# A Multi-Modal Model of Object Deformation under Robotic Pushing

Veronica E. Arriola-Rios, Jeremy L. Wyatt

***Abstract*—In this paper we present a multi-modal framework for learning generative models of object deformation under robotic pushing. The model is multi-modal in that it is based on integrating force and visual information. The framework consists of several sub-models that are independently learned from the same data. These component models can be sequenced to provide many-step prediction and classification. When presented with a test example–a robot finger pushing a deformable object made of an unidentified, but previously learned, material–the predictions of modules for different materials are compared so as to classify the unknown material. Our approach, which consists of learning and combining multiple models, goes beyond previous techniques by enabling i) predictions over many steps, ii) learning of plastic and elastic deformation from real data, iii) prediction of forces experienced by the robot, iv) classification of materials from both force and visual data, v) prediction of object behaviour after contact by the robot terminates. While previous work on deformable object behaviour in robotics has offered one or two of these features none has offered a way to achieve them all, and none has offered classification from a generative model. We do so through separately learned models which can be combined in different ways for different purposes.**

***Index Terms*—deformable objects, prediction, classification, learning.**

## I. Introduction

A Major challenge in robot manipulation is to plan actions with objects so as to deform them into new shapes. Most objects that are manipulated by animals in natural environments are deformable, including objects that are elastic, plastic, breakable, tearable, and spreadable. The work reported here is based on the premise that in order to plan manipulations of deformable objects, the sensorimotor contingencies governing object deformation are required. A sensorimotor contingency is simply defined as the relationship between the robot's actions and the resultant changes in its observations. Different materials (here, elastic and plastic materials) have different sensorimotor contingencies. The main hypothesis is that predictive models of these sensorimotor contingencies can be learned from data. The predictions of a learned model can be fed back into itself, so as to predict an object's deformation behaviour over many time steps. These learned models can then be used to make the predictions required for planning manipulation.

There are various models of deformation used in engineering, such as finite element models, but these typically require careful optimisation by hand, and cannot run in real time. In graphics there are many methods used for generating qualitatively plausible simulations of deformable objects, but these are not easy to fit to real data. In contrast, robots must learn model parameters from data, on-line, in reasonable amounts of training time. This includes learning from the limited sensing typically available, such as partial visual views of an object, and force-torque monitoring only at contact points. Thus, tackling the prediction problem in robotics adds additional difficulties to an already challenging problem.

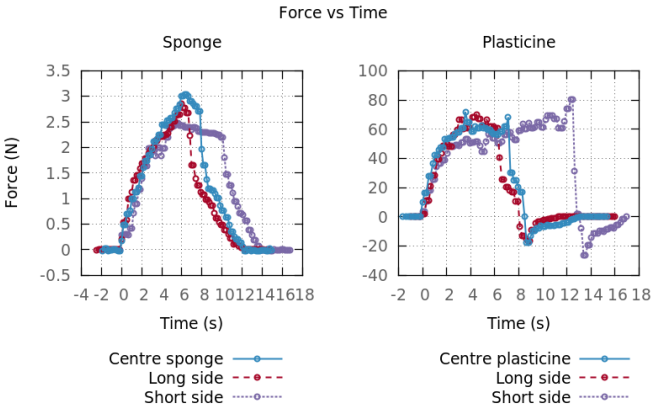



Fig. 1. Force information acquired while working with a block of sponge (elastic) and a piece of plasticine (plastic). Notice the difference in magnitude between the force used with the sponge and the plasticine.

Previous work on learning sensorimotor contingencies for deformable objects in robotics has been limited. Table I summarises the features of the major efforts to date. In each case some abilities are missing: either models cannot be learned from data; or only address one type of deformation (e.g. elastic); or cannot perform classification; or cannot predict the forces felt; or cannot predict what will happen when the robot loses contact with the object (will it retain or recover from the deformation?). In this paper we present an approach for solving all these problems. Although our framework requires two separate learned models, they are combined into a single machine for prediction of the behaviour of deformable material. In a second machine, multiple copies of this predictor can be compared to an actual outcome deformation, so as to classify the material.

There are several novel contributions. First, we show how the learning problem can be decomposed into two separate learning problems, one for predicting forces from finger motions, and one for predicting object deformations from forces. Both models are generative, and we show how to learn each

V.E. Arriola-Rios is with Facultad de Ciencias, UNAM, Mexico, e-mail: v.arriola@ciencias.unam.mx

J.L. Wyatt is with the University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK. e-mail: jeremy.l.wyatt@gmail.com

from data. Third, we show how to use these learned generative models not only to predict deformation, but also to classify an unidentified material.

In summary, our approach enables us to learn models of sensorimotor contingencies, and then to use these as components to build different machines. This approach enables i) predictions over many steps, ii) learning of plastic and elastic deformation from real data, iii) prediction of forces experienced by the robot, iv) classification of materials from either or both force and visual data, v) prediction of object behaviour after contact by the robot is removed. In the rest of the paper we present related work (Section II); give an overview of the framework (Section III); describe the two component models in detail (Section IV); describe how the model parameters are learned from real data (Section V); report experiments on prediction and classification with elastic and plastic objects (Section VII); and finish with a short discussion (Section VIII).

TABLE I
ADVANCES OVER PREVIOUS WORK

| | Model type | Learned from GT | GT=Real data | Prediction of Shape | Prediction of Forces | Actuator Retrieval | Elastic | Plastic | Classification |
|---|---|---|---|---|---|---|---|---|---|
| Howard | MS | ✓ | ✓ | x | ✓ | x | ✓ | x | x |
| Teschner | MS | x | x | ✓ | x | ✓ | ✓ | ✓ | x |
| Morris | MS | ✓ | x | ✓ | x | x | ✓ | x | x |
| Frank | FEM | ✓ | ✓ | ✓ | x | QSA | ✓ | x | x |
| This work | MS + Regression | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

*MS = *Mass-spring*
*QSA = *Quasi-Static Assumption*

## II. BACKGROUND

It is known that the human motor system uses predictive (or forward) models of the effects that motor actions have on sensory state [15], [14], [22]. These are believed to be particularly important for dexterous manipulation. In robotics, prediction of physical interaction is a well studied but incompletely solved problem. There is a large body of work on the effects of pushing on rigid objects. This includes physics based models [27], [41], [30] and learning approaches [36], [34], [13], [26], [45], [33], [43], [24], [25], [3]. These predictive models have many applications, including visual tracking [39], [12], [42] and push planning [44], [11], [28], [47], [8].

There is much less work in robotics on predicting the behaviour of deformable objects under manipulation. This paper focuses on learning to predict what happens when an elastic or plastic object is pushed by a robot finger. These predictions include the response forces and the deformations of the object's shape over many future steps. The predictive models can be used to classify materials, even when the interactions (e.g. the contact point) are novel. Although there is a large body of work on deformable object tracking, some of which makes use of forces, models and predictions to improve tracking accuracy [6], [19], [1], those predictions are typically used only for the next frame and can not be used for robotic planning, where predictions must be made many steps into the future [10]. Others like [29], [9] make use of templates or make assumptions about the shape of the objects that would not work with highly deformable objects like plasticine. There is also an abundant literature in industrial robotics for modelling 1D and 2D materials (e.g. string, hair, metallic sheet and textile), but there is little research about predicting the deformation of 3D deformable objects [23]. We now review the most closely related work to that presented here.

First, Howard and Bekey [21] address robotic grasping and lifting of viscoelastic objects. Their model of the material can predict a net amount of deformation given the applied forces, but does not evaluate details of the deformed shape. This model is inspired by the crystalline structure of atoms in solids. The space lattice of the crystal is approximated by a particle system where elements with mass are connected by springs and dampers, thus representing the attraction and repulsion between atoms *in equilibrium* with a damped mass-spring system. The masses are calculated by measuring the force required to lift the object without sliding. The deformation and damping coefficients are estimated as functions of:

- the force applied while pushing against the material with two robot arms in opposite directions and,
- the global displacement and velocity vectors resulting from adding the displacements and velocities of all particles.

The estimated values of the mass, deformability and damping coefficients are used as the input to a neural network that outputs the minimum necessary force to lift the object without sliding. This function is used to compliantly adjust the force during an interaction. However, they do not evaluate the quality of the overall shape prediction, and use only the final values of the prediction after the system has stabilised; that is, they do not evaluate the predictions during the dynamic phase.

Teschner et al. [46] proposed an enhanced 3D mass-spring model, in which constraints are expressed as potential energy terms. These terms encode the tendency of springs to recover their length (with damping), of triangular faces of a mesh to recover their area and of tetrahedrons to recover their volume. He sketches the inclusion of plastic deformation. Morris and Salisbury [38] developed a method to calibrate Teschner's model automatically with respect to a FEM model. The search begins with a uniform random sampling of possible values for the elasticity constants, which are latter modified using adaptive simulated annealing to minimize the difference between the deformations in the FEM mesh and those in the mass-spring mesh. However, they only compare the steady states and do not include plastic deformations. We extend this work, by replacing the FEM model with force and 2D visual data gathered from real materials. Furthermore, we analyse the quality of the predictions while the deformations take place.

Frank et al. [16] use a force sensor and a bumblebee stereo

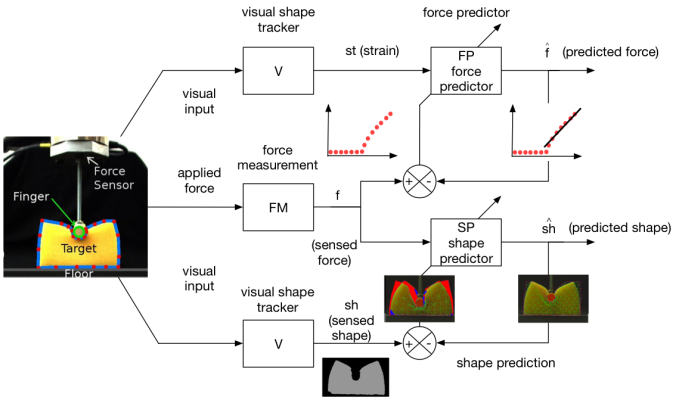



Fig. 2. The learning method comprises two learned models of sensorimotor contingencies. Both are learned from the same data. The first (FP) learns to predicts the force experienced for some motion and material. The second (SP) predicts the deformation of the object given the applied force. The strain $st$ is calculated directly from the interpenetration of the finger into the object. This can be calculated by vision (V) or from the planned motion of the finger.

camera to measure the Young modulus and the Poisson ratio of unknown deformable objects. By assuming a homogeneous material, they propose candidate values for these constitutive parameters and simulate the expected behaviour of the object with a FEM method. The volume of the prediction is compared with a 3D reconstruction of the 3D surface of the object. A gradient descent search is used to minimize the difference. Once the parameters are selected, a collision detection algorithm informs the model of the position of the actuator, thus implying the amount of local deformation. The FEM simulation provides estimates for the global deformation and response forces. However, a quasi-static assumption is made and a detailed study of the dynamic process is not offered.[1] The resulting model is applied to estimating costs of robotic navigation around deformable objects [17].

The advances of our work with respect to these related approaches is summarised in Table I.

Other interesting work includes that of Conti et al. [7] which introduced six-degree of freedom macroscopic elastic spheres described by mass, inertial and volumetric properties that are used to approximate the volume of deformable objects. The open source project Chai3D includes an implementation of this model. The spheres are placed along the skeleton of the object and are connected together with elastic links which model elongation, flexion and torsion properties. Each vertex of the surface mesh is attached to the nearest sphere or link with a damped spring. In [5], Burion and Baur, collaborated with Conti's group to automatically calibrate this model using particle filters. Again, there is no analysis of the dynamics and no real objects are used. Furthermore, this model is more adequate for objects that deform around fixed joints.

Finally, Cretu et al., used growing neural gas networks to learn to predictively track the deformation of objects. Even though they managed to make predictions about the deformation of the overall shape of the object, under previously

[1]With the quasi-static assumption a small force is applied and a state of equilibrium is reached before a new force is applied. Therefore the deformation process is approximated by a series of states of equilibrium.

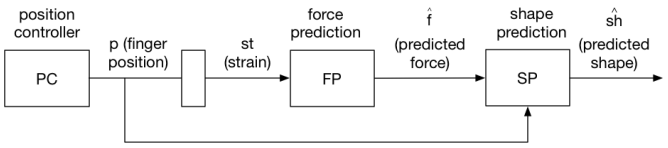



Fig. 3. The two learned models can be sequenced, so as to predict an object's shape and reaction forces when the robot finger will push in a particular way. The predicted shape is fed back on itself so as to predict over many steps for a planned sequence of finger motion. PC is the position controller for the robot, which outputs each planned finger position $p$ at each time. This is used to calculate the strain $st$ from the interpenetration using a fixed function, as well as providing an input to the shape predictor (SP).

unseen sets of forces, their predictions were evaluated only for the next frame.

Additionally, there is a vast literature on modeling of deformable shapes and deformation processes [18], [31], [35], [37], [40], particularly within the area of computer graphics. However, these types of models have not been successfully applied to robotics tasks beyond what is presented above.

## III. Model Overview

The framework consists of three separate systems: a learning system, a prediction system, and a classification system. The learning system is depicted in Figure 2. This enables learning of sensorimotor contingencies from real robot data. Learning of the force prediction (FP) model relies on comparing the predicted $\hat{f}$ and the experienced $f$ forces at the finger tip, as measured by a force-torque sensor (FM). Learning of the shape prediction (SP) model relies on comparing the predicted deformation $\hat{sh}$ with the actual deformation $sh$ measured by visual tracking (V).

The two learned models, FP and SP, can be sequenced to create a single, multi-modal predictor (Figure 3). This combined model takes a planned motion of the finger, and predicts the resulting forces that will be experienced at the contact points. It then uses both the planned positions and predicted forces to predict the resulting object deformations. Typically, a sequence of planned finger positions will be known. In prediction mode the models make predictions based on this sequence. To create the prediction for time $t+1$ the previous predicted state at time $t$ is required: thus the predictions are recursive. This enables predictions over arbitrarily long time periods, but makes learning a good predictor challenging.

The third system is for material classification (Figure 4). This simply comprises multiple copies of the prediction systems, one for each type of material that an unknown object might be made of. The predictors each make many-step predictions. At each step a classification is made, based on which predictor has made the better prediction of the shape at that step.

In summary, this way of learning and using models involves several choices. First, the learned models are generative. Second, our approach can be seen as one in which two contingencies are learned separately, and then sequenced to produce an overall prediction. Third, although here we only compare two material models for prediction, the scheme follows the principle of modular motor learning [25], [20],

where prediction and control is specialised into many modules, each of which covers a relatively small portion of the input space, such as an object or material type. Thus the aim of a generalised version of our scheme will be to acquire many specialist models of different deformable objects rather than a very few rather general models. The specific model forms we choose here could be replaced while retaining the overall scheme. In this paper, we implement the scheme using a regression approach to force prediction, and we follow several other authors in employing a mass-spring system with learnable parameters for the deformation prediction.

For clarity, the algorithm is described in exactly these three phases: training, simulation (i.e. the prediction phase), and classification. In empirically evaluating the model new data was used, acquired from new finger pushes. We now proceed to describe the shape predictor SP (Section IV), which is based on a mass-spring system, and then the force predictor (FP), which is a regression model (Section VI).

## IV. A Mass-Spring Model of Deformation

A mass-spring model is a very well established abstraction, commonly used to simulate the behaviour of certain types of deformable objects. As a physics based model, once its parameters are acquired, its behaviour is determined by the evolution of a set of differential equations. Therefore, by offering a method to automatically calibrate those parameters, this model can make long term predictions about the behaviour of deformable objects, under previously unseen interactions, if only the externally applied forces are known. The range of validity of these predictions greatly depends on the quality of the calibration, but also on the suitability of the mass-spring model as a model of the particular object. For this reason, there are variations that address specific issues, like [4] and [46]. In this work, we focus on the machinery required to achieve this automatic calibration of a physics based model and its integration into a system that exploits it to solve prediction and classification tasks. We emphasize that the physics based model could be easily substituted by another equivalent model. For example, the same machine will work if another type of mass-spring model is used, or a FEM model, even though numerical results will differ to some extent. For these experiments we chose to use a slightly modified version of the mass-spring model proposed by Teschner in [46].

In a simple mass-spring model, the shape of an object is approximated by a uniform geometric mesh, usually made of triangles (2D) or tetrahedrons (3D). At every vertex $i$ in the mesh, an ideal particle with no volume, but with mass $m_i$, is located, and is known as a *mass particle*. The edges between vertices roughly model the interactions between particles, which attract each other up to a certain critical distance, but repel each other if they become closer, therefore maintaining the cohesion of the solid, while avoiding the collapse of the material on itself. The simplest model associates a linear spring with each edge, as if those forces would occur only between connected neighbours. The resting state of the solid corresponds to a configuration of the masses and springs where the sum of all forces acting over each particle is zero, and they are said to be in equilibrium.

When one or more particles are displaced from their equilibirum positions, the forces that tend to restore an elastic body to its original shape are modeled by the forces of springs (along the edges of the mesh) that try to recover their original length). These forces act upon the particles modifying their positions in time. Traditionally, the relationship between the position of the particle $p_i$ and those forces $F$ obeys the second law of Newton:

$$F = m\frac{\partial \vec{p}}{\partial t} \tag{1}$$

where $t$ is time. However, it is possible to deviate from this equation if we need to model a different behaviour.

The above equation is a second order differential equation, which is continuous by definition. In order to perform a computer simulation we need to obtain the position $p$ of all vertices as a function of time $p(t)$ using this equation. A numerical method is selected to approximate a solution [40]. Time is discretized in intervals of length $h$, positions and forces at discrete past times are used to estimate the new positions at time $t + h$, according to an integration scheme which must be selected. The fact that the values of the forces remain constant in this approximation, for the whole interval $h$ (also know as *time step*), introduces errors which can be severe. For this reason, computer simulations may add procedures that help to alleviate these effects.

To simulate the effect of an external actuator (like the finger of the robot) interacting with the deformable object, external forces $F_{ext}$ acting on vertices of the mesh must be added to (1):

$$F + F_{ext} = m\frac{\partial \vec{p}}{\partial t} \tag{2}$$

Another option could be to induce response forces by displacing vertices from their position of equilibrium.

### A. Elastic Deformation

The mass-spring model proposed by Teschner et al. in [46], which is explained below, was reduced to a 2D model, so that the predictions could be compared with 2D images of the deformable object. The main idea being that an agent can make rough real time predictions of what it can see and touch and use those predictions to solve various tasks, like classification or grasping. Here, the surface of the deformable object is approximated by a regular triangular mesh of unitary masses and springs. Since the location of the masses and direction of the springs will bias the deformation response of the object, it is important to distribute the masses uniformly and place the springs with a symetric arrangement for an isotropic material. This placement can be done with an algorithm that takes into account the symmetries of the object at the begining of the experiment. If the material gets permanently deformed a properly callibrated physics based model, like the one we propose here, will produce an adequate new arrangement from which new interactions can be studied.

The potential energy terms for the preservation of length with damping and preservation of area were included in our system exactly as they were defined by Teschner. To

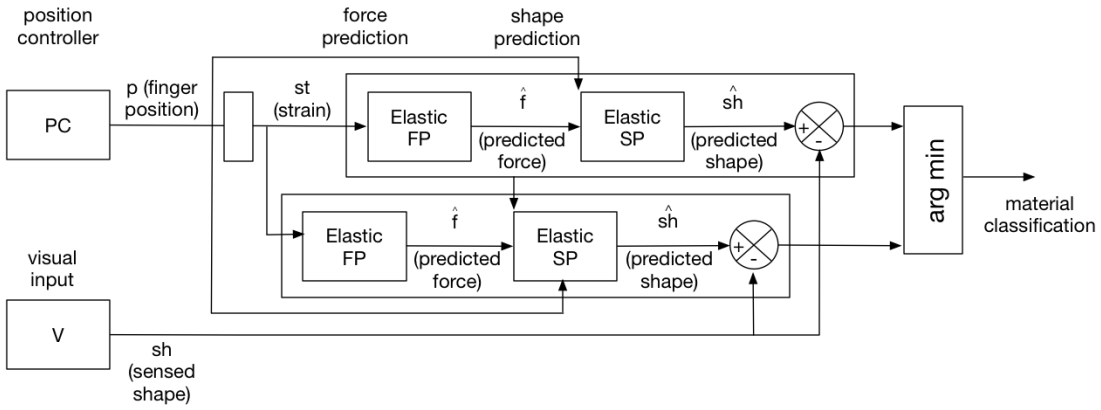



Fig. 4. Prediction sequences for several predictors–each of which has been learned for a different material–can be compared against real data. The most accurate predictor at any one moment gives the classification for the material type. The notation is explained in the text.

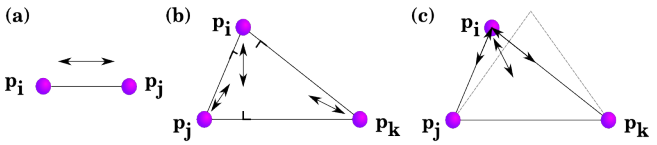



Fig. 5. Spring Forces. Recovery of: (a) Length, forces act along the spring. (b) Area, forces act along the heights of the triangles. (c) Angle, masses slide trying to restore the angle.

compensate for the lack of a term for preservation of volume, which tends to prevent distortions of the graph in the original model, we derived a new term for the preservation of the angles in every triangle. The relevant equations are presented in the following paragraphs, while detailed derivations can be found in [2].

Teschner defined a generalized version of the simple mass-spring model by using the concept of *constraint*. He makes use of different constraints to model the tendency of an object to recover several of its attributes. For every desired constraint $C(\vec{p}_1, ..., \vec{p}_n)$ acting over $n$ vertices, a potential energy $E$ is defined. When the value of the constraint is zero, the potential energy is zero as well. When the constraint is not satisfied, the potential energy increases. This behaviour is obtained through the following definition for the energy:

$$E(\vec{p}_1, ..., \vec{p}_n) = \frac{1}{2} k C(\vec{p}_1, ..., \vec{p}_n)^2, \tag{3}$$

where $k$ is a proportionality constant and $p_i$ is the position of particle $i$.

In order to reduce the energy of the system when a constraint is not satisfied, the force $\vec{F}^i$ acting on the $i^{th}$ mass particle is defined as the negative gradient of this potential energy with respect to the position of the particle $\vec{p}_i$ (4).

$$\vec{F}^i(\vec{p}_1, ..., \vec{p}_n) = -\frac{\partial}{\partial \vec{p}_i} E = -kC \frac{\partial C}{\partial \vec{p}_i}. \tag{4}$$

Given the oscillatory nature of these equations, when using constraints that model the behaviour of springs, damping can be introduced to stop the oscillations. This is modelled with a force acting in the opposite direction to the velocity of the affected particles, thus slowing down motion. Teschner wrote this as a function of the constraints as well [46]:

$$\vec{F}^i(\vec{p}_1, ..., \vec{p}_n, \vec{v}_1, ..., \vec{v}_n) = \left( -kC - k_d \sum_{1 \le j \le n} \frac{\partial C}{\partial \vec{p}_j} \vec{v}_j \right) \frac{\partial C}{\partial \vec{p}_i}, \tag{5}$$

where $\vec{v}_j = \frac{\partial \vec{p}_j}{\partial t}$ is the velocity of the $j^{th}$ particle connected to $i$, when the constraint acts on $n$ particles.

*1) Preservation of Length:* The constraint for the potential energy is given by:

$$C = \frac{|\vec{p}_j - \vec{p}_i| - D_0}{D_0}, \tag{6}$$

where $\vec{p}_i$ and $\vec{p}_j$ are connected by a spring and $D_0$ is the rest length of that spring. The force derived from this constraint pulls the masses in the direction of the spring that joins them [Fig. 5(a)]. Teschner normalised the value dividing by $D_0$ to make the elasticity constants scale independent [46]. Morris [38] omitted that in his final equations for the force. Here we use the normalised force $\vec{F}^i_D$ (7), which has the effect that, if we obtain a good set of parameters for a certain mesh of high resolution, the same set can be used for a mesh of lesser resolution that preserves the same type of symmetries. Therefore, substituting (6) in (5), after some manipulation we get:

$$\vec{F}^i_D(p_i, p_j, v_i, v_j) = \frac{k_L}{D_0^2} \left( |\vec{p}_j - \vec{p}_i| - D_0 \right) \frac{\vec{p}_j - \vec{p}_i}{|\vec{p}_j - \vec{p}_i|} + \frac{k_d}{D_0^2} \left( \frac{\vec{p}_j - \vec{p}_i}{|\vec{p}_j - \vec{p}_i|} \cdot (\vec{v}_j - \vec{v}_i) \right) \frac{\vec{p}_j - \vec{p}_i}{|\vec{p}_j - \vec{p}_i|} \tag{7}$$

where $k_L$ is the linear spring constant and $k_d$ the damping constant.

*2) Preservation of Area:* This constraint is applied per triangle in the mesh. Since Teschner and Morris found that the use of damping for the preservation of areas does not significantly improve the stability of the simulation, it is not included here either. Therefore the force $\vec{F}^i_A$ is derived as

follows:

$$C = \frac{\frac{1}{2}|(\vec{p}_j - \vec{p}_i) \times (\vec{p}_k - \vec{p}_i)| - A_0}{A_0} \tag{8}$$

$$Area = \frac{1}{2}|(\vec{p}_j - \vec{p}_i) \times (\vec{p}_k - \vec{p}_i)|, \tag{9}$$

$$\vec{F}_A^i(p_i, p_j, p_k) = -\frac{k_A}{2A_0^2}(Area - A_0) \frac{(\vec{p}_j - \vec{p}_i) \times (\vec{p}_k - \vec{p}_i)}{|(\vec{p}_j - \vec{p}_i) \times (\vec{p}_k - \vec{p}_i)|}[\vec{1} \times (\vec{p}_j - \vec{p}_k)] \tag{10}$$

where $A_0$ is the initial area of the triangle and $k_A$ the proportionality constant.

It is possible to rewrite (10) in a more geometrically intuitive manner by rewriting the direction of the gradient according to Morris' geometric derivation [38], but keeping the area normalisation constant[2]. It becomes evident that the forces pull along the heights of the triangles [Fig. 5(b)]. Then the magnitude of the force is given by:

$$forcemag_A(\vec{p}_i) = \frac{\frac{1}{2}|(\vec{p}_j - \vec{p}_i) \times (\vec{p}_k - \vec{p}_i)| - A_0}{A_0^2} \tag{11}$$

While the rest becomes:

$$\vec{F}_A(\vec{p}_i) = k_A \cdot forcemag_A(\vec{p}_i) \cdot forcedir_A(\vec{p}_i)$$

$$forcedir_A(\vec{p}_i) = \frac{\vec{F}_A(\vec{p}_i)}{|\vec{F}_A(\vec{p}_i)|} = \frac{areagradient(\vec{p}_i)}{|areagradient(\vec{p}_i)|}$$

$$areagradient(\vec{p}_i) = (\vec{p}_i - \vec{p}_j) - \left((\vec{p}_k - \vec{p}_j) \cdot \frac{(\vec{p}_k - \vec{p}_j) \cdot (\vec{p}_i - \vec{p}_j)}{(\vec{p}_k - \vec{p}_j) \cdot (\vec{p}_k - \vec{p}_j)}\right) \tag{12}$$

*3) Preservation of Angles:* The previous terms can not do anything to restore the mesh if triangles get flipped during deformation. For this reason we included a new term that enforces the preservation of the original angles of the triangles. [Fig. 5(c)] gives an idea of how these forces look. Here the energy depends on the difference between the current angle, between adjacent edges, and the angle between them at the equilibrium position. Energy is given by[3]:

$$E_\varphi(\varphi) = \frac{1}{2}k_\varphi(\varphi - \varphi_0)^2 \tag{13}$$

$$\varphi(p_i, p_j, p_k) = \arccos\left(\frac{(p_j - p_i) \cdot (p_k - p_i)}{\|p_j - p_i\| \, \|p_k - p_i\|}\right)^2$$

where $\varphi$ is the angle between adjacent edges, $E_\varphi$ is the energy associated with changes in the angle, $k_\varphi$ is the corresponding stiffness constant, and the $p_i$s are the Cartesian coordinates of the mass particles.

The force emerging from this term is a linear combination of the vectors along the edges that form the angle of interest, acting in the direction of the gradient. It tends to restore the angles in the most efficient way, but does not take the original size into account [Fig. 5(c)]. Therefore, it helps to recover a similar triangle, but it may produce tiny or very big triangles if it is not accompanied by some of the other terms that tend to restore the original dimensions, in addition to the angles. The force is derived from:

$$F_\varphi(p_i) = k_\varphi(\varphi - \varphi_0)\frac{\partial \varphi}{\partial p_i} \tag{14}$$

$$\frac{\partial \varphi}{\partial p_i} = \frac{\partial}{\partial p_i} arccos\left(\frac{(\vec{p}_j - \vec{p}_i) \cdot (\vec{p}_k - \vec{p}_i)}{\|\vec{p}_j - \vec{p}_i\| \, \|\vec{p}_k - \vec{p}_i\|}\right) \tag{15}$$

$$\tag{16}$$

The force is:

$$F_\varphi(p_i) = k_\varphi(\varphi - \varphi_0)\frac{\partial \varphi}{\partial p_i} \tag{17}$$

$$\frac{\partial \varphi}{\partial p_i}(p_i) = \frac{1}{d_{ji}d_{ki}\sqrt{1 - \left[\frac{pp}{(d_{ji})(d_{ki})}\right]^2}} \left\{\left[1 - \frac{pp}{d_{ki}^2}\right](p_k - p_i) + \left[1 - \frac{pp}{d_{ji}^2}\right](p_j - p_i)\right\}$$

$$pp(p_i) = (p_j - p_i) \cdot (p_k - p_i)$$

$$d_{ji}(p_i) = \|p_j - p_i\|$$

$$d_{ki}(p_i) = \|p_k - p_i\| \tag{18}$$

The terms for the preservation of length and angle model elastic deformations. Even though the preservation of area by itself would allow some plasticity, it is still necessary to incorporate permanent deformations in the rest lengths of the springs, to stop the triangles of the mesh from trying to recover their original shape.

**Algorithm 1:** *Plastic Deformation of an Edge*

Let $l$ be the length of the edge at time $t + h$, after an integration step where new positions of its vertices $p_i(t + h)$ where calculated. And $max\_\alpha$ the maximum proportional permanent deformation being allowed.
**if** $\alpha > max\_\alpha$ **then**
  $l_0 \leftarrow l$
**else**
  $l_0 \leftarrow (1.0 - \alpha)D_0$
**end if**
$elastic\_deformation \leftarrow \frac{l_0 - l}{D_0}$
**if** $elastic\_deformation > yield$ **then**
  $\alpha \leftarrow \alpha + creep * elastic\_deformation$
  **if** $\alpha > max\_\alpha$ **then**
    $\alpha \leftarrow max\_\alpha$
  **end if**
**end if**

Fig. 6. Algorithm to calculate the permanent deformation of an edge.

## B. Plastic Deformation

The simplest method to model plastic deformation consists in changing the length at rest $D_0$ of the springs, in (7). The effect of the permanent deformation can be either to permanently compress this length, or to permanently expand it (up to the breaking point). However, arbitrarily changing the value of $D_0$ can produce geometrical oddities like collapsed edges. For this reason, in this work, the permanent modification is expressed

[2]This substitution is valid because both are expressions for the gradients of the area of the triangle formed by the three verticies.

[3]It was also considered to multiply $E_\varphi$ by the lengths of the edges, but it hasn't improved the performance of the model.

in terms of a constant of compression $\alpha$, with respect to the original length $D_0$, so that the effective value of the length at rest becomes $l_0 = (1.0 - \alpha)D_0$. The value of $\alpha$ changes every time the spring is deformed beyond the threshold value *yield*. A *maximum deformation* limit is introduced to avoid collapsing edges due to compression: if the new length $l$ is smaller than a given fraction of the original length at rest (e.g. $l = 0.2D_0$), the length at rest suffers no further modifications. See Algorithm 1.

### C. Integration Scheme

Rather than using traditional Newtonian equations with a numeric integration scheme like Euler or Verlet, used twice to solve the second order differential equiations, we made the force proportional to the velocity:

$$F(t) = \vec{p} \tag{19}$$

which implies the following assumptions:

$$\frac{\partial \vec{p}}{\partial t} = \vec{p} \tag{20}$$

this equation can only be satisfied with an exponential, thus avoiding the oscillatory behaviour of springs in favour of forcing critically damped springs to model the elastic materials.

Then we applied Euler only once to solve for $p$:

$$\vec{p}(t+h) = \vec{p}(t) + h\frac{\vec{F(t)}}{m} \tag{21}$$

with $\vec{F}(t) = \vec{F}_D(t) + \vec{F}_A(t) + F_\varphi(t)$ being the sum of the forces for the preservation of length, area and angle respectively and $\vec{p}(t+h)$, the position of the vertex at time $t+h$. We acknowledge that this is not the proper Newtonian model, however the simulations obtained were closer to the observed behaviour, because there is no inertia in the oscillation. When experimenting with different integration steps, force values were interpolated between measurements.

### D. Collision Detection

In the simulation, rigid objects (like the robotic finger and the table) are treated as geometric obstacles. When the simulation of the mesh deformation runs—used to model the sponge—the predicted positions of the vertices may indicate that the mesh penetrates an obstacle. This should be interpreted as a collision between the sponge and an obstacle. Since the sponge can not penetrate an obstacle, these overlaps must be eliminated.

In order to model this, we model the finger and table using geometric primitives. For the table obstacle, any interpenetration by the sponge is solved by pushing vertices of the mesh of the sponge back to the border of the table. The springs will then propagate the effect to their neighbours. For the finger obstacle, resolution is in two stages: by vertex and then by edge. A vertex will be pushed out to a distance $\epsilon$ of the finger, in the direction of the radius. For edges, the standard algorithm in Algorithm 2 is used to estimate the shortest distance between the centre of the finger and the edge. Given this distance, the algorithm in Algorithm 3 indicates how far and in which direction the edge must be displaced.

**Algorithm 2:** *Closest Point on a Segment to Any Other Point*

Let $a$, $b$ be the ends of the edge $\overline{AB}$, $c$ the external point, and $x$ the closest point to $c$ in the line.
Let the vectors $\vec{AB} = b - a$ and $\vec{AC} = c - a$.
The projection of $\vec{AC}$ over $\vec{AB}$ is $\|x\| = \dfrac{\vec{AC} \cdot \vec{AB}}{\|\vec{AB}\|}$
**if** $\|x\| < 0$ **then**
  $closest \leftarrow a$
**else**
  **if** $\|x\| > \|\vec{AB}\|$ **then**
    $closest \leftarrow b$
  **else**
    $closest \leftarrow a + \|x\| \cdot \dfrac{\vec{AB}}{\|\vec{AB}\|}$
  **end if**
**end if**

Fig. 7. Algorithm to find the closest point on a segment to any other point.

**Algorithm 3:** *Push Edge Out*

Let $a$, $b$ be the ends of the edge $\overline{AB}$, $c$ the centre of the circle. Use algorithm 2 to find the closest point to the centre in the edge.
**if** $\|(closest - c)\| < radius$ (There is an intersection) **then**
  $displacement = (radius - \|(closest - c)\| + \epsilon) * \dfrac{(closest - c)}{\|(closest - c)\|}$
  $a \leftarrow a + displacement$
  $b \leftarrow b + displacement$
**end if**

Fig. 8. Algorithm to push an edge out of a circular obstacle.

### E. Additional Geometric Constraints

While the mass-spring model incorporates constraints based on mesh geometry, these are soft constraints implemented in the form of an energy function. Even with several of these terms incorporated, such as a term for preservation of angles, the mass-spring model can cause triangles of the mesh to overlap one another and be reversed or flattened. All of these situations are innadecuate since this mesh is used to model one face of a 3D object. It is thus necessary to employ additional hard geometric constraints to make the mesh internally consistent. We achieve this through geometric checking subroutines. There are two geometric constraints that the mesh is enforced to maintain:

1) For a reversed triangle, (one vertex crossed over an edge): the vertex gets pushed more than half the distance between the vertex and the edge in the direction of the "reversed" height[4], while both ends of the edge get pushed the same amount in the opposite direction. This is enough for most cases, but triangles in the border of the mesh can still overlap each other which would render a bad numerical approximation to the behaviour of a surface.

[4] 3/4 of this distance was used in our implementation.

2) If a triangle becomes flat: the longest edge and its opposite vertex along the perpendicular to the edge are pushed in opposite directions, to form a tiny triangle[5].

### F. Many Step Prediction

Given the initial shape of the object and a set of parameters, the mass-spring system allows predictions of the deformation of an object for various interactions. Given the procedures explained in previous sections, there are three ways to calculate the subsequent deformations of the shape:

1) **Collision detection only.** Collision detection is used to determine the compression of the springs around the actuator. In this case, the neighbouring springs are compressed and their preservation forces propagate the deformation towards their neighbours, and so on. There is no need to use the force sensor. It can be equated with observing someone else pushing an object we have touched before.
2) **Forces on vertices.** The forces to be applied on the vertices around the actuator are approximated by the measured reaction forces. Both the series of positions of the actuator and the reaction forces are necessary. These forces will be solely responsible for the deformation of the mesh. If the deformation is insufficient, the actuator may interpenetrate the border of the mesh. The quality of the simulation is much more sensitive to the values of the elasticity-plasticity constants. This corresponds to the robot itself pushing the object and feeling the response, while using vision solely to determine the position of the actuator and for evaluation purposes.
3) **Collisions and forces.** The previous two approaches can be combined. Visual constraints (collision detection) and forces are used to determine the displacement of nodes around the actuator. Again, the springs propagate the deformation to the rest of the mesh.

The mesh as predicted one step ahead is then fed back as the current shape for the next step and the cycle is repeated until the simulation is stopped.

The next stage is the learning process which consists in identifying sets of parameters that produce simulations similar to the ground truth. This problem is addressed in the following section.

## V. Learning to Predict Deformations

### A. Tracking to Provide Ground Truth

The contour of the deformable object is used to obtain the ground truth for the training phase, which is the area enclosed by the contour. Since we only require a tracking algorithm which is good enough to evaluate the predictions of the mass-spring system, a very simple one was used. Geometrically, the contour is represented by a polygon with hundreds of sides, which can also be called a *linear spline* [32]. The vertices of the polygon work as control points which can be moved to adjust the polygon to the shape of the object as it gets deformed. Even though this implementation is inspired by the theory of active contours, it works with a much simpler and faster, but less accurate, algorithm, instead of minimizing a global energy. This causes some shadowed corners of the material to be missed at some frames, but as Fig. 9(b) at its bottom left shows, these errors are not significant in comparison to the errors made by the mass-spring model. This is explained below. The tracking of the deformable target in the ground truth has two stages:

[5]An area of 2.0 units was good enough.

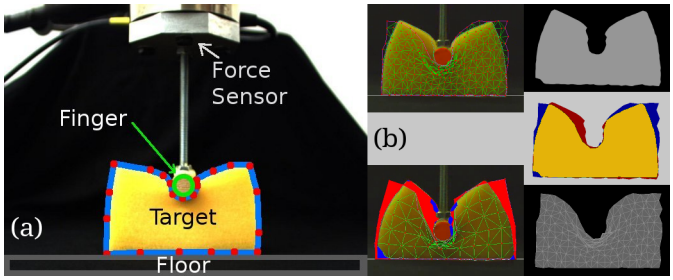



Fig. 9. (a) Experimental set up. A robot finger pushes a block of deformable material sitting on a table. A camera registers the events of the plane where the main deformations take place. (b) Evaluation function. Left top: Mesh over real block of material. Left bottom: The typical error of the model vs. the ground truth (red) and the typical error of the tracker (blue). Right top: tracked ground data. Middle: TP in yellow, FP in blue, FN in red and TN in gray. Bottom: simulated mesh.

1) The target object is segmented from the environment. The Canny algorithm is used to extract the edges of the image. Since the object of interest has a distinctive colour, its characteristic hue range is extracted from the area enclosed by the initial contour. Then the hue of the pixels on every edge are used to keep only edges that could belong to it. If the average hue of the pixels on an edge falls outside the calibrated range, the edge is eliminated.
2) The internal representation of its contour, the linear snake, is adapted to its new shape. The algorithm assumes that no occlusions take place and that the contour does not bend over itself.

For the first frame, the snake is initialised as a rectangle around the target. This rectangle fragments itself into smaller pieces. Each new vertex adheres itself to the closest point on an edge, as reached by growing rings of pixels around its current position, up to a certain threshold distance [Fig. 9(a)]. If there is no edge within that area, the control point remains where it is. This criterion was selected to make the representation more robust against edges that are easier to detect on some frames than in others. Once the edge reappears, the control point adapts itself again. Also, the distance between two consecutive control points is kept within the interval $[min\_distance, max\_distance]$ and two adjacent edges do not form angles smaller than $min\_angle$. See Algorithm 4. This allows the snake to adapt very quickly to the complexity of the contour it represents.

### B. Evolutionary Algorithm

Since a mass-spring model is not a constitutive model, its parameters are not directly derived from the physical properties of the material, they instead depend on the geometry

**Algorithm 4:** *Regularise Linear Spline*

Let $L$ be a linear spline representing the contour of a $2D$ shape, and $L_i$ the $i^{th}$ control point in a cyclic order.
**for all** $L_i$ in $L$ **do**
  **if** $dist(L_i, L_{i-1}) > max\_distance$ **then**
    Insert $midpoint(L_i, L_{i-1})$ after $L_i$
  **else if** $dist(L_i, L_{i-1}) < min\_distance$ **then**
    erase $L_i$
  **else if** $angle(L_{i-1}, L_i, L_{i+1}) < min\_angle$ **then**
    erase $L_i$
  **end if**
**end for**

Fig. 10. Algorithm used to regularise the number of elements of a linear spline, in accordance with the level of detail required.

of the mesh used to model the object. There can thus be several sets of parameters in different regions of the parameter space that produce adequate simulations [38]. We therefore chose to use an evolutionary algorithm to both perform global search, and local improvement.

The evolutionary algorithm described in Algorithm 5 was used to search the parameter space of the mass-spring model presented in the previous section. From the parameters that must be chosen for the simulation to work, some were fixed while others had to be searched for. They are:

- The mass per vertex. *(For the moment we consider a unitary mass for all vertices.)*
- The elasticity constants: for the preservation of length $k_L$, area $k_A$, angles $k_\varphi$ and linear damping $k_d$. *(All springs will have the same values to favour generalisation.)*
- The plastic parameters: $yield$, $creep$ and the maximum amount of permanent deformation $max\_\alpha$. *(All springs will have the same values to favour generalisation.)*
- The integration time step $\Delta t$. *(It has been set to 0.1 seconds, even though values as small as 0.01 were tried as well; however, since they didn't improve stability or accuracy in a visible way, but made the process much slower, they were disregarded as impractical.)*
- An upper limit for the magnitude of the total estimated forces per vertex $max\_Force$.
- The minimum area before a triangle is considered *flat*. (It has been set to 2.0 units.)

To evaluate the suitability of each set of parameters, the simulation is compared with the ground truth tracked from real objects.

### C. Evaluation Function

To evaluate the quality of the predicted deformations, the pixels inside the tracked contour are used as ground truth. These are compared against those inside the mesh of the simulation. In machine learning terms, these pixels can be divided into four classes: the *true positives* $TP$, where the material is present, and the simulation predicted it would be present; the *true negatives* $TN$, where the absence of material is in accordance with the prediction; the *false positives* $FP$, where the prediction said there would be material, but there is not;

**Algorithm 5:** *Evolutionary Search*

Let $N$ be the number of sets of parameters to be evaluated per generation and $M$ the number of generations.
Let $\alpha + \beta + \gamma + \zeta = N$ denote four portions of the $N$ elements, with:
$\alpha \leftarrow$ number of elements that survive for the next generation as they are,
$\beta \leftarrow$ mutated individuals,
$\gamma \leftarrow$ crossed over, and
$\zeta \leftarrow$ new individuals.
which will be chosen as follows:
Let $\sigma_{max} > \sigma_{min}$ both be standard deviations of a Gaussian
Let $\Delta\sigma \leftarrow (\sigma_{max} - \sigma_{min})/M$.
**for** $i = 1 \rightarrow M$ **do**
  Run $N$ simulations and compare prediction with ground truth (real object), by comparing the area occupied by the real object with the area occupied by the simulated mesh [Fig. 9(b)].
  Keep the best $\alpha$ from all simulations.
  $\sigma \leftarrow \sigma - \Delta\sigma$
  Generate $\beta$ by adding random Gaussian noise, with standard deviation $\sigma$, to the best candidates.
  Generate $\gamma$ by selecting values of the parameters from two different sets at random, after eliminating the $\epsilon$ worst.
  **for** $\zeta$ elements **do**
    Generate completely new sample with:
    **for** every parameter **do**
      $n \leftarrow randomNumber \in [0, 1)$
      $min \leftarrow \log(minValue)$
      $max \leftarrow \log(maxValue)$
      $param \leftarrow e^{(n(max-min)+min)}$
    **end for**
  **end for**
  $i \leftarrow i + 1$
**end for**

Fig. 11. Evolutionary Search for Automatic Calibration

and the *false negatives* $FN$, where there is material, but the prediction said there wouldn't be [Fig. 9(b)]. Given that maintaining a record of the true negatives (the empty/background space around the material) is mostly irrelevant and highly inefficient, the score for a whole video was derived from the following quantities:

$$Precision = \frac{TP}{TP + FP}, \tag{22}$$

$$Recall = \frac{TP}{TP + FN} \tag{23}$$

From which the harmonic average $F$ is derived:

$$F = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{1}{1 + \frac{FN+FP}{2TP}} \tag{24}$$

This measurement is know as the F-score and is used in the computer vision community to evaluate segmentation algorithms. $F = 1$ means that there were no false predictions

($FN + FP = 0$). $F \to 0$[6], if there are no good predictions ($TP \to 0$) or if there are too many errors ($FN + FP \to \infty$). $F$ is calculated for every frame and the score for the video at frame $i$ is the mean over all the frames so far, $\mu(F)$:

$$\mu(F) = \sum_{t=0}^{t=i} F(frame = t) \tag{25}$$

If the predictions for all the frames were perfect, $\mu(F)$ would be equal to 1; the closer to zero, the more errors there were. The simulation stops sometimes, if a model has become unstable; from that frame on, the value of $F$ is zero. The model with the highest final value of $\mu(F)$ is considered the best.

## VI. Learning to predict forces

Additionally and independently from the mass-spring system, it is possible to learn to make predictions about the expected reaction force of the material, by using information from the stress-strain diagram obtained from the same training data set as the mass-spring model. This diagram is obtained from the video recorded by the visual system of the robot and the forces recorded by a sensor located at the wrist. When the readings of the force sensor start to increase and the figer collides with the block of material, the point of zero deformation is identified. From here on, the displacement of the finger is considered equivalent to the amount of deformation in the material. Even though this method is not as rigorous as an engineering test, the estimation is good enough for robot manipulation of common materials.

The stress strain curves in [Fig. 14] are approximated by segmented regression curves. Several candidates are proposed using the least squares technique and the best are chosen as follows. Since there is only one independent variable the equations for adjusting a line are straightforward:

$$y = mx + y_0 + Error \tag{26}$$

$$m = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \tag{27}$$

$$y_0 = \frac{\sum y_i}{n} - y_0 \frac{\sum x_i}{n} \tag{28}$$

$$Error = \sum (y_i - (mx_i + y0))^2 \tag{29}$$

Where $y$ is the dependent variable, $x$ the independent one, $m$ is the slope of the line, and $y_0$ the $y$ intercept. To adjust logarithmic curves a change of variable is used. Before computing the line, the natural logarithm of the independent variable is calculated, that is: $x_i' = \ln(x_i)$. In this case, the final equation will be of the form: $y = m \ln x + y_0$. It would be possible to add other types of curves, but these were enough for the materials covered.

For automatic calibration of the force prediction model for the pushing (as opposed to retraction) phase both families of lines (logarithmic curves and straight lines) are considered. To determine the points of discontinuity between segments, each family is evaluated incrementally. The candidate line for a particular family is obtained by fitting a growing subset of data points and calculating the fitting error. We begin with a small number of points (e.g. five). To detect the discontinuity new points are added until the corresponding normalized error increases beyond a threshold value. This marks the point of discontiuity. Having discovered one best parameterized line for each family, the two resulting candidates are compared. We select the type of curve that covers more points in a single segment. This model is used to predict the forces for the mass-spring simulation Fig. 12 and Fig. 14.

## VII. Experimental Results

### A. Scenario

For the experimental scenario a rectangular block of deformable material, of $8.5\,\mathrm{cm}$ long, $5\,\mathrm{cm}$ wide and $1.8\,\mathrm{cm}$ depth, is placed on a table; the robot pushes the object with a cylindrical finger, followed by a retrieval movement. A colour firewire camera records the action perpendicularly, while a force sensor registers the reaction forces in synchrony with each photograph taken [Fig. 9(a)]. This set up allowed us to study the effect of using an automatically calibrated physics based model as model for the material, while using 2D meshes, even though the same principles can be used if we use a 3D point cloud and a 3D mesh. To generate the initial mesh for the mass-spring model a routine placed the nodes for the masses evenly spaced on a rectangular grid, given the number of squares we wanted per side. We made experiments with grids of $3x2, 10x5$ and $20x10$ squares. To aliviate the bias produced by the springs they were placed in a star pattern, thus compensating forces on eight directions.

The force sensor was a DAQ-FT-Gamma, that can measure forces up to 200 Newtons in the $z$ direction (up to 65N in the others) and torques up to 5 Newton-metre. The size of the photographs is $800 \times 600$ pixels. The scale between the real world measurements and the pixels in the image were: $4600 \frac{\text{pixels}}{\text{m}}$ [7] for the sponge and $4350 \frac{\text{pixels}}{\text{m}}$ for the plasticine (they are different because the camera was placed at slightly different positions).

For both sets of experiments, each of the blocks was pushed in different positions:

1) In the middle of its longest side.
2) Close to the corner of its longest side.
3) In the middle of its shortest side.

The first set of data was used to train the model for the respective material. The set of parameters that allowed the model to behave most like the real object, according to the criterion explained in Section V-C, was selected. The other two videos were used to test the quality of the predictions made by the model for the new interactions.

To take into account the cylindrical shape of the actuator, the radial component of the registered force is applied at any vertex that happens to be located within the radius of the robotic finger. For collision detection routines the finger is considered a circular obstacle.

[6] This is read: the value of $F$ goes to zero or $F$ tends to zero.

[7] $\mathrm{m} = metre$

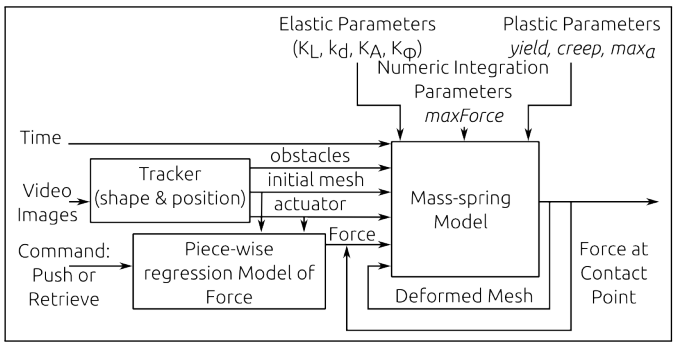



Fig. 12. Detail of the simulation system with shape and force prediction. The position of the actuator is extracted from the image, while the force is predicted by the piece-wise regression model (FP). This information is used by the calibrated mass-spring system (SP) to predict the shape of the object for several frames.

TABLE II
SCORES OF SETS OF PARAMETERS OBTAINED WITH DIFFERENT INTEGRATION SCHEMES

| Scheme | Best | Worst | Average |
|---|---|---|---|
| Euler | $0.88 \pm 0.04$ | $0.44 \pm 0.13$ | $0.75 \pm 0.08$ |
| Verlet | $0.76 \pm 0.19$ | $0.53 \pm 0.18$ | $0.68 \pm 0.19$ |
| Customised | $0.921 \pm 0.008$ | $0.60 \pm 0.13$ | $0.86 \pm 0.02$ |

Where the mark is $\mu(F)$ as explained in Section V-C. Mean $\mu(F)$ and standard deviation across 24 training sesions are reported for the best performing models of every training session, worst model and average model.

### B. Integration Scheme

Sets of 24 training sessions were run, using different combinations of elasticity and plasticity terms. For example, we used damped preservation of length only, preservation of area and angle only, or all of them together. Cases were tried including plastic deformation, and removing the collision resolution routines and the aid of geometric constraints. For this first round of experiments three different integration schemes were used: Euler, Verlet and our customised version. The resulting marks are summarised in Table II.

The traditional equations of movement, estimated with Euler or Verlet show big oscillations and frequently become unstable. Verlet seems to be badly affected by the collision resolution routines and geometric aids, that drastically affect the values of velocity and acceleration. On the other hand, the customised first degree equations suggested in Section IV-C produce smooth movements that more closely resemble the ground truth, thus obtaining better marks. From this group of experiments it was also found that evolutionary learning is as efficient when considering all elastic and plastic terms, as when using only a few of them[8].

### C. Results of Automatic Parameter Calibration

*1) Performance for the Learned Shape Prediction Model:* The performance of the genetic learning algorithm was evaluated using the customised integration scheme and the mass-spring elastic-plastic model. At this point no prediction of the force was included, rather, the reaction forces measured with the force sensor were used. Fig. 13 shows how the quality of the simulation throughout the whole video improved with each generation. Both graphs show that even though the enforcing of geometric constraints with additional routines may produce simulations with lower scores at the begining of the training, they perform just as well after enough generations. These scores reflect the degree to which areas covered by the ground truth and the mesh of the simulation match; however, when looking at the videos produced, it becomes evident that simulations with additional geometric constraints succeed more in keeping the mesh flat.

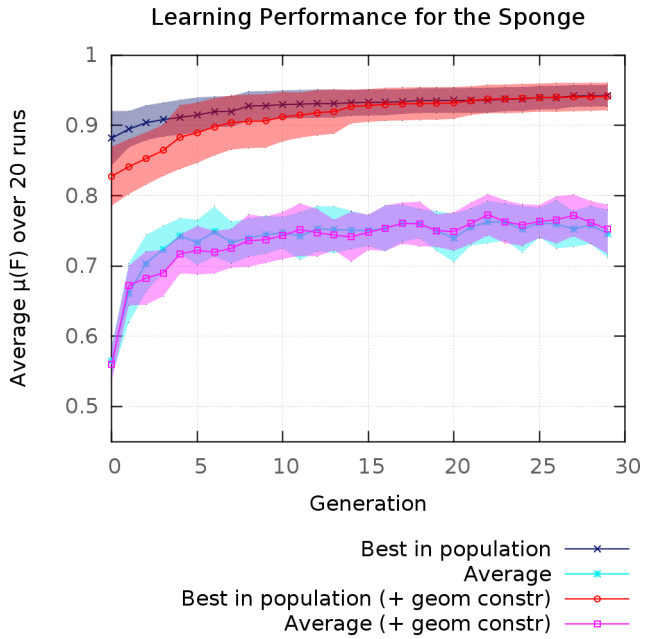
*2) Performance for the Learned Force Prediction Model:* Using the readings of the force sensor, and associating the deformation of the material with the displacement of the robotic finger, a regression curve was fitted to the stress-strain graph for the pushing and retrieving movements. See Fig. 14. As was expected from typical stress-strain diagrams of elastic and plastic phases of materials, a linear regression is characteristic of an elastic material, while the logarithmic regression better fits a plastic material. For the retrieving movement only a linear model was considered, since the shape deformation of the plastic material makes it harder to measure the strain from the visual information. The curves obtained are used to approximately predict the reaction force for unknown interactions. These forces are used for the mass-spring model as in Fig. 12.

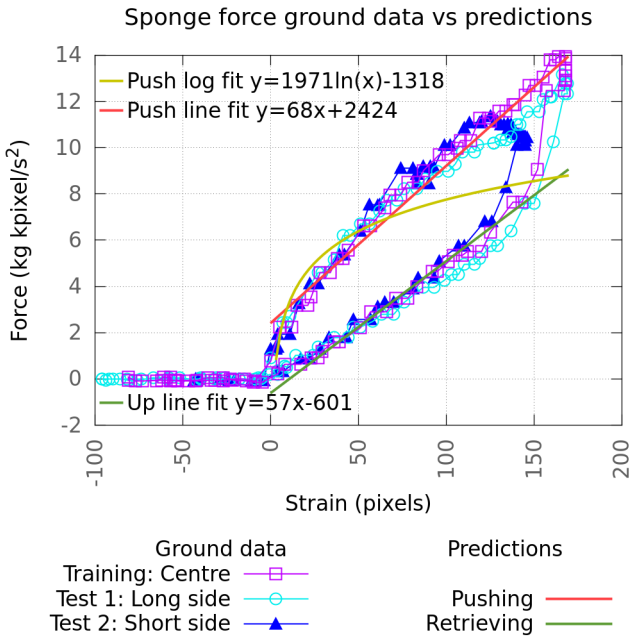
### D. Prediction

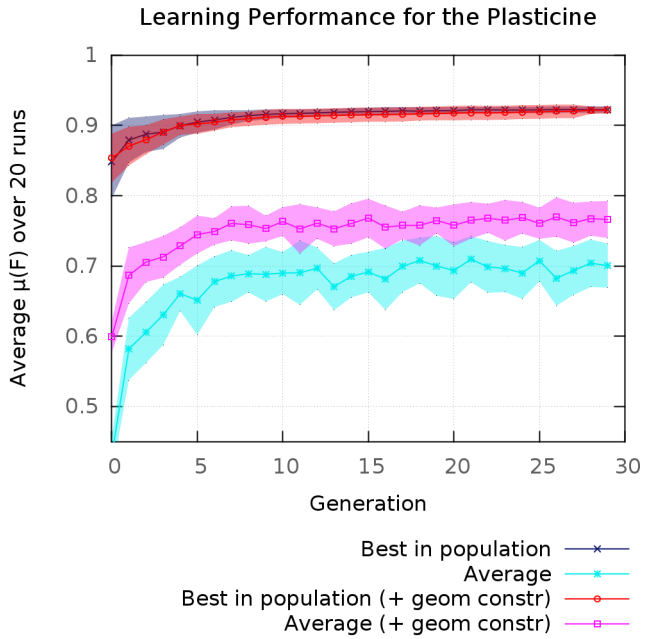
One hundred photographs and force readings were taken every $0.17 \pm 0.007s$ in average. The mass spring model runs with an integration step $h = 0.1s$, where forces that were not measured directly are interpolated. The push down movement lasts approximately 50 frames and the retrieval movement the other 50. The continuous lines in Fig. 14 show the predicted forces, given the amount of compression of the material (strain), while the point-lines show the actual readings. Fig. 15 and Fig. 16 show the actual shapes predicted by the combined FP and SP model.

We also compared the generalisation capabilities of our model against the system proposed in [10]. This system uses a Growing Neural Gas (GNG) Network to calibrate a colour segmentation algorithm. We couldn't use the HSV space as suggested because our videos have lots of shadows which produce holes inside the material regions which cause problems in further stages. We got better results with Luv color space. We clarify that, even though we used the HSV space in our tracking algorithm, we didn't have the same problems because the nodes of our linear snake search for nearby edges and are not affected by shadows in the middle of the object or temporary failures in detecting an edge. Another GNG network is used to select the control points adequate for approximating the contour in the first frame. Later, Cretu et al. use a Neural Gas (NG) algorithm to adjust that fixed number of control points to the contour of the material at every frame. Using the coordinates of the finger, the sensed force and the coordinates of those control points through time, they train a Feedforward Neural Network (FFNN) to predict

[8]The full list of terms is summarised at the end of Section V-B.



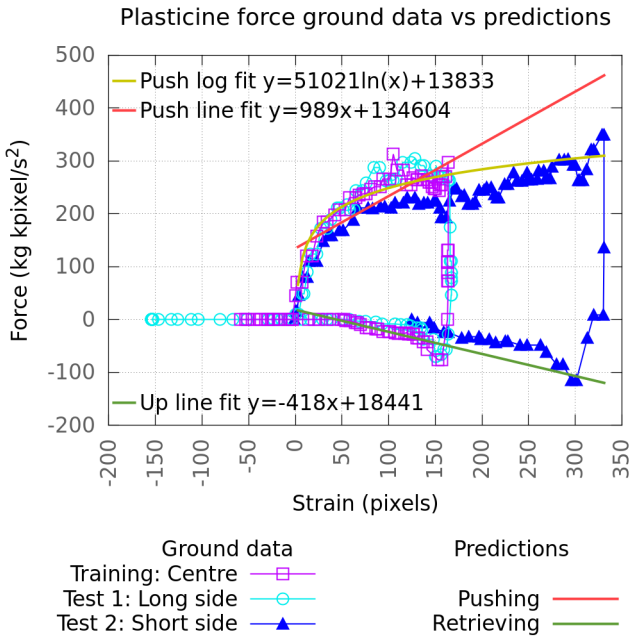




Fig. 13. Final marks of the videos after each generation of the genetic search. The filled areas correspond to the standard deviation $\sigma(\bar{\mu}(F))$, across 15 runs of the learning algorithm. Using the customised integration scheme, the elastic-plastic model with all preservation terms, with and without the aid of geometric recovery routines. The block of material was represented by a mesh of $20 \times 10$ elements.

Fig. 14. The response force during the interaction can be characterised in three phases: pushing, transition and retrieval of the finger. A different regression curve can be used to approximate the force during the pushing and retrieving phases. Linear and logarithmic regressions were suggested for the pushing movement and the best was chosen autonomously. Only linear regression was used while retrieving.

the contour of the material given the position of the finger and sensed force. We reproduced those steps and their training method on our training video. However, when we traid to test the generalisation capabilities of their method to new pushing actions, the weakness of the FFNN method became evident: it got tied to the specific location and type of interaction it was trained with, while our physics based model had no problem at making predictions for new pushing actions. The results with this other method can be seen on Fig. 18.

Since the videos with the plasticine have even greater shadows it was not posible to use their tracking algorithm on them. However, even if we could change that algorithm, we would expect the system to have the same problems and predict the deformation at the middle of the material even if is being pushed somewhere else.

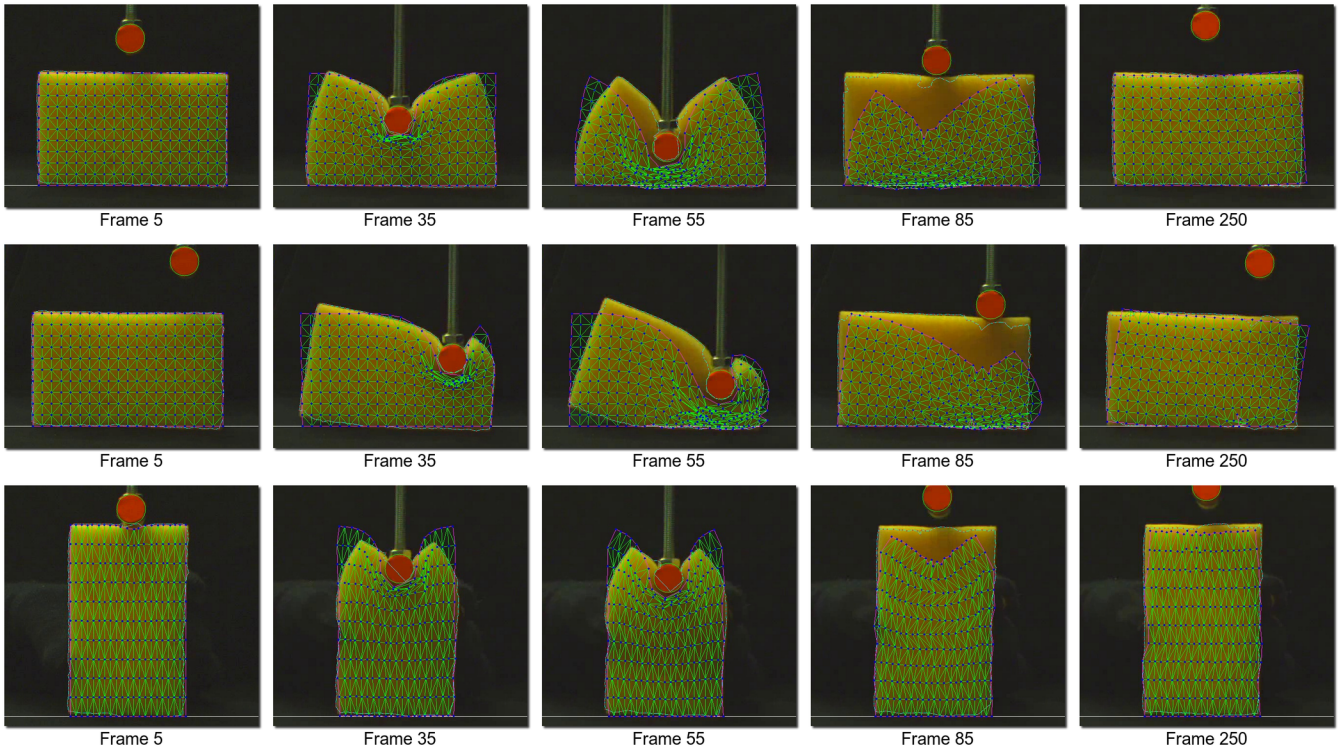



Fig. 15. Simulation of the sponge obtained with parameters $k_L = 3034.48, k_d = 0.000237, k_A = 103.568 k_\phi = 66.774, max\_Force = 1218.81, yield = 0.000434973, creep = 0.00890343, max\_\alpha = 0.45204$. Using the customised integration scheme, the elastic-plastic model with all preservation terms, with the aid of geometric recovery routines and collision detection with the finger. The block of material was represented by a mesh of $20 \times 10$ elements. Top: Training. Middle: Test 1. Bottom: Test 2

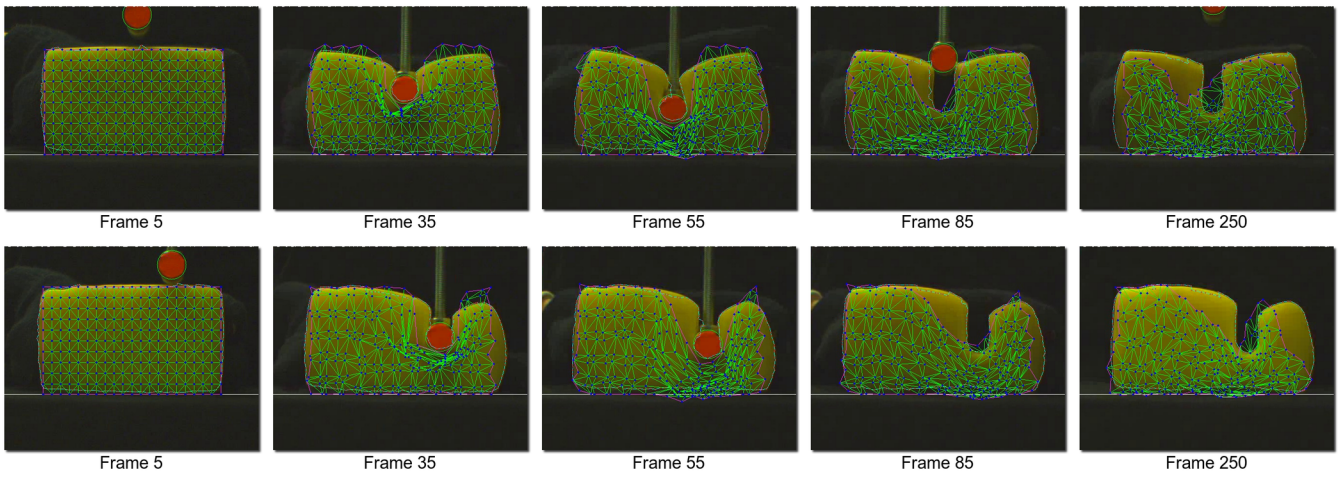



Fig. 16. Simulation of the plasticine obtained with parameters $k_L = 642.328, k_d = 0.562201, k_A = 29.2776 k_\phi = 72.6202, max\_Force = 765.912, yield = 0.0790105, creep = 0.664469, max\_\alpha = 0.48842$. Using the customised integration scheme, the elastic-plastic model with all preservation terms, with the aid of geometric recovery routines and collision detection with the finger. The block of material was represented by a mesh of $20 \times 10$ elements. Top: Training. Middle: Test 1. Bottom: Test 2

### *E. Classification*

The mass-spring models calibrated for the prediction task were used to classify materials through recognition of their behaviour, even when the objects where pushed and deformed at novel locations. The models make predictions as if the material of the object was known. The model that makes the best predictions indicates which type of object is being observed. There are two criteria for selecting an object/material for the video being shown to the system:

1) Per frame: a material is selected for each frame of the interaction according to whichever model made the best prediction for that particular frame, according to the F-score. This test was performed for every frame of the six videos, where the robot finger pushes the block of material.
2) Global: The system makes a vote about which material

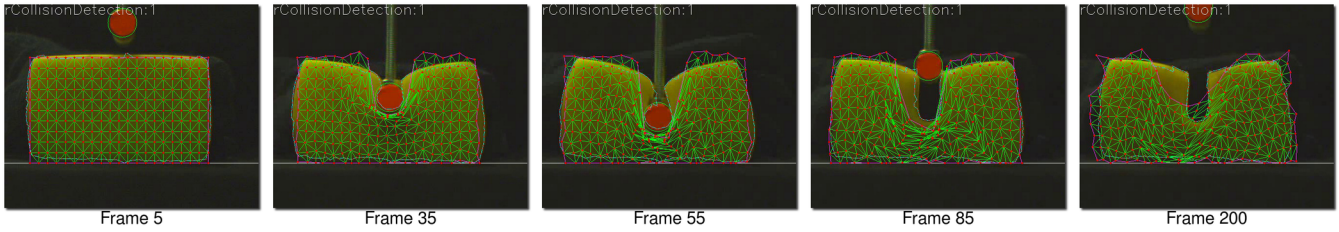



Fig. 17. Simulation of the plasticine obtained with parameters $k_L = 0, k_d = 0, k_A = 28202.1 k_\phi = 34.5491, max\_Force = 780.825, yield = 0.772727, creep = 0.67466, max\_\alpha = 0.656572$. Using the customised integration scheme, the elastic-plastic model with some preservation terms, furtherly tuned by hand, with the aid of geometric recovery routines with the finger. The block of material was represented by a mesh of $20 \times 10$ elements.

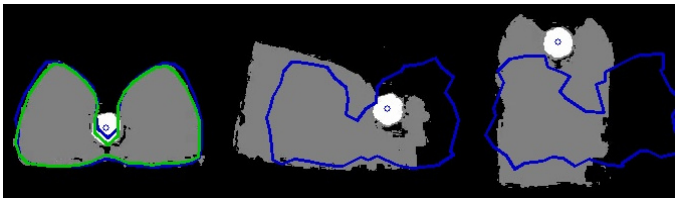


Fig. 18. Neural network approach fits quite well the training data set, but can not generalise to new pushing actions. The green line shows the border used as ground truth, while blue lines show prediction from the FFNN.

TABLE III
CLASSIFICATION PER FRAME

| | Sponge | | | Plasticine | | |
|---|---|---|---|---|---|---|
| Data Set | TP | FN | Precision | TP | FN | Precision |
| Trained with forces on vertices and no collision resolution. | | | | | | |
| training | 167 | 132 | 0.56 | 186 | 113 | 0.62 |
| test1 | 167 | 132 | 0.56 | 271 | 28 | 0.91 |
| test2 | 242 | 57 | 0.81 | 289 | 10 | 0.97 |

* TP and FN measured in number of frames. TP + FN = total number of frames. Final class assigned according to the performance of the models' predictions across the whole video.

the object was made of, for the whole duration of the video, by making use of a weighted score. The difference in the F-score is added for all frames. Frames where the distinction is more noticeable will receive a greater weight.

## VIII. DISCUSSION

We note the following points about the results and the method in general.

### A. The Learning Method

Even though the evolutionary algorithm allowed for a uniform exploration of the logarithmic space of parameters, it failed to find the optimal solution. Better simulations have been obtained by hand tuning some of the parameters.

### B. Using Additional Geometric Constraints

Adding extra routines to enforce the planarity of the mesh had good aesthetic results, however that did not reflect much on the mark for the best videos of each generation during training. Also, when attempting to model the plastic material, the geometric constraints provoked a much larger effect caused by the plasticity term, even if its corresponding constant was small. As a consequence the plastic material also recovered its original shape after some extra frames. It has been seen that, if the constant for linear preservation is set to zero, the problem disappears. Even if the automatic calibration selects these extreme values with probability $\epsilon > 0$, in an attempt to reproduce this finding, their derivatives obtained after applying gaussian noise to their parameters obtain better marks, even though they eventually recover their shapes, and get selected in a better position for the next generation. We hypothesise that humans give greater weight to specific aspects of the simulation, for example: to the beginning, to the instant of greatest deformation and to the final shape. Meanwhile, the evaluation function gives the same weight to all frames in the sequence. It is possible to use the force learning algorithm and the commands given to the robot (push/retrieve) to detect these points, since they are correlated with discontinuities in these other modalities. Future work will evaluate the impact of these modifications on the evaluation function.

### C. Training With Collision Resolution with the Finger vs. Without

It was found that the overall score of the best simulations was similar in both cases, since only geometric similarities are under consideration. However, there was a noticeable difference in the rate of success for the classification problem. If no collision detection with the finger is used during training, the springs react only to the forces and, if the spring coefficients are not adequate, the shape of the mesh will be very different from the shape of the ground truth.

On the other hand, if collision detection is used, the geometric constraints strongly favour a correct configuration of the mesh in the vicinity of the actuator, even though the springs did not react properly. For subsequent frames, it is more likely, when using collision detection, that the neighbours will succeed in propagating the deformation to the rest of the material, thus producing better simulations than without it. However, this lessens the effectiveness of the mass-spring model in distinguishing the materials.

### D. Classification

Table III shows the results for the classification task, frame per frame, for the videos that were used as training and test sets for the prediction task. When the sponge model performed better for a frame of the sponge video it is marked as a true positive, when plasticine does better it is marked as a false negative (since it didn't recognize it as a sponge). The same process is used for the other material. Since the concept was proved with only two materials, true positives and false negatives provide all the information.

It calls for attention that in some of the experiments, like the one illustrated above, the performance of classification for the sponge on the training and test1 sets are the same. We interpret that the mass-spring physics based model generalizes naturally when the force/deformation conditions are similar, even if the place for pushing is different. Between pushing in the middle and pushing closer to the border the qualitative behaviour of the elastic material was no so different for these models; unlike the data from the third set, where the distribution of mass and maximum force/deformation is more different from the ones seen on the training set of the model. It is interesting to confirm that the difference in predictions between the models is bigger here in favour of a better classification.

It is also remarkable that, even though, when modeling the sponge, the parameters for the plasticine were close in performance to the parameters for sponge (thus causing more problems to establish a good classification), the opposite does not happen. That is, the parameters for the sponge are not good at modeling the plasticine on the test sets, thus yielding better results at the classification task. This could happen because the plastic behaviour (that is, the permanent deformation of a spring) gets activated with stronger forces after the $yield$ limit is reached, which only appear when pushing the plasticine. With the small forces feedback by the sponge, the model for the plasticine does not reach its range of plastic behaviour and thus the big differences in behaviour do not manifest.

The best results for classification with the mass-spring model are obtained when the training of the models uses only forces to displace the nodes in the mass-spring model[9], but the combination of force readings and collision resolution is used during classification[10]. This finding agrees with the fact that the objects are more distinguishable from the data obtained from the force sensor, than from the visual data. The fact that the spring constants must respond to the applied forces during the training and not only to the visual appearance is reflected in a better differentiation of the behaviour of the models. When the calibration is more sensitive to the constants of the springs, it is more robust for novel interactions, therefore the difference in quality of the predictions is a better indicator of which material is being used.

### E. Conclusions

In summary, we have presented a framework and specific algorithms for learning the sensorimotor contingencies for the deformable behaviour of objects under robot pushing. We have developed a learning model that learns the parameters of two contingencies that are sequenced, one for kinematic motion to force (FP) and the other for force to deformation (SP). By sequencing these we obtained a compound contingency that predicts deformation given motion. The main properties of the model are as follows. First, the parameters for the model components were learned from data from real objects under robotic manipulation. Second, we have attempted the automatic calibration of a simulation of plasticity. Third, we use regression to predict the reaction forces in the force prediction model. Fourth, we include a new energy term for the preservation of angles in a mass-spring model of shape deformation. Fifth, we employ a regularization algorithm for a linear snake, that increases or decreases the number of its control points, according to the level of detail required to represent the deformed object. Sixth, we provide a detailed evaluation of the dynamic simulation, instead of only during its stable states as in previous work. Finally we showed the application of the learned prediction models to the problem of classifying deformable materials.

There is considerable future work to be done. The first extension is due to the fact that we restricted the deformation model to 2D, and this can easily revert to the 3D meshes as used in Teschner's original work. The second issue is to explore new methods for the learning phase. The current search procedure doesn't always find good solutions for the plastic deformation model. But a more basic problem is that mass-spring models restrict the range of deformations that can be modelled. While the framework presented here is promising, we regard the actual deformation model representation as an open research problem.

[9]Case 2 in Section IV-F

[10]Case 3 in Section IV-F

## ACKNOWLEDGMENT

We gratefully acknowledge funding from the PhD Scholarship program of CONACYT, Mexico and from the European Community's FP7 program, under grant agreements 215181 CogX, and 600918 PaCMan.

## REFERENCES

[1] O. Alkkiomki, V. Kyrki, H. Klviinen, Y. Liu, and H. Handroos, "Complementing visual tracking of moving targets by fusion of tactile sensing," *Robot Auton Syst*, vol. 57, no. 11, pp. 1129–1139, 2009.

[2] V. E. Arriola-Rios, "Learning to predict the behaviour of deformable objects through and for robotic interaction," Ph.D. dissertation, University of Birmingham, 2013.

[3] D. Belter, M. Kopicki, S. Zurek, and J. Wyatt, "Kinematically optimised predictions of object motion," in *IEEE Int Conf on Intell Robot Sys*, 2014, pp. 4422–4427.

[4] D. Bourguignon and M.-P. Cani, "Controlling anisotropy in mass-spring systems," in *Proceedings of the Eleventh Eurographics Workshop on Computer Animation and Simulation*, ser. Springer Computer Science. Springer-Verlag, Berlin, Aug. 2000, pp. 113–123.

[5] S. Burion, F. Conti, A. Petrovskaya, C. Baur, and O. Khatib, "Identifying physical properties of deformable objects by using particle filters," *2008 IEEE Int Conf Robot Autom*, vol. 1, no. 9, pp. 1112–1117, May 2008.

[6] M. Chan, D. Metaxas, and S. Dickinson, "Physics-based tracking of 3d objects in 2d image sequences," in *Proc 12th Int Conf on Patt Recog*, 1994.

[7] F. Conti, O. Khatib, and C. Baur, "Interactive rendering of deformable objects based on a filling sphere modeling approach," in *Proc IEEE Int Conf Robot Autom*, vol. 3, Sept. 2003, pp. 3716 – 3721.

[8] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push planning for object placement on cluttered table surfaces," in *Proc IEEE/RSJ Int Conf Intell Robot Sys*. IEEE, 2011, pp. 4627–4632.

[9] D. Cremers, "Dynamical statistical shape priors for level set-based tracking," *IEEE T Pattern Anal*, vol. 28, no. 8, pp. 1262–1273, Aug. 2006.

[10] A.-M. Cretu, P. Payeur, and E. M. Petriu, "Soft object deformation monitoring and learning for model-based robotic hand manipulation," *IEEE Trans Syst Man Cyb*, vol. 42, no. 3, pp. 740–753, June 2012.

[11] M. Dogar and S. Srinivasa, "Push-grasping with dexterous hands: Mechanics and a method," in *Proc IEEE/RSJ Int Conf Intell Robot Sys*, October 2010, pp. 2123–2130.

[12] D. J. Duff, T. Mörwald, R. Stolkin, and J. Wyatt, "Physical simulation for monocular 3d model based tracking," in *IEEE Int Conf Robot Autom*. IEEE, 2011, pp. 5218–5225.

[13] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning about objects through action-initial steps towards artificial cognition," in *IEEE Int Conf Robot Autom*, vol. 3, 2003.

[14] J. R. Flanagan, M. C. Bowman, and R. S. Johansson, "Control strategies in object manipulation tasks," *2006*, vol. 16, pp. 650–659, 2006.

[15] J. R. Flanagan, P. Vetter, R. S. Johansson, and D. M. Wolpert, "Prediction precedes control in motor learning," *Current Biology*, vol. 13, pp. 146–150, 2003.

[16] B. Frank, R. Schmedding, C. Stachniss, M. Teschner, and W. Burgard, "Learning the elasticity parameters of deformable objects with a manipulation robot." in *Proc IEEE/RSJ Int Conf Intell Robot Sys*, 2010.

[17] B. Frank, C. Stachniss, N. Abdo, and W. Burgard, "Efficient motion planning for manipulation robots in environments with deformable objects." in *Proc IEEE/RSJ Int Conf Intell Robot Sys*, San Francisco, CA, USA, 2011.

[18] S. F. F. Gibson and B. Mirtich, "A survey of deformable modeling in computer graphics," MERL (Mitsubishi Electric Research Laboratory), Tech. Rep., 1997.

[19] M. A. Greminger and B. J. Nelson, "A deformable object tracking algorithm based on the boundary element method that is robust to occlusions and spurious edges," *Int. J. Comput. Vision*, vol. 78, pp. 29–45, June 2008.

[20] M. Haruno, D. Wolpert, and M. Kawato, "Mosaic model for sensorimotor learning and control," *Neural comput*, vol. 13, no. 10, pp. 2201–2220, 2001.

[21] A. Howard and G. Bekey, "Intelligent learning for deformable object manipulation," *Auton Robot*, vol. 9, no. 1, pp. 51–58, AUG 2000.

[22] R. J. Johansson and K. J. Cole, "Sensory-motor coordination during grasping and manipulative actions," *Current Opinion in Neurobiology*, vol. 2, pp. 815–823, 1992.

[23] F. F. Khalil and P. Payeur, "Dexterous robotic manipulation of deformable objects with multi-sensory feedback – a review," in *Robot Manipulators, Trends and Development*, 2010, pp. 587–619, a. Jimenez and B. M. Al Hadithi, Eds.

[24] M. Kopicki, S. Zurek, T. Mörwald, and J. Wyatt, "Learning to predict how rigid objects behave under simple manipulation," in *IEEE Int Conf Robot Autom*, 2011.

[25] M. Kopicki, S. Zurek, R. Stolkin, T. Moerwald, and J. L. Wyatt, "Learning modular and transferable forward models of the motions of push manipulated objects," *Auton Robot*, pp. 1–22, 2016.

[26] O. Kroemer and J. Peters, "Predicting object interactions from contact distributions," in *Proc IEEE/RSJ Int Conf Intell Robot Sys*, 2014.

[27] K. Lynch, "The mechanics of fine manipulation by pushing," in *IEEE Int Conf Robot Autom*, 1992, pp. 2269–2276.

[28] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning," *Int J Robot Res*, vol. 15, pp. 533–556, 1996.

[29] S. Malassiotis and M. G. Strintzis, "Tracking textured deformable objects using a finite-element mesh," *IEEE T Circ Syst Vid*, vol. 8, no. 6, pp. 756–774, 1998.

[30] M. T. Mason, *Mechanics of robotic manipulation*. MIT press, 2001.

[31] T. McInerney and D. Terzopoulos, "Deformable models in medical image analysis: a survey." *Med Image Anal*, vol. 1, no. 2, pp. 91–108, 1996.

[32] S.-M. Menet and Medioni, "B-snakes: Implementation and application to stereo," in *Artificial Intelligence and Computer Vision*, Y. A. Feldman and A. Bruckstein, Eds., vol. Proceedings of the Seventh Israeli Conference. Elsevier Science Publishers B. V., 1991.

[33] T. Meriçli, M. Veloso, and H. L. Akın, "Push-manipulation of complex passive mobile objects using experimentally acquired motion models," *Auton Robot*, vol. 38, no. 3, pp. 317–329, 2014.

[34] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. D. Raedt, "Learning relational affordance models for robots in multi-object manipulation tasks," in *Proc IEEE Int Conf Robot Autom*, 2012, pp. 4373–4378.

[35] J. Montagnat, H. Delingette, and N. Ayache, "A review of deformable surfaces: topology, geometry and deformation," *Image Vision Comput*, vol. 19, no. 14, pp. 1023–1040, 2001.

[36] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory-motor coordination to imitation," *IEEE T Robotics*, vol. 24, no. 1, pp. 15–26, 2008.

[37] P. Moore and D. Molloy, "A survey of computer-based deformable models," in *IMVIP 2007: Int Mach Vision Image Proc Conf*, J. McDonald, C. Markham, and J. Ghent, Eds. IEEE, 2007, Proceedings Paper, pp. 55–64.

[38] D. Morris and K. Salisbury, "Automatic preparation, calibration, and simulation of deformable objects," *Computer Methods In Biomechanics and Biomedical Engineering*, vol. 11, no. 3, pp. 263–279, 2008.

[39] T. Mörwald, M. Kopicki, R. Stolkin, J. Wyatt, S. Zurek, M. Zillich, and M. Vincze, "Predicting the unobservable visual 3d tracking with a probabilistic motion model," in *IEEE Int Conf Robot Autom*. IEEE, 2011, pp. 1849–1855.

[40] A. Nealen, M. Mueller, R. Keiser, E. Boxerman, and M. Carlson, "Physically based deformable models in computer graphics," *Comput. Graph. Forum*, vol. 25, no. 4, pp. 809–836, December 2006, 18th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2005).

[41] M. A. Peshkin and A. C. Sanderson, "The motion of a pushed, sliding workpiece," *IEEE J Robot Autom*, vol. 4, pp. 569–598, 1988.

[42] T.-H. Pham, A. Kheddar, A. Qammaz, and A. Argyros, "Capturing and reproducing hand-object interactions through vision-based force sensing," in *IEEE Int Conf on Computer Vision, Workshops (ICCVW 2015)*, 2015.

[43] J. Scholz and M. Stilman, "Combining motion planning and optimization for flexible robot manipulation," in *10th IEEE-RAS Int Conf on Humanoid Robots*. IEEE, 2010, pp. 80–85.

[44] M. Stilman and J. J. Kuffner, "Planning among movable obstacles with artificial constraints," *Int J Robot Res*, vol. 27, no. 12, pp. 1296–1307, 2008.

[45] A. Stoytchev, "Learning the affordances of tools using a behavior-grounded approach," in *LNAI 4760* (E. Rome et al.), 2008, pp. 140–158.

[46] M. Teschner, B. Heidelberger, M. Muller, and M. Gross, "A versatile and robust model for geometrically complex deformable solids," in *Proceedings of Computer Graphics Int. (CGI'04)*, Crete, Greece, June 16-19 2004, pp. 312–319.

[47] C. Zito, R. Stolkin, M. Kopicki, and J. L. Wyatt, "Two-level RRT planning for robotic push manipulation," in *Proc IEEE/RSJ Int Conf Intell Robot Sys*. IEEE, 2012, pp. 678–685.


**Veronica E. Arriola-Rios** has recently obtained her PhD in Computer Science from the University of Birmingham. She works as full time lecturer at the National Autonomous University of Mexico (UNAM). Veronica holds undergraduate degrees in Physics and Computer Science, and a masters degree in Computer Sciences, in the area of Computer Graphics and Virtual Environments. Her research interests include artificial inteligence, cognitive robotics and natural cognition.

**Jeremy L. Wyatt** is Professor of Robotics and Artificial Intelligence at the University of Birmingham. He obtained a BA in Theology from the University of Bristol, an MSc in Knowledge Based Systems from the University of Sussex, and his Ph.D. in Artificial Intelligence from the University of Edinburgh in 1997. He was a Leverhulme Fellow from 2006-8. He has won two best paper prizes, published more than 100 refereed articles, led two international research projects on robot planning and learning (CogX) and robot manipulation (PaCMan), and edited three books. His interests include machine learning, planning, architectures, vision, mobile robotics and robot manipulation.