.011

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.