A Multi-Modal Model for Prediction and Classification of Object Deformation during Robotic Manipulation

Veronica E. Arriola-Rios, Jeremy L. Wyatt

***Abstract*—In this paper we present a new framework for learning generative models of object deformation under robotic action. The model is multi-modal in that it is based on integrating force and visual information. Our framework goes beyond previous techniques by simultaneously 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 them all in a single learning model framework, and none has offered classification from a generative predictable model.**

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

I. Introduction

A Major challenge in robotic 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, and these include objects that are elastic, plastic, breakable, tearable, and spreadable. This work is based on the premise that in order to plan manipulations of deformable objects, the sensorimotor contingencies governing object deformation are required. These sensorimotor contingencies can then be used to feedback on themselves to predict over long timescales. Without such models planners cannot reason about the effects of actions, or sequences of actions, to achieve desired effects.

There is a wide range of models of deformation used in engineering, that are capable of producing predictions, such as finite element models, but these typically require careful optimisation by hand over many weeks or months. In graphics there are many models used for generating qualitatively plausible simulations of objects, but these are not necessarily easy to tune to real data. By contrast robots must perform system identification of their model parameters from data, online, in reasonable amounts of training time. This includes building models from the usually very limited sensing typically available, such as partial visual views of an object, and force-torque monitoring at contact points. Thus tackling this problem in robotics adds additional difficulties to an already challenging problem.

Previous work on learning sensorimotor contingencies for deformable objects in robotics has been suprisingly limited. Table I summarises the features of the major efforts to date. In each case critical 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 a framework for solving all these problems simultaneously. The key novel contribution we make is to bring together a model of what forces the robot will feel at a contact point with a model of how an object will deform under such applied forces over time. This formulation allows the robot to make predictions about the type of object, and thus it's future behaviour using the resistive forces and the object deformations Fig. 1 and Fig. 12. A second critical contribution is that our model is both generative and generalisable with respect to contact point. Finally the third major feature of the model is that it is adaptive: a set of model parameters is learned for each material presented from data, thus creating a model library of materials. This generative property enables both prediction suitable for use in planning, and also classification of materials. The overall model structure is summarised in Fig. 2. Essentially the approach is to use visual tracking of shape deformations to create feedback that is used to train the parameters of a model (Model A) that predicts deformation when coupled with contact forces. The contact forces are themselves predicted using a second model (Model B). In the specific implementation of our system we use a mass spring system for Model A, and a regression model for Model B. These models are learned from the same data, and

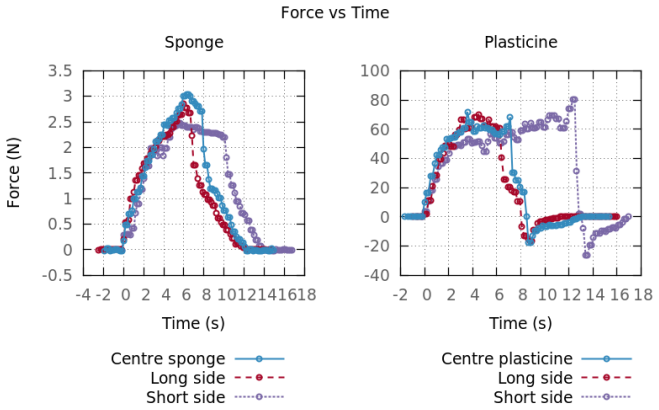

Fig. 1. Force information acquired while working with a block of sponge (elastic) and a piece of plasticine (plastic). Notice the difference in magnitude between the force used with the sponge and the plasticine.

are both indexable by a known material type from the library. It is important to note, however, that the model structure is separate from the specific model forms used, and that other model types could be substituted for one or both stages.

In summary the two stage model we present gives us the ability to solve *simultaneously* for the first time the following problems. Our model 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 an overview of the model in more detail (Section II); related work (Section III); describe the model (Section IV); describe how the model parameters are learned from real data (Section V); and the experiments on prediction and classification with both elastic and plastic objects (Section VII); finishing with a short discussion (Section VIII).

II. Model Overview

As has been mentioned above there are several choices made in our modeling approach. The key choice is that the models are generative, which means that the models can be used to generate predicted sequences of behaviour. The specific approach taken here separates the predictive model out into components in two ways (see Figure 2). First we decompose models according to the modality of the information. In this case it is important to note that deformation has a particular causal structure related to the modes of sensing that are most easily employed. The forces and deformations can be measured separately, using force-torque and visual sensors respectively. The sensed forces are the cause of the observed deformation, and this can be exploited by sequencing the models by mode. Thus our approach can be seen as a sensorimotor contingency in which two contigencies are sequenced to produce an overall prediction. The input to the first predictor is the planned motion of the robot (or the joint torques), and this predictor predicts the resulting sequence of contact forces. The second predictor takes this sequence of forces as input and predicts a deformation sequence. Both models must be conditioned on the material and the object. This is where we decompose the model space in the second way, by specialising models to particular material and object combinations. This follows the principle of modular motor learning [11], [8], 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 we aim 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.

In each case the models can be run in two modes, prediction and classification. In prediction mode the models make predictions based on a sequence of robot finger positions. 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. In filtering mode the predictions of different models are compared to the actual outcomes, and this is used to classify the object as being one of the modular models in the set of models. Since the model is multi-modal we can compare both force and deformation data to the corresponding model predictions.

The training of the model is performed offline using ground truthed data. For clarity the algorithm is given in three phases: training, simulation (i.e. the prediction phase), and classification (i.e. matching prediction to new data). In testing the model new data is used, where this is typically new deformation actions on an object of known material and approximate shape. Having given an overview of the model we now relate our approach to those in the literature, before proceeding to describe the details of the spring-mass model of deformation (Section IV), and then the regression model of contact forces (Section VI).

III. Background

This work focuses on prediction of response forces and of the deformation of the shape over several frames of elastic and plastic objects when they are pushed and released by a robotic finger. Acquired models are used to classify materials through recognition of trained samples, even when the interactions are novel. Even though there is abundant literature in the industrial robotics area for 1D and 2D materials (e.g. strings, hair, metallic sheets and textiles), there is little research about predicting the deformation of 3D deformable objects [10].

First, Howard and Bekey [9] 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 in 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 robotic 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 entry of 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. [17] proposed an enhanced 3D mass-spring model, whose constraints are expressed as potential energy

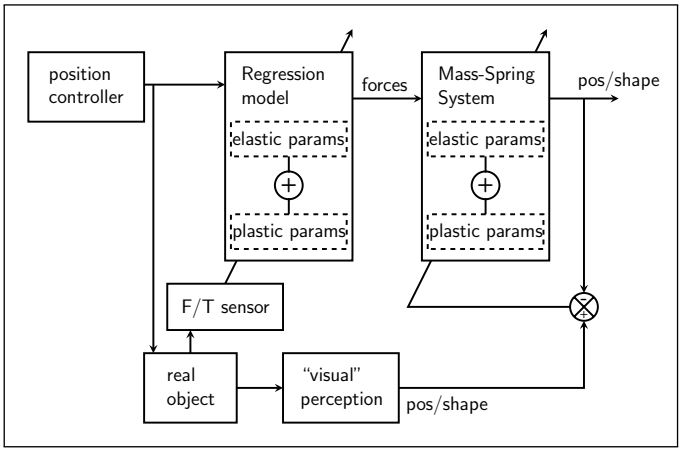

Fig. 2. During training time, the system learns to predict the object's shape and reaction forces from real world data.

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*

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 [15] 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 on this work, by changing the FEM model with force and 2D visual data gathered from real materials. Furthermore, we analysed the quality of the predictions while the deformations take place.

Frank et al. [5] use a force sensor and a bumblebee stereo 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 [6].

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

Other interesting variants are: Conti et al. [4] introduce six-degrees 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 [3], 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 preferably 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 unseen sets of forces, their predictions were evaluated only for the next frame.

Additionally, there is a vast amount of literature on modeling of deformable shapes and deformation processes [7], [12], [13], [14], [16], particularly within the area of computer graphics. However, these types of models have not been successfully applied to robotic tasks beyond what was presented above.

IV. A MASS-SPRING MODEL OF DEFORMATION

A mass-spring model is a very well established abstraction used frequently to simulate the behaviour of some deformable objects. As a physics based model, once its parameters are properly assigned, 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 interacting forces are know. 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 for the particular object. For this reason, there are variations that address specific issues, like [2] and [17]. In this work we focus on the machinery required to achieve this automatic calibration of a physics based model and its integration to a

[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 succession of states of equilibrium.

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 [17].

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, that 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 over 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 positions of equilibrium, 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:

$$F = m\frac{\partial^2 p}{\partial t^2} \tag{1}$$

where t is time. However, it is possible to modify this equation according to need if what we need is to model a different behaviour.

The previous 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 [16]. 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 value 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):

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

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

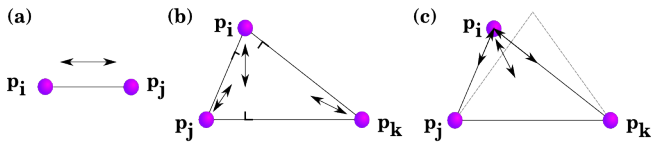

Fig. 3. Spring Forces. Recovery of: (a) Length, forces act along the spring. (b) Area, forces act along the heights of the triangles. (c) Angle, masses slide trying to restore the angle.

A. Elastic Deformation

The mass-spring model proposed by Teschner et al. in [17], 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. 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 compensate for the lack of a preservation of volume, which tends to avoid distortions of the graph in the original model, we derived a new term for the preservation of the angles in every triangle ~~was included~~. The relevant equations are presented in the following paragraphs, while detailed derivations can be found in [1].

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:

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

where k is a proportionality constant ~~(3)~~ 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 oscilatory 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 [17]:

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

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:

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

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. 3(a)]. Teschner normalised the value dividing by D_0 to make the elasticity constants scale independent [17]. Morris [15] 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:

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

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 ~~didn't find it helpful to add damping to the preservation of areas, it is avoided here.~~ found that the use of damping for the preservation of areas does not improve significantly the stability of the simulation, it is not included here either. Therefore the force $\vec{F}^i_A$ is derived as follows:

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

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

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

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 [15], but keeping the area normalisation constant[2]. It becomes evident that the forces pull along the heights of the triangles [Fig. 3(b)]. Then the magnitude of the force is given by:

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

While the rest becomes:

$$\vec{F}_A(\vec{p}_i) = k_A \cdot forcemag_A(\vec{p}_i)\cdot forcedir_A(\vec{p}_i)$$
$$forcedir_A(\vec{p}_i) = \frac{\vec{F}_A(\vec{p}_i)}{|\vec{F}_A(\vec{p}_i)|} = \frac{areagradient(\vec{p}_i)}{|areagradient(\vec{p}_i)|}$$
$$areagradient(\vec{p}_i) = (\vec{p}_i - \vec{p}_j) - \left((\vec{p}_k - \vec{p}_j)\cdot\frac{(\vec{p}_k - \vec{p}_j)\cdot(\vec{p}_i - \vec{p}_j)}{(\vec{p}_k - \vec{p}_j)\cdot(\vec{p}_k - \vec{p}_j)}\right) \quad (12)$$

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. 3(c)] gives an idea of how these forces look. Here the energy depends on the difference ~~of the angles between adjacent edges.~~ between the current angle, between adjacent edges, and the angle between them at the equilibrium position.

Energy[3]:

$$E_\varphi(\varphi) = \frac{1}{2}k_\varphi(\varphi - \varphi_0)^2 \quad (13)$$
$$\varphi(p_i,p_j,p_k) = \arccos\left(\frac{(p_j - p_i)\cdot(p_k - p_i)}{\|p_j - p_i\|\,\|p_k - p_i\|}\right)^2$$

Where φ is the angle between adjacent edges, E_φ is the energy associated to changes in the angle, k_φ is the corresponding stiffness constant and the p_is 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. 3(c)]. Therefore, it helps to recover a similar triangle, but it also 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:

$$F_\varphi(p_i) = k_\varphi(\varphi - \varphi_0)\frac{\partial\varphi}{\partial p_i} \quad (14)$$
$$\frac{\partial\varphi}{\partial p_i} = \frac{\partial}{\partial p_i}arccos\left(\frac{(\vec{p}_j - \vec{p}_i)\cdot(\vec{p}_k - \vec{p}_i)}{\|\vec{p}_j - \vec{p}_i\|\,\|\vec{p}_k - \vec{p}_i\|}\right) \quad (15)$$
$$(16)$$

[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_φ by the lengths of the edges, but it hasn't improved the performance of the model.

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_α 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. 4. Algorithm to calculate the permanent deformation of an edge.

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.

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 in terms of a constant of compression α, 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 α 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 the traditional Newton equations with a numeric integration scheme like Euler or Verlet, used twice to solve the second order differential equiations, we made the ~~velocities proportional to the forces:~~ force proportional to the velocity:

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

and applied Euler only once to solve for p:

$$\vec{p}(t+h) = \vec{p}(t) + h\frac{\vec{F(t)}}{m} \tag{20}$$

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 not inertia in the oscillation. When experimenting with different integration steps, force values were interpolated between measurements.

D. Collision Detection

In the scenario, rigid objects (like the robotic finger and the table) are treated as geometric obstacles. When the simulation of the behaviour of the mesh, that models the image of the sponge, is carried on, the predicted positions of the vertices may indicate that the mesh penetrates the obstacles. This should be interpreted as a collision between the sponge and the obstacle. Since the sponge can not penetrate an obstacle, the overlapping between geometries must be solved, and its physical consequences must affect the simulation. For example: if vertices of the mesh overlap the table, the sponge will remain on top of the table, with its material being more compact, which will cause the response force propagated towards the finger to be increased.

In order to model this, two types of solid obstacles are considered: circles (for the finger) and a table line that can not be crossed. For the table obstacle, the collision of parts of the sponge with it is roughly solved by pushing vertices of the mesh of the sponge back to the border of the table. The springs will eventually propagate the effect to the neighbours. For the circle the resolution is in two stages: vertices and edges. A vertex will be pushed out to a distance ϵ of the finger, in the direction of the radius. For edges, the common algorithm in Algorithm 2 is used to estimate the shortest distance between the centre of the circle and the edge. Given this distance, the algorithm in Algorithm 3 indicates how and in which direction the edge must be displaced.

E. Geometric Constraints

Even with the addition of the term for preservation of angles, the discretized mass-spring forces can cause triangles of the mesh to overlap each other and be reversed or to be flattened. The option of giving additional aid through small geometric subroutines was evaluated as well. There are two geometric constraints that the mesh is enforced to maintain:

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

1: Be a, b the ends of the edge $\overline{AB}$, c the external point, and x the closest point to c in the line.
2: Be the vectors $\vec{AB} = b - a$ and $\vec{AC} = c - a$.
3: The projection of $\vec{AC}$ over $\vec{AB}$ is $\|x\| = \dfrac{\vec{AC} \cdot \vec{AB}}{\|\vec{AB}\|}$
4: **if** $\|x\| < 0$ **then**
5: $closest \leftarrow a$
6: **else**
7: **if** $\|x\| > \|\vec{AB}\|$ **then**
8: $closest \leftarrow b$
9: **else**
10: $closest \leftarrow a + \|x\| \cdot \dfrac{\vec{AB}}{\|\vec{AB}\|}$
11: **end if**
12: **end if**

Fig. 5. Algorithm to find the closest point on a segment to any other point.

Algorithm 3: *Push Edge Out*

1: Be a, b 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.
2: **if** $\|(closest - c)\| < radius$ (There is an intersection) **then**
3: $displacement = (radius - \|(closest - c)\| + \epsilon) * \dfrac{(closest - c)}{\|(closest - c)\|}$
4: $a \leftarrow a + displacement$
5: $b \leftarrow b + displacement$
6: **end if**

Fig. 6. Algorithm to push an edge out of a circular obstacle.

1) For a reversed triangle, (one vertex crossed over an edge): the vertex gets pushed beyond 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.
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 Steps Prediction

Given the initial shape of the object and a set of parameters, the mass-spring system allows to make predictions of the deformation of an object for various interactions. Given the procedures explained in the 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 trespass invalidly 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) **Both.** The previous two are 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.

[4] 3/4 this distance was used in our implementation.
[5] An area of 2.0 units was good enough.

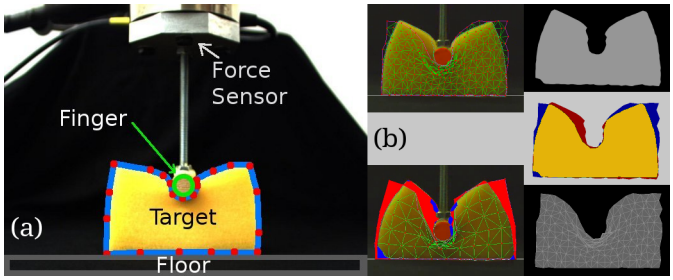

Fig. 7. (a) Experimental set up. A robotic finger pushes a block of deformable material over a table. A camera registers the events of the plane where the main deformations take place. (b) Evaluation function. Left top: Mesh over real block of material. Left bottom: Tipical error of the model vs the ground truth (red) and tipical 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.

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

The next stage ~~of~~ 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 the Deformation

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. Geometrically, the contour is represented by a polygon with hundreds of sides, which can also be called a *linear spline*. The tracking of the deformable target in the ground truth has two stages:

1) The target object is segmented from the environment. The Canny algorithm is used to extract the edges of the image. Next, a simple colour segmentation is used to

Algorithm 4: *Regularise Linear Spline*

Be L 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. 8. Algorithm used to regularise the number of elements of a linear spline, in accordance with the level of detail required.

remove edges that are not likely to belong to the object of interest.

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. 7(a)]. If there is no edge within that area, the control point remains were 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

The evolutionary algorithm listed in Algorithm 5 was used to search the space of parameters 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_φ 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_α. *(All springs will have the same values to favour generalisation.)*
- The integration time step Δt. *(It has been set to 0.1 seconds, even though values as small as 0.01 were tried as well.)*
- An upper limit for the magnitude of the total estimated forces per vertex max_Force.

Algorithm 5: *Evolutionary Search*

Be N the number of sets of parameters to be evaluated per generation and M the number of generations to try.
Let $\alpha + \beta + \gamma + \zeta = N$ denote four portions of the N elements.
Be $\sigma_{max} > \sigma_{min}$ standard deviations for a Gaussian
Let $\Delta\sigma \leftarrow (\sigma_{max} - \sigma_{min})/M$.
for $i = 1 \rightarrow M$ **do**
  Run the N simulations and compare prediction with the ground truth (real object), by comparing the area occupied by the real object with the area occupied by the mesh of the simulation [Fig. 7(b)].
  Keep the best α of all simulations.
  $\sigma \leftarrow \sigma - \Delta\sigma$
  Generate β by adding random Gaussian noise, with standard deviation σ, to the best candidates.
  Generate γ by selecting values of the parameters from two different sets at random, after eliminating the ϵ worst.
  for ζ 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. 9. Evolutionary Search for Automatic Calibration

- 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 shapes, 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 among 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; and the *false negatives* FN, where there is material, but the prediction said there wouldn't [Fig. 7(b)]. Given that maintaining a record of the true negatives (the empty/background space around the material) is mostly irrelevant and highly inefficient, the mark for a whole video was derived from the following

quantities:

$$Precision = \frac{TP}{TP+FP}, \tag{21}$$

$$Recall = \frac{TP}{TP+FN} \tag{22}$$

From which the harmonic average F is derived:

$$F = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{1}{1 + \frac{FN+FP}{2TP}} \tag{23}$$

This measurement 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 mark for the video is the mean over all the frames so far, $\mu(F)$. If the predictions for all the frames were perfect, $\mu(F)$ would be equal to 1; the closest 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. Models of Force Learning

Additionally and independently from the mass-spring system, it is possible to make predictions about the expected reaction force of the material, by using information from the stress-strain diagram obtained from the training data set [Fig. 12]. The curve in this diagram is 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:

$$y = mx + y_0 + Error \tag{24}$$

$$m = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \tag{25}$$

$$y_0 = \frac{\sum y_i}{n} - y_0 \frac{\sum x_i}{n} \tag{26}$$

$$Error = \sum (y_i - (mx_i + y0))^2 \tag{27}$$

Where y is the dependent variable, x the independent one, m is the sloop 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 the automatic calibration of the pushing phase both types of curves are considered. To determine the place of the discontinuities between the segmentes, several candidates are calculated. Each candidate is obtained by adjusting a curve through different amounts of data points and calculating the error. Beginning with a small number of points (e.g. 5), as new measurements are taken, new curves are adjusted until the corresponding normalized error increases beyond a threshold value. The type of curve that can cover more points with a low fit error is adopted for the model. This model is used to predict the forces for the mass-spring simulation Fig. 10 and Fig. 12.

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

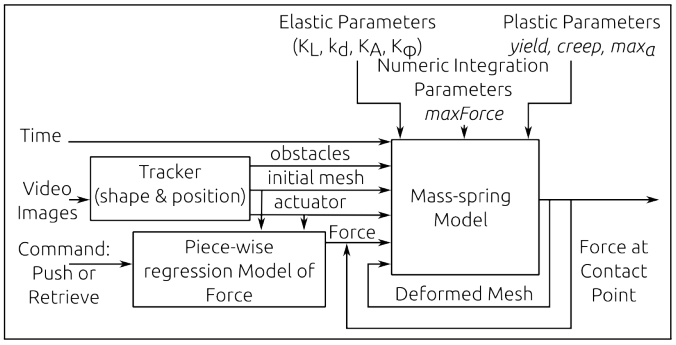

Fig. 10. The simulation time system with shape and force prediction. The position of the actuator is extracted from the image, while the force is predicted by a piece-wise regression model. This information is used by the calibrated mass-spring system to predict the shape of the object for several frames.

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. 7(a)]. 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 800×600 pixels. The scale between the real world measurements and the pixels in the image were: $4600\frac{\mathrm{pixels}}{\mathrm{m}}$ [7] for the sponge and $4350\frac{\mathrm{pixels}}{\mathrm{m}}$ 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 more 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 on 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.

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

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

Scheme	Best	Worst	Average
Euler	0.88 ± 0.04	0.44 ± 0.13	0.75 ± 0.08
Verlet	0.76 ± 0.19	0.53 ± 0.18	0.68 ± 0.19
Customised	0.921 ± 0.008	0.60 ± 0.13	0.86 ± 0.02

Where the mark is $\mu(F)$ as explained in Section V-C.

B. Integration Scheme

Sets of 24 training sessions were run, using different combinations of elasticity and plasticity terms. For example: using damped preservation of length only, using preservation of area and angle only or all of them together; including plastic deformation or removing the collision resolution routines and the aid of geometric constraints. For this first rounds 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 affect drastically the values of the velocity and acceleration. On the contrary, the customised first degree equations suggested in Section IV-C produce smooth movements that resemble more the ground truth, thus obtaining better marks. From this group of experiments it was also found that the evolutive learning is as efficient when considering all elastic and plastic terms, as when using only a few of them.

C. Parameters Calibrated Automatically

1) Performance of the Learning Algorithm for the Mass-spring 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. 11 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 aid routines may produce simulations with lower marks at the begining of the training, they perform just as well after enough generations. These marks reflect the degree at 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 geometric aids succeed more in keeping the mesh flat.

2) Prediction of the Force: 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. 12. As it 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 fits better to a plastic one. 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 the diagram of Fig. 10.

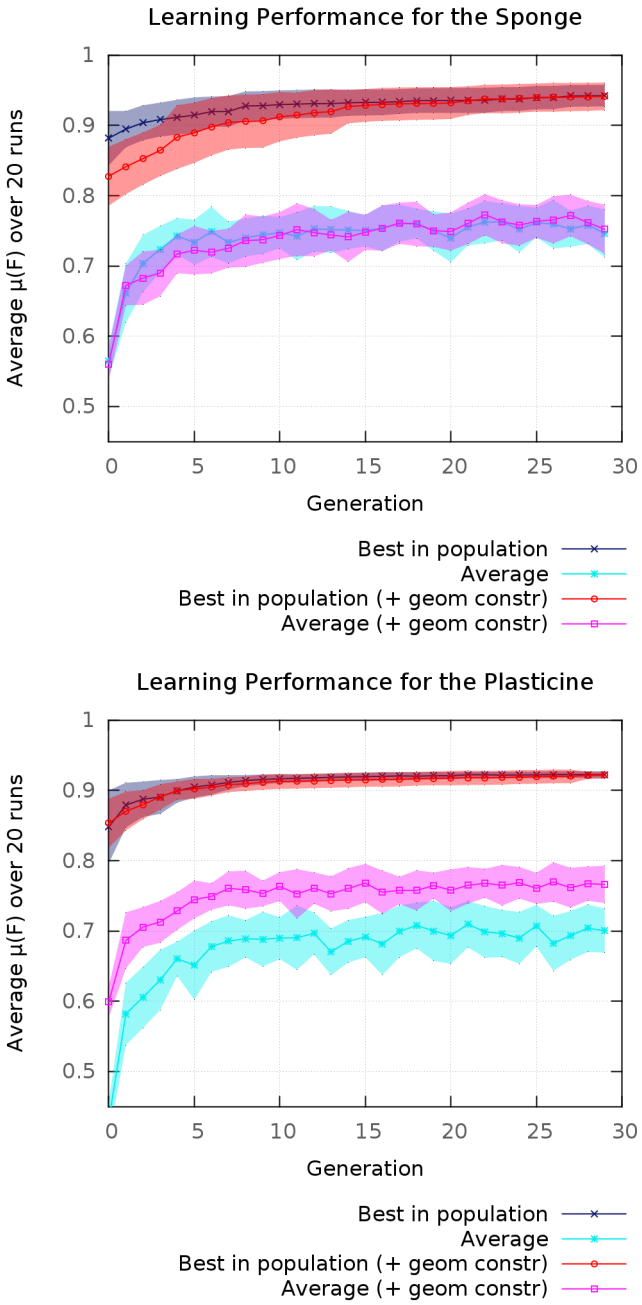

Fig. 11. Final marks of the videos after each generation of the genetic search. The filled areas correspond to the standard deviation $\sigma(\bar{\mu}(F))$, across 15 runs of the learning algorithm. Using the customised integration scheme, the elastic-plastic model with all preservation terms, with and without the aid of geometric recovery routines. The block of material was represented by a mesh of 20×10 elements.

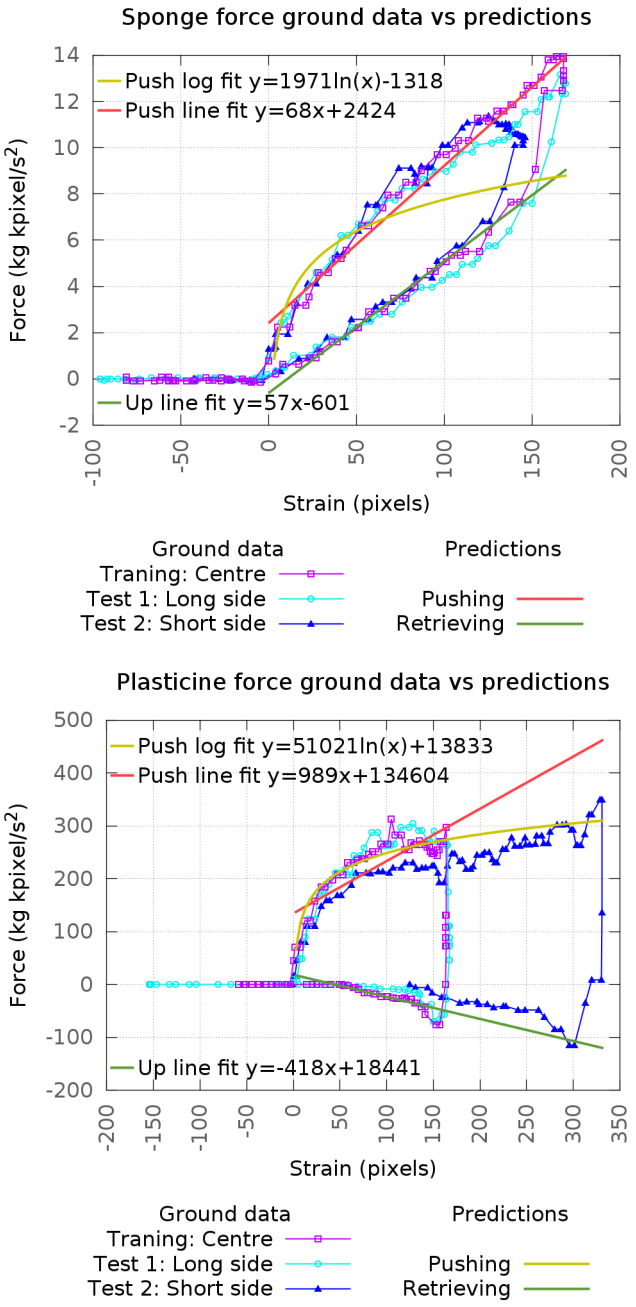

Fig. 12. The response force during the interaction can be characterised in three phases: pushing, transition and retrieval of the finger. A different regression curve can be used to approximate the force during the pushing and retrieving phases. Linear and logarithmic regressions were suggested for the pushing movement and the best was chosen autonomously. Only linear regression was used while retrieving.

D. Prediction

One hundred photographs and force readings where taken every $0.17 \pm 0.007s$ in average. The mass spring model runs with an integration step of $0.1s$, where forces that were not measured directly are interpolated. The push down movement lasts approximately 50 frames and the retrieval, the other 50. The continuous lines in Fig. 12 show the predicted forces, given the amount of compression of the material (strain), while the point-lines show the actual readings. Fig. 13 and Fig. 14 show the shapes predicted by the mesh.

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

E. Classification

The calibrated mass-spring models were used to classify objects through recognition of their behaviour, even for novel interactions. The models make predictions as if the object was known. The model that makes the best predictions indicates which object is being observed. There are two criteria for selecting an object:

1) Per frame: an object is selected for each frame of the interaction. See Table III.
2) Global: The difference in the marks 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 improve the quality of the simulations as much as it could. Better simulations have been obtained by hand tunning some of the parameters.

B. Using 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 provoqued a much more notorious effect comming from any 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 the 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

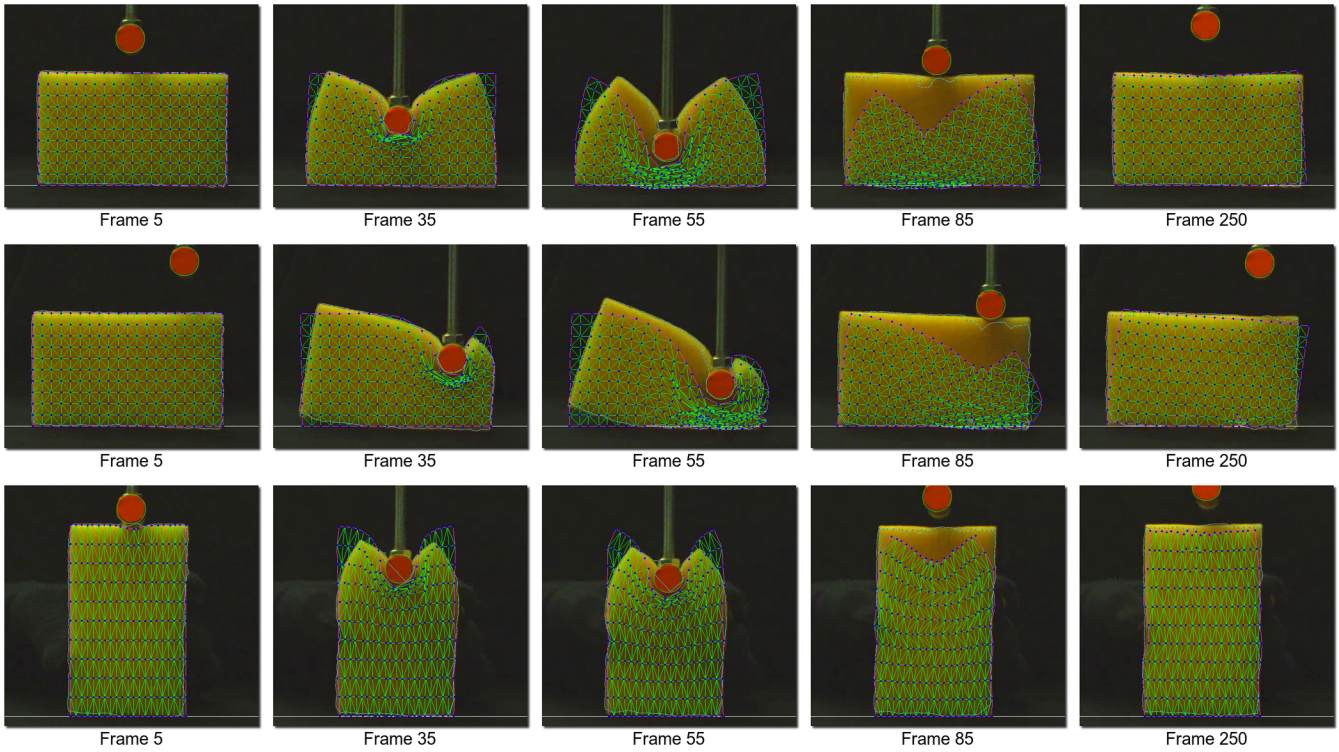

Fig. 13. Simulation of the sponge obtained with parameters $k_L = 3034.48, k_d = 0.000237, k_A = 103.568 k_\phi = 66.774, max\_Force = 1218.81, yield = 0.000434973, creep = 0.00890343, max\_\alpha = 0.45204$. Using the customised integration scheme, the elastic-plastic model with all preservation terms, with the aid of geometric recovery routines and collision detection with the finger. The block of material was represented by a mesh of 20×10 elements. Top: Training. Middle: Test 1. Bottom: Test 2

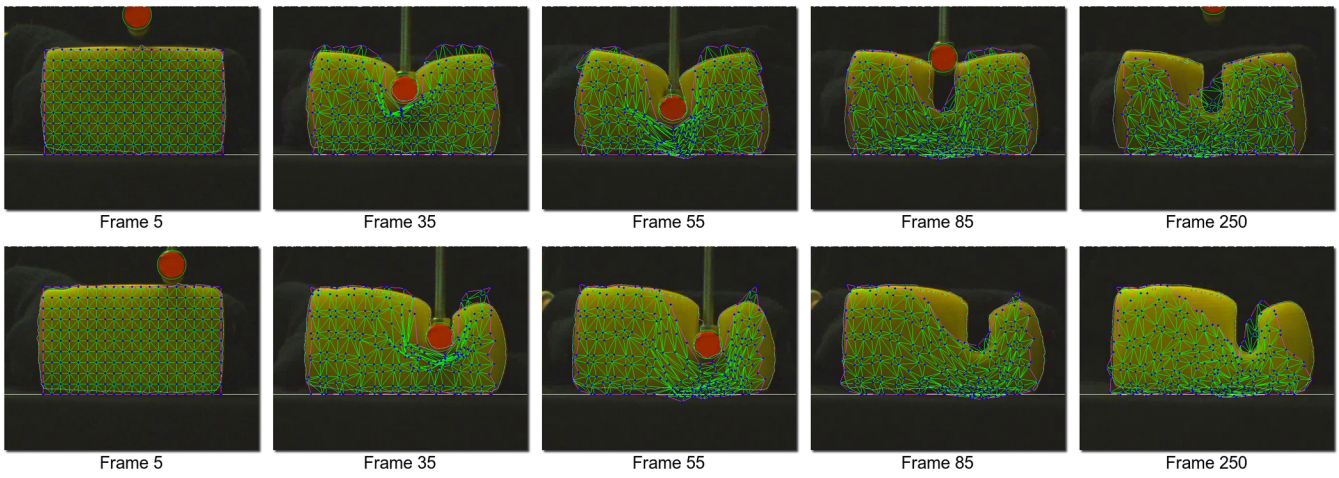

Fig. 14. Simulation of the plasticine obtained with parameters $k_L = 642.328, k_d = 0.562201, k_A = 29.2776 k_\phi = 72.6202, max\_Force = 765.912, yield = 0.0790105, creep = 0.664469, max\_\alpha = 0.48842$. Using the customised integration scheme, the elastic-plastic model with all preservation terms, with the aid of geometric recovery routines and collision detection with the finger. The block of material was represented by a mesh of 20×10 elements. Top: Training. Middle: Test 1. Bottom: Test 2

the begining, instant of greatest deformation and 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/retreive) to detect these points, since they are corelated to 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 mark of the best simulations was similar in both cases, since only geometric similarities are under consideration. However, there was a notorious difference on 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

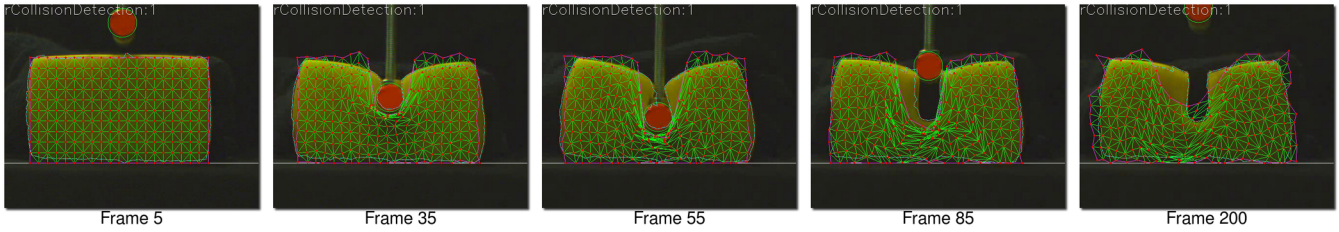

Fig. 15. Simulation of the plasticine obtained with parameters $k_L = 0, k_d = 0, k_A = 28202.1 k_\phi = 34.5491, max\_Force = 780.825, yield = 0.772727, creep = 0.67466, max\_\alpha = 0.656572$. Using the customised integration scheme, the elastic-plastic model with some preservation terms, furtherly tuned by hand, with the aid of geometric recovery routines with the finger. The block of material was represented by a mesh of 20×10 elements.

are not adequate, the shape of the mesh will be very different from the shape of the ground truth.

On the other side, 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 that the neighbours will succeed in propagating the deformation to the rest of the material, than if the positions of the springs had not been tampered by the collision resolution, thus producing better simulations. However, this lessens the effectiveness of the mass-spring model in distinguishing the materals.

D. Classification

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[8], but the combination of force readings and collision resolution is used during classification[9]. 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 beeing used.

E. Conclusions

In summary we have presented a framework, together with specific algorithms for learning the sensorimotor contingencies for the deformable behaviour of objects under robot manipulation. We have developed a learning model that learns the parameters of two contingencies that are sequenced, one for kinematic motion to force and the other for force to deformation. 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 were adjusted to 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. Fourth we include a new energy term for the preservation of angles of the mesh's triangles. 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 model 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.

[8]Case 2 in Section IV-F

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

[2] 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.

[3] S. Burion, F. Conti, A. Petrovskaya, C. Baur, and O. Khatib, "Identifying physical properties of deformable objects by using particle filters," *2008 Ieee International Conference On Robotics And Automation*, vol. 1, no. 9, pp. 1112–1117, May 2008.

[4] F. Conti, O. Khatib, and C. Baur, "Interactive rendering of deformable objects based on a filling sphere modeling approach," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 3, sept. 2003, pp. 3716 – 3721.

[5] B. Frank, R. Schmedding, C. Stachniss, M. Teschner, and W. Burgard, "Learning the elasticity parameters of deformable objects with a manipulation robot." in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[6] B. Frank, C. Stachniss, N. Abdo, and W. Burgard, "Efficient motion planning for manipulation robots in environments with deformable objects." in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, 2011.

[7] S. F. F. Gibson and B. Mirtich, "A survey of deformable modeling in computer graphics," MERL (Mitsubishi Electric Research Laboratory), Tech. Rep., 1997.
[8] M. Haruno, D. Wolpert, and M. Kawato, "Mosaic model for sensorimotor learning and control," *Neural computation*, vol. 13, no. 10, pp. 2201–2220, 2001.
[9] A. Howard and G. Bekey, "Intelligent learning for deformable object manipulation," *Autonomous Robots*, vol. 9, no. 1, pp. 51–58, AUG 2000, 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA 99), MONTEREY, CALIFORNIA, NOV 08-09, 1999.
[10] F. F. Khalil and P. Payeur, "Dexterous robotic manipulation of deformable objects with multi-sensory feedback–a review," *Robot Manipulators, Trends and Development*, vol. Vukovar, Croatia: In-Tech, pp. 587–619, 2010, a. Jimenez and B. M. Al Hadithi, Eds.
[11] M. Kopicki, M. Zurek, R. Stolkin, T. Morwald, and J. Wyatt, "Learning to predict how rigid objects behave under simple manipulation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 5722–5729.
[12] T. McInerney and D. Terzopoulos, "Deformable models in medical image analysis: a survey." *Med Image Anal*, vol. 1, no. 2, pp. 91–108, 1996. [Online]. Available: http://www.cs.ucla.edu/ dt/papers/mia96/mia96.pdf
[13] J. Montagnat, H. Delingette, and N. Ayache, "A review of deformable surfaces: topology, geometry and deformation," *Image and Vision Computing*, vol. 19, no. 14, pp. 1023–1040, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885601000646
[14] P. Moore and D. Molloy, "A survey of computer-based deformable models," in *IMVIP 2007: INTERNATIONAL MACHINE VISION AND IMAGE PROCESSING CONFERENCE, PROCEEDINGS*, J. McDonald, C. Markham, and J. Ghent, Eds., Irish Pattern Recognit & Classificat Soc. 10662 LOS VAQUEROS CIRCLE, PO BOX 3014, LOS ALAMITOS, CA 90720-1264 USA: IEEE COMPUTER SOC, 2007, Proceedings Paper, pp. 55–64, international Machine Vision and Image Processing Conference, Natl Univ Ireland Maynooth, Maynooth, IRELAND, SEP 05-07, 2007.
[15] 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.
[16] A. Nealen, M. Mueller, R. Keiser, E. Boxerman, and M. Carlson, "Physically based deformable models in computer graphics," *COMPUTER GRAPHICS FORUM*, vol. 25, no. 4, pp. 809–836, December 2006, 18th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2005), Natal, BRAZIL, OCT 09-12, 2005.
[17] M. Teschner, B. Heidelberger, M. Muller, and M. Gross, "A versatile and robust model for geometrically complex deformable solids," in *Proceedings of Computer Graphics International (CGI'04)*, Crete, Greece, June 16-19 2004, pp. 312–319.

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 has published more than 80 refereed articles. His interests include machine learning, planning, architectures, vision, mobile robotics and robot manipulation.