frontiers in Robotics and AI

REVIEW
published: 17 September 2020
doi: 10.3389/frobt.2020.00082

Modeling of Deformable Objects for Robotic Manipulation: A Tutorial and Review

Veronica E. Arriola-Rios[1], Puren Guler[2]*, Fanny Ficuciello[3], Danica Kragic[4], Bruno Siciliano[3] and Jeremy L. Wyatt[5]*

[1] Department of Mathematics, Faculty of Science, UNAM Universidad Nacional Autonoma de Mexico, Ciudad de México, Mexico, [2] Autonomous Mobile Manipulation Laboratory, Centre for Applied Autonomous Sensor Systems, Orebro University, Orebro, Sweden, [3] PRISMA Laboratory, Department of Electrical Engineering and Information Technology, University of Naples Federico II, Naples, Italy, [4] Robotics, Learning and Perception Laboratory, Centre for Autonomous Systems, EECS, KTH Royal Institute of Technology, Stockholm, Sweden, [5] School of Computer Science, University of Birmingham, Birmingham, United Kingdom

Manipulation of deformable objects has given rise to an important set of open problems in the field of robotics. Application areas include robotic surgery, household robotics, manufacturing, logistics, and agriculture, to name a few. Related research problems span modeling and estimation of an object's shape, estimation of an object's material properties, such as elasticity and plasticity, object tracking and state estimation during manipulation, and manipulation planning and control. In this survey article, we start by providing a tutorial on foundational aspects of models of shape and shape dynamics. We then use this as the basis for a review of existing work on learning and estimation of these models and on motion planning and control to achieve desired deformations. We also discuss potential future lines of work.

Keywords: deformable objects, shape representation, learning of deformation, control of deformable objects, registration of shape deformation, tracking of deformation

OPEN ACCESS

Edited by:
Hakan Karaoguz, Independent Researcher, Stockholm, Sweden

Reviewed by:
Farah Bouakrif, University of Jijel, Algeria
Dongming Gan, Purdue University, United States

****Correspondence:***
Veronica E. Arriola-Rios
v.arriola@ciencias.unam.mx
Puren Guler
puren.guler@oru.se

Specialty section:
This article was submitted to Robot and Machine Vision, a section of the journal Frontiers in Robotics and AI

Received: *13 February 2020*
Accepted: *19 May 2020*
Published: *17 September 2020*

Citation:
Arriola-Rios VE, Guler P, Ficuciello F, Kragic D, Siciliano B and Wyatt JL (2020) Modeling of Deformable Objects for Robotic Manipulation: A Tutorial and Review. Front. Robot. AI 7:82. doi: 10.3389/frobt.2020.00082

1. INTRODUCTION

Robotic manipulation work tends to focus on rigid objects (Bohg et al., 2014; Billard and Kragic, 2019). However, most objects manipulated by animals and humans change shape upon contact. Manipulating a deformable object presents a quite different set of challenges from those that arise when manipulating a rigid object. For example, forces applied to a rigid object simply sum to determine the external wrench and, when integrated over time, result in a sequence of rigid body transformations in SE(3). This is relatively simple dynamic model, albeit still difficult to estimate for a given object, manipulator, and set of environment contacts.

Forces applied to a deformable body, by contrast, both move the object and change its *shape*. The exact combination of deformation and motion depends on the precise material composition. Thus, *material properties* become a critical part of the system dynamics, and consequently the underlying physics of deformation is complex and hard to capture. In addition, the *dynamics* models typically employed in high-fidelity mechanical modeling—such as finite element models—while precise, require detailed knowledge of the material properties, which would be unavailable to a robot in the wild. Yet such a lack of detailed physics knowledge does not prevent humans and other animals from performing dexterous manipulation of deformable objects. Consider the way a New Caledonian crow shapes a tool from a branch (Weir and Kacelnik, 2006) or how a pizzaiolo

dexterously transforms a ball of dough into a pizza. Clearly, robots have a long way to go to match these abilities.

Even though a great deal of work has been done on developing solutions for each of these stages (Cootes et al., 1992; Montagnat et al., 2001; Nealen et al., 2006; Moore and Molloy, 2007), only a few combinations have actually been tried in robotics to date (Nadon et al., 2018; Sanchez et al., 2018). In contrast to earlier review articles, this paper consists of both a tutorial and a review of related work, aimed at newcomers to robotic modeling and manipulation of deformable objects who need a quick introduction to the basic methods, which are adopted from various other fields and backgrounds. We provide, in a single place, a menu of possible approaches motivated by computer vision, computer graphics, physics, machine learning, and other fields, in the hope that readers can find new material and inspiration for creatively developing their work. We then review how these methods are applied in practice. In this way, we take a more holistic approach than existing reviews. Since the literature in the fields we touch on is abundant, we cannot be exhaustive and will focus mostly on manipulation of volumetric solid objects.

This review is motivated by a future goal in the form of an ideal scenario. In this scenario a general purpose robot would be capable of perceiving *shape*, *dynamics*, and necessary *material properties* (e.g., elasticity, plasticity) of deformable objects to implement manipulation strategies relying on planning and control methods. Currently no robot has all these capabilities. Therefore, we decompose the problem space into five main parts that work like pieces in a puzzle: Because representational choices are fundamental, we explain, in a tutorial style, (1) the modeling of shape (section 2) and (2) the modeling of deformation dynamics (section 3) to provide the reader with the necessary mathematical background; then we discuss, in survey form, (3) learning and estimation of the parameters of these models that are related to deformability of objects (e.g., material properties, such as elasticity, or shape properties, such as resolution of a mesh; section 4), (4) the application of the models to perception and prediction, and (5) planning and control of manipulation actions (section 5), since these topics build on the models explained in (1) and (2) and there is such a wide range of different approaches that it would be impossible to cover them all in depth. **Figure 1** shows our guideline processing stream: (i) The robot perceives the object, segments it from its environment, and selects an adequate representation for its shape and intended task; the desired type of representation will determine which algorithms must be used to recognize the object. (ii) As the robot interacts with the object, it deforms the object and must register the deformation by modifying the shape representation accordingly; while doing so, it can make use of tracking/registration techniques or enhanced predictive tracking (which requires a model of the dynamics). (iii) A suitable model for the dynamics is selected to predict new configurations of the shape representation as the robot interacts with the object. (iv) Information from the previous stages is used to integrate

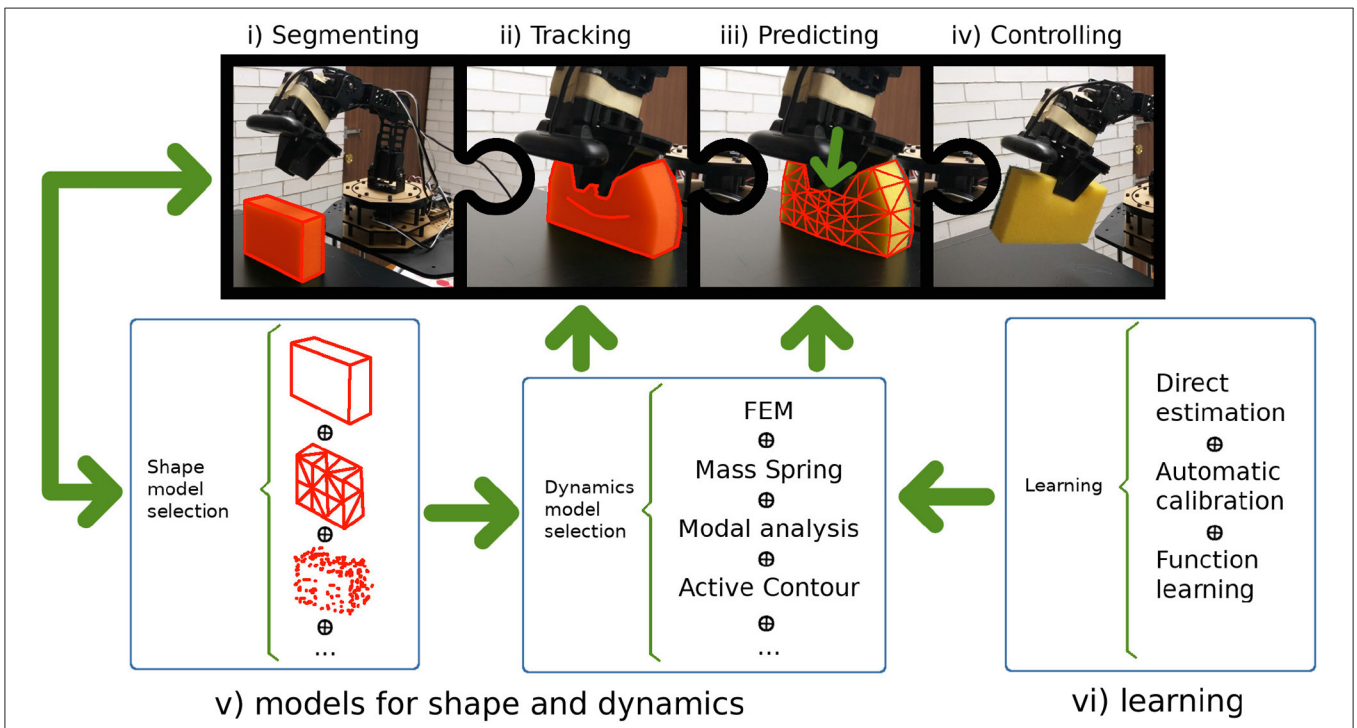

FIGURE 1 | A general purpose robot must be capable of: **(i)** perceiving and segmenting the object from the scene; **(ii)** tracking the object's motion; **(iii)** predicting the object's behavior; **(iv)** planning and controlling new manipulation strategies based on the predictions; **(v)** selecting the appropriate model for each task, since all models for shape and dynamics have limitations; and **(vi)** learning new models for previously unknown shapes and materials.

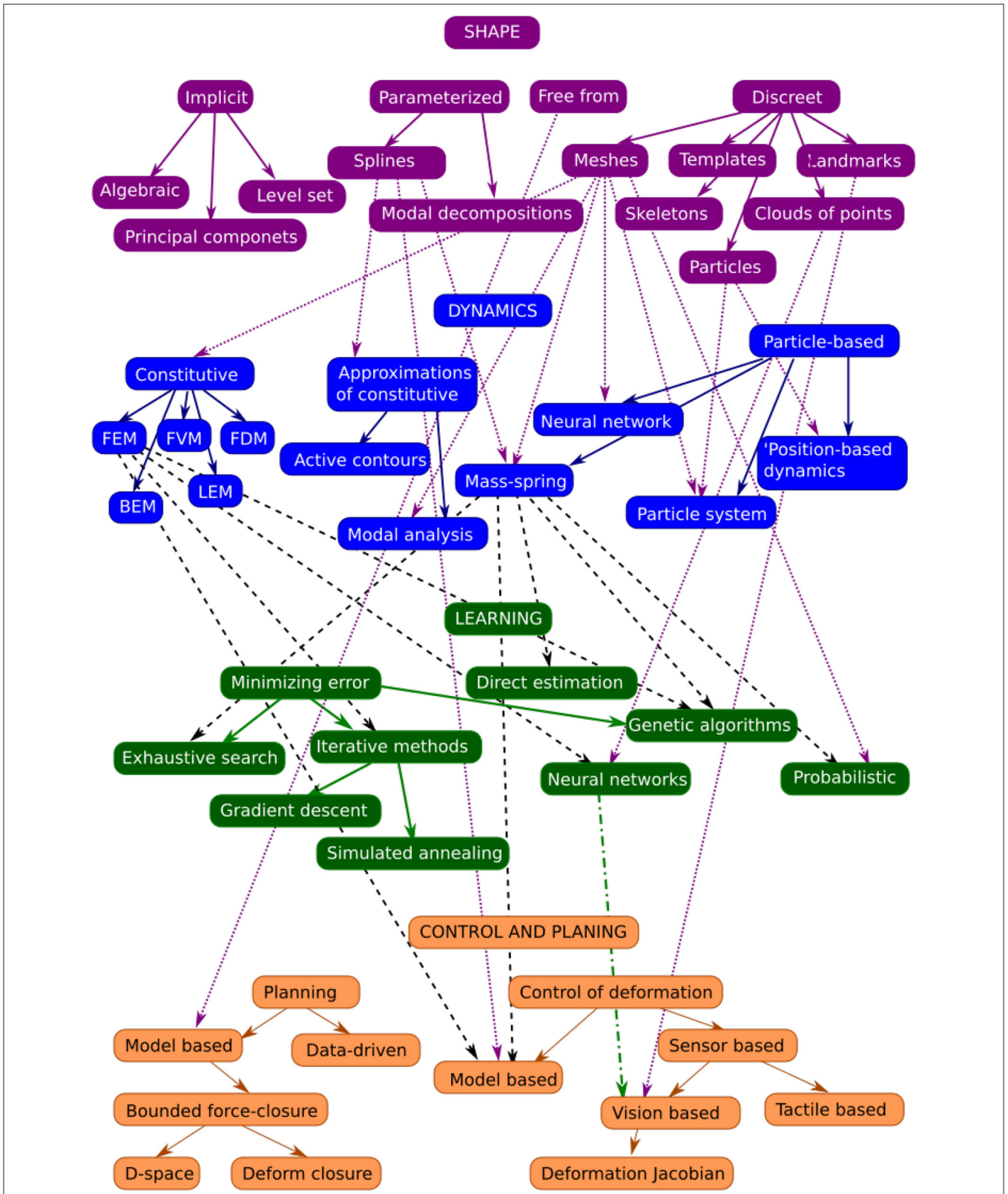

FIGURE 2 | Relationships between shape, dynamics, learning models, and control methodologies as they have been used in the literature. Colored arrows indicate subcategories, while black arrows show when a methodology in one level has been used for the next one.

the inputs and rules for the control strategies. Finally, all this information can be fed into learning algorithms capable of increasing the repertoire of known objects. At this stage, there are three types of strategies: (a) estimating new parameters directly (which is rarely viable); (b) calibrating known physics-based models automatically by allowing the robot to take some measurements that will help to determine the value of model parameters; and (c) approximating new functions that describe the dynamics, as is done with neural networks.

Figure 2 shows the connections between the different models covered in this review as they have been used in the publications mentioned. **Table 1** gives a summary of the publications discussed in each section. The following describes the notation that will be used throughout the paper:

- Scalars: italic lower-case letters, e.g., x, y, z.
- Vectors: bold lower-case letters, e.g., $\mathbf{p} = \{x, y, z\}^T$.
- Matrices: bold upper-case letters, e.g., $\mathbf{R}$.

TABLE 1 | Publication summary based on some papers from each section.

Shape	Implicit	Algebraic	Gascuel, 1993; Kumar et al., 1995; Jaklic et al., 2000
		Level set	Sethian, 1997; Cremers, 2006; Sun et al., 2008
		Eigenmodes	Cootes et al., 1995; Blake et al., 1998; Leventon et al., 2000; Tsai et al., 2001; Cremers, 2006
	Parameterized	Splines	de Boor, 1976; Catmull and Clack, 1978; Kass et al., 1988; Gibson and Mirtich, 1997; Unser, 1999; Cordero Valle and Cortes Parejo, 2003; Sederberg et al., 2003; Maraffi, 2004; Song and Bai, 2008; Prasad et al., 2010
		Modal decomposition	Szekely et al., 1995
	Free-form		Sederberg and Parry, 1986; Moore and Molloy, 2007
		Multigrid	Xian et al., 2019
	Discrete	Meshes	Delingette, 1999; Montagnat et al., 2001; Arvanitis et al., 2019
		Skeletons	Schaefer and Yuksel, 2007
		Templates	Yuille et al., 1992; Basri et al., 1998; Ravishankar et al., 2008; Arriola-Rios et al., 2013; Gallardo et al., 2020
		Landmarks	Blake et al., 1998; Cootes and Taylor, 2004
		Particles	Nealen et al., 2006
		Cloud of points	Cretu et al., 2009; Newcombe and Davison, 2010; Martínez et al., 2019; Makovetskii et al., 2020
Dynamics	Particle-based	Particle systems	Tonnesen and Terzopoulos, 2000
		Mass-spring systems	Bianchi et al., 2004; Teschner et al., 2004; Morris and Salisbury, 2008; Schulman et al., 2013; Arriola-Rios and Wyatt, 2017
		Neural networks	Nurnberger et al., 1998; Zhang et al., 2019
		Position-based	Müller et al., 2005; Zhu et al., 2008; Tian et al., 2013; Macklin et al., 2014; Sidorov and Marshall, 2014; Guler et al., 2017; Romeo et al., 2020
	Constitutive	FEM	Essa et al., 1992; Frank et al., 2014; Petit et al., 2018
		FVM	Teran et al., 2003; Barth et al., 2018
		FDM	Terzopoulos et al., 1987
		BEM	Greminger and Nelson, 2008
		LEM	Balaniuk and Salisbury, 2002
	Approximations	Modal analysis	Pentland and Williams, 1989; Barbič and James, 2005; Fulton et al., 2019
		Active contours	Kass et al., 1988; Ahlberg, 1996; Nisirat, 2019
Learning	Discrete		Gelder, 1998
	Minimizing error	Exhaustive search	Guler et al., 2015
		Iterative methods	Teschner et al., 2004; Frank et al., 2014
		Genetic algorithms	Bianchi et al., 2004
		Neural networks	Cretu et al., 2012
	Probability		Risholm et al., 2010; Schulman et al., 2013
Control and planning	Planning	Model-based	Gopalakrishnan and Goldberg, 2004; Das and Sarkar, 2011; Frank et al., 2014
		Data-driven	Mira et al., 2015; Li et al., 2016
	Control	Model-based	Largilliere et al., 2015; Lin et al., 2015; Zaidi et al., 2017; Ficuciello et al., 2018
		Sensor-based	Wada et al., 2001; Smolen and Patriciu, 2009; Berenson, 2013; Navarro-Alarcon et al., 2016; Delgado et al., 2017b; Hu et al., 2019; Cherubini et al., 2020

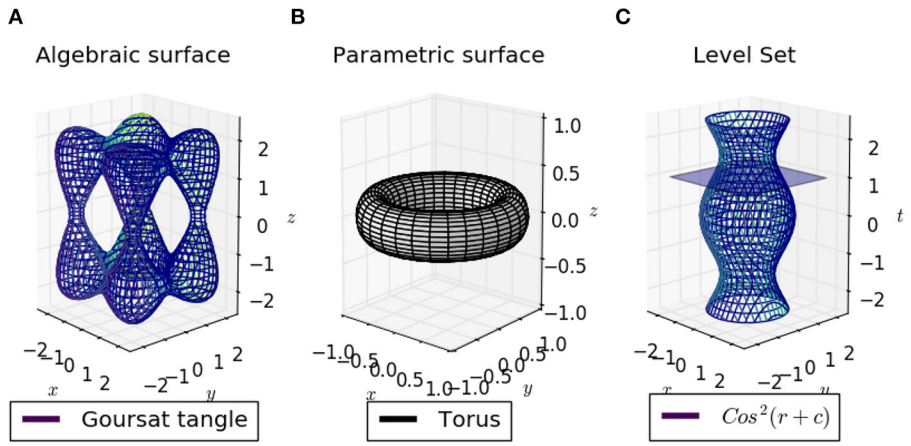

FIGURE 3 | (A) Algebraic surfaces can represent basic and complex shapes, such as circles or the tangled cube; a limited but useful set of deformations is straightforward to define, while other deformations that are not so intuitive tend to be used in combination with skeletons and skinning techniques. **(B)** Parametric surfaces can represent a wider variety of shapes and their great flexibility allows deformation to be controlled with more intuitive parameters; therefore their use in connection with models of dynamics is considerable. **(C)** Level set curves can show the evolution of a 2D shape in time; the intersection of a time plane with the surface in 3D is a level curve, which represents the 2D contour shape of the object at a given time t. In this example $\cos^2(r+c)$ is the mathematical representation of the 2D contour shape's evolution through time.

- Scalar functions whose range is $\mathbb{R}$: italic lower-case letters followed by parentheses, e.g., $f(\cdot)$.
- Vector functions whose range is $\mathbb{R}^n$ with $n > 1$: bold italic letters followed by parentheses, e.g., $\boldsymbol{S}(\cdot)$.
- Sets: calligraphic letters, e.g., $\mathcal{S}$.
- Number sets: blackboard bold upper-case letters, e.g., $\mathbb{R}, \mathbb{N}$.

2. REPRESENTING SHAPE FOR DEFORMABLE OBJECTS

The initial problem of manipulating a deformable object is to perceive and segment the object shape from the scene as it deforms (**Figure 1i**). The main difficulty with this problem is the number of degrees of freedom required to model the object shape. The expressiveness, accuracy, and flexibility of the model can ease the modeling of the dynamics or make it more difficult in different scenarios. For this reason, this section introduces a variety of models from mathematics, computer graphics, and computer vision for representation of the shape of deformable objects.

2.1. Implicit Curves and Surfaces

An implicit curve or surface of dimension $n-1$ is generally defined as the zero set of a function $f : \mathbb{R}^n \to \mathbb{R}$,

$$\mathcal{S}_f = \{\mathbf{p} \in \mathbb{R}^n \mid f(\mathbf{p}) = 0\}, \tag{1}$$

where $\mathbf{p}$ is a coordinate in an n-dimensional space in which the surface is embedded (Montagnat et al., 2001). Therefore, the set $\mathcal{S}_f$ defines the surface formed by all points $\mathbf{p}$ in $\mathbb{R}^n$ such that the function f, when evaluated at $\mathbf{p}$, is equal to zero. The implicit function is used to locate surface points by solving the equation $f(\mathbf{p}) = 0$ (**Figure 3A**). In robotics, n is usually 3 to represent Cartesian coordinates, but sometimes an extra dimension can be used to represent time. Representations that fall within this category are explained in the rest of this subsection.

2.1.1. Algebraic Curves and Surfaces

Algebraic curves and surfaces satisfy (1) with $f(\mathbf{p})$ being a polynomial. First-degree polynomials define *planes* and *hyperplanes*; second-degree polynomials define *conics*, which include circles, ellipses, parabolas, and hyperbolas, and *quadrics*, which include ellipsoids, paraboloids, hyperboloids, toroids, cones, and cylinders; their $(n-1)$-dimensional extensions are surfaces in an n-dimensional space that satisfy the equation

$$f(\mathbf{p}) = \mathbf{p}^T\mathbf{A}\mathbf{p} + \mathbf{b}\mathbf{p} + c = 0 \tag{2}$$

where $\mathbf{p} = \{x_1, x_2, \ldots, x_n\}^T \in \mathbb{R}^n$ is a column vector, $\mathbf{p}^T$ denotes the transpose of $\mathbf{p}$ (a row vector), $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a matrix, $\mathbf{b} \in \mathbb{R}^n$ is a row vector, and c is a scalar constant. Note that all the aforementioned shapes are included as particular cases of this definition. To define a particular shape, which satisfies a given set of constraints, the constant values in $\mathbf{A}, \mathbf{b}$, and c must be determined; for example, with $n = 2$, the constants that define a circle passing through a given set of three points can be found by solving the system of three equations where $f(\mathbf{p}_i) = 0$ for all i and $f(\mathbf{p}_i)$ is a second-degree polynomial. In some contexts the same equation can be rewritten to facilitate this estimation; for example, it is easy to determine the circle centered at (x_c, y_c) with radius r if the second-degree polynomial is written as $(x-x_c)^2 + (y-y_c)^2 = r^2$ with $\mathbf{p} = \{x, y\}$.

Superquadrics are defined by second-degree polynomials, while *hyperquadrics* are the most general form and allow the

representation of complex non-symmetric shapes; they are given by the equation

$$\sum_{i=1}^{N} |a_i x + b_i y + c_i z + d_i|^{\gamma_i} = 1 \tag{3}$$

where $\mathbf{p} = \{x, y, z\}^T \in \mathbb{R}^3$, N is an arbitrary number of planes whose intersection surrounds the object, a, b, c, and d are shape parameters of these planes, and $\gamma_i \in \mathbb{R}$ with $\gamma_i \geq 0$ for all i. Kumar et al. (1995) presented a method for fitting hyperquadrics to very complex deformable shapes registered as range data. A model like (3) could be used as the base representation for a model of the dynamics if the deformations are small. It can be used in combination with local surface deformations to represent a wider set of surfaces. For example, if the superquadric is the set of points $\mathcal{Q}$ that satisfy the corresponding equation, a deformed model could be given by $\boldsymbol{S} = \mathbf{c} + \mathbf{R}(\mathcal{Q} + \mathbf{d})$ where $\mathbf{c}$ represents the inertial center of the superquadric, $\mathbf{R}$ is a rotation matrix, and $\mathbf{d}$ is a vectorial displacement field. Other types of deformation can be defined as well (Montagnat et al., 2001). For more information about superquadrics, see Jaklic et al. (2000).

Algebraic curves, surfaces, and volumes can be used as 1D, 2D, and 3D skeletons, or to represent objects of similar shapes and deformations (Gascuel, 1993). They are easy to deform in certain cases, but the types of deformations that are straightforward to apply are very limited, such as matrix transformations that bend, pitch, twist, stretch, or translate all the points on a curve or the space in which the curve is embedded. For this reason, algebraic curves and surfaces are more suited to representing articulated or semi-articulated objects. Objects can also be composed of several algebraic curves, where each component is easy to manipulate with this representation. To model more complex deformed shapes, Raposo and Gomes (2019) introduced products of primitive algebraic surfaces, such as spheres and cylinders, which enable both local and global deformations to be controlled more easily than in traditional algebraic shape models. Moreover, these models can be combined with skinning techniques to emulate soft deformable objects and also have parameterized representations.

2.1.2. Level Set Methods

In level set methods, the deformable model is embedded in a higher-dimensional space, where the extra dimension represents time (Sethian, 1997; Montagnat et al., 2001). A hypersurface $\boldsymbol{\Psi}$ is defined by $\boldsymbol{\Psi}(\mathbf{p}, 0) = \text{dist}(\mathbf{p}, \boldsymbol{S}_0)$, where $\boldsymbol{S}_0$ is the initial surface and dist can be the signed Euclidean distance between a point $\mathbf{p}$ and the surface. The distance is positive if the point lies outside the surface and negative otherwise. The evolution of the surface $\boldsymbol{S}$ is governed by the partial differential equation $\boldsymbol{\Psi}_t + |\nabla \boldsymbol{\Psi}| F = 0$ involving the function $\boldsymbol{\Psi}(\mathbf{p}, t)$ and a speed function F, which determines the speed at which each point of the surface must be moved. Thus, the function $\boldsymbol{\Psi}$ evolves in time and the current shape corresponds to the surface given by $\boldsymbol{\Psi}(\mathbf{p}, t) = 0$ (**Figure 3C**). The function $\boldsymbol{\Psi}$ could have any form, such as a parameterized one, $\boldsymbol{\Psi} = \boldsymbol{\Psi}(x(u, v), y(u, v), z(u, v), t)$, with the surface still defined in implicit form, $\boldsymbol{\Psi}(x(u, v), y(u, v), z(u, v), t) = \mathbf{0}$. Unfortunately, it could also be that $\boldsymbol{\Psi}$ does not even have an algebraic expression and may need to be approximated with numerical methods. The main advantage of level set methods is that they allow changes of surface topology implicitly. The set $\mathcal{S}$ may split into several connected components, or several distinct components may merge, while $\boldsymbol{\Psi}$ remains a function.

In computer vision applications, such as human tracking and medical imaging, level set methods have been used successfully in tracking deformable objects (Sethian, 1997; Cremers, 2006). For example, Sun et al. (2008) recursively segmented deformable objects across a sequence of frames using a low-dimensional modal representation and applied their technique to left ventricular segmentation across a cardiac cycle. The dynamics are represented using a distance level set function, whose representation is simplified using *principal component analysis* (PCA). Sun et al. used methods of particle-based smoothing as well as non-parametric belief propagation on a loopy graphical model capturing the temporal periodicity of the heart, with the objective being to estimate the current state of the object not only from the data observed at that instant but also from predictions based on past and future boundary estimates. Even though we did not find examples of this method being used in robotics, it seems a suitable candidate since it models the shape change through time implicitly and would thus allow the robot to keep track of the evolving shape of an object during manipulation.

2.1.3. Gaussian Principal Component Eigenmodes

This kind of representation is valid when the types of deformations can be described with a single mathematical formulation. Given a representative set $\mathcal{S}_N = \{\boldsymbol{S}_0, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_{N-1}\}$ of the types of surface deformation that objects can undergo, it is possible to use PCA to detect the main modes of deformation (i.e., the eigenmodes $\boldsymbol{\Phi}_n$) and thus re-express the shapes as a linear combination of those modes. Hence a new shape estimation can be done using

$$\bar{\boldsymbol{S}} = \boldsymbol{S}_\mu + \alpha \boldsymbol{\Phi}_n \tag{4}$$

where $\boldsymbol{\Phi}_n$ ($n \ll N$) is the largest eigenmode of shape variations in $\boldsymbol{S}_N$, $\boldsymbol{S}_\mu$ is the mean of the representative set of shapes $\boldsymbol{S}_N$, and α is a set of coefficients. Such an eigenmode representation is useful for dealing with missing or misleading information (e.g., noise or occlusions) coming from sensory data while constructing the shape of the object (Cootes et al., 1995; Blake et al., 1998; Cremers, 2006; Sinha et al., 2019).

Employing combinations of previously cited methods, Leventon et al. (2000) used eigenmode representation with level set curves to segment images, such as medical images of the femur and corpus callosum, by defining a probability distribution over the variances of a set of training shapes. The segmentation process embeds an initial curve as the zero level set of a higher-dimensional surface, and then evolves the surface such that the zero level set converges on the boundary of the object to be segmented. At each step of the surface evolution, the maximum *a posteriori* position and shape of the object in the image were estimated based on the prior shape information and the image information. The surface was then evolved globally

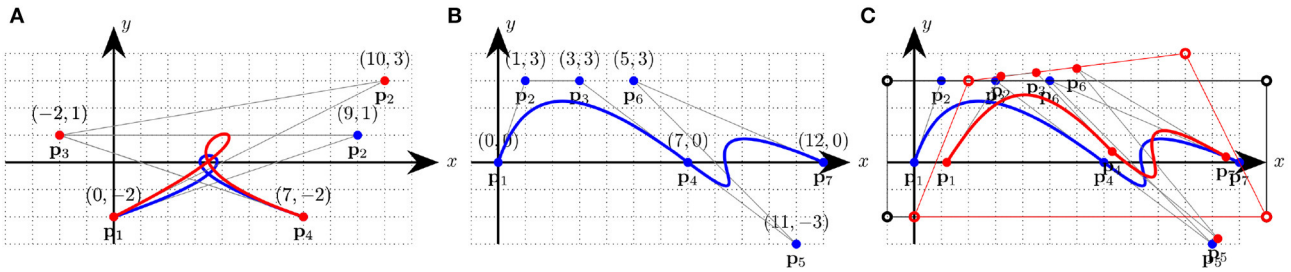

FIGURE 4 | (A) Two Bezier curves with control points; displacing one control point (e.g., $\mathbf{p}_2$) affects the entire curve. In this case, the line that joins the control points $\mathbf{p}_1$ and $\mathbf{p}_2$ is the tangent of the polynomial at $\mathbf{p}_1$; and the line that joins $\mathbf{p}_3$ and $\mathbf{p}_4$ is the tangent at $\mathbf{p}_4$. **(B)** Two enchained Bezier curves; by placing $\mathbf{p}_3$, $\mathbf{p}_4$, and $\mathbf{p}_5$ on a line it is possible to make the curve C^1 at $\mathbf{p}_4$. **(C)** Free-form deformation of a spline; the black circles form a lattice in a new local coordinate system, the red circles show the deformed lattice, and the red spline is the result of shifting the blue spline in accordance with this deformation.

toward the maximum *a posteriori* estimate and locally based on image gradients and curvature. The results were demonstrated on synthetic data and medical images, in both 2D and 3D. Tsai et al. (2001) further developed this idea.

2.2. Explicit Parameterized Representations

Explicit parameterized representations are evaluated directly from their functional definition (**Figure 3B**). In 3D, they are of the form $\boldsymbol{C}(u) = \{x(u), y(u), z(u)\}^T$ for curves and $\boldsymbol{S}(u, v) = \{x(u, v), y(u, v), z(u, v)\}^T$ for surfaces, where u and v are parameters. The curve or surface is traced out as the values of the parameters are varied. For example, if $u \in [0, 1]$, the shape is traced out as u varies from 0 to 1, as happens with the circle

$$\boldsymbol{C}(u) = \begin{Bmatrix} x = cos(u) \\ y = sin(u) \end{Bmatrix}. \tag{5}$$

It is common practice to parameterize a curve by time t if it will represent a trajectory, or by arc length l[1].

2.2.1. Splines

A mathematical spline $\boldsymbol{S}$ is a piecewise-defined real function, with k polynomial pieces $s_i(u)$ parameterized by $u \in [u_0, u_k]$, used to represent curves or surfaces (Cordero Valle and Cortes Parejo, 2003). The order n of the spline corresponds to the highest order of the polynomials. The values $u_0, u_1, \ldots, u_{k-1}, u_k$ where the polynomial pieces connect are called *knots*.

Frequently, for a spline of order n, $\boldsymbol{S}$ is required to be differentiable up to order $n - 1$, that is, to be C^{n-1} at knots and C^∞ everywhere else. However, it is also possible to reduce its differentiability to take into account discontinuities.

In general, any spline function $\boldsymbol{S}(u)$ of order n with knots $u_0, \ldots, u_k$ can be expressed as

$$\boldsymbol{S}(u) = \sum_{j=1}^{k+n+1} \mathbf{p}_j s_j(u), \tag{6}$$

where the coefficients $\mathbf{p}_j$ are interpreted geometrically as the coordinates of the *control points* that determine the shape of the spline, and

$$\begin{aligned} s_j(u) &= (u - u_j)^n \quad \text{for } j = 1, \ldots, k, \\ s_{k+j}(u) &= u^{j-1} \quad \text{for } j = 1, \ldots, (n+1) \end{aligned} \tag{7}$$

constitute a basis for the space of all spline functions with knots $u_0, \ldots, u_k$, called the *power basis*. This space of functions is a $(k + (n + 1))$-dimensional linear space. By using other bases, a large family of spline variations is generated (Gibson and Mirtich, 1997); the most important ones are the following.

Bezier splines have each segment being a Bezier curve given by

$$\mathbf{B}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_i^n(u), \tag{8}$$

$$B_i^n(u) = \binom{n}{i} (1 - u)^{n-i} u^i, \tag{9}$$

where each $\mathbf{p}_i$ is a control point, the B_i^n are the Bernstein polynomials of degree n, $\binom{n}{i}$ are the binomial coefficients, and $u \in [0, 1]$. The curve passes through its first and last control points, $\mathbf{p}_0$ and $\mathbf{p}_n$, and remains close to the control polygon obtained by joining all the control points, in order, with straight lines. Also, at its extremes it is tangent to the line segment defined by $\overline{\mathbf{p}_0\mathbf{p}_1}$ and $\overline{\mathbf{p}_{n-1}\mathbf{p}_n}$. It is easy to add and remove control points from a Bezier curve, but displacing one causes the entire curve to change, which is why usually only third-degree polynomial segments are used (see **Figure 4**). A two-dimensional Bezier surface is obtained as the tensor product of two Bezier curves:

$$\mathbf{B}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u) B_j^m(v) \mathbf{p}_{i,j}. \tag{10}$$

B-splines are more stable, since changes to the positions of control points induce only local changes around that control point, and the polynomials pass through the control points (de Boor, 1976). They are particularly suitable for 3D

[1]The arc length is the distance between a starting point α on the curve and the current point β. For example, the length of a 1D curve embedded in 3D space, parameterized by u, is given by $L = \int_\alpha^\beta \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}\, du$, where $\dot{x} = \frac{\partial x}{\partial u}$ and similarly for $\dot{y}$ and $\dot{z}$.

reconstructions. Song and Bai (2008) show how they can be used to fill holes and smooth surfaces originally captured as dense clouds of points, while producing a much more compact and manipulable representation.

Catmull-Clark surfaces approximate points lying on a mesh of arbitrary topology (Catmull and Clack, 1978).

Non-uniform rational B-splines (NURBS) are notable because they can represent circles, ellipses, spheres, and other curves which, in spite of their commonness and simplicity, cannot be represented by polynomial splines. They achieve this by introducing quotients of polynomials in the basis[2].

T-splines and non-uniform rational Catmull-Clarck surfaces with T-junctions (T-NURCCs) allow for high-resolution representations of 3D deformable objects with a highly reduced number of faces and improved representations of joints between surface patches, by introducing connections with the shape of a T between edges of the shape (Sederberg et al., 2003).

Splines are a very flexible tool for representing all sorts of deformable shapes. They are extremely useful for signal and image processing (Unser, 1999) as well as for computer animation (Maraffi, 2004) and shape reconstruction of 2D and 3D deformable objects (Song and Bai, 2008; Prasad et al., 2010). *Active contours* (see section 3.4.2 and Kass et al., 1988), also known as *snakes*, are splines governed by an energy function that introduces dynamic elements to the shape representation and are used for tasks, such as object segmentation (Marcos et al., 2018; Chen et al., 2019; Hatamizadeh et al., 2019).

The advantage of splines is that compact representations of deformable objects can be built on them in accordance with the complexity of their shape at each time. When new corners or points of high curvature appear, more control points can be added for an adequate representation, and if the shape becomes simplified these points can be removed. However, this flexibility also makes splines sensitive to noise and can lead to computation of erroneous deformations. Hence, learning the dynamics of such a representation is difficult with current learning algorithms, and a general solution remains an open problem.

2.2.2. Modal Decompositions

In modal decomposition, a curve or a surface is expressed as the sum of terms in a basis, whose elements correspond to frequency harmonics. The sum of the first modes constituting the surface gives a good rough approximation of its shape, which becomes more detailed as more modes are included (Montagnat et al., 2001). Among methods for modal decomposition, Fourier decomposition is in widespread use. A curve may be represented as a sum of sinusoidal terms and a surface as a combination of spherical harmonics $\boldsymbol{Y}_l^m(\theta, \varphi)$ which are explicitly parameterized:

$$\boldsymbol{S}(r, \theta, \varphi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} c_l^m r^l \boldsymbol{Y}_l^m(\theta, \varphi), \tag{11}$$

where r^l is a normalization factor for $\boldsymbol{Y}$ and the c_l^m are constants. It is also possible to use other bases that may be more suitable for other shapes, such as surfaces homeomorphic to a sphere, torus, cylinder, or plane.

Modal decomposition has mostly been used in model-based segmentation and recognition of 2D and 3D medical images (Szekely et al., 1995). Its main advantage is that it creates a compact and easy-to-manipulate representation of objects whose shape can be described as a linear combination of a few dominant modes of deformation. The disadvantage of such methods is that it is easy for them to miss details in objects, such as small dents, because shapes are approximated by a limited number of terms.

2.3. Free-Forms

Free-form deformation is a method whereby the space in which a figure is embedded is deformed according to a set of control points of the deformation (Moore and Molloy, 2007; see **Figure 4C**). It can be used to deform primitives, such as planes, quadrics, parametric surface patches, or implicitly defined surfaces. The deformation can be applied either globally or locally. A local coordinate system is defined using a parallelpiped so that the coordinates inside it are $\mathbf{p} = \{x_1, x_2, x_3\}$ with $0 < x_i < 1$ for all i. A set of control points $\mathbf{p}_{ijk}$ lie on a lattice. When they are displaced from their original positions, they define a deformation of the original space with new coordinates $\mathbf{p}'$. The new position of any coordinate is interpolated by applying a transformation formula that maps $\mathbf{p}$ into $\mathbf{p}'$. For some transformations it is enough to estimate the new coordinates of the nodes of a mesh or control points of a spline with respect to the new positions of the control points of the deformed space, and the rest of the shape will follow them, as in Sederberg and Parry (1986), where a trivariate tensor product Bernstein polynomial was proposed as the transformation function. Loosely related are multigrid representations, which also allow for local management of deformation (Xian et al., 2019).

2.4. Discrete Representations

Discrete representations contain only a finite fixed number of key elements describing them, mainly points and lines. Representations that fall into this category include the following:

Meshes are collections of vertices connected through edges that form a graph. Common shapes for their faces are triangles (*triangulations*), quadrilaterals, and hexagons for surfaces, and tetrahedrons for volumes. A special case consists of the *simplex meshes*, which have a constant vertex connectivity. This type of shape representation permits smooth deformations in a simple and efficient manner (Delingette, 1999; Montagnat et al., 2001). Therefore, meshes are used for various tasks, such as 3D object recognition (e.g., Madi et al., 2019) and simulation of the dynamics of deformable objects (see section 3.3.1) with efficient coding (Arvanitis et al., 2019).

Skeletons are made of rigid edges connected by joints that allow bending. The position and deformation of elements attached to the skeleton are defined with respect to their assigned bone. Skeletons tend to be used together with the method known as *skinning*, where a deformable surface

[2]https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/

is attached to the bone and softens the visual appearance of the articulated joints through interpolation techniques. By nature skeletons are designed to model articulated deformations. Schaefer and Yuksel (2007) proposed a method to automatically extract skeletons by detecting articulated deformations.

Deformable templates are parameterized representations of a shape that specify key values of salient features in an image. They deform with greater ease around these features. The features can be peaks and valleys in the image intensity, edges, and the intensity itself, as well as points of high curvature[3] (sharp turns or bends; see Yuille et al., 1992; Basri et al., 1998). Deformable templates are mainly used for object recognition and object tracking (e.g., Ravishankar et al., 2008; Xia et al., 2019; Gallardo et al., 2020). They support the particular relevance of critical points in modeling deformations, which could make them key to developing a robot's ability to generate its own optimal representation of a deformable object. The use of deformable templates is explored and illustrated by experiments on natural and artificial agents in Arriola-Rios et al. (2013).

Landmark points are points that would remain stable across deformations. They can be corners, T-junctions, or points of high curvature. For example, when a rectangular sponge is pushed, its corners will still be corners after the deformation, while the point of contact with the external force will become a point of high curvature during the process and will remain as such; these are all stable points. In particular, landmark points can correspond to control points of splines (Blake et al., 1998). Methods for further processing, such as the application of deformations, can work more efficiently if they focus only (or mainly) on landmark points rather than on the whole representation (Cootes and Taylor, 2004).

Particles are idealized zero-dimensional dots. Their positions are specified as a vector function parameterized by time, $\boldsymbol{P}(t)$. They can store a set of attributes, such as mass, temperature, shape (for visualization purposes), age, lifetime, and so on. These attributes influence the dynamical behavior of the particles over time and are subject to change due to procedural stochastic processes. The particles can pass through three different phases during their lifetime: generation, dynamics, and death. However, manipulating them and maintaining constraints, such as boundaries between them can become non-trivial. For this reason, particles are used mainly to represent gases or visual effects in animations, where the interaction between them is very limited (Nealen et al., 2006).

Clouds of points are formed by large collections of coordinates that belong on a surface. They are frequently obtained from 3D scanned data and may include the color of each point. A typical problem consists in reconstructing 3D surfaces from such clouds (Newcombe and Davison, 2010; Makovetskii et al., 2020). Cretu et al. (2009) gives a comparative review of several methods for efficiently processing clouds of points and introduces the use of self-organizing neural gas networks for this purpose. Clouds of points can be structured and enriched with orientations of the 3D normals, as well as other feature descriptors for perceptual applications (Martínez et al., 2019).

[3]The curvature of a function is defined as $\kappa = \frac{d\phi}{ds}$ where ϕ is the tangential angle, with $\tan\phi = \frac{dy}{dx}$, and s is the arc length as defined earlier.

3. REPRESENTING DYNAMICS FOR DEFORMABLE OBJECTS

After an object's shape is defined as described in section 2, a suitable model of the dynamics can be used to register and predict deformations as a robot interacts with the object (**Figures 1ii,iii**). In this section, we introduce some of the most commonly used models from different fields (e.g., computer graphics; see Gibson and Mirtich, 1997; Nealen et al., 2006; Moore and Molloy, 2007; Bender et al., 2014) for predicting the dynamics of deformable objects. In robotics, the important features used to select an appropriate model are computational complexity (e.g., for real-time perception and manipulation), physical accuracy or visual plausibility, and simplicity or intuitiveness (i.e., the ability to implement simple cases easily and to be built on iteratively to accommodate more complex cases). Therefore, we divide the models into three classes: (1) particle-based models, which are usually computationally efficient and intuitive but physically not very accurate; (2) constitutive models, which are physically accurate but computationally complex and not very intuitive; and (3) approximations of constitutive models, which aim to decrease the computational complexity of constitutive models through approximations.

3.1. Background Knowledge of Deformation

First, we briefly review some background information about the physics and dynamics of deformation. Initially, the object is in a rest shape $\boldsymbol{S}_0$. In the discrete case it could be $\mathcal{S}_0 = \{\mathbf{p}_i^0 = \{x_i^0, y_i^0, z_i^0\} \in \mathbb{R}^{n=3}, i \in N\}$ where N is the number of points constituting the shape of the object. Then, when an external force $\mathbf{f}_{\text{ext}}$ acts on the object, such as gravitational force or force applied by a manipulator, the object deforms and its points move to a new position $\mathbf{p}^{\text{new}}$. In physics-based models, the resulting deformation is typically defined using a displacement vector field $\mathbf{u} = \mathbf{p}^{\text{new}} - \mathbf{p}^0$. From this displacement, the deformation can be computed through the stress σ (i.e., the force applied per area of the object shape) and the strain ϵ (i.e., the ratio of deformation to the original size of the object shape). The stress tensor σ is usually calculated for each point on the object shape using Hooke's law, $\sigma = \mathbf{E}\epsilon$, where ϵ can be calculated as $\epsilon = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$ with $\nabla\mathbf{u}$ denoting the spatial derivative of the displacement field,

$$\nabla\mathbf{u} = \begin{pmatrix} \partial u_x/\partial x & \partial u_x/\partial y & \partial u_x/\partial z \\ \partial u_y/\partial x & \partial u_y/\partial y & \partial u_y/\partial z \\ \partial u_z/\partial x & \partial u_z/\partial y & \partial u_y/\partial z \end{pmatrix}; \tag{12}$$

$\mathbf{E}$ is a tensor that is dependent on the real physical material properties of the object, such as Young's modulus E and Poisson's ratio υ. These properties are parameters in constitutive models

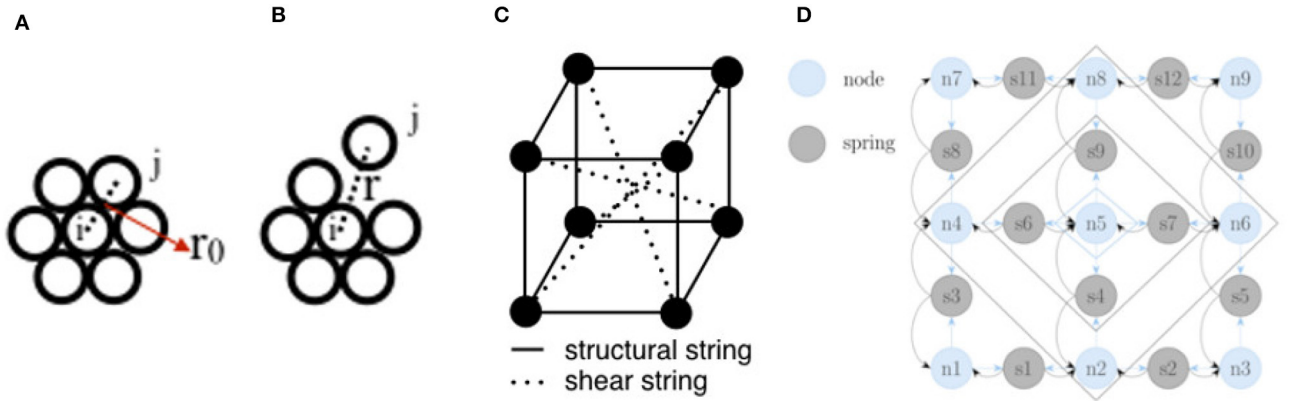

FIGURE 5 | (A) A simple circular deformable object represented in 2D with $N = 7$ particles in the particle system; the particles are in static equilibrium (initial rest state). **(B)** When the distance r between the particles i and j increases, new forces exerted on i are calculated and the particles move. **(C)** A mass-spring model of a simple cubic deformable object with $N = 8$ particles in 3D; the particles are connected with structural and shear springs that enable the object to resist longitudinal and shear deformations. **(D)** In Nurnberger et al. (1998), the neurons of a recurrent neural network represent the mass nodes (light blue) and springs (gray) of a mass-spring model; the activation functions between the neurons of the network were devised to reproduce the mass-spring system equations.

(e.g., finite element models). Constitutive models describe strain-stress relationships as the response of materials (e.g., elastic or plastic) to different loads (e.g., forces applied) based on the material properties; they are commonly used to simulate deformation because of their high physical accuracy.

To simulate the dynamical behavior of deformation over time, Newton's second law of motion is employed. Let $\mathbf{p}_i^t$ be the position of particle i at time t:

$$\mathbf{v}_i^t = \dot{\mathbf{p}}_i^t, \qquad \mathbf{a}_i^t = \dot{\mathbf{v}}_i^t, \qquad m_i \mathbf{a}_i^t = \mathbf{f}_{\text{ext}_i}^t, \tag{13}$$

where m_i, $\mathbf{f}_{\text{ext}_i}^t \in \mathbb{R}^3$, $\mathbf{a}_i^t \in \mathbb{R}^3$, and $\mathbf{v}^t \in \mathbb{R}^3$ are respectively the mass, external forces, acceleration, and velocity at time t, and $\dot{\mathbf{p}}_i^t = \frac{(\mathbf{p}_i^{t+\Delta t} - \mathbf{p}_i^t)}{\Delta t}$ and $\dot{\mathbf{v}}_i^t = \frac{(\mathbf{v}_i^{t+\Delta t} - \mathbf{v}_i^t)}{\Delta t}$ are first-order time derivatives of the position and velocity, respectively, which are approximated using finite differences. Then, according to these derivative approximations, in each time step Δt the points move according to a time integration scheme. The simplest such scheme is explicit Euler integration:

$$\mathbf{p}_i^{t+\Delta t} = \mathbf{p}_i^t + \mathbf{v}_i^t \Delta t, \tag{14}$$

$$\mathbf{v}_i^{t+\Delta t} = \mathbf{v}^t + \frac{1}{m} \mathbf{f}_{\text{ext}i} \Delta t, \tag{15}$$

where $\mathbf{v}_i^t$ and $\mathbf{p}_i^t$ are the velocity and position of point i at time t. We remark that explicit Euler integration can cause problems, such as unrealistic deformation behavior (e.g., overshooting). There are other more stable integration schemes (e.g., implicit integration, Verlet, Runge-Kutta; see Hauth et al., 2003) that can be used.

Such a dynamical model can be represented simply as $G(\boldsymbol{S}_0, \mathbf{f}_{\text{ext}}, \theta)$, where input to the model G includes the initial state of points $\mathbf{p}^0 \in \boldsymbol{S}_0$ of the object and the external forces $\mathbf{f}_{\text{ext}}$; θ represents model parameters that could be related to material properties (e.g., E, υ), as in constitutive models, to simulate desired deformations. Then, within G, the deformation is computed and the points $\mathbf{p}^t$ are iterated to time state t using an integration scheme, such as (14) and (15).

3.2. Particle-Based Models

3.2.1. Particle Systems

In a particle system, a solid object shape $\boldsymbol{S}$ is represented as a collection of N particles (see section 2.4). These particles are initially in an equilibrium position, $\mathbf{p}_i^0 = \{x_i^0, y_i^0, z_i^0\} \in \mathbb{R}^3$, which can be regarded as the initial coordinates of each particle $i \in \{1, \ldots, N\}$ (**Figure 5A**). When an external force is applied, the object deforms and the particles move to new coordinates $\mathbf{p}_i^t$ based on physics laws, in particular Newton's second law of motion (13), according to a time integration scheme, such as (14) and (15) (**Figure 5B**).

Although particles are usually used to model objects, such as clouds or liquids, there are also particle frameworks for simulation of solids. These frameworks are based on so-called dynamically coupled particles that represent the volume of an object (Tonnesen and Terzopoulos, 2000). The advantage of particle systems is their simplicity, which allows simulation of a huge number of particles to represent complex scenes. A disadvantage of particle systems is that the surface is not explicitly defined. Therefore, maintaining the initial shape of the deforming object is difficult, and this can be problematic for applications, such as tracking the return of elastic objects to their original shape after deformation during robotic manipulation. Hence, for objects that are supposed to maintain a given structure, particle-based models with fixed particle couplings are more appropriate, such as models that employ meshes for shape representation.

3.2.2. Mass-Spring Systems

Mass-spring (MS) models use meshes for shape representation (see section 2.4). In such a model N particles are connected by a network of springs (**Figure 5C**). As in particle systems, particle motion is simulated using Newton's second law of motion (13).

However, there are other forces between the connected particles, say i and j, that affect their motion, in particular the spring force $\mathbf{f}_s(\mathbf{p}_i) = k_s(|(\mathbf{p}_j - \mathbf{p}_i)| - l_{ij})\frac{(\mathbf{p}_j - \mathbf{p}_i)}{|(\mathbf{p}_j - \mathbf{p}_i)|}$, where k_s is the spring's stiffness and l_{ij} is the rest length of the spring, and the damping force $\mathbf{f}_d(\mathbf{p}) = k_d(\mathbf{v}_j - \mathbf{v}_i)$ of the spring, where k_d is the damping coefficient. Then, the equation of motion (13) becomes

$$m_i\mathbf{a}_i = \mathbf{f}_{ext}(\mathbf{p}_i) + \mathbf{f}_d(\mathbf{p}_i) + \mathbf{f}_s(\mathbf{p}_i). \tag{16}$$

For the entire particle system, this can be expressed in matrix form as

$$\mathbf{Ma} + \mathbf{Dv} + \mathbf{Ku} = \mathbf{f}_{ext}, \tag{17}$$

where $\mathbf{M} \in \mathbb{R}^{3N\times 3N}$, $\mathbf{D} \in \mathbb{R}^{3N\times 3N}$, and $\mathbf{K} \in \mathbb{R}^{3N\times 3N}$ are a diagonal mass matrix, a diagonal damping matrix, and a stiffness matrix for $n = 3$ dimensions. The MS system can be represented as a model $G_{MS}(\boldsymbol{S}_0, \mathbf{f}_{ext}, \theta)$, where input to G_{MS} consists of the initial state of the mesh shape $\mathbf{p}^0 \in \boldsymbol{S}_0$, the external forces $\mathbf{f}_{ext}$, and the model parameters $\theta = \{k_s, k_d\}$, which can be changed (tuned) to determine object deformability.

MS systems are a widely used type of physics-based model for predicting and tracking object states during robotic manipulation, since they are intuitive and computationally efficient (Schulman et al., 2013). However, the spring constants are difficult to tune according to the material properties to obtain the desired deformation behavior. One way to overcome this tuning problem is to use learning algorithms and reference solutions (Bianchi et al., 2004; Morris and Salisbury, 2008; Arriola-Rios and Wyatt, 2017). Another disadvantage of MS models is that they cannot directly simulate volumetric effects, such as volume conservation in its basic formulation. To simulate such effects, Teschner et al. (2004) introduced additional energy formulations. Also, the behavior of an MS model is affected by the directions in which the springs are placed; to deal with this issue, Bourguignon and Cani (2000) added virtual springs to compensate for this effect. In addition, Xu et al. (2018) proposed a new method by introducing extra elastic forces into the traditional MS model to integrate more complex mechanical behaviors, such as viscoelasticity, non-linearity, and incompressibility.

3.2.3. Neural Networks

Nurnberger et al. (1998) designed a method for controlling the dynamics of an MS model using a recurrent neural network (NN). Different types of neurons are used to represent the positions **p**, velocities **v**, and accelerations **a** of the mass points (nodes) and the springs (spring nodes) of the mesh shape $\boldsymbol{S}$ (**Figure 5D**). The differential equations governing the behavior of the MS system are codified in the structure of the network. The spring functions are used as activation functions for the corresponding neurons. The whole system poses the simulation as a problem of minimization of energy. The information is propagated to the neurons in stages, starting from the mass points where the applied force is greatest, and an equilibrium point must be reached to obtain the new configuration of the nodes at each time t. The training is carried out with gradient descent (*backpropagation*) for the NN. In addition, Zhang et al. (2019) employed a convolutional neural network (CNN) to model propagation of mechanical load using the Poisson equation rather than an MS model.

The advantage of using an NN to control deformation is the method's flexibility, such as being able to modify the network structure during simulation (e.g., by removing springs as in Nurnberger et al., 1998) and simulate large deformations efficiently (e.g., Zhang et al., 2019).

3.2.4. Position-Based Dynamics

Particle systems and MS models are force-based models where, based on given forces, the velocities and positions of particles are determined by a time integration scheme. In contrast, position-based dynamics (PBD) models compute the positions directly by applying geometrical constraints in each simulation step. PBD methods can be used for various purposes, such as simulating liquids, gases, and melting or visco-elastic objects undergoing topological changes (Bender et al., 2014). Here we focus on a special PBD method, called meshless shape matching (MSM; see Müller et al., 2005), that is used to simulate volumetric solid objects while preserving their topological shape.

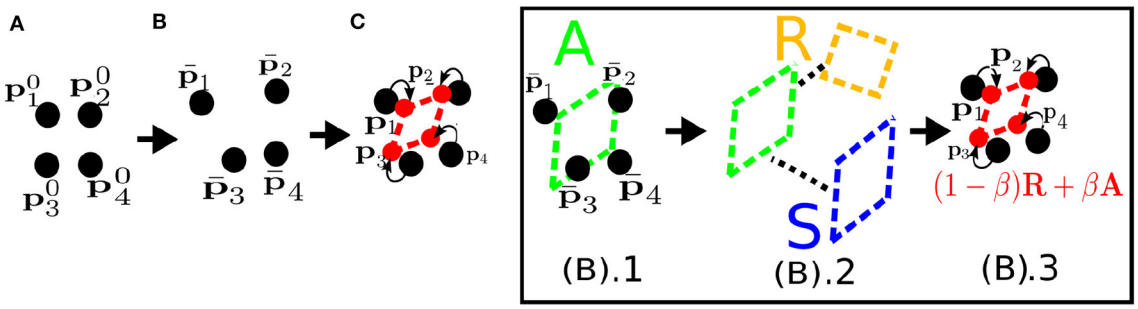

FIGURE 6 | (A) Meshless shape matching applied to a simple object consisting of $N = 4$ particles. **(B).1** The method estimates the optimal linear transformation **A** that allows the particles to move to the actual deformed positions **p** with respect to the rest state $\mathbf{p}^0$, as in **(C)**. **(B).2** Then **A** is decomposed into a rotational (rigid) part **R** and a symmetric (deformation) part **S**. **(B).3** The **R** transformation is used to simulate rigid motion; to simulate deformation, **A** and **R** are combined using a parameter, β.

In MSM, as in particle systems, an object is represented by a set of N particles without any connectivity (**Figure 6A**). Since there is no connectivity information between the particles, when they are disturbed by external forces (**Figure 6B**) they tend to adopt a configuration that does not respect the original shape $\mathbf{p}^0$ of the object. We call this disturbed configuration the intermediate deformed shape, $\bar{\mathbf{p}}_i = \mathbf{p}_i^{t-1} + \bar{\mathbf{v}}_i^t \Delta t$, where $\bar{\mathbf{v}}_i = \mathbf{v}_i^{t-1} + \mathbf{f}_{\text{ext}} \frac{\Delta t}{m_i}$. MSM calculates an optimal linear transformation, $\mathbf{A} = (\sum_i m_i \mathbf{r}_i \mathbf{q}_i)(\sum_i m_i \mathbf{q}_i \mathbf{q}_i)^{-1}$, between the initial shape $\mathbf{p}^0$ and the intermediate deformed shape $\bar{\mathbf{p}}$ that allows preservation of the original shape of the object; here $\mathbf{r}_i = \bar{\mathbf{p}}_i - \bar{\mathbf{c}}$ and $\mathbf{q}_i = \mathbf{p}_i^0 - \mathbf{c}_0$, with $\mathbf{c} = \frac{1}{\sum_i^N m_i} \sum_i^N m_i \mathbf{p}_i$ being the center of mass of the object (**Figure 6B**.1–3). Then, the linear transformation $\mathbf{A}$ is separated into rotational and symmetric parts: $\mathbf{A} = \mathbf{RS}$ where $\mathbf{R}$ represents rigid behavior and $\mathbf{S}$ represents deformable behavior. Hence, to simulate rigid behavior, the goal (actual) position of the particles is

$$\mathbf{p}_i = \mathbf{R}\mathbf{q}_i + \mathbf{t} \tag{18}$$

where $\mathbf{t} = \mathbf{c}$ is the translation of the object. If the object is deformable, then $\mathbf{S}$ is also included and the goal position is

$$\begin{aligned} \mathbf{p}_i &= (\mathbf{R}((1-\beta)\mathbf{I} - \beta\mathbf{S}))\mathbf{q}_i + \mathbf{t} \\ &= ((1-\beta)\mathbf{R} + \beta\mathbf{A})\mathbf{q}_i + \mathbf{t}, \end{aligned} \tag{19}$$

where β is a control parameter that determines the degree of deformation coming from the $\mathbf{S}$ matrix. If $\beta = 0$, (19) becomes (18). If β approaches 1, then the range of deformation increases. Subsequently, using the following integration scheme, the new position and velocity at time t are updated:

$$\mathbf{p}_i^t = \bar{\mathbf{p}}_i + \alpha(\mathbf{p}_i - \bar{\mathbf{p}}_i), \tag{20}$$

$$\mathbf{v}_i^t = (\mathbf{p}_i^t - \mathbf{p}_i^{t-1})/\Delta t, \tag{21}$$

where α affects the stiffness of the model (similar to the MS model) and determines the speed of convergence of the intermediate positions to the goal positions. In simplest form this model can be represented as $G_{\text{MSM}}(\mathbf{S}^0, \mathbf{f}_{\text{ext}}, \theta)$, where input to the model G_{MSM} consists of the initial state $\mathbf{S}^0$, external forces $\mathbf{f}_{\text{ext}}$, and model parameters $\theta = \{\beta, \alpha\}$, which can be tuned to decide the range of deformability and stiffness of the model.

The main advantages of PBD methods are their simplicity, computational and memory-wise (i.e., not needing a mesh model) efficiency, and scalability owing to their particle-based parallel nature. Also, they are able to calculate more visually plausible deformations than MS models. Hence, they have been used in a wide range of interactive graphical applications (Tian et al., 2013; Macklin et al., 2014), particularly for modeling the deformation of human body parts (Zhu et al., 2008; Sidorov and Marshall, 2014; Romeo et al., 2020), and robotic manipulation tasks (Caccamo et al., 2016; Guler et al., 2017). A disadvantage of PBD methods is that they simulate physical deformation less accurately than constitutive models, since they are geometrically motivated.

3.3. Constitutive Models

To simulate more physically accurate deformations, constitutive models, which incorporate real physical material properties, are used. In this subsection, we start by introducing the most commonly used constitutive models, namely finite element models, and then briefly mention other models that simplify finite element models to increase computational efficiency.

3.3.1. Finite Element Method

The finite element method (FEM) aims to approximate the true physical behavior of a deformable object by dividing its body into smaller and simpler parts called finite elements. These elements are connected through N nodes that make up an irregular grid mesh (**Figure 7A**). Thus, instead of particles, we work with node displacements. The mesh deformation is calculated through the displacement vector field $\mathbf{u}$. For simulation, an equation of motion similar to (17) is used for an entire mesh. Usually, to decrease the computational complexity, the dynamical parts of the equation are skipped and the deformation is calculated for a static state in equilibrium ($\mathbf{a} = \mathbf{v} = \mathbf{0}$). Then, the relationship

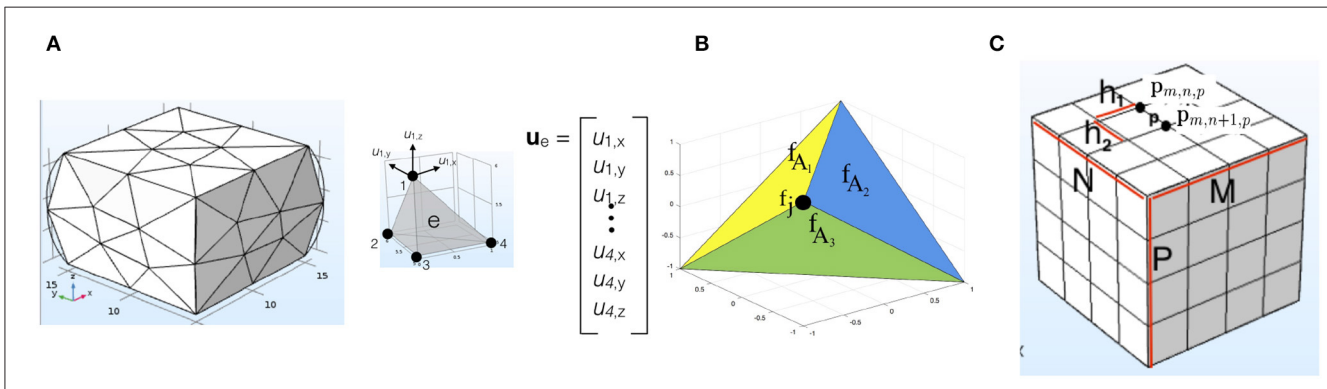

FIGURE 7 | (A) An irregular grid as the mesh of a cubic deformed object in 3D using the finite element method (left) and an element e of the mesh with its $N_e = 4$ nodes (right); the arrows show the displacement fields $\mathbf{u}_1 = \{u_{1,x}, u_{1,y}, u_{1,z}\}$ at node $i = 1$, and $\mathbf{u}_e$ is the nodal displacement vector of the element e. **(B)** An element with node j and forces applied on three adjacent faces in the finite volume model. **(C)** A regular discrete mesh to be used in calculations of the finite difference method.

between a finite element e and its N_e nodes (e.g., $N_e = 4$ for a tetrahedron as in **Figure 7A**) can be expressed as

$$\mathbf{K}_e \mathbf{u}_e = \mathbf{f}_e \tag{22}$$

where $\mathbf{f}_e \in \mathbb{R}^{3\times N_e}$ contains the N_e nodal forces, $\mathbf{u}_e \in \mathbb{R}^{3\times N_e}$ is the displacement of an element between the actual and the deformed positions, and $\mathbf{K}_e \in \mathbb{R}^{3N_e\times 3N_e}$ is the stiffness matrix of the element. The stiffness matrices of the different elements are assembled into a single matrix $\mathbf{K} \in \mathbb{R}^{3N\times 3N}$ for an entire mesh with N nodes:

$$\mathbf{K} = \sum_e \mathbf{K}_e, \tag{23}$$

$$\mathbf{K}\mathbf{u} = \mathbf{f}. \tag{24}$$

The matrix $\mathbf{K}$ relies on the nodal displacement $\mathbf{u} \in \mathbb{R}^{3\times N}$ and the constitutive material properties (e.g., E and υ) to compute the nodal forces $\mathbf{f} \in \mathbb{R}^{3\times N}$ of the entire mesh. Therefore, this huge matrix $\mathbf{K}$ should be calculated at every time step t.

The FEM can be represented as a model $G_{\text{FEM}}(\boldsymbol{S}^0, \mathbf{f}_{\text{ext}}, \theta)$, which takes as input the initial state $\boldsymbol{S}^0$, external forces $\mathbf{f}_{\text{ext}}$, and constitutive model parameters θ, such as E and υ. Then, within model G_{FEM}, the new positions and velocities of the nodes are updated using a time integration scheme, such as (14) and (15).

The FEM can produce physically realistic simulations and model complex deformed configurations. Owing to these properties, FEM models have been used in many robotics applications, such as tracking (Essa et al., 1992; Petit et al., 2018; Sengupta et al., 2019) and planning manipulation around deformable objects (Frank et al., 2014). However, they can have a heavy computational burden due to re-evaluating $\mathbf{K}$ at each time step. This can be avoided by using linear FEM models where $\mathbf{K}$ in (24) stays constant[4]. The drawback is that this assumption limits the model to simulating only small deformations. Other methods that can decrease computational complexity, such as co-rotational FEM (Müller and Gross, 2004), can also be used.

3.3.2. Finite Volume Method

In the finite volume method (FVM), instead of calculating the nodal forces of the mesh shape $\boldsymbol{S}$ individually as in FEM, the force per unit area with respect to a certain plane orientation is calculated. This is done by using the constitutive law for the computation of the stress tensor σ (section 3.1). Then, the total force acting on face i of a finite element can be calculated using the formula

$$\mathbf{f}_{A_i} = A_i \sigma \mathbf{n}_i \tag{25}$$

where A_i is a scalar representing the area of face i and $\mathbf{n}_i$ is its normal vector. To calculate the nodal forces, the forces of surfaces adjacent to node j are summed and distributed evenly to each node (Teran et al., 2003; see **Figure 7B**):

$$\mathbf{f}_j = -\frac{1}{n}(\mathbf{f}_{A_1} + \mathbf{f}_{A_2} + \mathbf{f}_{A_3}). \tag{26}$$

[4]For a detailed tutorial on how to efficiently compute the $\mathbf{K}$ matrix, we direct the reader to the notes of Müller et al. (2008).

The FVM model can be represented as $G_{\text{FVM}}(\boldsymbol{S}^0, \mathbf{f}_{\text{ext}}, \theta)$, which takes as input the initial state $\boldsymbol{S}^0$, in which areas A and normals $\mathbf{n}$ can be calculated, the external forces $\mathbf{f}_{\text{ext}}$, and the constitutive model parameters E and υ. Then, within model G_{FVM}, the new positions $\mathbf{p}$ of nodes of $\boldsymbol{S}^0$ are updated at each time step t using a time integration scheme. Since this method is computationally more efficient than the FEM, it has been used in many computer graphics applications (Barth et al., 2018; Cardiff and Demirdžić, 2018). However, it restricts the types of deformation that can be simulated, such as the deformation of irregular meshes.

3.3.3. Finite Difference Method

In the finite difference method (FDM), the volume of the object is defined as a regular $M \times N \times P$ discrete mesh of nodes with horizontal, vertical, and stacked inter-node spacings h_1, h_2, and h_3, respectively (**Figure 7C**). The nodes are indexed as $[m, n, p]$ where $1 \le m \le M$ (parallel to the x-axis), $1 \le n \le N$ (parallel to the y-axis), and $1 \le p \le P$ (parallel to the z-axis), and $\mathbf{p}_{m,n,p} \in \mathbb{R}^3$ is the position of the node in 3D space. The object is deformed when an external force is applied. To calculate the nodal forces, a displacement vector $\mathbf{u}$ should be calculated using spatial derivatives. This is done by defining finite difference operators between the new node positions in the deformed mesh. For example, for $\mathbf{p}_{m,n,p}$ the first-order finite difference operator along the x-axis can be defined as $d_x(\mathbf{p}_{m,n,p}) = (\mathbf{p}_{m+1,n,p} - \mathbf{p}_{m,n,p})/h_1$. Using the finite difference operators, the nodal forces are calculated and the deformation of the object can be computed as in the FEM (section 3.3.1).

The FDM is one of the alternative methods suggested for decreasing the computational complexity of the FEM (Terzopoulos et al., 1987). A disadvantage of this method is that it is more difficult to approximate the boundaries of objects using a regular grid for the mesh (Nealen et al., 2006), and hence the accuracy is decreased.

3.3.4. Boundary Element Method

The boundary element method (BEM) computes the deformation of $\boldsymbol{S}$ by calculating the equation of motion (17) over a surface rather than over a volume as in the FEM. The boundary (surface) $\boldsymbol{S}$ is discretized into a set of N non-overlapping elements (e.g., mesh elements) e, whose node coordinates $\mathbf{p}_i$, $i = 1, \ldots, N$, are the centroids of the elements. These elements represent displacements and tractions, and $\boldsymbol{S}_u$ and $\boldsymbol{S}_r$ are surface parts where the displacement and traction boundary conditions are defined, respectively.

The BEM provides a significant speedup compared to the FEM because it requires fewer nodes and elements. However, it only works for objects whose interior consists of homogeneous material. It has been used in the ArtDefo System (James and Pai, 1999) to simulate volumetric models in real-time. Also, it has been used to improve tracking accuracy against occlusions and spurious edges in (Greminger and Nelson, 2008).

3.3.5. Long Elements Method

In the long elements method (LEM), a solid object is considered to be filled with incompressible fluid as in biological tissues. The volume of object shape $\boldsymbol{S}$ is discretized into Cartesian meshes (i.e.,

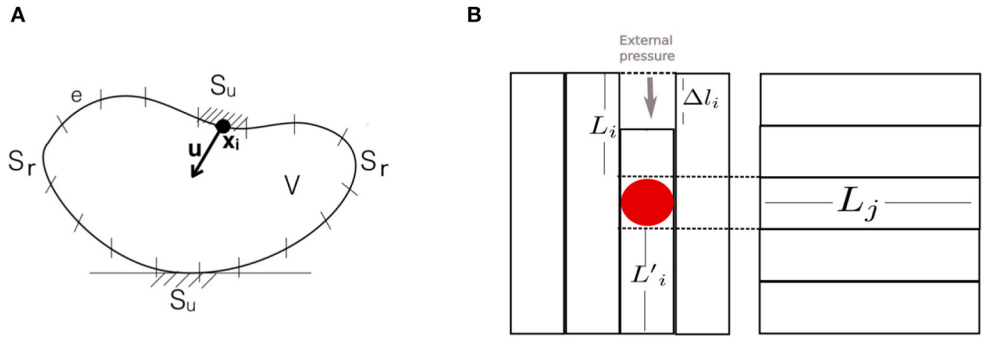

FIGURE 8 | (A) In the boundary element method, the deformable object's shape $\boldsymbol{S}$ is discretized into a set of elements e; $\boldsymbol{S}$ is disturbed by a force that displaces the centroid node coordinates $\mathbf{p}$, and the deformation is calculated based on the displacement and traction boundary conditions defined on $\boldsymbol{S}_u$ and $\boldsymbol{S}_r$, respectively. **(B)** A virtual object discretized as Cartesian meshes according to the long elements method. Here, for the sake of simplicity, we show a 2D object with only two axes (i and j). On the left and right figures are meshes of the object discretized with long elements L_i and L_j parallel to the i and j axes, respectively. External pressure is applied at a particular point on the surface of the object, and the resulting force f on the particle (red dot) is calculated using the deformations Δl_i of long elements crossing the particle.

one mesh for each axis), and each mesh contains long elements (LEs) $e \in \{1, \ldots, N_i\}$, where N_i is the number of elements in the mesh shape $\boldsymbol{S}_i$ discretized parallel to axis i (**Figure 8B**). The crossings of the LEs of the different axes define cells, each of which contains a particle. By calculating the state of these particles (e.g., position $\mathbf{p}$ and velocity $\mathbf{v}$), the deformation of the object is simulated.

The state of each particle is calculated using the laws of fluid mechanics (e.g., Pascal's law). Then, a system of linear equations for each e that fills the object volume is created. By solving this system of equations by numerical methods, the deformation of e, Δl_i, is calculated. From Δl_i the forces occurring due to deformation are computed. Here, the LE is regarded as a spring attached to a particle with known mass. As an example, in **Figure 8B**, pressure is applied to an object. As a result of this pressure, a force f_i acts on the particle along the ith axis and is calculated using the displacements of the crossing LEs attached to the particle:

$$f_i = k_{L_i}(\Delta l_i - \Delta l_{i'}) + k_{L_j}(\Delta l_j - \Delta l_{j'}) \tag{27}$$

where k is the spring constant. Therefore, an LEM model can be represented as $G_{\text{LEM}}(\boldsymbol{S}_i, \boldsymbol{S}_j, \mathbf{f}_{\text{ext}}, \theta)$, which takes as input meshes of axes i and j for 2D space, external forces $\mathbf{f}_{\text{ext}}$, and model parameters, such as spring constants that determine object deformability. Subsequently, the force obtained is used to calculate the velocities and positions of the particles along each axis.

The LEM was developed for modeling soft tissues, especially for surgical simulation (Balaniuk and Salisbury, 2002). It uses a smaller number of elements than tetrahedral (e.g., FEM) and cubic (e.g., FD) meshing, so the computational complexity of the model is reduced as well. It is therefore capable of interactive real-time soft tissue simulation for haptic and graphic applications, such as robotic surgery. However, it provides only an approximation of real physical deformation and so presents a trade-off between physical accuracy and computational efficiency.

3.4. Approximations of Constitutive Models

3.4.1. Modal Analysis

What makes constitutive models, such as the FEM expensive is calculation of the motion with large matrices $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$ in Equation (17); for example, with $N = 20$ nodal points of a mesh shape $\mathbf{p} = (x, y, z)$, the calculation would involve three matrices of size 60×60. Pentland and Williams (1989) proposed a way of reducing this computational complexity based on a method called modal analysis. Modal analysis is used for identifying an object's vibrational modes (**Figure 9A**) by decoupling (17). This is done by using linear algebraic formulations (Nealen et al., 2006). Below, we outline the steps of modal analysis using these formulations, while skipping the detailed derivations.

First, the matrices are diagonalized by solving the following eigenvalue problem (i.e., *whitening transition*):

$$\mathbf{M}\Phi\Lambda = \mathbf{K}\Phi, \tag{28}$$

where Λ and Φ are matrices containing the eigenvalues and eigenvectors of $\mathbf{M}\mathbf{K}^{-1}$. Then, the eigenvectors of Φ are used to transform the displacement vector $\mathbf{u}$:

$$\mathbf{u} = \Phi\mathbf{q}. \tag{29}$$

By substituting (29) into (17) and multiplying by Φ^T, the following system of equations is constructed:

$$\Phi^T\mathbf{M}\Phi\ddot{\mathbf{q}} + \Phi^T\mathbf{D}\Phi\dot{\mathbf{q}} + \Phi^T\mathbf{K}\Phi\mathbf{q} = \Phi^T\mathbf{f}_{\text{ext}}, \tag{30}$$

$$\nabla\mathbf{M}\ddot{\mathbf{q}} + \nabla\mathbf{D}\dot{\mathbf{q}} + \nabla\mathbf{K}\mathbf{q} = \nabla\mathbf{f}_{\text{ext}}, \tag{31}$$

where $\nabla\mathbf{M}$, $\nabla\mathbf{D}$, and $\nabla\mathbf{K}$ are all diagonal matrices. This generates $3N$ independent equations of motion for the modes:

$$\nabla\mathbf{M}_i\ddot{\mathbf{q}}_i + \nabla\mathbf{D}_i\dot{\mathbf{q}}_i + \nabla\mathbf{K}_i\mathbf{q}_i = \nabla\mathbf{f}_{\text{ext}i}. \tag{32}$$

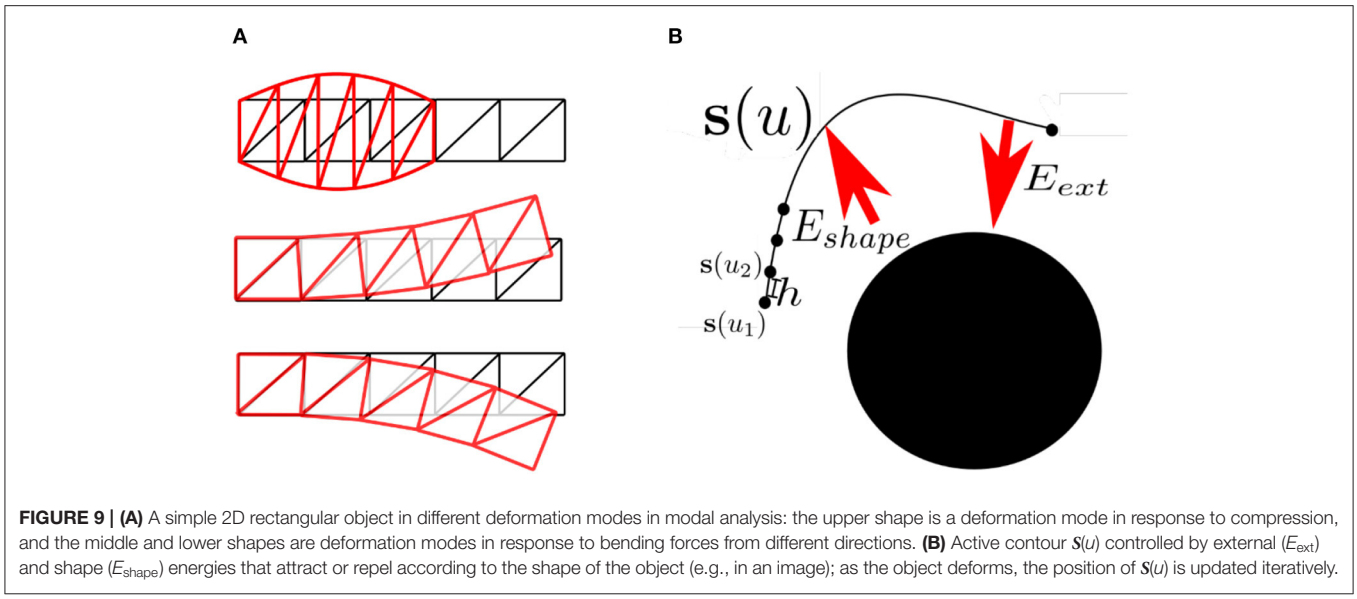

FIGURE 9 | (A) A simple 2D rectangular object in different deformation modes in modal analysis: the upper shape is a deformation mode in response to compression, and the middle and lower shapes are deformation modes in response to bending forces from different directions. **(B)** Active contour $\boldsymbol{S}(u)$ controlled by external (E_{ext}) and shape (E_{shape}) energies that attract or repel according to the shape of the object (e.g., in an image); as the object deforms, the position of $\boldsymbol{S}(u)$ is updated iteratively.

Then, (32) can be solved analytically for $\mathbf{q}_i$ to compute the motion of each mode $i \in \{1, 2, \ldots, N\}$. The matrix Φ contains a different mode shape in each of its columns (**Figure 9A**), i.e., $\Phi = [\Phi_1, \Phi_2, \ldots, \Phi_N]$. Hence, by analyzing the eigenvalues in Λ, high-frequency modes in Φ can be eliminated so that only the most dominant modes are updated using (32). This reduces the number of equations in (32) and hence lowers the computational cost significantly.

In modal analysis, as in constitutive models, taking $\mathbf{K}$ to be constant can increase the computational efficiency. However, this assumption is valid only when simulating small linear deformations and leads to errors when dynamically simulating large deformations. To overcome this problem, some methods use different formulations of the strain tensor (e.g., the Green strain as in Barbič and James, 2005) to enable simulation of larger non-linear deformations (An et al., 2008; Pan and Manocha, 2018), or adopt more data-driven approaches (e.g., by employing CNNs as in Fulton et al., 2019).

3.4.2. Active Contours

Active contour (AC) models are approximations of constitutive models, such as the FEM. They were first introduced by Kass et al. (1988) in the form of the snakes model. In their simplest form they can be described as a function of a spline shape (section 2.2.1), $\boldsymbol{S}(u) \in \mathbb{R}^n$ for $u \in [0, 1]$, in an n-dimensional space, for example $\boldsymbol{S}(u) = \{\mathbf{x}(u), \mathbf{y}(u)\} \in \mathbb{R}^2$ in an image $I : \mathbb{R}^2 \to \mathbb{R}$ (**Figure 9B**). This spline is fitted to the shape of an object in the image by minimizing the following energy formulation:

$$E_{\text{snake}} = \int_0^1 E_{\text{ext}}(\boldsymbol{S}(u)) + E_{\text{shape}}(\boldsymbol{S}(u))\, du. \tag{33}$$

Here E_{ext} depends on the contour position with respect to an attractor function f:

$$E_{\text{ext}} = f(\mathbf{S}(u)), \tag{34}$$

where in the $n = 2$ case the $f(x, y)$ function could be the image intensity $I(x, y)$, which would attract the snake to the brightest regions, or an edge detector, which would attract the snake to the edges (Moore and Molloy, 2007).

In (33), E_{shape} is the internal energy of the contour, which depends on the shape of the contour:

$$E_{\text{shape}} = \alpha(n)|\boldsymbol{S}'(u)^2| + \beta(i)|\boldsymbol{S}''(u)|^2. \tag{35}$$

The first-order derivative $|\boldsymbol{S}'(u)^2|$ controls the length of the contour, and the goal is to minimize the total length. The second-order derivative $|\boldsymbol{S}''(u)|^2$ controls the smoothness of the contour; this term enables the contour to resist stretching or bending by external forces due to $f(\boldsymbol{S}'(u))$ and is used to regularize the contour. The weight parameters α and β determine elasticity and rigidity, respectively.

The energy E_{snake} can be discretized into N parts as $\mathbf{s}_i(u)$ where the $u_i = ih$, for $i \in \{1, \ldots, N\}$, are knots and $h = \frac{1}{N}$:

$$E^*_{\text{snake}} = \sum_{i=1}^{N} E_{\text{ext}}(\mathbf{s}_i(u)) + E_{\text{shape}}(\mathbf{s}_i(u)). \tag{36}$$

Then, from the discretization, the derivatives in E_{shape} can be approximated using finite difference operators:

$$E_{\text{shape}}(i) = \alpha_i \frac{|\mathbf{s}(u_i) - \mathbf{s}_{i-1}(u)|^2}{2h^2} + \beta_i \frac{|\mathbf{s}_{i-1}(u) - 2\mathbf{s}_i(u) + \mathbf{s}_{i+1}(u)|^2}{2h^4}. \tag{37}$$

To find the contour that minimizes the total energy, E^*_{snake} is minimized. The resulting expression is then put into matrix form and used to update the position of the contour iteratively in time by using a time integration scheme as demonstrated in Kass et al. (1988). To represent an AC in 3D, some additional parameters are included in the shape energy formulation: the elasticity parameter

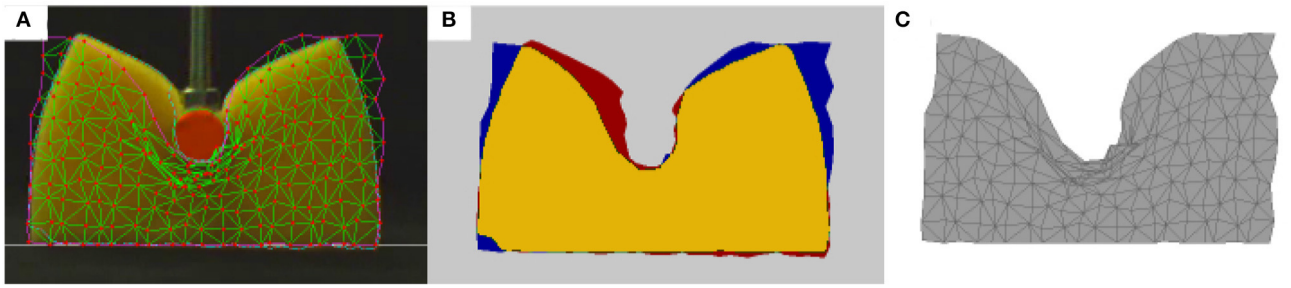

FIGURE 10 | The general schema for learning: **(A)** A type of model and ground truth are selected, such as 2D images used as ground truth to calibrate a mass-spring model. **(B)** If the model's parameters have not been calibrated correctly, there will be a difference between the observed ground truth and the simulation; the observed ground truth is shown in red, the simulation in blue, and their intersection in yellow. **(C)** When the model is properly calibrated, it should be able to predict the behavior of the object if given the corresponding interaction parameters, such as external forces (e.g., contact forces of the manipulator) or geometric constraints (e.g., not crossing the floor).

β is defined along the third axis as well, and an extra parameter is added to control the resistance to twisting (Ahlberg, 1996).

AC models have been widely used, especially in medical imaging, for motion tracking and shape registration tasks (Williams and Shah, 1992; Leventon et al., 2000; Das and Banerjee, 2004), and they can also be combined with constitutive models to achieve greater physical accuracy (Luo and Nelson, 2001). The main disadvantage of AC models is their reliance on a good initialization of the snake contour near the desired shape in the image. To overcome this drawback, attractor functions other than image intensity, such as edge maps, have been proposed in recent years (e.g., Nisirat, 2019).

This concludes our tutorial-style description of models to provide some technical grounding in the basic mathematical approaches to deformable object modeling. The content of sections 4 (learning and estimation) and 5 (planning and control) builds on the models we have described. These sections are written in the form of broad surveys, as there is such a wide range of different approaches that we cannot cover them all in depth.

4. LEARNING AND ESTIMATION OF MODEL PARAMETERS

In the previous sections, we introduced computational models of deformable objects that have numerous applications. However, for the models to be useful, several parameters must be known beforehand (**Figure 1vi**), so these models should be calibrated carefully. In this section we give an overview of some representative cases of applying various learning algorithms, which can make the calibration process autonomous (**Figure 10**). The methods we review can be grouped into three types of strategies: (a) estimating parameters directly, which is rarely feasible (section 4.1); (b) calibrating known physics-based models, G, automatically by allowing the robot to take some measurements that will help to determine the values of model parameters, θ (section 4.2.2); and (c) approximating new functions that describe the dynamics, as is done with neural networks (section 4.2.4).

4.1. Direct Estimation

For some models, it is possible to derive a formula to directly calculate the parameters. For example, Gelder (1998) obtained a formula for the parameter k_s of an MS model (section 3.2.2) in a static state, where the materials are non-uniform but isotropic. An isotropic material is a material whose local deformation in response to force is independent of the direction in which the force is applied. However, in a non-uniform material the response varies with the position where the force is applied. For 3D tetrahedral meshes, the spring constant k_s can be obtained from the formula

$$k_s = \frac{E \sum_e V_e}{|c|^2} \tag{38}$$

where the sum is over the volume V_e of a triangular element e of a 3D mesh shape $\boldsymbol{S}^0$ on its edge c. Young's modulus, E, is chosen empirically to give the desired amount of elasticity.

Direct estimation is a computationally efficient method. However, often it is not possible to do such calculations for models that rely on complex constitutive material laws as in the FEM.

4.2. Minimizing Error

This group of methods relies on the definition of an error function $\text{Err}(\mathbf{p}^\theta, \hat{\mathbf{p}}) = \text{dist}(\mathbf{p}^\theta, \hat{\mathbf{p}})$ that measures the difference (e.g., Euclidean distance) between the deformation of some ground truth $\hat{\mathbf{p}}$ and the simulated virtual deformable object position $\mathbf{p}^\theta \in G(\boldsymbol{S}^0, \mathbf{f}_{\text{ext}}, \mathbf{p}_c, \theta)$, where $\mathbf{p}_c$ is the point of contact. The ground truth $\hat{\mathbf{p}}$ can be obtained from camera observations of a real-world deformable object or from another, more reliable, simulation, usually an FEM simulation. Then, $\mathbf{p}^\theta \in \boldsymbol{S}^\theta$ is simulated with various θ values and the same interaction parameters as in the ground truth observations, such as the contact forces $\mathbf{f}_{\text{ext}}$ and positions $\mathbf{p}_c \in \boldsymbol{S}^\theta$ of the manipulator or geometric constraints (boundary conditions) like the object not crossing the bottom surface. The objective of the learning algorithm is to find a set of parameters θ for the model G that minimizes the error function $\text{Err}(\mathbf{p}^\theta, \hat{\mathbf{p}})$ with the given interaction parameters.

Error minimization methods are usually successful at calibrating models that are difficult to tune, such as non-constitutive models (e.g., MS models) for which there is no direct link between the model parameters (e.g., k_s) and the material properties (e.g., E and υ). However, they require the simulations to be run multiple times with different parameter values, which can be computationally expensive. Therefore, most of the time, offline estimation is performed (e.g., Leizea et al., 2017). Thus, these methods are not adaptable for online or for more dynamic deformable object manipulations. Also, in cases where the error function is not exactly convex, the algorithm may get stuck at a local minimum and result in deformations that are physically not very accurate or visually implausible.

4.2.1. Exhaustive Searches

Guler et al. (2015) used exhaustive search to obtain the β value for the MSM model (section 3.2.4) $G_{\text{MSM}}(\boldsymbol{S}^0, \mathbf{f}_{\text{ext}}, \mathbf{p}_c, \beta, \alpha)$. This parameter determines the degree of deformability of a material. Camera images of real-world objects are used as the ground truth, with $\hat{\mathbf{p}}$ being the points detected on the surface of the object shape in the images. For β, uniform samples are taken from the interval $[0, 1]$, and the deformed shape $\boldsymbol{S}_\beta$ is simulated for these different β values with the corresponding interaction parameters (e.g., $\mathbf{f}_{\text{ext}}$ and $\mathbf{p}_c \in \boldsymbol{S}_\beta$) coming from ground truth observations. The β value that gives the minimum error between $\mathbf{p}^\beta \in \boldsymbol{S}_\beta$ and $\hat{\mathbf{p}}$ is selected as the deformation parameter that represents the ground truth deformation in the best way:

$$\min_{\beta} \text{Err}(\mathbf{p}^\beta, \hat{\mathbf{p}}). \tag{39}$$

4.2.2. Iterative Methods

Iterative methods try to decrease $\text{Err}(\mathbf{p}, \hat{\mathbf{p}})$ by moving the parameters step by step through the error function space. Techniques that help to ease the finding of the global minimum, such as *gradient descent* and *simulated annealing*, are frequently used to minimize the error of the model $G(\boldsymbol{S}^0, \mathbf{f}_{\text{ext}}, \mathbf{p}_c, \theta)$ being calibrated.

For example, Frank et al. (2014) used gradient descent to find the E and υ values required by an FEM model (section 3.3.1) $G_{\text{FEM}}(\boldsymbol{S}^0, \mathbf{f}_{\text{ext}}, \mathbf{p}_c, E, \upsilon)$. The applied forces $\mathbf{f}_{\text{ext}}$ are measured with a sensor located in the robotic manipulator; $\mathbf{p}_c$ is identified based on the collision point of the manipulator and the object in camera observations. Then, $\boldsymbol{S}^0$ is deformed according to these interaction parameters. To evaluate the simulation, 3D point clouds obtained from real objects through a depth camera are used as the ground truth $\hat{\mathbf{p}}$ and compared with the simulated FEM mesh, $\mathbf{p}^{E,\upsilon} \in \boldsymbol{S}^{E,\upsilon}$. Since the error function involves a computationally complex FEM model, it is difficult to compute the gradient directly. It is therefore approximated numerically by carrying out a sequence of deformation simulations that fix the interaction parameters in G_{FEM} and vary E and υ. The model thus obtained is good enough to be used for the planning of motion around deformable objects.

In contrast, Morris and Salisbury (2008) used simulated annealing to automatically calibrate an MS model and a mesh shape model based on Teschner's linear, planar, and volumetric spring energies (Teschner et al., 2004). An FEM simulation is used as the ground truth $\hat{\mathbf{p}}$. The forces $\mathbf{f}_{\text{ext}}$ are applied at vertices of a 3D mesh $\boldsymbol{S}^0$ that represents the object, producing deformations and resulting in a new state of static equilibrium (in which the object does not move but its shape is deformed under the influence of the forces). The objective is to reconstruct the FEM result with the MS model. To obtain the right set of parameters (e.g., k_s), pools of sets of parameters are generated by randomly assigning values to the parameters of the springs. The simulations are run with these parameter sets, and those whose behavior is unstable or which do not reach static equilibrium after a certain number of steps are eliminated. For every surviving set, an error function (e.g., $\text{Err}(\mathbf{p}^{k_s}, \hat{\mathbf{p}})$) is defined, which measures the distance between the nodes in the FEM simulation and the same nodes when displaced by MS simulation after a stable static state has been reached. Simulated annealing is used to minimize this error function.

4.2.3. Genetic Algorithms

Genetic and evolutive algorithms are optimization strategies inspired by natural evolution. Combinations of elements (e.g., the spring stiffness k_s of an MS model) that constitute solutions to a defined problem (e.g., calibrating the MS model) are codified as individuals in a population. A fitness or cost function, such as $\text{Err}(\mathbf{p}^{k_s}, \hat{\mathbf{p}})$, is defined to evaluate the potential solution represented by each individual; $\hat{\mathbf{p}}$ may come from an FEM simulation (Bianchi et al., 2004) or from camera images of a deforming object (Arriola-Rios and Wyatt, 2017). The objective of the algorithm is to find the individuals with the highest fitness value. To accomplish this, the population is evolved by means of genetic operators, such as mutation and crossover. Mutation applies random changes to each member of the population with a certain probability (e.g., sampling new k_s values from a Gaussian distribution). Crossover creates offspring by selecting genes from a pair of individuals and combining them into a new one, also with a predefined probability (e.g., combining some k_s values sampled in the previous generation with newly sampled values in the current generation). Different criteria can be used to select which members of the population are passed onto each new generation. Although genetic algorithms do not guarantee convergence to the global optimum, local optima that are attained can still be good approximations.

4.2.4. Neural Network Estimations

NNs can be used to estimate the parameters of different types of models (e.g., shape models $\boldsymbol{S}$ or dynamics models G). An NN is made up of several convolutional or fully connected layers. The weights of the connections between layers enable the observable parameters (e.g., the force $\mathbf{f}_{\text{ext}}$ applied by the external manipulator to deform the object) to be mapped to an output (e.g., the predicted deformed positions $\mathbf{p} \in \boldsymbol{S}$ of points constituting the shape of the object).

In several studies, Cretu et al. used neural gas networks to segment and track the shape of deformed objects (Cretu et al., 2010, 2012; Tawbe and Cretu, 2017). The shape of the object is represented using a mesh and neural gas, which allows a more adaptable representation (e.g., by increasing the resolution of the mesh around the manipulator, which needs a higher accuracy in the deformation computation, and decreasing the resolution

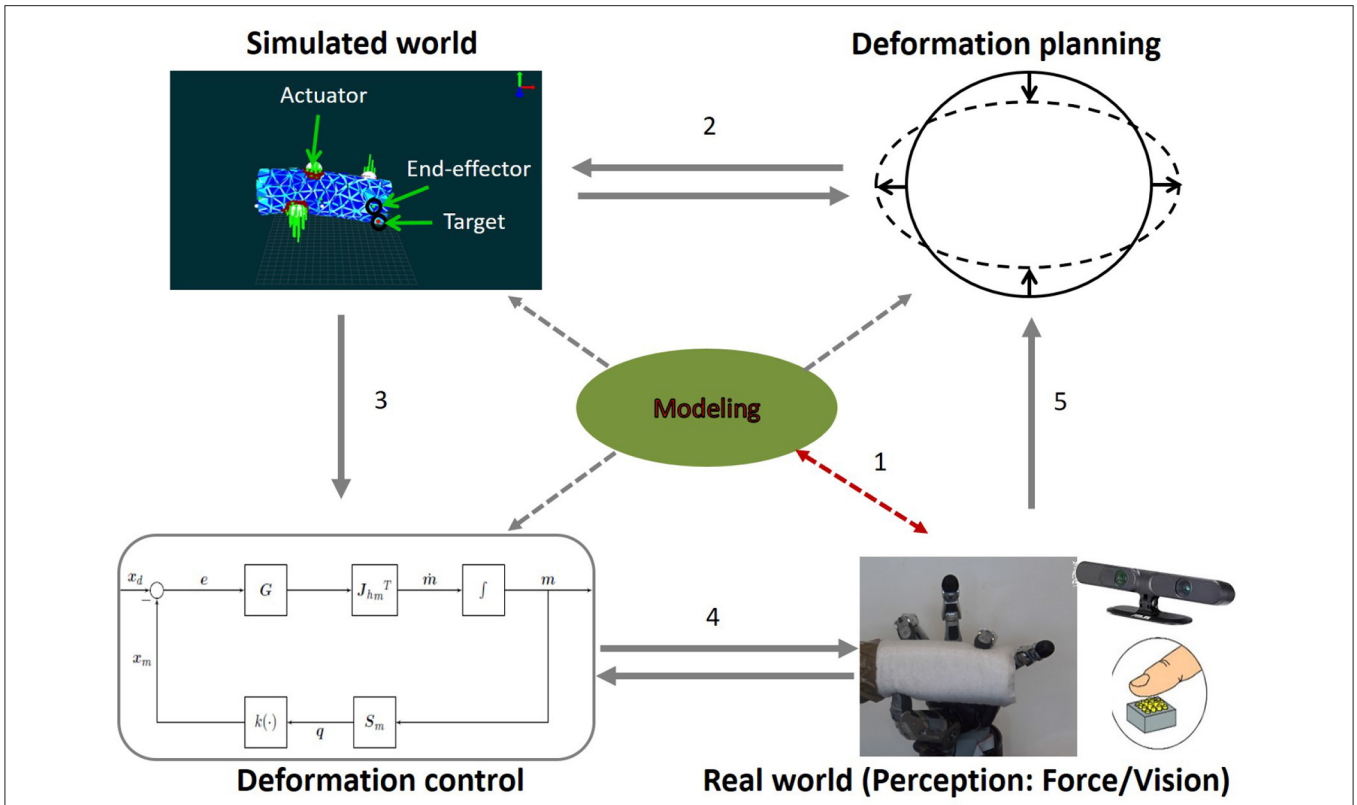

FIGURE 11 | A block schema representing connections between different elements of a complex architecture for robotic manipulation of 3D deformable objects, namely modeling, perception, planning, simulation, and control. In this representation a cylindrical object and an anthropomorphic hand are taken as an example of a practical application.

in the rest of the areas for computational efficiency). An NN model is used to predict the positions of the nodal points of the mesh $\mathbf{p}$ with respect to the applied force $\mathbf{f}_{\text{ext}}$. The NN is trained with data captured through force sensors and a depth camera. There are also works in which an NN is used with an FEM model to estimate parameters, such as E and υ and to predict the deformed positions of nodal points of the mesh of an object (e.g., Wang et al., 2018). The NN is trained offline using reference deformations with known parameters obtained from an FEM simulation and then used to predict online the deformation of real-world objects observed through a depth camera.

The advantage of such methods is that NNs provide efficient estimation similar to direct estimation, via an offline-trained NN mapping the observed input (e.g., forces applied) to an output (e.g., deformation). However, NNs require a lot of annotated training data, which may not be available or possible to obtain for every type of object that a robot might encounter.

4.3. Probabilistic Methods

In probabilistic approaches, usually one attempts to characterize a posterior probability, such as $p(\mathbf{p} \mid \hat{\mathbf{p}})$ or $p(\mathbf{u} \mid \hat{\mathbf{p}})$. The posterior probability represents the position $\mathbf{p}$ or the displacement $\mathbf{u}$ of the deformed shape with the highest probability relative to the erroneous (e.g., noisy or missing) observed deformation $\hat{\mathbf{p}}$ coming from a sensor, such as a depth camera. Shape and dynamics models are used to represent and calculate $\mathbf{p}$ or $\mathbf{u}$. To compute the posterior probability, methods, such as expectation maximization (Schulman et al., 2013) or sampling techniques, such as Markov chain Monte Carlo (Risholm et al., 2010) are used.

5. MANIPULATION PLANNING AND CONTROL

Accurate tracking and prediction of deformations, supported by learning models, are needed to control the actively changing shape and motion of objects during manipulation (**Figure 1iv**). Active control of shape and motion consists of five main modules (**Figure 11**). The core is (1) the modeling of deformation (i.e., shape representation, section 2) and dynamics (section 3) and the estimation of parameters (section 4), which contributes to (2) simulation, (3) planning, and (4) control, while receiving input from (5) perception. Developing a system that can perform precise information exchanges between all these modules is still an open research problem.

Figure 11 shows an architecture for the control of deformable objects inspired by the work of Ficuciello et al. (2018), in which only some aspects of the manipulation problem are

addressed. In the schema, the example of an anthropomorphic hand deforming a soft cylindrical object is considered. The manipulation is based primarily on modeling the deformation, with the model being refined and updated according to sensory perception that involves touch and vision (connection 1 in **Figure 11**). Then, based on the modeled shape $\boldsymbol{S}$ and dynamics G, planning consists in establishing the points/areas on $\boldsymbol{S}$ where concentrated/distributed external forces (called actuators) must be exerted by an end-effector to obtain desired (target) deformations. The choice of the points at which to exert the forces and the intensity of the forces are established during the planning phase, using G and a simulation that involves G and the object behavior, and this information is sent to the control (connection 3). Once the action is planned, the control module makes sure that the desired actions are correctly executed and that the expected deformations are achieved. To do this, it is necessary to consider a closed-loop control in which measurements of the state of the object and of the robot are used to correct errors (connection 4). Finally, based on the result of the action measured in the real world, a subsequent action can be planned (connection 5).

By taking into account this ideal architecture, in this section we survey the literature on robotic manipulation of deformable objects and organize the topic into two main categories, *planning of object manipulation* and *control of deformation*. The field of deformable object manipulation is still at an early stage of development. Therefore, as in Ficuciello et al. (2018), the works we review in the following cover only some of the modules depicted in **Figures 1**, **11** and do not describe a full system for complex deformation control of 3D objects.

5.1. Planning Object Manipulation

Only recently has the robotics community begun to publish results on manipulation of soft 3D objects. Most of the studies on motion planning for deformable object manipulation relate to deformable linear objects studied using mainly probabilistic roadmaps (Holleman et al., 1998; Moll and Kavraki, 2004; Saha and Isto, 2007) or deformable surgical tools and snake robots (Anshelevich et al., 2000; Bayazit et al., 2002; Teschner et al., 2004; Gayle et al., 2005). We identify two main bodies of work in the literature, namely work on deformable robots interacting with the environment and work on robots interacting with deformable environments. A few works concern planning in a complex deformable environment (Alterovitz et al., 2009; Maris et al., 2010; Patil et al., 2011).

As evidenced by the two blocks at the top of **Figure 11**, path planning for deformable object manipulation requires simulation of deformation, either when the deformation is the goal or when it is simply a side-effect of a planned action. In Jiménez (2012), a survey of model-based, offline planning strategies for deformable objects is presented and organized according to the type of manipulation, namely motion planning, folding/unfolding, topological modifications, and assembly. The paper is organized based on the geometric and physical characteristics of the object to be manipulated.

One of the methods adopting a model-based approach is that of Das and Sarkar (2011). In their method, three actuators are used to move the position of a point lying within the deformable object toward a goal location (target deformation). First an optimization technique, which minimizes the total force on the object to plan the location of the actuation points, is applied, and then a proportional-integral (PI) controller determines the motions of the actuators. The method uses a spline-like shape representation $\boldsymbol{S}$ (section 2.2.1) and an MS model (e.g., G_{MS}) to simulate the dynamics of the deformable object (section 3.2.2).

Li et al. (2016) also used a model-based approach. However, rather than simulating the deformation using a dynamics model, they used a database of deformed shapes $\boldsymbol{S}$ simulated by a commercial physics simulator called Maya (Autodesk, INC., 2019). A predictive model-driven approach is used to generate a large number of instances that can be used for learning and to optimize trajectories for manipulation of deformable objects, so as to create a bridge between the simulated and real worlds. Another work that uses a database approach is Mira et al. (2015), but rather than shape models, their database contains previous grasps, which are used by their grasp planner to determine the contact points. The grasp planner was designed for flexible objects that can be grasped by exploiting their inherent flexibility. Due to similarities between the robot's hand and the human hand, the planner reproduces the action of the human hand.

A method for efficient planning of trajectories for interaction with deformable objects that is closest to the ideal schema in **Figure 11** is that of Frank et al. (2014), which uses an FEM model (G_{FEM}, section 3.3.1) and a mesh representation (section 2.4). First, the robot perceives the object through a depth camera and the model $\boldsymbol{S}$ as a mesh. Then, it calibrates the model parameters of G_{FEM} with the gradient descent approach by estimating the material properties (i.e., E and υ) through physical interaction using a force sensor (section 4.2.2). However, to avoid time-consuming computation of the FEM during online planning of manipulation, an approximate, object-specific deformation cost function is created by means of Gaussian process regression.

A few works are also available that study the problem of finding the optimal contact location for handling a deformable object. In Wakamatsu et al. (1996), the concept of bounded force-closure, which is an extension of the concept of force-closure, is introduced, and stable grasping of deformable objects based on the concept is analyzed. In Gopalakrishnan and Goldberg (2004), a new framework for holding deformable parts is developed, building on the well-established form-closure framework for holding rigid parts. Deformable parts are modeled as linearly elastic polygons with a triangular mesh in an FEM model and a given stiffness matrix. New concepts, such as D-space (deformation-space) and deform closure grasps are introduced in that work, inspired by rigid-body grasp theory.

5.2. Control of Deformation

Works on deformation control mostly adopt one of two approaches: model-based and real-time sensor-based.

5.2.1. Model-Based Deformation Control

The methods discussed in the following require a deformation model of the target object. A simplified solution is often needed in order to derive a real-time control policy for manipulation that

reduces the computational cost. In this direction, Smolen and Patriciu (2009) introduced a model-based deformation control algorithm that relies on non-linear elasticity. In their work, the boundary of a deformable object is manipulated by defining a set of control points on the object surface which must converge to desired positions. In addition, Nanayakkara et al. (2016) focus on maintaining a stable grip on a soft object. In their study a soft object, which undergoes temporal variations in its internal impedance, is presented. A control law is derived based on a relaxed stability criterion, with the aim of maintaining a stable grip on the object. The proposed controller uses only three parameters to interpret the probability of failure, which is estimated using a history of grip forces.

There are also methods that use constitutive models. In Largilliere et al. (2015), the inverse solution of an FEM model (e.g., G_{FEM}) is used in the setting of soft robots that deform by the actions of a certain number of actuators. The goal is to generate proper actuator displacements so as to produce desired object deformations by means of desired displacements of selected control points. The desired actuator displacements are computed from the real-time inverse solution of the FEM equations, obtained by reducing the size of the constraint problem. For this purpose, a set of control points on the object is obtained using Lagrange multipliers. In Ficuciello et al. (2018), this framework is adapted to perform an active deformation control task with an anthropomorphic robot hand grasping a soft object. The method relies on G_{FEM} for real-time deformation control during dexterous manipulation of the 3D soft object. The goal is to generate proper forces at the fingertips during in-hand manipulation, so as to produce desired displacements of the selected control points on the object shape $\boldsymbol{S}_2$. The desired motions of the fingers are computed in real-time as an inverse solution of G_{FEM}, with the forces applied (e.g., $\mathbf{f}_{\text{ext}}$) by the fingertips at the contact points $\mathbf{p}_{\text{c}}$ being modeled by Lagrange multipliers (section 3.3.1). The model parameters (e.g., E and υ of G_{FEM}) are initially estimated using a vision system and a force sensor. Similarly, in Lin et al. (2015) an FEM formulation, G_{FEM}, is adopted to simulate the deformation of the object shape $\boldsymbol{S}$ based on the displacement caused by the finger contacts (e.g., $\mathbf{p}_{\text{c}}$ as the contact points and $\mathbf{f}_{\text{ext}}$ as the force applied). The simulation is used to check that the contact points do not slide while lifting the object. In this way, it is possible to properly control the motion of the fingers, so as to properly squeeze and lift the object. Moreover, in Zaidi et al. (2017), an FEM simulation based on a non-linear MS system is used to compute the deformation of an object grasped by three fingers. The contact forces are computed using the contact model based on the fingers' positions and velocities. After the contact forces are computed by simulation of the contact model guaranteeing equilibrium of the grasp, they are used as references for the grasping control strategy.

5.2.2. Real-Time Sensor-Based Deformation Control

Approaches that do not use any modeling cannot predict whether the desired object shape will be achieved, but they generally have lower computational costs. Here we briefly review such approaches that do not fully conform to our ideal schema described in **Figure 11** but do provide innovative and efficient control strategies for deformable objects using only sensory information. We divide them into two main categories, namely vision-based and tactile-based control strategies.

In recent work, visual servoing has been demonstrated to be a promising solution for accurate manipulation of deformation. Inspired by Smolen and Patriciu (2009), Berenson (2013) present a method of manipulating deformable objects that does not require modeling and simulation of deformation. They use a Jacobian-based method to drive the points within the deformable object toward a set of targets, assuming that they are able to sense the geometry of the object. Similar approaches are used in Cherubini et al. (2020), Hirai et al. (2001), and Wada et al. (2001). These methods align points of interest on the deformable object to targets, using a visual-servoing controller or a proportional-integral-derivative (PID) controller.

In addition, a series of works by Navarro-Alarcon et al. are based on a vision system (e.g., stereo vision) that tracks specific control points on the object and directs motion of a robotic manipulator to achieve a desired configuration (Navarro-Alarcon and Liu, 2013; Navarro-Alarcon et al., 2014; Navarro-Alarcon et al., 2016). In these papers, the deformation features based on a set of control points for describing different types of deformations are introduced for the first time. These features constitute the principal innovation of the work, together with the introduction of the *deformation Jacobian*. The deformation Jacobian defines the relationship between the manipulator's motion and the deformation feature vector, and it is estimated using Broyden's method.

In contrast, Hu et al. (2019) follows the general framework of visual servoing but combines it with a deep NN-based controller to servo-control the position and shape of a deformable object with unknown deformation properties. To describe the deformable object's deformation, a novel feature based on the fast point feature histogram (FPFH) is directly extracted from the 3D point cloud.

Apart from vision, the tactile sense is also widely used to control the shape of deformable objects during manipulation. For instance, recent work on sensor-based control by Delgado et al. uses tactile control for in-hand manipulation of elastic objects (Delgado et al., 2015; Delgado et al., 2017a,b). In their work, they adapt contact forces to different elastic properties of the object by estimating the deformability degree during the grasping process using only tactile sensors. The deformability degree can be a relative value between 0 and 1 (where 1 represents no deformation) or a classification, such as rigid body, soft elastic body, or soft plastic object, and it is related to the relationship between finger position displacement and the applied forces after the grasping process. Afterwards, contact points and forces are regulated according to the tactile information.

6. DISCUSSION

Manipulating deformable objects presents more challenges than manipulating rigid objects. Developing methods to specify models that take into account the dynamics of shape change is important for robotic manipulation. In this paper we have

introduced, using a combination of tutorial and review styles, some of the methods for modeling deformation. We have presented an ideal scenario for autonomous manipulation of deformable objects, which is summarized in **Figure 1**. According to this scenario, the problem consists of five main components.

6.1. Future Lines of Work

The five components of manipulating deformable objects still need improvements in the areas of shape perception, dynamics modeling, and planning/control of manipulation. Techniques from outside of robotics can help with this.

6.1.1. Modeling Shape

In most state-of-the-art work, the representation of shape deformation using various models is a missing aspect that could bring in more information for controlling the shape of the object. In this direction, the different shape models described in section 2 that are not yet widely used in robotics should be explored more. For instance, the level set method, which is mostly used in medical imaging or human tracking, could be developed further to include the temporal aspect of deformation. Also, in many works, deformed object detection and segmentation from the scene are done either by hand or using simplistic color segmentation techniques (e.g., Hu et al., 2019), which may not work for complex, cluttered scenes. Hence, more autonomous methods should be investigated, such as the deformable-templates method introduced in Ravishankar et al. (2008), which is mostly used in medical imaging, or neural network-based methods that have had success in many computer vision problems, such as segmentation (Marcos et al., 2018; Chen et al., 2019; Hatamizadeh et al., 2019).

6.1.2. Simultaneous Modeling, Learning, and Control

As described in sections 2 and 3, shape and dynamics models and their various parameters need to be known in order to perceive, track, and predict deformable behavior during manipulation. Currently, for realistic and accurate computation of deformation during planning and control, these models must be calibrated or learned beforehand. We have mentioned some methods for learning model parameters in section 4, but in most of the planning and control strategies (section 5), these learning techniques are applied in an offline setting because of their computational complexity. However, offline calibration may not be possible for every single object that a robot might encounter during online manipulation in real-world settings. In contrast, some control strategies do not rely on any modeling, and control of the object shape is based only on sensory measurements (e.g., Berenson, 2013), but such methods cannot predict whether the desired object shape will be achieved (section 5.2.2). To overcome this difficulty, manipulation could be controlled with a combination of model-building to incrementally learn the shape and dynamics of a previously unseen object—where the model includes the object's dynamics (e.g., force calculations and time integration scheme for updating the position and velocity)—and knowledge of properties, such as shape, mass, elasticity, etc.

In this direction, an interesting vein of work has started to emerge in recent years that we would like to mention briefly here: learning the physics of objects using deep NNs (Battaglia et al., 2016; Ajay et al., 2018). Deep NN models are similar to the NN dynamics models described in section 3.2.3 but are more scalable; for example, they can be used to model interactions between multiple objects in a single scene. We call these methods deep learning-based dynamics models. An advantage of these models is that they can learn directly from sensory observations (e.g., images of the observed scene) with high accuracy by virtue of their end-to-end differentiability, unlike the more computationally expensive analytical dynamics models in section 3, which are mostly too complex to differentiate. However, deep learning-based models have two limitations. First, most methods based on differentiable dynamics models are tested on limited scenarios with a few interacting rigid objects (e.g., balls colliding), which makes them difficult to generalize to more complicated behaviors of deformable objects. There are differentiable dynamics models that can learn more complex deformable object behavior, as in Mrowca et al. (2018) and Li et al. (2018); however, they may exhibit visually implausible deformations, such as the loss of shape preservation over time, or may not be able to deal with visual perception that includes partial or noisy observations of the state of objects, for instance due to the occlusion of a robot manipulator with an object. Furthermore, NN-based methods mostly work on entities of fixed dimensionality and do not exploit characteristics, such as the flexibility of splines.

In conclusion, for the future we need algorithms that can deal with various challenging manipulation scenarios where previously unseen objects appear in a scene (e.g., in contexts, such as cooking or nursing) by incrementally learning the shape and dynamics of a previously unseen object simultaneously while planning and controlling manipulation. To develop such methods, different approaches, such as integrating efficient differentiable models with physically accurate dynamics models, should be investigated. Also, models of dynamics should be adapted to work on different families of shape representations.

AUTHOR CONTRIBUTIONS

VA-R and PG were the main authors who contributed to each section. Additionally, VA-R was the main contributor to section 2, PG was the main contributor to section 3, and FF was the main contributor to section 5. JW, DK, and BS have made the substantial contributions to the conception or design of the work, revising the intellectual content critically, and providing the approval for publication.

FUNDING

This research has been funded by SIP-STRIM project TracMac (2017-02205). SIP-STRIM was a strategic research initiative funded by Vinnova, Formas and the Swedish Energy Agency. Also, it has been partially funded by the POR FESR 2014-2020 Italian National programme within BARTOLO project (CUP B41C17000090007) and by the PNR 2015-2020 Italian National programme within PROSCAN project (CUP E26C18000170005). In addition, it has been supported by The Swedish Research Council and Swedish foundation for strategic research.

REFERENCES

Ahlberg, J. (1996). *Active Contours in Three Dimensions* (thesis), Linkoping University, Sweden. Available online at: https://pdfs.semanticscholar.org/2cc5/13dbfef8ec6d7b4992adae3ff236e3e137a1.pdf?_ga=2.236805827.391043127.1596479054-447384472.1596479054.

Ajay, A., Wu, J., Fazeli, N., Bauza, M., Kaelbling, L. P., Tenenbaum, J. B., et al. (2018). "Augmenting physical simulators with stochastic neural networks: case study of planar pushing and bouncing," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Madrid: IEEE), 3066–3073. doi: 10.1109/IROS.2018.8593995

Alterovitz, R., Goldberg, K. Y., Pouliot, J., and Hsu, I. (2009). Sensorless motion planning for medical needle insertion in deformable tissues. *IEEE Trans. Inform. Technol. Biomed.* 13, 217–225. doi: 10.1109/TITB.2008.2008393

An, S. S., Kim, T., and James, D. L. (2008). Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph.* 27, 1–10. doi: 10.1145/1409060.1409118

Anshelevich, E., Owens, S., Lamiraux, F., and Kavraki, L. (2000). "Deformable volumes in path planning applications," in *IEEE International Conference on Robotics and Automation* (San Francisco, CA), 2290–2295. doi: 10.1109/ROBOT.2000.846368

Arriola-Rios, V. E., Demery, Z. P., Wyatt, J., Sloman, A., and Chappell, J. (2013). *Salient Features and Snapshots in Time: An Interdisciplinary Perspective on Object Representation*. Berlin; Heidelberg: Springer Berlin Heidelberg.

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

Arvanitis, G., Lalos, A., and Moustakas, K. (2019). Adaptive representation of dynamic 3d meshes for low-latency applications. *Comput. Aided Geometr. Des.* 73, 70–85. doi: 10.1016/j.cagd.2019.07.005

Autodesk, INC. (2019). *Maya*.

Balaniuk, R., and Salisbury, K. (2002). "Dynamic simulation of deformable objects using the long elements method," in *10th Symposium On Haptic Interfaces for Virtual Environment and Teleoperator Systems, Proceedings* (Orlando, FL), 58–65. doi: 10.1109/HAPTIC.2002.998941

Barbič, J., and James, D. L. (2005). Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM Trans. Graph.* 24, 982–990. doi: 10.1145/1073204.1073300

Barth, T., Herbin, R., and Ohlberger, M. (2018). "Finite volume methods: foundation and analysis," in *Encyclopedia of Computational Mechanics, 2nd Edn.*, eds E. Stein, R. de Borst, and T. J. R. Hughes, (New York, NY: John Wiley & Sons, Ltd.), 1–60. doi: 10.1002/9781119176817.ecm2010

Basri, R., Costa, L., Geiger, D., and Jacobs, D. (1998). Determining the similarity of deformable shapes. *Vision Res.* 38, 2365–2385. doi: 10.1016/S0042-6989(98)00043-1

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

Bayazit, O., Lien, J.-M., and Amato, N. (2002). "Probabilistic roadmap motionplanning for deformable objects," in *IEEE International Conference on Robotics and Automation* (Washington, DC), 2126–2135.

Bender, J., Müller, M., Otaduy, M., Teschner, M., and Macklin, M. (2014). A survey on position-based simulation methods in computer graphics. *Comput. Graph. Forum* 33, 228–251. doi: 10.1111/cgf.12346

Berenson, D. (2013). "Manipulation of deformable objects without modeling and simulating deformation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Tokyo), 4525–4532. doi: 10.1109/IROS.2013.6697007

Bianchi, G., Solenthaler, B., Székely, G., and Harders, M. (2004). "Simultaneous topology and stiffness identification for mass-spring models based on fem reference deformations," in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (Saint-Malo: Springer), 293–301. doi: 10.1007/978-3-540-30136-3_37

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

Blake, A., Bascle, B., Isard, M., and MacCormick, J. (1998). Statistical models of visual shape and motion. *Proc. R. Soc. Lond.* 356, 1283–1302. doi: 10.1098/rsta.1998.0222

Bohg, J., Morales, A., Asfour, T., and Kragic, D. (2014). Data-driven grasp synthesis–a survey. *IEEE Trans. Robot.* 30, 289–309. doi: 10.1109/TRO.2013.2289018

Bourguignon, D., and Cani, M.-P. (2000). "Controlling anisotropy in mass-spring systems," in *11th Eurographics Workshop on Computer Animation and Simulation, EGCAS 2000, August, 2000, Springer Computer Science*, eds N. Magnenat-Thalmann, D. Thalmann, and B. Arnaldi (Interlaken: Springer-Verlag), 113–123. doi: 10.1007/978-3-7091-6344-3_9

Caccamo, S., Bekiroglu, Y., Ek, C. H., and Kragic, D. (2016). "Active exploration using gaussian random fields and gaussian process implicit surfaces," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Daejeon: IEEE), 582–589. doi: 10.1109/IROS.2016.7759112

Cardiff, P., and Demirdžić, I. (2018). Thirty years of the finite volume method for solid mechanics. *arXiv [Preprint] arxiv* 1810.02105.

Catmull, E., and Clack, J. (1978). Recursively generated b-spline surfaces on arbitrary topological meshes. *Comput. Aided Des.* 10, 350–355. doi: 10.1016/0010-4485(78)90110-0

Chen, X., Williams, B. M., Vallabhaneni, S. R., Czanner, G., Williams, R., and Zheng, Y. (2019). "Learning active contour models for medical image segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Long Beach, CA), 11632–11640. doi: 10.1109/CVPR.2019.01190

Cherubini, A., Ortenzi, V., Cosgun, A., Lee, R., and Corke, P. (2020). Model-free vision-based shaping of deformable plastic materials. *Int. J. Robot. Res.* doi: 10.1177/0278364920907684

Cootes, T., Taylor, C., Cooper, D., and Graham, J. (1992). "Training models of shape from sets of examples," in *Proceedings of the British Machine Vision Conference* (London: Springer-Verlag), 9–18. doi: 10.5244/C.6.2

Cootes, T., Taylor, C., Cooper, D., and Graham, J. (1995). Active shape models-their training and application. *Comput. Vis. Image Understand.* 61, 38–59. doi: 10.1006/cviu.1995.1004

Cootes, T. F., and Taylor, C. J. (2004). *Statistical Models of Appearance for Computer Vision*, University of Manchester, Manchester, United Kingdom. Available online at: https://www.face-rec.org/algorithms/AAM/app_models.pdf.

Cordero Valle, J. M., and Cortes Parejo, J. (2003). *Curvas y Superficies para Modelado Geometrico*. Madrid: Alfaomega-Ra-Ma.

Cremers, D. (2006). Dynamical statistical shape priors for level set-based tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* 28, 1262–1273. doi: 10.1109/TPAMI.2006.161

Cretu, A.-M., Payeur, P., and Petriu, E. M. (2009). Neural gas and growing neural gas networks for selective 3D sensing: a comparative study. *Sens. Transd. J*, (Phoenix, AZ), 5, 119–134. doi: 10.1109/ROSE.2008.4669190

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

Cretu, A.-M., Petriu, E., Payeur, P., and Khalil, F. (2010). "Estimation of deformable object properties from shape and force measurements for virtualized reality applications," in *2010 IEEE International Symposium on Haptic Audio-Visual Environments and Games (HAVE)*, 1–6. doi: 10.1109/HAVE.2010.5623970

Das, B., and Banerjee, S. (2004). Inertial snake for contour detection in ultrasonography images. *IEE Proc. Vis. Image Signal Process.* 151, 235–240. doi: 10.1049/ip-vis:20040310

Das, J., and Sarkar, N. (2011). Autonomous shape control of a deformable object by multiple manipulators. *J. Intell. Robot. Syst.* 62, 3–27. doi: 10.1007/s10846-010-9436-5

de Boor, C. (1976). Splines as linear combinations of b-splines. A survey. *Approx. Theory* II, 1–47.

Delgado, A., Corrales, J., Mezouar, Y., Lequievre, L., Jara, C., and Torres, F. (2017b). Tactile control based on gaussian images and its application in bi-manual manipulation of deformable objects. *Robot. Auton. Syst.* 94, 148–161. doi: 10.1016/j.robot.2017.04.017

Delgado, A., Jara, C. A., Mira, D., and Torres, F. (2015). "A tactile-based grasping strategy for deformable objects' manipulation and deformability estimation," in *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)* (Colmar), Vol. 2, 369–374. doi: 10.5220/0005562103690374

Delgado, A., Jara, C. A., and Torres, F. (2017a). Adaptive tactile control for in-hand manipulation tasks of deformable objects. *Int. J. Adv. Manuf. Technol.* 91, 4127–4140. doi: 10.1007/s00170-017-0046-2

Delingette, H. (1999). General object reconstruction based on simplex meshes. *Int. J. Comput. Vis.* 32, 111–142. doi: 10.1023/A:1008157432188

Essa, I.A., Sclaroff, S., and Pentland, A. (1992). A unified approach for physical and geometric modeling for graphics and animation. *Comp. Graphics Forum* 11, 129–138. doi: 10.1111/1467-8659.1130129

Ficuciello, F., Migliozzi, A., Coevoet, E., Petit, A., and Duriez, C. (2018). "FEM-based deformation control for dexterous manipulation of 3D soft objects," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Madrid), 4007–4013. doi: 10.1109/IROS.2018.8593512

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

Fulton, L., Modi, V., Duvenaud, D., Levin, D. I., and Jacobson, A. (2019). Latent-space dynamics for reduced deformable simulation. *Comput. Graph. Forum* 38, 379–391. doi: 10.1111/cgf.13645

Gallardo, M., Pizarro, D., Collins, T., and Bartoli, A. (2020). Shape-from-template with curves. *Int. J. Comput. Vis.* 128, 121–165. doi: 10.1007/s11263-019-01214-z

Gascuel, M.-P. (1993). "An implicit formulation for precise contact modeling between flexible solids," in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '93* (New York, NY: ACM), 313–320. doi: 10.1145/166117.166157

Gayle, R., Segars, P., Lin, M. C., and Manocha, D. (2005). Path planning for deformable robots in complex environments. *Robot. Syst. Sci.* doi: 10.15607/RSS.2005.I.030

Gelder, A. V. (1998). Approximate simulation of elastic membranes by triangulated spring meshes. *J. Graph. Tools* 3, 21–41. doi: 10.1080/10867651.1998.10487490

Gibson, S. F. F., and Mirtich, B. (1997). *A Survey of Deformable Modeling in Computer Graphics*. Technical report, MERL (Mitsubishi Electric Research Laboratory).

Gopalakrishnan, K., and Goldberg, K. Y. (2004). "D-space and deform closure: a framework for holding deformable parts," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, Vol. 1, 345–350. doi: 10.1109/ROBOT.2004.1307174

Greminger, M. A., and Nelson, B. J. (2008). A deformable object tracking algorithm based on the boundary element method that is robust to occlusions and spurious edges. *Int. J. Comput. Vis.* 78, 29–45. doi: 10.1007/s11263-007-0076-6

Guler, P., Pauwels, K., Pieropan, A., Kjellstrom, 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

Guler, P., Pieropan, A., Ishikawa, M., and Kragic, D. (2017). "Estimating deformability of objects using meshless shape matching," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE), 5941–5948. doi: 10.1109/IROS.2017.8206489

Hatamizadeh, A., Hoogi, A., Sengupta, D., Lu, W., Wilcox, B., Rubin, D., et al. (2019). "Deep active lesion segmentation," in *International Workshop on Machine Learning in Medical Imaging* (Shenzhen: Springer), 98–105. doi: 10.1007/978-3-030-32692-0_12

Hauth, M., Etzmuß, O., and Straßer, W. (2003). Analysis of numerical methods for the simulation of deformable models. *Visual Comput.* 19, 581–600. doi: 10.1007/s00371-003-0206-2

Hirai, S., Tsuboi, T., and T., W. (2001). "Robust grasping manipulation of deformable objects," in *Proceeding of the IEEE Symposium on Assembly and Task Planning* (Fukuoka), 411–416. doi: 10.1109/ISATP.2001.929069

Holleman, C., Kavraki, L., and Warren, J. (1998). "Planning paths for a flexible surface patch," in *IEEE International Conference on Robotics and Automation* (Leuven), 21–26. doi: 10.1109/ROBOT.1998.676243

Hu, Z., Han, T., Sun, P., Pan, J., and Manocha, D. (2019). 3-D deformable object manipulation using deep neural networks. *IEEE Robot. Autom. Lett.* 4, 4255–4261. doi: 10.1109/LRA.2019.2930476

Jaklic, A., Leonardis, A., and Solina, F. (2000). *Segmentation and Recovery of Superquadrics*. Heidelberg: Springer.

James, D., and Pai, D. (1999). "Artdefo: accurate real time deformable objects," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY: ACM Press; Addison-Wesley Publishing Co.), 65–72. doi: 10.1145/311535.311542

Jiménez, P. (2012). Survey on model-based manipulation planning of deformable objects. *Robot. Comput. Integr. Manuf.* 28, 154–163. doi: 10.1016/j.rcim.2011.08.002

Kass, M., Witkin, A., and Terzopuolos, D. (1988). Snakes: active contour models. *Int. J. Comput. Vis.* 1, 321–331. doi: 10.1007/BF00133570

Kumar, S., Han, S., Goldgof, D., and Bowyer, K. (1995). On recovering hyperquadrics from range data. *IEEE Trans. Pattern Anal. Mach. Intell.* 17, 1079–1083. doi: 10.1109/34.473234

Largilliere, F., Verona, V., Coevoet, E., Sanz-Lopez, M., Dequidt, J., and Duriez, C. (2015). "Real-time control of soft-robots using asynchronous finite element modeling," in *2015 IEEE International Conference on Robotics and Automation (ICRA)* (Seattle, WA), 2550–2555. doi: 10.1109/ICRA.2015.7139541

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

Leventon, M. E., Grimson, W. E. L., and Faugeras, O. D. (2000). "Statistical shape influence in geodesic active contours," in *2000 Conference on Computer Vision and Pattern Recognition (CVPR 2000)* (Berder), 1316–1323. doi: 10.1109/CVPR.2000.855835

Li, Y., Wang, Y., Yue, Y., Xu, D., Case, M., Chang, S.-F., et al. (2016). *Model-Driven Feed-Forward Prediction for Manipulation of Deformable Objects*.

Li, Y., Wu, J., Tedrake, R., Tenenbaum, J. B., and Torralba, A. (2018). Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv [Preprint] arxiv* 1810.01566.

Lin, H., Guo, F., Wang, F., and Jia, Y.-B. (2015). Picking up a soft 3D object by "feeling" the grip. *Int. J. Robot. Res.* 34, 1361–1384. doi: 10.1177/0278364914564232

Luo, Y. H., and Nelson, B. J. (2001). Fusing force and vision feedback for manipulating deformable objects. *J. Robot. Syst.* 18, 103–117. doi: 10.1002/rob.1009

Macklin, M., Müller, M., Chentanez, N., and Kim, T.-Y. (2014). Unified particle physics for real-time applications. *ACM Trans. Graph.* 33:104. doi: 10.1145/2601097.2601152

Madi, K., Paquet, E., and Kheddouci, H. (2019). New graph distance for deformable 3D objects recognition based on triangle-stars decomposition. *Pattern Recogn.* 90, 297–307. doi: 10.1016/j.patcog.2019.01.040

Makovetskii, A., Voronin, S., Kober, V., and Voronin, A. (2020). An efficient algorithm for non-rigid object registration. *Comput. Opt.* 44, 67–73. doi: 10.18287/2412-6179-CO-586

Maraffi, C. (2004). *Maya Character Creation, Modeling and Animation Controls*. Indianapolis, IN: Stephanie Wall; New Riders Publisher.

Marcos, D., Tuia, D., Kellenberger, B., Zhang, L., Bai, M., Liao, R., et al. (2018). "Learning deep structured active contours end-to-end," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Salt Lake City, UT), 8877–8885.

Maris, B., Botturi, D., and Fiorini, P. (2010). "Trajectory planning with task constraints in densely filled environments," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Taipei), 2333–2338. doi: 10.1109/IROS.2010.5650483

Martínez, L., Ruiz-del Solar, J., Sun, L., Siebert, J., and Aragon-Camarasa, G. (2019). Continuous perception for deformable objects understanding. *Robot. Auton. Syst.* 118, 220–230. doi: 10.1016/j.robot.2019.05.010

Mira, D., Delgado, A., Mateo, C. M., Puente, S. T., Candelas, F. A., and Torres, F. (2015). "Study of dexterous robotic grasping for deformable objects manipulation," in *2015 23rd Mediterranean Conference on Control and Automation (MED)* (Torremolinos), 262–266. doi: 10.1109/MED.2015.7158760

Moll, M., and Kavraki, L. (2004). "Path planning for minimal energy curves of constant length," in *IEEE International Conference on Robotics and Automation* (New Orleans, LA), 2826–2831. doi: 10.1109/ROBOT.2004.1307489

Montagnat, J., Delingette, H., and Ayache, N. (2001). A review of deformable surfaces: topology, geometry and deformation. *Image Vis. Comput.* 19, 1023–1040. doi: 10.1016/S0262-8856(01)00064-6

Moore, P., and Molloy, D. (2007). "A survey of computer-based deformable models," in *IMVIP 2007: International Machine Vision and Image Processing Conference, Proceedings, Irish Pattern Recognit & Classificat Soc, IEEE Computer Soc. International Machine Vision and Image Processing Conference*, eds J. McDonald, C. Markham, and J. Ghent (Los Alamitos, CA; Maynooth: Natl Univ Ireland Maynooth), 55–64.

Morris, D., and Salisbury, K. (2008). Automatic preparation, calibration, and simulation of deformable objects. *Comput. Methods Biomech. Biomed. Eng.* 11, 263–279. doi: 10.1080/10255840701769606

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*, eds S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett, (Montreal: Neural Information Processing Systems Foundation, Inc.), 8799–8810.

Müller, M., and Gross, M. (2004). "Interactive virtual materials," in *Proceedings of Graphics Interface 2004* (Waterloo: Canadian Human-Computer Communications Society), 239–246.

Müller, M., Heidelberger, B., Teschner, M., and Gross, M. (2005). "Meshless deformations based on shape matching," in *SIGGRAPH* (Los Angeles, CA: ACM). doi: 10.1145/1186822.1073216

Müller, M., Stam, J., James, D., and Thürey, N. (2008). "Real time physics: class notes," in *ACM SIGGRAPH 2008 Classes* (Los Angeles, CA: ACM), 88. doi: 10.1145/1401132.1401245

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

Nanayakkara, T., Jiang, A., del Rocío Armas Fernández, M., Liu, H., Althoefer, K., and Bimbo, J. (2016). Stable grip control on soft objects with time-varying stiffness. *IEEE Trans. Robot.* 32, 626–637. doi: 10.1109/TRO.2016.2549545

Navarro-Alarcon, D., hui Liu, Y., Romero, J. G., and Li, P. (2014). On the visual deformation servoing of compliant objects: uncalibrated control methods and experiments. *Int. J. Robot. Res.* 33, 1462–1480. doi: 10.1177/0278364914529355

Navarro-Alarcon, D., and Liu, Y. (2013). "Uncalibrated vision-based deformation control of compliant objects with online estimation of the Jacobian matrix," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Tokyo), 4977–4982. doi: 10.1109/IROS.2013.6697075

Navarro-Alarcon, D., Yip, H. M., Wang, Z., Liu, Y., Zhong, F., Zhang, T., et al. (2016). Automatic 3-D manipulation of soft objects by robotic arms with an adaptive deformation model. *IEEE Trans. Robot.* 32, 429–441. doi: 10.1109/TRO.2016.2533639

Nealen, A., Mueller, 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

Newcombe, R. A., and Davison, A. J. (2010). "Live dense reconstruction with a single moving camera," in *CVPR* (San Francisco, CA). doi: 10.1109/CVPR.2010.5539794

Nisirat, M. A. (2019). A new external force for snake algorithm based on energy diffusion. *Int. J. Mach. Learn. Comput.* 9, 316–321. doi: 10.18178/ijmlc.2019.9.3.804

Nurnberger, A., Radetzky, A., and Kruse, R. (1998). "A problem specific recurrent neural network for the description and simulation of dynamic spring models," in *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)* (Anchorage, AK), Vol. 1, 468–473. doi: 10.1109/IJCNN.1998.682312

Pan, Z., and Manocha, D. (2018). Active animations of reduced deformable models with environment interactions. *ACM Trans. Graph.* 37, 1–17. doi: 10.1145/3197565

Patil, S., van den Berg, J., and Alterovitz, R. (2011). "Motion planning under uncertainty in highly deformable environments," in *Robotics: Science and Systems VII* (Los Angeles, CA: University of Southern California). doi: 10.15607/RSS.2011.VII.033

Pentland, A., and Williams, J. (1989). "Good vibrations: modal dynamics for graphics and animation," in *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, SIGGRAPH '89* (New York, NY: ACM), 215–222. doi: 10.1145/74333.74355

Petit, A., Cotin, S., Lippiello, V., and Siciliano, B. (2018). "Capturing deformations of interacting non-rigid objects using RGB-D data," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Madrid: IEEE), 491–497. doi: 10.1109/IROS.2018.8593756

Prasad, M., Fitzgibbon, A., Zisserman, A., and Van Gool, L. (2010). "Finding nemo: deformable object class modelling using curve matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (San Francisco, CA). doi: 10.1109/CVPR.2010.5539840

Raposo, A. N., and Gomes, A. J. (2019). "Pi-surfaces: products of implicit surfaces towards constructive composition of 3D objects," in *Proceedings of WSCG 2019 WSCG 2019 27. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (Pilsen). doi: 10.24132/CSRN.2019.2901.1.13

Ravishankar, S., Jain, A., and Mittal, A. (2008). "Multi-stage contour based detection of deformable objects," in *Computer Vision–ECCV 2008, Pt I, Proceedings* (Marseille), Vol. 5302, 483–496. doi: 10.1007/978-3-540-88682-2_37

Risholm, P., Samset, E., and Wells, W. (2010). "Bayesian estimation of deformation and elastic parameters in non-rigid registration," in *International Workshop on Biomedical Image Registration* (Lübeck: Springer), 104–115. doi: 10.1007/978-3-642-14366-3_10

Romeo, M., Monteagudo, C., and Sánchez-Quirós, D. (2020). Muscle and fascia simulation with extended position based dynamics. *Comput. Graph. Forum*, 39, 134–146. doi: 10.1111/cgf.13734

Saha, M., and Isto, P. (2007). Manipulation planning for deformable linear objects. *IEEE Trans. Robot.* 23, 1141–1150. doi: 10.1109/TRO.2007.907486

Sanchez, J., Corrales Ramon, 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

Schaefer, S., and Yuksel, C. (2007). "Example-based skeleton extraction," in *ACM International Conference Proceeding Series* (New York, NY), Vol. 257, 153–162.

Schulman, J., Lee, A., Ho, J., and Abbeel, P. (2013). "Tracking deformable objects with point clouds," in *2013 IEEE International Conference on Robotics and Automation (ICRA)* (Karlsruhe: IEEE), 1130–1137. doi: 10.1109/ICRA.2013.6630714

Sederberg, T. W., and Parry, S. R. (1986). "Free-form deformation of solid geometric models," in *SIGGRAPH '86 Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (Dallas, TX), Vol. 20, 151–160. doi: 10.1145/15922.15903

Sederberg, T. W., Zheng, J., Bakenov, A., and Nasri, A. (2003). T-splines and t-nurccs. *ACM Trans. Graph.* 22, 477–484. doi: 10.1145/882262.882295

Sengupta, A., Krupa, A., and Marchand, E. (2019). "Tracking of non-rigid objects using RGB-D camera," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)* (Bari: IEEE), 3310–3317. doi: 10.1109/SMC.2019.8914543

Sethian, J. A. (1997). Tracking interfaces with level sets: an "act of violence" helps solve evolving interface problems in geometry, fluid mechanics, robotic navigation and materials sciences. *Am. Sci.* 85, 254–263.

Sidorov, K., and Marshall, A. (2014). Learnt real-time meshless simulation. *Comput. Graph. Forum* 33, 147–156. doi: 10.1111/cgf.12440

Sinha, A., Billings, S. D., Reiter, A., Liu, X., Ishii, M., Hager, G. D., et al. (2019). The deformable most-likely-point paradigm. *Med. Image Anal.* 55, 148–164. doi: 10.1016/j.media.2019.04.013

Smolen, J., and Patriciu, A. (2009). "Deformation planning for robotic soft tissue manipulation," in *2009 Second International Conferences on Advances in Computer-Human Interactions* (Cancun), 199–204. doi: 10.1109/ACHI.2009.31

Song, Y., and Bai, L. (2008). "3D modeling for deformable objects," in *Articulated Motion and Deformable Objects, Proceedings* (Mallorca), Vol. 5098, 175–187. doi: 10.1007/978-3-540-70517-8_18

Sun, W., Cetin, M., Chan, R., and Willsky, A. S. (2008). Learning the dynamics and time-recursive boundary detection of deformable objects. *IEEE Trans. Image Process.* 17, 2186–2200. doi: 10.1109/TIP.2008.2004638

Szekely, G., Kelemen, A., Brechbuhler, C., and Gerig, G. (1995). "Segmentation of 3D objects from mri volume data using constrained elastic deformations of flexible fourier surface models," in *Proceedings of the First International Conference on Computer Vision, Virtual Reality and Robotics in Medicine, CVRMed'95* (Nice). doi: 10.1007/978-3-540-49197-2_66

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

Teran, J., Blemker, S., Hing, V., and Fedkiw, R. (2003). "Finite volume methods for the simulation of skeletal muscle," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, CA: Eurographics Association), 68–74.

Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. (1987). Elastically deformable models. *SIGGRAPH Comput. Graph*. 21, 205–214. doi: 10.1145/37402.37427

Teschner, M., Heidelberger, B., Muller, M., and Gross, M. (2004). "A versatile and robust model for geometrically complex deformable solids," in *Proceedings of Computer Graphics International (CGI'04)* (Crete), 312–319. doi: 10.1109/CGI.2004.1309227

Tian, Y., Yang, Y., Guo, X., and Prabhakaran, B. (2013). "Haptic-enabled interactive rendering of deformable objects based on shape matching," in *2013 IEEE International Symposium on Haptic Audio Visual Environments and Games (HAVE)* (Istanbul: IEEE), 75–80. doi: 10.1109/HAVE.2013.6679614

Tonnesen, D. L., and Terzopoulos, D. (2000). *Dynamically Coupled Particle Systems for Geometric Modeling, Reconstruction, and Animation*. Toronto, ON: University of Toronto.

Tsai, A. Jr., Yezzi, A., Wells, W., Tempany, C., Tucker, D., Fan, A., Grimson, W., et al. (2001). "Model-based curve evolution technique for image segmentation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (Kauai, HI), 463–468. doi: 10.1109/CVPR.2001.990511

Unser, M. (1999). Splines: a perfect fit for signal and image processing. *IEEE Signal Process. Mag.* 16, 22–38. doi: 10.1109/79.799930

Wada, T., Hirai, S., Kawamura, S., and Kamiji, N. (2001). "Robust manipulation of deformable objects by a simple pid feedback," in *IEEE International Conference on Robotics and Automation* (Seoul), 85–90. doi: 10.1109/ROBOT.2001.932534

Wakamatsu, H., Hirai, S., and Iwata, K. (1996). "Static analysis of deformable object grasping based on bounded force closure," in *Proceedings of IEEE International Conference on Robotics and Automation* (Minneapolis, MN), Vol. 4, 3324–3329. doi: 10.1109/ROBOT.1996.509219

Wang, Z., Rosa, S., Yang, B., Wang, S., Trigoni, N., and Markham A. (2018). "3D-PhysNet: learning the intuitive physics of non-rigid object deformations," in *27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence IJCAI-ECAI*, ed M. Hansson (Stockholm: International Joint Conferences on Artificial Intelligence).

Weir, A. A., and Kacelnik, A. (2006). A new caledonian crow (corvus moneduloides) creatively re-designs tools by bending or unbending aluminium strips. *Anim. Cogn.* 9:317. doi: 10.1007/s10071-006-0052-5

Williams, D., and Shah, M. (1992). A fast algorithm for active contours and curvature estimation. *CVGIP-Image Understand.* 55, 14–26. doi: 10.1016/1049-9660(92)90003-L

Xia, H., Zhao, W., Jiang, F., Li, H., Xin, J., and Zhou, Z. (2019). Fast template matching based on deformable best-buddies similarity measure. *Multimed. Tools Appl.* 78, 11905–11925. doi: 10.1007/s11042-018-6722-x

Xian, Z., Tong, X., and Liu, T. (2019). A scalable galerkin multigrid method for real-time simulation of deformable objects. *ACM Trans. Graph.* 38:162. doi: 10.1145/3355089.3356486

Xu, L., Lu, Y., and Liu, Q. (2018). Integrating viscoelastic mass spring dampers into position-based dynamics to simulate soft tissue deformation in real time. *R. Soc. Open Sci.* 5:171587. doi: 10.1098/rsos.171587

Yuille, A. L., Hallinan, P. W., and Cohen, D. S. (1992). Feature extraction from faces using deformable templates. *Int. J. Comput. Vis.* 8, 99–111. doi: 10.1007/BF00127169

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

Zhang, J., Zhong, Y., Smith, J., and Gu, C. (2019). Neural dynamics-based poisson propagation for deformable modelling. *Neural Comput. Appl.* 31, 1091–1101. doi: 10.1007/s00521-017-3132-3

Zhu, B., Gu, L., Zhang, J., Yan, Z., Pan, L., and Zhao, Q. (2008). "Simulation of organ deformation using boundary element method and meshless shape matching," in *EMBS 2008. 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (Vancouver: IEEE), 3253–3256. doi: 10.1109/IEMBS.2008.4649898

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.

The handling editor declared a past co-authorship with one of the authors DK.

Copyright © 2020 Arriola-Rios, Guler, Ficuciello, Kragic, Siciliano and Wyatt. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.