Learning to Predict the Behaviour of Deformable Objects through and for Robotic Interaction

by

Veronica Esther Arriola Rios

A thesis submitted to the

UNIVERSITY OF BIRMINGHAM

for the degree of

DOCTOR OF PHILOSOPHY

School of Computer Science

College of Engineering and Physical Sciences

University of Birmingham

April 2013

✠ ✠ ✠

"Eternal God,
who are the light of the minds that know you,
the joy of the hearts that love you,
and the strength of the wills that serve you;
grant us so to know you,
that we may truly love you,
and so to love you
that we may fully serve you,
whom to serve is perfect freedom,
in Jesus Christ our Lord."

St Augustine

UNIVERSITY OF BIRMINGHAM

Abstract

College of Engineering and Physical Sciences

School of Computer Science

Doctor of Philosophy

by Veronica Esther Arriola Rios

Every day environments contain a great variety of deformable objects and it is not possible to program a robot in advance to know about their characteristic behaviours. For this reason, robots have been highly successful in manoeuvring deformable objects mainly in the industrial sector, where the types of interactions are predictable and highly restricted, but research in everyday environments remains largely unexplored. The contributions of this thesis are: i) the application of an elastic/plastic mass-spring method to model and predict the behaviour of deformable objects manipulated by a robot; ii) the automatic calibration of the parameters of the model, using images of real objects as ground truth; iii) the use of piece-wise regression curves to predict the reaction forces, and iv) the use of the output of this force prediction model as input for the mass-spring model which in turn predicts object deformations; v) the use of the obtained models to solve a material classification problem, where the robot must recognise a material based on interaction with it.

To my family,
who have supported me through good and bad.

Acknowledgements

First I express my sincere gratitude to my supervisor Jeremy Wyatt, whose guidance, patience and support through all these years made this work possible.

To my second supervisor Aaron Sloman, for the always challenging problems and questions that will keep me working beyond this PhD.

To Jackie Chappell, for the lessons and fruitful collaboration.

To Zoe Demery, for all I learned from her, for the joint works and overall for her friendship and support.

To José Nuñez, Javier Juárez and Lucy Ramírez, for keeping the nation always at hand, for the long lasting friendship that shall continue. See you back in Mexico.

To Rustam Stolkin, whose aid during the data gathering sessions made all the difference for this research.

To my ice-skating coaches Catherine Harvey, Alexander Markuntsov (Sasha), Colin Sturgess and Tina Tedder-Graham who, unknowingly, encouraged me to keep going.

To all the fellows from my generations, with whom I shared cookie breaks, Research Skills, office and so many other things.

This research would not have been possible without the sponsorship of CONACYT Mexico. Also thank you very much to the NHS, the ORSAS scheme and to COG-X.

Thank you to all my friends, especially to the people from Newman House and the Spinney Flat 16, who made all this adventure worth.

Thank you to Jesus, without His teachings I would not be here today.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Objective: Prediction and Classification . . . 5
- 1.3 Research Hypothesis . . . 5
- 1.4 Contributions . . . 6
- 1.5 Overview . . . 9

2 Prediction for Deformable Objects: Problem Definition **13**
- 2.1 Stages in the Process of Robotic Interaction . . . 13
- 2.2 Problem Statement . . . 17
  - 2.2.1 General Goals for the Robot . . . 18
  - 2.2.2 Requirements . . . 18
    - 2.2.2.1 Forces . . . 19
    - 2.2.2.2 Shape . . . 20
  - 2.2.3 Materials . . . 22
  - 2.2.4 Learning . . . 22
- 2.3 Research Scenario . . . 23
- 2.4 Relevant Disciplines . . . 24
- 2.5 Summary . . . 26

3 The Difficulty of Studying Deformable Materials **29**
- 3.1 Understanding Different Materials . . . 29
- 3.2 Segmentation of the Environment into Objects . . . 30
- 3.3 Characterization of Substances in Terms of Their Properties . . . 31
- 3.4 Inference of an Underlying Structure . . . 31
- 3.5 Emergence of the Properties of Materials from the Structure of Matter . . . 33
- 3.6 Design of Materials . . . 34
- 3.7 Identification and use of Properties of Materials . . . 35
- 3.8 The Static Case: Equilibrium . . . 36
  - 3.8.1 Equation of State and Constitutive Equations . . . 37

3.9 Quasi-static Processes . . . 39
3.9.1 Characterising the Material with Differentials . . . 40
3.10 Summary . . . 41

4 Mechanical Behaviours of Deformable Materials 43
4.1 Definition of Stress and Strain . . . 43
4.2 Types of Deformation . . . 45
4.3 The Phases of a Material . . . 47
4.4 Stress/Strain Diagrams . . . 48
4.5 Types of Transitions between the Elastic and Plastic Phases . . . 49
4.6 The Perspective of a Robot . . . 52
4.7 Summary . . . 52

5 Mathematical Models of Deformable Objects 55
5.1 Mathematical Modelling for Physicists . . . 56
5.1.1 Modelling of Deformable Objects . . . 57
5.2 Particles . . . 58
5.3 Multiparticle Systems . . . 60
5.4 Inertial Systems: Force, Momentum and Kinetic Energy . . . 61
5.5 Forms of Interaction: Contact vs. Fields . . . 62
5.5.1 Potential Energy . . . 63
5.6 A Simple Model of Interactions: Ideal Springs . . . 65
5.6.1 Simple Harmonic Oscillator . . . 65
5.6.2 Interaction with Oscillatory Forces: Resonance . . . 68
5.6.3 Numerical Integration . . . 69
5.6.3.1 Euler . . . 69
5.6.3.2 Verlet . . . 71
5.6.3.3 RK4 . . . 72
5.7 A Model of Friction . . . 74
5.7.1 Viscous Damping . . . 75
5.7.2 Spring Damping: Dampers . . . 76
5.8 Damped Harmonic Oscillator . . . 77
5.8.1 Case 1: Overdamping . . . 78
5.8.2 Case 2: Critical damping . . . 79
5.8.3 Case 3: Underdamping . . . 80
5.8.4 A One Dimensional Model . . . 80
5.9 A Model for Complex Objects or Materials: Mass-spring Systems . . . 81
5.9.1 Representation of the Shape: Meshes . . . 82
5.9.2 The Dynamics: Particles Connected with Springs . . . 83
5.9.3 An Extended use of Springs: Teschner's Model . . . 85

5.9.3.1 Preservation of Length . . . 86
5.9.3.2 Preservation of Area . . . 87
5.9.3.3 Preservation of Volume . . . 88
5.9.3.4 Damping . . . 89
5.9.4 Plastic Deformation . . . 89
5.10 Summary . . . 90

6 Related Work I: Modelling of the Shape 91
6.1 Continuous vs Discrete Models . . . 92
6.2 Implicit Representations . . . 94
6.2.1 Algebraic Curves and Surfaces . . . 94
6.2.1.1 Superquadrics . . . 95
6.2.1.2 Hyperquadrics . . . 97
6.2.2 Level Set Methods . . . 97
6.2.3 Gaussian PCA eigenmodes . . . 99
6.3 Explicit/Parameterised Representations . . . 102
6.3.1 Splines . . . 103
6.3.2 Active Contours (Snakes) . . . 110
6.3.3 Modal Decomposition . . . 111
6.4 Discrete Models . . . 112
6.4.1 Meshes . . . 112
6.4.1.1 Triangulations . . . 113
6.4.1.2 Simplex Meshes . . . 114
6.4.2 Skeletons . . . 114
6.4.3 Deformable Templates . . . 115
6.4.4 Landmark Points . . . 116
6.4.5 Particles . . . 117
6.4.6 Clouds of Points . . . 117
6.5 Free-Form Deformation . . . 118
6.6 Summary . . . 119

7 Related Work II: Methods for the Simulation of Deformation Dynamics 121
7.1 Dynamics of Elasto-Plastic Continuous Media . . . 122
7.1.1 Stress Tensor . . . 123
7.2 Finite Element Methods (FEM) . . . 124
7.2.1 Automatic Calibration of FEM models . . . 126
7.2.2 FEM + Snakes = Visual Force Sensors . . . 127
7.2.3 Finite Difference Method . . . 128
7.2.4 Finite Volume Method . . . 128
7.2.5 Boundary Element Methods (BEM) . . . 128

7.2.6 Long Element Method . . . 129
7.3 Mass-Spring Methods . . . 129
7.3.1 Handling Anisotropic Behaviour of Mass-Spring Systems . . . 129
7.3.2 Use of Spheres for Volume Preservation in Mass-Spring Systems 131
7.3.3 Automatic Calibration of Mass-Spring Models . . . 132
7.4 Particle Systems . . . 132
7.5 Neural Networks . . . 133
7.5.1 Cellular Neural Networks . . . 133
7.5.2 Neural Gas Networks . . . 134
7.6 Probabilistic Approaches . . . 136
7.7 Modal Analysis . . . 138
7.7.1 Modal Dynamics . . . 138
7.7.2 Geometric Modal Analysis . . . 139
7.8 Summary . . . 141

8 Scenario 143
8.1 Experimental Set-up . . . 144
8.1.1 First Set of Experiments: Pressing a Sponge . . . 145
8.1.2 Second Set of Experiments: Pressing and Releasing a Sponge and Plasticine . . . 145
8.1.3 Training and Testing . . . 147
8.2 Input . . . 148
8.2.1 Image . . . 148
8.2.2 Force Sensor . . . 148
8.2.3 Timer . . . 149
8.3 Internal Representation of the Shape . . . 149
8.3.1 Contour: Linear Splines . . . 150
8.3.2 Surface: Meshes . . . 150
8.4 Deformation Process . . . 152
8.4.1 Tracking: Linear Snakes . . . 152
8.4.1.1 Identification . . . 153
8.4.1.2 The Tracking Algorithm . . . 154
8.4.2 Prediction . . . 156
8.4.2.1 Physics Based Model . . . 157
8.5 Learning: Automatic Calibration . . . 158
8.6 The System . . . 158
8.7 Summary . . . 160

9 Extensions to the Physics Based Model of Deformable Objects for Robotics 161
9.1 Research Hypothesis . . . 162

9.2 Extensions to Teschner's Model . . . 163
9.2.1 Preservation of Length with Damping Along the Spring . . . 163
9.2.2 Preservation of Areas . . . 165
9.2.3 Preservation of Angles . . . 167
9.3 Plastic Deformation . . . 170
9.3.1 Compression and Dilation . . . 171
9.4 Graphical Constraints . . . 172
9.4.1 Collision Detection . . . 173
9.4.2 Mesh Shape Preservation . . . 175
9.5 Integration Scheme . . . 176
9.6 Learning: Automatic Calibration . . . 177
9.6.1 Searching in the Space of Parameters with an Evolutionary Algorithm . . . 179
9.6.2 Evaluation Function: Difference of Areas . . . 183
9.7 Prediction of the Force . . . 187
9.8 Classification . . . 192
9.9 Summary . . . 194

10 Experiments on Prediction **195**
10.1 Training of the Mass-spring Model . . . 195
10.2 Performance of the Training Phase of the Genetic Algorithm . . . 202
10.3 Performance of the Predictions on the Test Data . . . 203
10.4 Adding Prediction of the Force . . . 212
10.5 Summary . . . 226

11 Experiments on Classification of the Materials by using the Learned Models **227**
11.1 Parameters for Sponge vs. Parameters for Plasticine . . . 228
11.2 Using the Quality of the Predictions . . . 233

12 Conclusions **247**
12.1 Summary . . . 247
12.2 Contributions . . . 248
12.3 Future Work . . . 249

A Architecture **251**

B Additional Experimental Data **255**
B.1 Rougher Meshes for Predictions on the Sponge . . . 255
B.2 Histogram of Parameters with the Finer Mesh . . . 255

Bibliography 261

Chapter 1

Introduction

1.1 Motivation

Currently, industrial robots are capable of highly complex automated manipulation of objects, solid or deformable (e.g. assembly of motor vehicles [45], automated sewing systems and unfolding of fabrics [64]). However, industrial research has focused on dedicated solutions for specific processes. Research on deformable materials has been concentrated mainly in the garment industry, followed by aerospace, automotive and shoe/leather industries where physics based models are tuned for specific problems [37, 72]. On the other hand, general purpose robots, which are the focus of this research, are limited to work on very simple scenarios containing mainly rigid solid objects [73]. However, most of the objects in the real world are not rigid solids. Deformable solids as well as the other states of matter populate any environment. Many complex animals, notably humans, have evolved to cope and handle deformable objects as part of their daily life. It is therefore of relevance to study how robots could be made able to

manipulate an ever growing variety of deformable objects. The main areas where some research is being carried out are: computer vision (tracking and recognition) [41], computer graphics [32], industrial robotics (for particular cases) [72] and medicine (organ recognition, tracking and haptics simulation, e.g for medical training) [53]. Most of this research focuses on particular materials and scenarios, while a robot should be able to deal with a variety of objects made of different materials. Clearly, even if a collection of methods for perception and control of deformable objects were to be programmed by hand in the robot, the robot will encounter new materials and new configurations, the behaviour of which is not exactly covered by the information it already has. This means that, in order to extend its capabilities, it must be able to learn from experience.

Furthermore, the robot must not only track the movement and deformation of objects, but it also must be capable of predicting their behaviour. Eventually, the predictions should be good enough to be used for the identification of possible affordances. This would allow the use of objects as tools for tasks that may be very different from the scenario where they were first introduced [80]. To achieve this, it is necessary for robots to be able to predict the behaviour of deformable materials under different circumstances with enough accuracy to make plans [27]. New scenarios may require actions and include configurations[i], or even materials that have not been experienced before. Therefore, the accuracy can be evaluated at different levels: regarding the overall configuration of the object, articulated movement or deformability of the shape.

An additional complication present in this scenario is the fact that no simple model can cover the behaviour of a material for all circumstances. There is always a trade off between generality and accuracy. Frequently, there are not only different models for

[i]Where *configuration* is defined as: form, as depending on the relative disposition of the parts of a thing's shape; figure. The way things are arranged or put together in order to achieve a result [http://en.wiktionary.org/wiki/configuration]

different materials, but also different piece-wise models for the different behaviours of one single material, under different conditions.

FIGURE 1.1: *Different stages in the deformation of a marshmallow covered with chocolate as it is increasingly stretched and allowed to recover its original shape. Top: The marshmallow is stretched beyond its elastic threshold and then photographed as it still tries to recover its original shape. Middle: The marshmallow is stretched beyond its fracture point and it begins to tear at an unpredictable point (the intention was to fracture it at the middle, but only the lower corner got separated). Bottom: Original marshmallow; marshmallow after being pulled on the elastic domain, the chocolate is already fractured, but the marshmallow got fully recovered; marshmallow after the photo in the middle, one corner got separated, but the rest of it almost recovered its shape!*

Figure 1.1 is used to illustrate the highly complex behaviour of a very common object. A little child, who begins to explore the world, would be interested in observing and learning about the detailed behaviour of a familiar object like a marshmallow covered with chocolate by performing actions like the one shown in the photograph. Technically, the joint behaviour of the marshmallow and the chocolate illustrates elasticity,

plasticity and fracture. The chocolate is a highly brittle material, which means that it fractures as soon as stress is applied. In contrast, the marshmallow exhibits great elasticity and, in spite of being stretched for a couple of centimetres, it fully recovers its original shape. The transition between its elastic and plastic deformation is hard to find, because it is only after relatively extreme tensions that the marshmallow begins to be torn (which is known as a ductile fracture) and thus loses its ability to fully recover; nevertheless it still compacts itself again.

This simple interaction clearly illustrates why the detailed behaviour of many deformable objects we encounter in everyday life can not be covered by one single mathematical model. However, such detailed and precise model is not required if a person desires to share the marshmallow equitably with a friend. In this case, it would be necessary to have some model of the behaviour of the marshmallow when forces are applied to it, to predict which forces must be applied to produce an even splitting or how to adjust quickly if the splitting deviates from the plan. The required characteristics of the model are task dependent. However, the flexibility to recall an adequate model for each situation is developed during playful activities during which a wide variety of behaviours have been characterised. It is possible that during these activities more than one model is generated, to be used in the future [12].

What a robot requires, is to acquire means to sufficiently approximate the behaviour of a real deformable object, according to the *requirements* of its current task. For this reason, this work focuses on the automatic calibration and analysis of the flexibility of a simple physics model; not with the aim of obtaining accurate quantitative predictions, but to evaluate its qualitative meaningfulness and to start exploring the stage of playful exploration. As an example, the models obtained are used to solve the problem of

classification; that is, the robot uses the models to identify which of the objects it has learned about corresponds to the object it is currently interacting with.

1.2 Objective: Prediction and Classification

The objective of this research is to provide the robot with the definition of a model, that can be calibrated to make predictions of the deformations an object will suffer, when the robot interacts with it in different ways. A direct application is the use of the calibrated models to identify the objects the robot is interacting with. The method explored here for the solution of these problems is proposed in the research hypothesis.

1.3 Research Hypothesis

By definition, a solid material resists changes in shape and volume, because the particles that constitute it remain at a fixed distance [34]. Real materials are only approximately solid. When the distance between the particles that constitute an object can change, but the object does not break apart, we say that the material is solid deformable. As it will be further developed in Chapter 4, there is no clear division between solid deformable objects (like plasticine) and heavily damped viscous fluids (like glass). For this reason, we work with materials that present a solid behaviour at rest[ii], but that allow for deformations of their shape when forces are applied to them (e.g. sponge and plasticine). The purpose of this research is to test the extent of the force/deformation domain where our proposed model can still make predictions and

[ii]That is, when no forces produce an acceleration of the particles that constitute the object, either with respect to other particles in the object or over the whole object.

the accuracy of these. We consider a simple scenario where the object's location is fixed, and the main deformations can be observed. In this context, the present thesis analyses the following hypotheses:

- Hypothesis 1:

  Can a Mass-spring model be learned from position & force data of the deformations of real solid deformable objects?

  - ⋆ Can it be calibrated automatically from real data?
  - ⋆ Can it make predictions of shape deformations given novel actions?
  - ⋆ Can it model both elastic & plastic deformations?
  - ⋆ Can it be used for classification of materials?

- Hypothesis 2:

  Can a regression model be learned from the stress-strain data generated by an action on a material?

  - ⋆ Can it predict the stress-strain data for a new action?
  - ⋆ Can it serve as input to the mass-spring model?
  - ⋆ Can this two-stage model be tuned for elastic and plastic objects?
  - ⋆ Can this two-stage model be used for classification of materials?

1.4 Contributions

There are few works on learning to predict arbitrary deformations of volumetric objects, even if their representation is in 2D. The closest ones were carried out by:

1. Teschner, who proposed some modifications to the traditional mass-spring system that improve the computational efficiency of the simulation [88]. This work covers elastic and plastic deformation. It can estimate the behaviour of the object while forces are applied on it and while they are retrieved. However, there is no calibration against ground data. See Figure 1.2(a).

2. Morris, who calibrated autonomously Teschner's model against a Finite Element Method (FEM) [55]. See Figure 1.2(b). However, only elastic deformations where covered and the value of the applied forces remained constant.

3. Frank, who calibrated autonomously a FEM model, but used only constant forces and elastic materials [26]. See Figure 1.2(c).

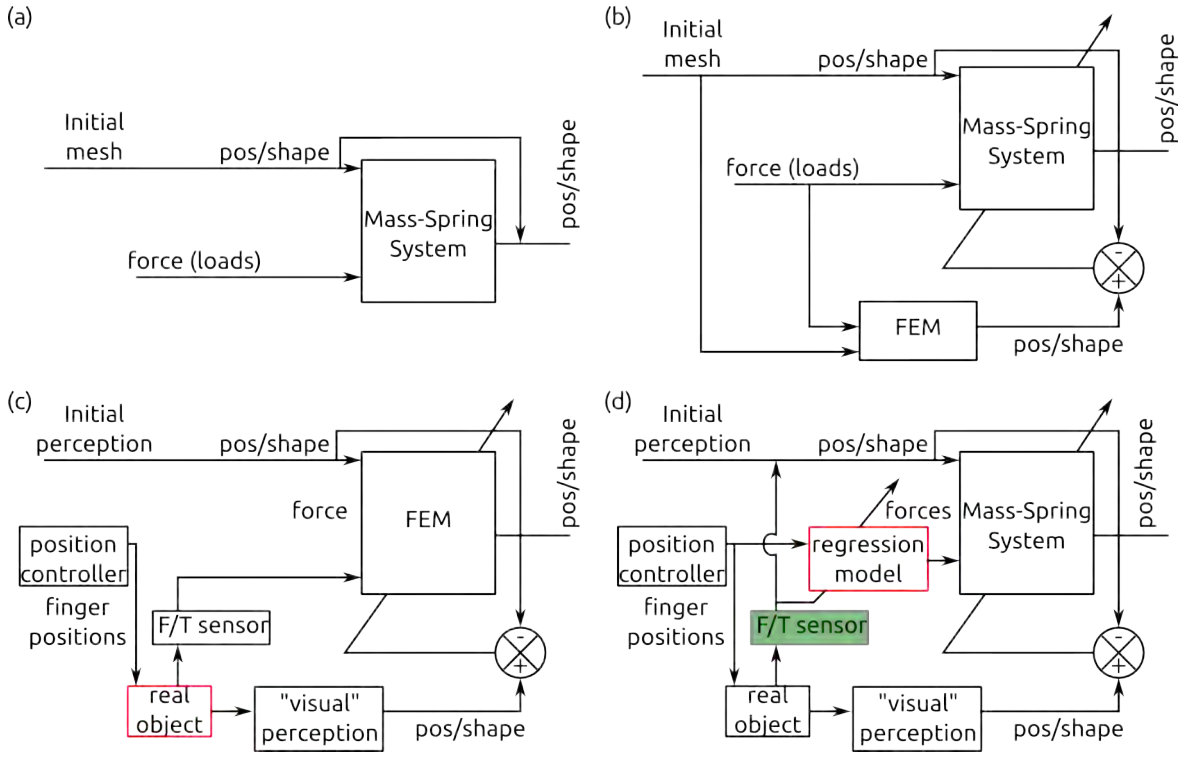

FIGURE 1.2: *Block diagram of the systems developed by (a) Teschner, (b) Morris, (c) Frank and (d) this thesis.*

This thesis builds on the first two. However, here real objects are used as ground truth; both elastic and plastic deformations are covered; and the models are evaluated while the actuator of the robot pushes and releases the target object. See Figure 1.2(d). Additionally, the calibrated models are used to recognise to which previously seen class they belong, even it the interaction with them is different. The main advances of this thesis over the cited work are summarized in Table 1.1. Furthermore, this research contributes to the field of robotics in various ways:

TABLE 1.1: *Advances over previous work.*

	Model type	Learned from GT	GT=Real data	Prediction of Positions	Prediction of Forces	Actuator Retrieval	Elastic	Plastic	Classification
Teschner	MS	x	x	✓	x	✓	✓	✓	x
Morris	MS	✓	x	✓	x	x	✓	x	x
Frank	FEM	✓	✓	✓	x	QSA	✓	x	x
Thesis	MS + Regression	✓	✓	✓	✓	✓	✓	✓	✓

*QSA Quasi-static assumption. Readings of the force sensor are not taken until the object stops moving.

1. It transfers knowledge about modelling of deformable objects from other fields to robotics: considerations on the difficulty of learning models for deformable objects, from material's science; a specific type of mass-spring model, from physics; numerical methods for solving the equations, from mathematics; and collision detection of obstacles and collision resolution, from computer geometry. Most of these techniques are better known in computer graphics and virtual reality.

2. A physics based mass-spring model is used by the system to make predictions about the changes in the shape of deformable objects that a robot pushes.
3. This model is calibrated autonomously by the system using real objects as ground truth, while most previous work did not use real objects or was not autonomous.
4. The application of the models to the problem of classification. The calibrated models are used to identify a previously known material, even if the interactions are different, because the correct model makes better predictions.
5. The outcome of the experiments justifies the use of a force sensor for this type of research. Those experiments where only visual information was used failed some times, where those with force information succeeded, most noticeably in the classification task.

There are also some minor contributions spread across the different aspects the system had to cover:

1. The sketching and use of a colour based tracking algorithm, with a technique called *snakes* [Section 6.3.2], that adjusts the number of its elements as the shape of the tracked object requires it.
2. The addition to the mass-spring model of a term that enforces the preservation of the angles of the elements of the mesh.

1.5 Overview

Chapter 2 presents the problem delineation in more detail. It splits the problem of robots interacting with deformable objects into different stages, which leads to the ability of learning to make predictions about how they deform. It also lists the different

disciplines that can contribute towards each stage and highlights the ones this research focuses on.

Chapter 3 delves into the matter and presents an overview of the problematic related to identifying, characterising and using materials with a wide variety of properties. A general framework taken from thermodynamics is presented since relevant methods to model deformable objects can be better understood within this frame.

Chapter 4 focuses on mechanical properties, since those are the target of our research. Relevant terminology, originating from the field of materials science, but in common use in the literature about modelling of deformable objects, is introduced here. This chapter also illustrates the wide variety of elastic, plastic and transitional behaviour that different materials exhibit. This shows graphically why one single function can not reproduce or predict the behaviour of all different deformable objects. Nevertheless, distinctive characteristics of the described behaviours point towards possible approaches.

Chapter 5 Introduces the most basic terms and equations used in physics to model particles, multiparticle systems, springs, dampers and meshes of masses and springs. It also presents a very quick review of the fundamental concepts used to describe movement and interactions: force, fields, kinetic and potential energy. These elements are used in this work to model the deformable objects the robot interacts with, so special care has been put into their description.

Chapter 6 presents the different techniques that appear throughout the literature to represent the shape of the object and its deformations. Chapter 7 presents the techniques used to animate the deformations of the objects, when the forces that act on them are know. Some representative works are given as illustration of these, but also relevant

developments that make use of combinations of the cited techniques are presented. However, even though these developments are relevant to each stage of the robotic process, there is very little work where they have actually been applied to robotic learning, as we do in this thesis.

The subsequent chapters present the original contributions of this work. Chapter 8 explains the concrete scenario where the hypothesis and methodologies are tested. It also indicates, from among the various options found during the literature review, which ones were chosen and why. Chapter 9 contains the details of the theoretical contributions, while Chapter 10 reports the experimental results. Chapter 11 explores results further, as the models obtained are applied to the classification problem.

Finally in Chapter 12 a summary is made, along with suggestions for future work and the conclusions of this research.

Chapter 2

Prediction for Deformable Objects: Problem Definition

The research problem of this thesis is delineated here in detail. To set the context, Section 2.1 begins with a small list of processes that a robot would need to implement when working with deformable objects. Section 2.2 focuses on those processes that are targeted in this thesis. The concrete scenarios, where our hypotheses were studied, are presented in Section 2.3. As a preparation for the literature review, Section 2.4 lists the disciplines that can potentially contribute to the problem of how a robot can learn to make predictions about deformable objects.

2.1 Stages in the Process of Robotic Interaction

There are several forms or stages of interaction with a deformable object that the program of a general purpose robot may need to address. These include:

1. Identification The object needs to be targeted in the scene. It begins with the acquisition of information about the object through the sensors of the robot. The most commonly used is vision, but force, pressure and temperature sensors could become just as important. These signals must be processed in such a way that the robot can identify the object of interest[i].Some characteristics of the object must be given to the robot, either through direct coding or though learning. They could be particular aspects, like its colour, texture or a more detailed model; or they could be generic principles like selecting the colour segment that is closer to the centre of the image.

For this research, a basic set of computer vision techniques are used to segment the environment and identify the object of interest, which is known to be bright yellow. Also, readings of the applied forces are recorded and used in later stages.

2. Representation The object must be represented internally somehow. There is a wide variety of options, each with advantages and disadvantages depending on the different tasks the robot may need to carry out with the object. The most representative are discussed in Chapter 6.

For example, here we make use of a 2D mesh for the representation of the surface of the original object [Section 8.3].

3. Classification In order to make information manageable, the information about several objects needs to be classified. Objects with similar characteristics should be grouped together.

For evaluation purposes, in this research, two classes are considered: elastic and plastic objects. By definition, their behaviour following a deformation determines to which

[i]Notice that *identification* here is different from *recognition*.

class they belong. Elastic objects recover their original shape, while plastic objects retain the deformation. We test the calibrated models by using them to distinguish between previously seen elastic and plastic objects.

4. Tracking and Predictive Tracking Since the deformation of an object is a dynamic process, it is necessary to follow the changes in their shape through time. It is more straightforward to carry out this task if the robot already has an expectation of how and where this shape will appear in the immediate future. Sometimes very good models are used to make these predictions. However, for tracking, frequently predictions for the next immediate frame are used, and tests for further predictions are not reported.

In this work we introduced a very simple variation to tracking with linear splines, so that the robot can detect the ground truth during its learning phase. We do not use predictive tracking.

5. Prediction A prerequisite for planning is the ability to predict the behaviour of objects, without the action being considered having actually taken place. To predict behaviours a robot needs models of the objects. For an industrial robot, it may be enough to have models of the limited types of materials and restricted conditions it will face. On the contrary, for a general purpose robot, the alternative is to learn new models as it encounters more and more types of materials.

As relevant and interesting as this topic is, there is not much literature in robotic **learning** of models of 3D deformable objects that would allow robots to make predictions. Therefore, the most developed research that can be applied to this problem is the automatic calibration of models based on physics [37]. In this area, there are very few works that have used real objects as ground truth for learning, apart from this research:

- Howard and Bekey, from the University of Southern California, used a couple of robotic arms and learned to predicted the force required to lift bags filled with various materials. They used a neural network to calibrate a damped mass-spring system [40]. However she did not evaluate the prediction of the deformation of the shape of the object.
- Frank and colleagues, from the University of Freiburg, developed their work in parallel with our work, but they used gradient descent to find appropriate values for the Young modulus and Poisson ratio for a Finite Element Model of the deformable object [26]. However, they were only interested in obtaining simulations good enough as to allow the robot to plan navigation trajectories around deformable objects [27]. In their experiments the robotic finger remained static during the simulation, they did not provide a detailed analysis of the behaviour of their model while the finger and the object were moving.
- Cretu and colleagues. from the University of Ottawa, used neural gas networks for predictive tracking and tested them on prediction as well, but for a single frame [21].

In our case, we chose to use a mass-spring model, for reasons that will be exposed below [Section 2.2.2]. Also, we were interested in evaluating how far, in terms of accuracy and generalisability, the predictions of a model calibrated automatically can be taken.

It is because of this lack of literature in robotics applied to deformable materials, that the literature review in Chapter 6 and Chapter 7 widely explores techniques highly developed in other fields. These techniques could eventually become options for robotics. There is also a collection of works in robotics that could be extended for our purposes and are presented in Chapter 7.

6. **Manipulation** Here it is possible to include the different types of interaction a robot can have with an object by using its effectors: pushing, pulling, grasping, throwing, etc.

   In this work the manipulation task consists of a pushing action, in such a way that the object becomes deformed.

   Some works address the problem of manipulating objects, without putting to much emphasis on modelling the behaviour of the object itself. In this area neural networks, fuzzy logic, genetic algorithms and hybrid intelligent techniques have been used for low-level control in the domain of robotic compliance tasks, for details see the book by Katic [44].

7. **Planning** When the robot is able to predict the behaviour of objects before acting, it is possible for it to make hypotheses and select sequences of actions in advance, to achieve a desired goal. It can also be possible to identify new affordances for new scenarios the robot had not encountered before.

   Even though we look at this stage as a future goal, we do not make any contribution to this point.

2.2 Problem Statement

From among the several stages mentioned in the previous section, our main focus is on the part of **prediction**. How can a robot learn to predict how objects deform? Furthermore, we chose 3D real objects, since there is so little research on them. However, because of the pioneering nature of our choice it was necessary to simplify the scenarios as much as possible, while preserving the challenging aspects of modelling

non-articulated deformable objects. The following sections explain the reasons behind each of our choices.

2.2.1 General Goals for the Robot

This research focuses on two general goals for the robot:

1. Given that the wide variety of behaviours for deformable objects, prevents robots from being preprogrammed to make predictions about them, robots require the ability to add knowledge about their mechanical properties as it encounters them. This involves identifying, representing and obtaining a model of the behaviour of the object. We kept the first stages extremely simple, while putting more emphasis on how to obtain the model.

2. To have the ability to predict its deforming behaviour beyond the observed domain.

To achieve this, there is a set of requirements for the system; particularly, on the types of models that could be considered as candidates.

2.2.2 Requirements

There are two sensory aspects to be considered: tactile and vision; in other words: forces and shapes. Much research in robotics focuses on the use of visual information, but it is proposed here that force information is highly relevant as well.

2.2.2.1 Forces

The process of deformation of an object is fundamentally dependent on the forces being applied to it, and the rate at which these forces are applied. Also, it would not be possible to determine whether the differences among behaviours of different objects are due to the forces being applied to them, or to the constitutive properties of the material, if there is no explicit information about either. This implies that, trying to learn to predict the kinematic behaviour of an object, without taking into account the forces being applied, is meaningless. These forces need to appear in the model either in an explicit or an implicit form. Given that different scenarios and different materials produce very different sets of forces, the model must be flexible enough to account for these differences. If we wanted to avoid mentioning the forces explicitly, they should either emerge from the model obtained as a measurable underlying cause for different behaviours, or be encoded within other types of equivalent constraints.

For a robot, we can consider the following ways in which it can perceive the forces:

1. As geometric constraints, derived from contacts between objects. E.g. a block of plasticine being unable to pass through a table, or a finger pushing the surface of a sponge, that it cannot penetrate, thus deforming it.

2. As measurements obtained from force sensors. These sensors could be force sensors placed directly in the actuators of the robot, or other sensors from which the forces can be derived (e.g. currents sensors placed in the joints of the actuators). Another technique, which would not be as functional for robotics, is to conveniently place the sensors around the object of interest, instead of on the robot. This technique is widely used to recreate objects for virtual environments, but is not practical for mobile robots [65].

3. They could be inferred from a model that has already been calibrated. It may be the case that the model has an explicit way to infer the forces (e.g. inverse kinematics [50]) or it could be possible to capture them in an implicit way (e.g. using neural networks).

4. For the particular case of the forces applied by the robot, it could be possible to know them because the program of the robot specified them. A more intricate way would be to have a model of how a device controls the forces to be applied (e.g. which forces should have been applied by a compliant arm, given that its reaction is known).

For this research, the use of a force sensor in the actuator and of graphic constraints for external obstacles [point 1 above] were selected. To simplify the problem a much as possible, we chose then to reduce the type of interaction to the robot pressing against a deformable object, whose movement is constrained by a fixed obstacle on the opposite side; see Figure 2.1. This set up allows the robot to acquire information about the properties of the material it is actively deforming, and simplifies the evaluation of the model.

2.2.2.2 Shape

The main inconvenience of deformable objects is that they can have almost any shape. Extremely intricate 3D-shapes pose a challenge from the initial stage of trying to represent them adequately[ii] in the memory of the computer; even more so, if the data must be acquired from sensors. The main forms of representation are described in Chapter 6. Also, there is a wide variety of devices that have been used to capture this data:

[ii] Here the measurement of adequateness is also highly task dependant.

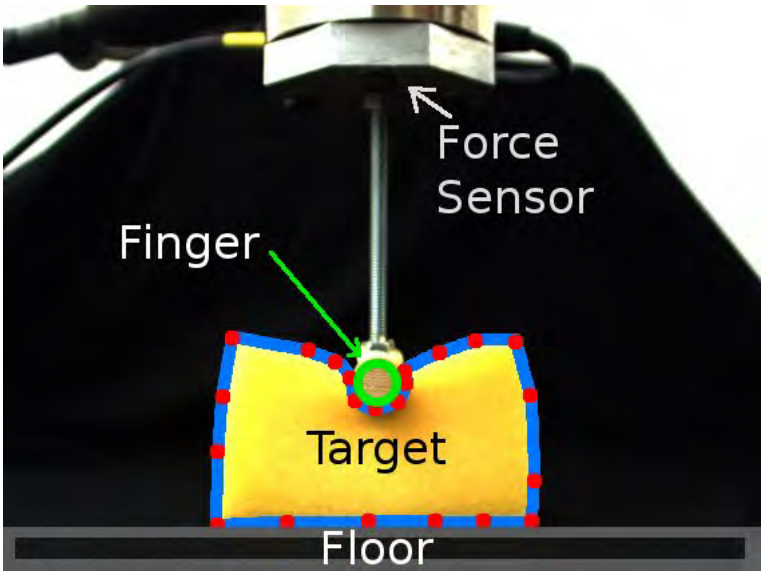

FIGURE 2.1: *Experimental setup. Perpendicular view. A finger with a force sensor pushes a sponge. The finger pushes the sponge away from it. On the other end, the floor is used as an obstacle always opposed to the finger. A camera observes and records the action perpendicularly to the target.*

stereo cameras, multiview-camera systems, 3D range scanners, etc. It can be possible as well to infer 3D structure from the proprioceptive information of a scanning probe. However, since our main focus is in the automatic calibration and evaluation of the dynamic model, it is more informative to make use of objects with a very simple initial shape, but that can freely deform, thus illustrating the lack of fixed or predictable number of elements required to represent them. For this reason, we chose to begin the experiments with rectangular blocks. This selection, joined to the type of interaction indicated in the previous section, also allows us to focus on only one frontal view of the object, so that the problem of capturing the information about the shape is highly reduced. Figure 2.1 shows the view that the robot has of the experiment. The choice of highly contrasting colours reduces the demands imposed on the segmentation algorithms, but the appearance of shadows due to the three dimensional deformations still

causes some problems. These were addressed with the tracking algorithm explained in Section 8.4.1.2. However, they are also an indicator of small deformations in the perpendicular direction that have been neglected and would be more properly addressed with a 3D model.

2.2.3 Materials

A robot must interact with real objects, not with engineered simulations. This implies that the materials are quite likely to exhibit behaviours that deviate from what a single model can cover. Since the most relevant deformable behaviours of solid materials are the elastic and plastic phases, according to the typical phase diagrams presented in Chapter 4, it is desirable for the robot to be able to model at least these. The materials chosen for the experiments therefore need to be highly illustrative of the main characteristics of each. It would also be interesting to include an intermediate material, but this had to be left for further research.

2.2.4 Learning

The robot should be able to calibrate models for elastic and plastic behaviours. It can be allowed to interact directly with the objects for the learning phase, so that it can have some access to information about the forces being involved. It should become capable of making predictions about their response beyond the training interaction (e.g. by pushing in a different point). At this stage, it will be aided by reducing the interaction in such a way that the deformability of the object is the dominant process. This is achieved here by blocking the movement of the object with some obstacle. In

an open scenario, a robot could also be required to actively set-up the object in such advantageous circumstances for learning purposes.

2.3 Research Scenario

Summarising, for this research we have a robot that:

- Captures data, with a force sensor in its actuator and a digital camera.
- Pushes a block of deformable material against an obstacle (applied force and corresponding shape).
- Interacts with highly deformable elastic and plastic material: a dish-washing sponge and a block of putty (plasticine).
- Uses the data to calibrate autonomously a physics based model (mass-spring like).
- Tests the calibrated model by pushing the material in a different position.

All the specific details of the implementation for this research will be given in Chapter 8, after the literature review of relevant work, since some concepts will be easier to explain after that.

Having this type of scenario in mind, the following section mentions the disciplines that can contribute more with ways to address the problems posed by each stage. The following two chapters introduce the most representative literature, thus better locating this particular research in the broader field.

2.4 Relevant Disciplines

The literature presented in the following chapters contributes to one or more of the stages mentioned in the previous section, and may come from very different scenarios and disciplines. It is not possible to cover everything, since the number of works that can contribute towards predicting the behaviour of deformable objects by a general purpose robot is massive. However, an effort will be made to organize representative works and present a clear overview of the general panorama.

There are several branches of computer science that have addressed these stages in different manners, principally:

Computer Vision Its name is self-descriptive. It focuses on the analysis of camera-acquired images and videos. It covers issues like segmentation, classification of objects and tracking. The techniques developed are commonly used for robot vision.

Computer Graphics The main aim of Computer Graphics consists in making computer generated videos, many of them being highly realistic. For this purpose a great variety of representations of all sorts of objects and materials have been designed. There has also been a huge effort in generating physically plausible simulations. Even though their main aim is not to generate exact predictions or generate the videos in real time, frequent optimisations of their algorithms are developed to improve in both directions. Their main techniques are used to visualize a wide variety of computer simulations in science and engineering.

Virtual Reality It could be considered an extension of computer graphics, however their simulations aim to be produced in real time and they incorporate more

senses like sound and touch. Their main fields of application are the video game industry and virtual surgery simulation in medicine.

Robotics The design and building of autonomous machines, used widely in the industry to handle all sorts of solid and deformable objects, specially textiles [45]. Here we are interested in robots of general purpose, with actuators (an arm) and sensors (camera and force sensor). We will take the required machinery and hardware as granted.

Additionally, for a long time, the **physical sciences** have specialized in studying all sorts of materials and phenomena occurring in nature, developing the most mature and varied techniques for modelling them, with branches like thermodynamics and material sciences. Consequently, the first approaches for modelling kinematics, dynamics and deformability of objects in computer graphics and virtual reality have been based in physics. It is important to highlight that physicist have developed methodologies for generating not only descriptive, but also causal models of phenomena, which empowers them to make predictions by nature. On the negative side, these models tend to oversimplify scenarios; their applicability to real situations, where better approximations are necessary, requires great analytic efforts, that force them to become specific.

As a response, there are techniques in computer science that aim to learn to make predictions, without recurring to explicit models, like neural networks or Bayesian approaches. However, when using them it is necessary to understand the nature and structure of the problem; a knowledge that can be fostered by physics. Because of the mature stage of physics, its study of modelling, and its successful integration into

virtual environments, we chose to begin with a physics based method, and leave the exploration of other techniques for future work.

2.5 Summary

There are several phases a robot can go through while interacting with a deformable object: identification, representation, classification, tracking, prediction, manipulation and planning. Of these, our main focus is prediction, concretely, prediction of the object's deforming behaviour, but will still require simplified versions of some of the others.

Our research scenario considers that the robot will learn about the properties of the materials by pushing them and observing their deformation. The robot will have access to information about the applied forces through a sensor in its effector, and about the deformation of the shape from a digital camera. The materials to be considered are representative of elastic and plastic behaviours: a dish-washing sponge and a block of putty.

There are several fields within computer science that can contribute to solve this research scenario, particularly: computer vision, computer graphics, virtual reality and robotics;but there are others like system optimisation, artificial intelligence and machine learning with relevant information.

The following three chapters will introduce the problem as studied in physics. In the following two, for practical purposes, the study of the simulation of deformable objects will be divided in to two aspects:

- Representation of the shape and its deformations.
- Simulation of the dynamics by taking into account interactions with external forces.

The most relevant works will be presented according to the type of contribution they make. Some may be mentioned more than once.

The following next chapters, Chapter 6 and Chapter 7, present the actual review of relevant work for robotics. They cover shape representation of deformable objects and methods for simulating the deformation, respectively. This review is far from being exhaustive. Its main purpose is to identify the main techniques that appear in the literature and that can contribute to a model that allows for making predictions. It is important to highlight the fact that a great majority of those works do not necessarily consider making predictions or consider it weakly. However, they can and some are used by further research that focuses on prediction. This is the reason why they are included. It is also quite possible that a good system for making predictions will require combinations of different works presented there, as will be seen in Chapter 7.

Chapter 3

The Difficulty of Studying Deformable Materials

This chapter presents an overview of the difficulties encountered when identifying [Section 3.1 and Section 3.2], characterising [Section 3.3 to Section 3.5] and using materials [Section 3.6] with a wide variety of properties [Section 3.7]. A general framework taken from thermodynamics is presented since relevant methods to model deformable objects can be better understood within this frame [Section 3.8 and Section 3.9].

3.1 Understanding Different Materials

Working with different materials is a challenging task because of the infinite variety of possible materials, states and phenomena, discrete and continuous, related to them.

When a cognitive agent, natural or artificial, encounters a complex natural environment, it needs a way to order, analyse and abstract information from the data it receives, before it can make use of it for planning (as opposed to agents that merely react) [12].

3.2 Segmentation of the Environment into Objects

One familiar approach for analysing a scene, is to begin by trying to segment the environment into objects, then analyse each object, in different degrees of detail, with its properties, behaviours and affordances and recombine those elements by considering the interactions between objects. However, for the general case, even the segmentation step is not straightforward. To be able to separate the environment into objects there must be a way to define a boundary between them. For some solid objects, even if deformable, this boundary is naturally established, as with balls of plasticine, dolls and towels. However, for other components of the environment, the borders are not so easy to identify, as in the case of liquids in motion such as honey, water or shampoo; aggregates of solid particles like sand, flour or salt; or combinations of these.

Moreover, depending on the sensors of the agent, other components are even hard to detect, but play an active role in the processes taking place. For example force sensors are required to distinguish materials that look similar but have different hardness due to their internal structure. Finally, other components are hard to categorise without some deep knowledge of their nature, like fire, light and shadow, lightning, and the electromagnetic waves that carry TV signals and produce deformations in the texture of a screen. Furthermore, it is possible to find substances in combined states, like water in a triple point where it coexists as ice, liquid and gas; or processes of combinations of

substances, like powders dissolving into liquids, melting solids, or liquids being mixed with other liquids.

3.3 Characterization of Substances in Terms of Their Properties

In order to make sense of all of the great variety of materials and processes, *humans have classified the wide variety of known substances, their properties and their interactions, according to various schemes. They have progressively looked for underlying structures and theories of interaction, that eventually could be used to explain all observed phenomena and, furthermore, make predictions which lead to a controlled manipulation of materials and their properties*.

3.4 Inference of an Underlying Structure

To make more explicit the difficulties that we face, when trying to make a robot learn *autonomously* about new materials, we explore further this human approach. Basically, a pattern that appears through human research consists in exploring the known materials, finding a simple set of elements (already visible or hypothetical) with their properties that, when combined in different arrangements, can generate all other observed materials with their properties. If there are exceptions, the model is considered to be incomplete, and the search will continue until the known exceptions are covered as well. If the set of elements is correct, new combinations of them should predict the existence of new compounds and their properties. Broadly speaking, these elements are

sought through finding regularities in the properties across a wide variety of materials. Frequently, it has been the case that those so called "elements" can also be explained in terms of smaller components.

However, the selection and evaluation of possible candidates is not easy. For example, in most civilizations it was considered, for several centuries, that all matter was made of five elements: earth, water, fire, air and aether or void; and everything else was understood in terms of combinations of these. Even more, Tibetan philosophy went further and affirmed that "every physical, sensual, mental and spiritual phenomenon was made of these" [70]. This illustrates how even the establishment of a division between domains can be a polemic and delicate cognitive process.

It is clear that it took a long time for humans to discover the atomic and molecular structure of matter, some time more to understand the interactions of atoms and molecules, how to control the creation of particular structures, and how their macroscopic attributes emerge from the underlying microscopic composition of the material. Now the study of materials has reached a mature status and humans have acquired the ability to design a material that will satisfy a predefined set of required characteristics, on the basis of a model of its atomic and molecular structure [35].

Currently, it has been found that the so called "elements" of the periodic table, are in fact made of smaller components[i]. Curiously, the new "table of elements" has been reduced in size and includes particles that can form all other seen particles and the forces of interactions. In principle, all the processes and materials that we can observe are derived from these *fundamental* particles and are a result of their interactions, although in practice that claim still requires to be verified. A common test to verify a theory like this, consists in demonstrating that other previously successful theories are contained

[i] Therefore the old table is now better called "table of chemical elements".

within this one, as particular cases, under the appropriate conditions. For example, when there are enough particles in an object to form a macroscopic structure, the equations should be reduced to the ones of the corresponding previous model. In addition, the new theory should explain or even predict phenomena that were not covered by the previous ones[ii].

3.5 Emergence of the Properties of Materials from the Structure of Matter

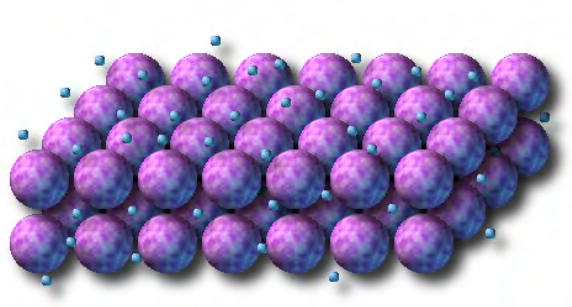

FIGURE 3.1: *The crystalline structure of a metal is given by its lattice of positive ions [purple spheres], which have been located in positions of quantic equilibrium; while the electrons of the last orbital of its atoms, are shared by the whole piece of metal and are easy to move [blue spheres]. This structure holds the metal together and grants it its high electric conductivity.*

Thank you to the theories developed, we understand that the properties of chemical elements, and of macroscopic materials like metals, can be derived from their quantic substructure, given by the arrangements of the particles that constitute them. For

[ii]However, for complex objects or systems, it is usually the case that it is not possible to solve the equations and give this step rigorously, due to their mathematical complexity [46].

example, the high electric conductivity of metals is due to the high number of free electrons in the lattice of atoms that form the material [Figure 3.1]. Furthermore, it has been possible to dissociate our perception of the qualities that made earth, water, air and fire seem to be elements, and instead relate them to states of aggregation of matter (solid, liquid, gas and plasma, respectively), which are due to differences in the nature of thermal motion of the molecules (or atoms), that constitute matter, and their interaction [1].

3.6 Design of Materials

With knowledge of the components of matter and their interactions, it is possible to analyse or even design materials with specific properties, as it is done in the field of *materials science*. However, for the most demanding tasks, this science makes use of models within the field of quantum mechanics, that are highly complex. In most cases, they will use what they call *mean-field techniques*. In material science a mean-field technique consist in using models that bypass the quantum nature of matter and yet can provide good approximations [48]. They do not model the structure of matter in detail, but instead merely approximate it with another structure, according to the level of detail that is required. These models are not complete, in the sense that not all the properties of the material will emerge from them, they approximate only a subset of them. When making use of these models it is important to remember the domain where they are valid. Because of this, there is no general solution for all problems and research about specific cases is done on demand.

3.7 Identification and use of Properties of Materials

Notwithstanding, it is important to remember that even in pre-history humans were highly capable of handling a great variety of materials, or even manufacturing substances with special characteristics, long before the discovery of modern models, as exemplified by Wadley et al. [91]. The precise set of underlying cognitive mechanisms that allowed this remains unknown. Furthermore, even now, a phenomenon like superconductivity has not been explained in full, and yet applications of it are already under consideration. This exemplifies that the identification and use of a property does not require a precise model of how it happens.

This implies that humans had the ability to sense[iii]:

- Physical Properties:
  - ⋆ Colour
  - ⋆ Weight
  - ⋆ Solidity
  - ⋆ Sound
  - ⋆ Friction
  - ⋆ Fluidity
  - ⋆ Pressure
  - ⋆ Viscosity
  - ⋆ Temperature
- Chemical Properties:
  - ⋆ Flavour
  - ⋆ Smell

Humans were also capable of identifying processes involving all of the previous aspects and, probably, even more. Thanks to those perceptions and the intellectual development that followed, humans can be capable of identifying, using and designing a wide variety of substances and materials on the basis of their personal experience

[iii] This list does not intent to be exhaustive.

(either direct or through teaching) or on abstract models. The question now is, how much of this can be included in the robot? Where to begin? The following sections expose traditional ways to divide the problem of studying matter, energy and their interactions in the environment, while delimiting the scope of this research. This vision focuses on the "description of large-scale characteristics by means of a few of its measurable properties, suggested more or less by our sensory perceptions, [which] constitutes a macroscopic description" while remembering that those "few directly measurable properties... are really averages, over a period of time, of a large number of microscopic characteristics" [95]. For the purpose of dividing the problem of studying materials and processes, it is highly useful to resort to the traditional distinction between systems in equilibrium (static) and dynamic systems.

3.8 The Static Case: Equilibrium

To describe a macroscopic system, like a solid undergoing deformation, it is required to determine the coordinates necessary and sufficient. The value of those coordinates at a given time describe the ***state of the system***. "When these coordinates change in anyway whatsoever, either spontaneously or by virtue of outside influence, the system is said to undergo a ***change of state***" [95]. Within thermodynamics it is possible to distinguish between three types of equilibrium:

1. Mechanical Equilibrium. When there is no unbalanced force or torque in the interior of a system and also none between the system and its surroundings.

2. Chemical Equilibrium. When a system in mechanical equilibrium does not tend to undergo a spontaneous change of internal structure, such as a chemical reaction, or a transfer of matter from one part of the system to another, such as diffusion or solution, however slow.

3. Thermal equilibrium. When there is no spontaneous change in the coordinates of a system in mechanical and chemical equilibrium when there is no exchange of heat between the system and its surroundings. In thermal equilibrium, all parts of a system are at the same temperature, and this temperature is the same as that of the surroundings[iv].

Changes of state cease when equilibrium is reached. When the conditions for all three types of equilibrium are satisfied, the system is said to be in a state of ***thermodynamic equilibrium***. Observe that, while the system is out of equilibrium, the macroscopic variables used to describe it may not be well defined for the system as a whole. For example, as a gas expands inside a tube, the pressure at different points will be different. A way to approach this problem is to model the substance as made of many small elements, each for which the macroscopic variable is well defined.

3.8.1 Equation of State and Constitutive Equations

The first advantage of working with systems in thermodynamic equilibrium is that it is possible to find an equation of equilibrium, called an ***equation of state***, which connects the macroscopic variables, that do not depend on time, and makes one of them dependent on the others [95]. These variables are called *thermodynamic variables*. Every

[iv] Taken from [95], pp 29.

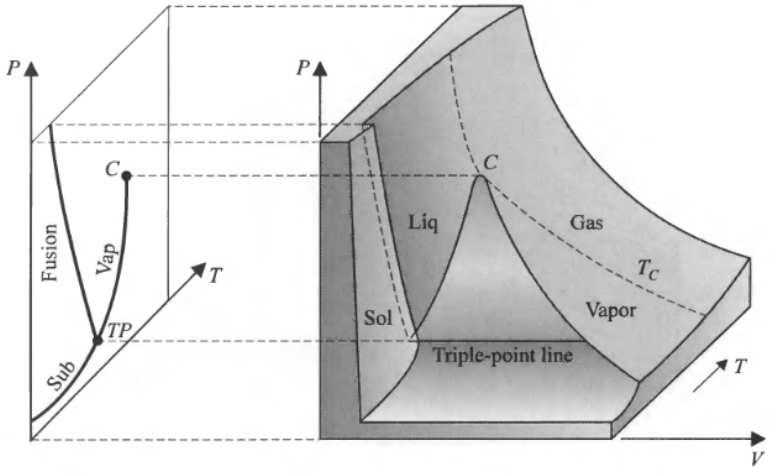

FIGURE 3.2: *PVT phase diagram for H_2O, which contracts while melting. Reproduced without permission from [95].*

thermodynamic system can be characterised by its equation of state, although in some cases, it may be so complicated that it can not be expressed in terms of simple mathematical functions. These equations are either determined by experiment or derived from molecular theory. A particular case are the ***constitutive equations***, which relate pairs of mechanical variables, one of them being a type of force and the other one a type of deformation [71].

A very illustrative example of an equation of state is the derived Pressure P– Volume V – Temperature T phase diagram of H_2O in Figure 3.2. The phase diagram clearly shows the relations between the thermodynamic variables PVT and illustrates the different aggregation states (phases) of the substance and transitions between them. Here, the behaviour in equilibrium of the substance is fully characterised. One way in which the robot could learn the behaviour of a material, could be to perform ordered experiments that would allow it to sample and approximate its equation of state. The figure above

has a very typical shape, which means that the robot could try to generalise, by interpolating between key measurements, the overall relation according to this or other templates. However, it may not be straightforward for the robot to perform enough experiments under controlled conditions. Nevertheless, this diagram is key to understanding many of the ideas explained below.

3.9 Quasi-static Processes

A quasi-static process is an idealised situation in which the external forces acting on a system are varied only slightly so that the unbalanced force is infinitesimal, and the process proceeds infinitesimally slowly. During a quasi-static process, the system is at all times infinitesimally near a state of thermodynamic equilibrium, and all the states through which the system passes can be described by an equation of state, in terms of thermodynamic variables. In this case, dissipative processes are ignored and processes are reversible. Under these conditions, it is possible to imagine the equation of state solved for any coordinate as a function of the others. Frequently, for simple systems, three variables will be involved:

1. An extensive property, that depends on the mass. For example, the length of a wire, area of a metal sheet or volume of a liquid.
2. Two intensive properties:
   (a) The temperature.
   (b) Another variable, like the tension or pressure.

For example, if the system is a wire, it could be desirable to express the length of the wire L, as a function of the tension in it $\mathfrak{T}$ and the temperature T:

$$L = L(T, \mathfrak{T}) \tag{3.1}$$

3.9.1 Characterising the Material with Differentials

If the wire undergoes a quasi-static process, which is an infinitesimal change from one state of equilibrium to another, that infinitesimal change can be written as an exact differential of the function L:

$$dL = \left(\frac{\partial L}{\partial T}\right)_{\mathfrak{T}} dT + \left(\frac{\partial L}{\partial \mathfrak{T}}\right)_{T} d\mathfrak{T}, \tag{3.2}$$

where the partial derivative $(\partial L/\partial T)_{\mathfrak{T}}$ means the infinitesimal change in L for an infinitesimal change in T *with* $\mathfrak{T}$ *held constant* during the operation of differentiation.

Observe that, if the value of the two partial derivatives can be obtained from experiments, then it is possible to integrate the state equation from the expression above. For this reason, there has been a big emphasis in estimating these values. For convenience, the following coefficients, from which it is possible to recover the partial derivatives, have been defined:

Linear expansivity α: It expresses the amount of change in the length of the wire as the temperature is changed:

$$\alpha = \frac{1}{L}\left(\frac{\partial L}{\partial T}\right)_{\mathfrak{T}} \tag{3.3}$$

Isothermal Young's Modulus Y: It expresses the tension produced in the wire when its length is modified by a certain amount, at a given temperature:

$$Y = \frac{L}{A}\left(\frac{\partial \mathfrak{T}}{\partial L}\right)_T \tag{3.4}$$

from where the ***average Young's modulus*** E is derived:

$$E = \frac{\Delta \mathfrak{T}/A}{\Delta L/L} \tag{3.5}$$

Analogous pairs of derivatives can be measured for other simple systems like electrochemical cells, paramagnetic rods or hydrostatic systems. For this research we have decided to focus on mechanical properties. In this manner, the following chapter will delve into their analysis, continuing with the approach developed so far. That is, it will be explained how to obtain as much information as possible about the mechanical variables of the equation of state of some deformable solids.

3.10 Summary

This chapter presented several difficulties encountered when trying to observe, characterise and make predictions about non-rigid materials. Some of these difficulties arise when defining the frontier between different types of materials, their different states of aggregation and the transitions between them. In order to characterise different materials according to their properties, their behaviour was divided in two cases: equilibrium and dynamic.

When a system is in equilibrium it can be characterised by its macroscopic thermodynamic variables, through the equation of state that relates them. To begin the study of dynamic systems a quasi-static assumption is made. With this assumption it is possible to consider that the system moves from one state of equilibrium to another one. The partial derivatives of the equation of state can thus be defined. Of these, the Young's modulus is of particular interest for the study of mechanical properties of deformable solids.

The following chapter deepens the characterisation of deformable solids through their equation of state and their partial derivatives, an approach used widely in materials science. The study of these properties will aid us to select and define appropriate models that may allow prediction of the deformation of the materials upon interaction.

Chapter 4

Mechanical Behaviours of Deformable Materials

This chapter presents how to characterise deformable solids according to their mechanical properties, since those are the target of our research. Relevant terminology, originating from the field of materials science, but of common use in the literature about modelling of deformable objects, is introduced here. This chapter also illustrates the wide variety of elastic, plastic and transitional behaviour that different materials exhibit.

4.1 Definition of Stress and Strain

The behaviour of deformable objects can be characterised by their response to the application of particular types of forces [35, 38]:

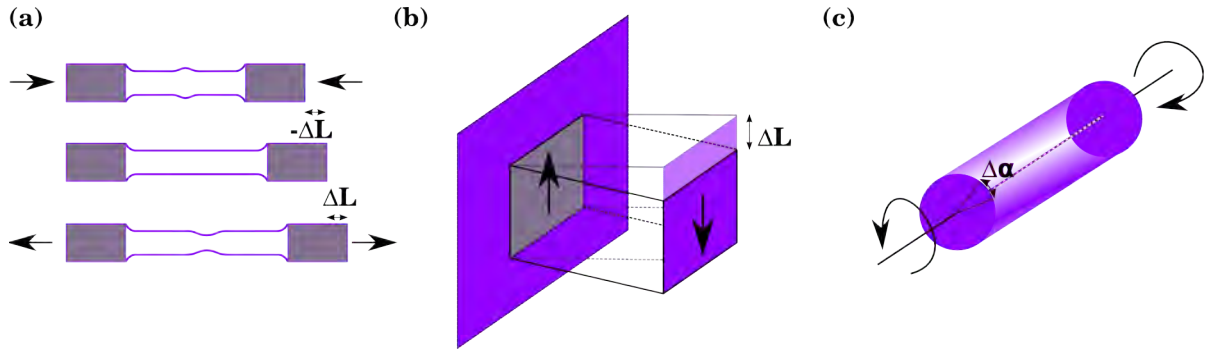

FIGURE 4.1: *(a) Top: compression stress; middle: test piece; bottom: tensile stress. (b) Shear stress. (c) Torsion stress.*

Stress is the applied mechanical force per unit area. Three types of stress tests are distinguished:

1. Tensile or compression stress – the stress is equivalent to a pressure (either positive or negative) [Figure 4.1(a)].
2. Shear stress – Two forces are applied in opposite directions but in different ways and at different points of application [Figure 4.1(b)].
3. Torsion stress – Two torques are applied in opposite directions but in different ways and at different points of application [Figure 4.1(c)].

Strain refers to the proportional deformation produced in a material under the influence of *stress*. It is the numerical ratio of deformation length divided by original length. We will represent it:

$$Strain = \frac{\Delta L}{L} \tag{4.1}$$

To characterise a material properly, by acquiring its equation of state from experiments, these quantities are measured on a predefined standard test piece [38] under controlled conditions [Figure 4.1(a)]. Whereas any animal or robot encounters new materials

mostly under poorly controlled situations, consequently the measurements they can take to generate their model of the material can not be equally precise. Nevertheless, the basic variables to be measured can be quite similar and the variation in the outcome should not affect significantly the performance of the agent.

4.2 Types of Deformation

Upon the application of stress, the same material may show different qualitative behaviours at different stress levels. The separations between these behaviours may or may not be evident at all. At one extreme, the response may be **elastic** – strain disappears when the stress is removed – while, on the other, it may be **plastic** – the material may remain as a coherent whole but fail to return to its original form when the stress has been removed. The atomic nature of each type of behaviour is different, each corresponding to a different type of strain [38]:

Elastic strain, which is reversible and disappears when the stress is removed, the material goes back to its original shape. Atoms are displaced from their initial positions by the application of stress, but when this stress is removed they return to their initial positions relative to their neighbours, provided that the strain has been of an elastic nature. The amount of elastic strain undergone by a particular material is characterised by its **Young's Modulus**, as defined by Equation 3.4. Strain is roughly proportional to the applied stress and, for practical purposes, the material obeys **Hooke's Law** [Figure 4.2(a)]. This states that, *for an elastic body, the strain produced is directly proportional to the stress applied*. In this case, the constant of proportionality is the inverse of the Young's Modulus.

Elastic strain has its microscopic origin in the electric forces that appear in the lattice of ions or molecules, of which the material is composed[i], when these are moved from their equilibrium positions. Thus, the stretch modulus is proportional to the force (stress) produced when an atom or ion is displaced by a fixed distance (strain).

Plastic strain, which results when a material is stressed to the extent where its elastic limit is exceeded. It coincides with a movement of the atoms within the structure of the material into permanent new positions of equilibrium, with respect to neighbouring atoms. When the stress is removed, only elastic strain disappears and any plastic strain produced is retained [Figure 4.2(b)].

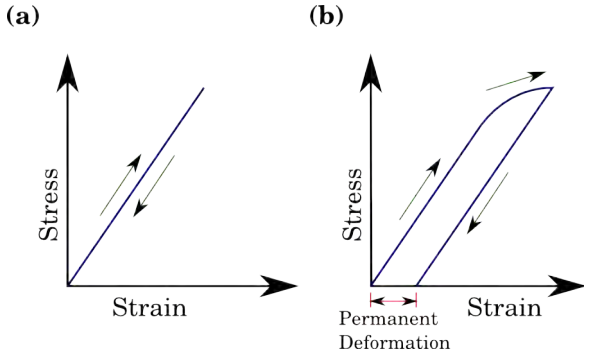

FIGURE 4.2: *Typical stress-strain diagrams for elastic and inelastic materials. The arrows indicate the direction in which the magnitude of the stress is modified during a test. (a) Shows the typical elastic behaviour: as the stress is removed, the material recovers its shape. (b) Shows an inelastic material; when the stress is removed, part of it recovers its shape, but part remains permanently deformed.*

One material can undergo both types of strain when stress is applied, thus giving rise to mixed behaviours. When the material is not fully elastic it is said to be **inelastic**. It can

[i] Ions are particles with net electric charge.

show *hysteresis*[ii], because part of it is elastic, while the other one "flows", permanently altering its attributes. An inelastic material may be identified either as a *viscoelastic material* – when deviations from the elastic behaviour are large, and it can become anything between an elastic material and a viscous fluid – or as a *plastic material*. If the stress is high enough, the material will *break*, *fail* or have a *fracture*.

4.3 The Phases of a Material

Each distinctive behaviour of a material is known as a *phase*. Each phase of a material corresponds to a form of macroscopic presentation or type (elastic, plastic, fluid, etc.). The phase allows a characterisation of the material's properties, that depends on external macroscopic variables such as specific environmental conditions or the material's history. Therefore, a phase can be modified when external parameters are changed, but remember that its characterisation will only be well defined if the process is quasi-static [Section 3.9].

To assert that a sample of a material is of a certain type, it is necessary to specify the interval of environmental conditions in which its current phase is *stable*. Often even this is not enough. A material's degradation may allow two samples of the same material, in different phases, to coexist under the same environmental conditions. In this case, data about the sample's history is also required.

[ii]Hysteresis is a phenomenon where the value of a physical property of a material depends on the processes to which it has been submitted before; that is, on its history.

4.4 Stress/Strain Diagrams

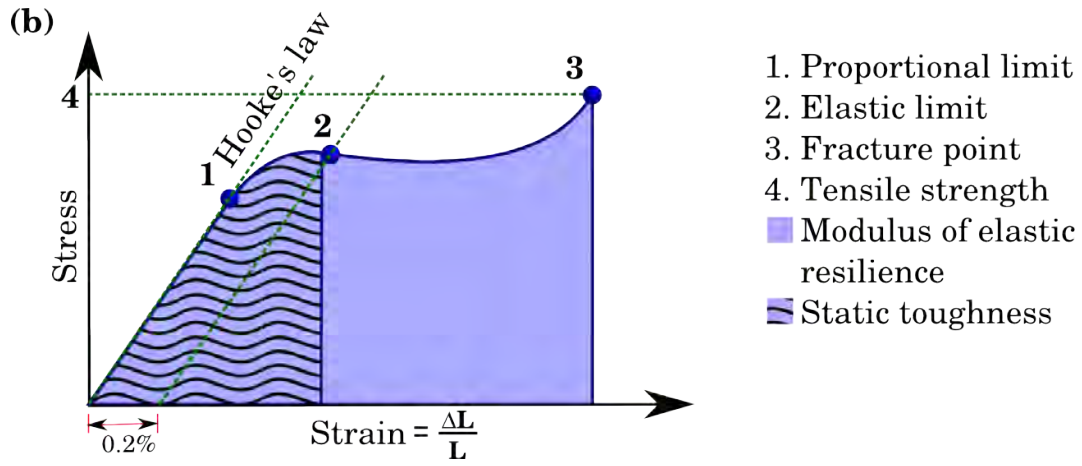

FIGURE 4.3: *Ideal generic stress/strain curve. 1. Limit to the elastic behaviour according to Hooke's Law. 2. Stress from which the material is no longer elastic. 3. Stress at the maximum of the stress-strain curve. 4. Value of the stress at point 3. [wavy]: Resilience. [coloured]: Toughness [35].*

A stress/strain diagram shows the amount of stress that a material undergoes given its current strain. It illustrates part of its equation of state, obtained experimentally [Section 3.8.1]. These diagrams are particularly useful to study the phases of a material and the transitions between them. These transitions are different for every material. Nevertheless, it is still possible to depict an ideal generic stress/strain curve to illustrate the most relevant phases and their transitions [Figure 4.3]. The first section of the curve is the domain of Hooke's Law. Between the points 1 and 2 of Figure 4.3 there is a region where the material still behaves elastically, but the deformation is not directly proportional to the applied tension. In material science the end of this region is, by definition, marked at the stress at which the stress-strain curve intersects a parallel to the line determining the elastic behaviour, that passes through the point of zero stress,

and a fixed strain (usually 0.2%). Between points 2 and 3, in addition to the elastic deformation, there is plastic deformation, until the material begins to *fracture* at point 3, with the value of the tension marked with point 4 [Figure 4.3]. The wavy and coloured areas below the curve show the energy absorbed by the material during the deformation process. The *resilience* is the capacity of a material to absorb elastic energy when it is strained and to release it when the stress is released. The *toughness* is the capacity of a material to absorb energy, elastic or not, before the fracture point [35].

4.5 Types of Transitions between the Elastic and Plastic Phases

Some materials have been classified according to characteristics in their stress/strain diagrams as **ductile** and **non-ductile**. Ductile materials exhibit an **elastic limit** beyond which plastic deformation occurs. Non-ductile materials undergo little or no plastic deformation before fractures appear, they are **brittle** and break easily [Figure 4.4(i)]. The maximum stress which a material can withstand before plastic flow sets in is known as **yield strength** Y [Figure 4.4(iii)]. Some materials have a very marked yield point, that separates the elastic phase from the plastic one [Figure 4.4(iii)]; for others this limit is not very well defined. When such unclear yield stress is relevant, a substitute value is derived (the **proof stress** σP), and it is that stress which will produce a permanent (plastic) extension of a constant percentage (0.1% or up to 0.5%[iii]) [Figure 4.4(ii) and Figure 4.4(iv)]. After the force being applied reaches the substance's elastic limit, the Young's Modulus is not a constant but, if a value is required, an equivalent section of

[iii] Remember that the strain is given by Equation 4.1 and the values are measured in the test piece made of the material under study.

the stress/strain curve can be approximated by a secant or a tangent, and the value of the modulus will be the value of the slope. The tensile strength [Figure 4.3] is the maximum stress that a material can withstand while being stretched or pulled before necking, which is when the sample's cross-section starts to significantly contract [See Figure 4.1(a) piece at the bottom].

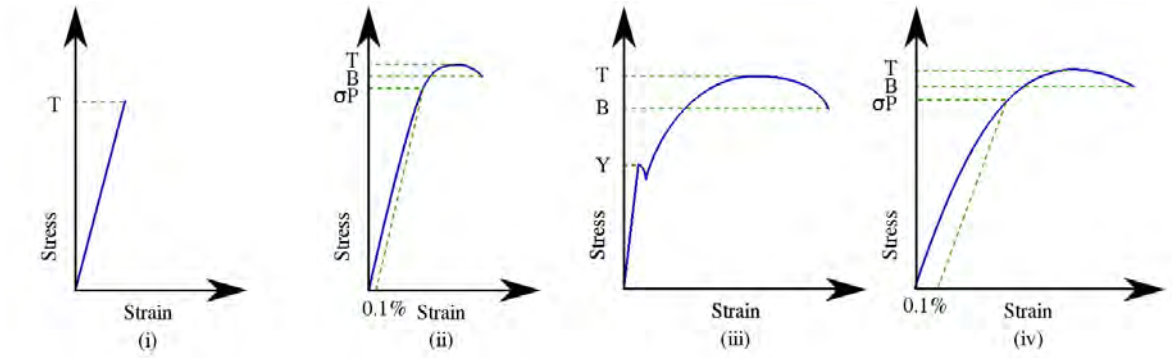

FIGURE 4.4: *Representative stress/strain diagrams for various types of materials. T: tensile strength; B: breaking strength; Y = yield stress and σP = proof stress. (i) Non-ductile material. (ii) Semi-ductile material. (iii) Ductile materials where the onset of plastic flow is marked by a very definite yield point Y , and (iv) Ductile materials where the elastic limit is not well defined and proof of stress is used instead. [38]*

Observe that, in the general case, only a piecewise continuous mathematical function will be able to describe all of these possibilities, specially due to discontinuities like the yield point [Figure 4.4(iii)]. Is there a simplified structural model that can predict all phases and transitions with a sufficient degree of accuracy for the tasks of a robot? Most characteristics and transitions depend on the specific atomic structure of the material (e.g. the arrangement of its atoms or molecules in a crystalline structure and the presence of impurities in it), which makes this approach likely to succeed only in part. Alternatively, the robot could make rough and piecewise approximations of the observed behaviours, instead of deriving them from the structure of the material.

However, the approximations may not be fully adequate for making generalisations and predictions.

Contributing to the diversity of the problem, there exist some materials that do not obey Hooke's Law in any phase, their stress-strain diagram shows no region where a linear relationship applies. In these cases, engineers still make a linear approximation, by approaching the first part of the curve with its secant or its tangent [Figure 4.5 or Figure 4.4(ii) respectively] [38]. This case adequately illustrates how the real behaviour can be approximated with a much simpler model. Still it is necessary to make the remark that, even though the precise behaviour is not included in the model, humans were still able to perceive/measure, identify and approximate the behaviour by the simplifying model.

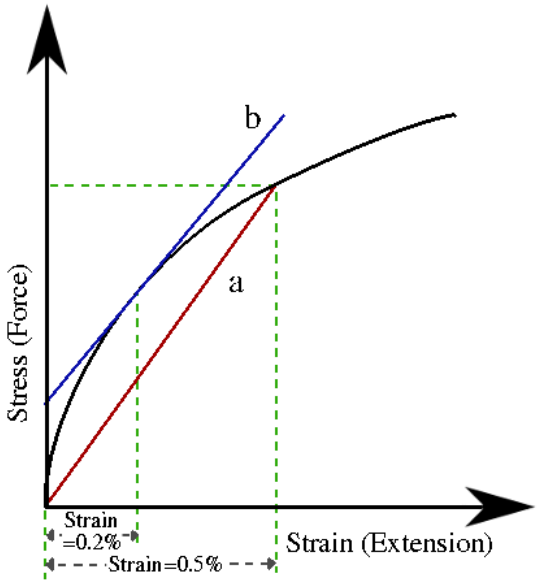

FIGURE 4.5: *a = Secant modulus, b = Tangent modulus.*

4.6 The Perspective of a Robot

A robot for domestic use, for example, should be prepared to perceive and manipulate objects that exhibit the different behaviours presented above, even though it may not be able to generate an exact model. In this work we explore how it could perceive and learn a model, that it could use later, to predict some of the behaviours presented in this chapter. Particularly, we will be concerned with the modelling of the pure elastic and pure plastic behaviours, but being conscious that the region of transition between these two remains unmodelled. Certainly not all possible behaviours have been presented in this chapter. The production of stress/strain diagrams presupposes mostly homogeneous materials. Most real life objects are made of non-homogeneous materials. Therefore, more complex behaviours are expected from real objects and our models are expected to fail; specially in uncontrolled interactions like a normal pushing/deforming action performed on everyday objects, as opposed to a test piece under quasi-static stresses.

To the sources of error due to the approximations made by the models, we must add sensor noise, bad measurements due to uncontrolled experimental situations and non-homogeneity in the materials. All of these aspects will influence the different stages of processing in the robot: perception, learning and modelling. However, it is outside the scope of this thesis to model these aspects as well.

4.7 Summary

The most relevant terminology for the characterisation of deformable solids has been introduced. That includes the definition of phases of a material, stress, strain, elasticity,

plasticity and viscoelasticity. The stress/strain diagrams were used to illustrate the types of transitions between the different phases of the material, and how, on some occasions, more complex behaviours can be approximated by simpler ones (e.g. when elastic materials without a linear region are approximated, up to a predefined threshold value, with a linear equation, anyway).

The next step is to present how some of these behaviours can be modelled to make predictions. The following chapter introduces the mathematical elements required to build and understand the most relevant models, giving special attention to the mass-spring models that were selected for this thesis. Later, Chapter 7 makes a general review of the related research by presenting the different families of dynamic models found in the literature and how they have been used.

Chapter 5

Mathematical Models of Deformable Objects

Chapter 3 exposed informally how the behaviours of different materials can emerge from current models of matter. These models pay close attention to the particles that compose a material, the arrangements that emerge from their interactions (that is, the structure of the material) and how these explain its macroscopic properties. This approach can be particularly useful when designing new materials. However, it was mentioned as well, that sometimes the most exact models are so complex, mathematically speaking, that they can not be used in practice to obtain all behaviours for all cases. Frequently it is necessary to reduce the level of detail, in accordance with the properties or behaviours of interest. This is the approach that will be taken here.

In this research we are interested in robots that can learn to make predictions about the macroscopic mechanical behaviours of deformable solids, from macroscopic sensory observations. For this reason, Chapter 4 presented a review of the most noticeable

ones, most specially elasticity and plasticity. This chapter introduces a set of useful abstract elements for building mathematical models of materials with those properties and behaviours. Many of these concepts appear in the related literature, often unexplained, and will be in constant use for the rest of this thesis.

5.1 Mathematical Modelling for Physicists

Creating mathematical models of physical phenomena, in general, can be described as detecting and exploiting analogies between those phenomena and mathematical entities and procedures. Each term in an equation represents something in the phenomenon to be modelled. That something can depend on elements/actors that interact with each other (e.g. objects colliding with each other, surfaces producing friction or wind slowing down movement) or approximations of the characteristics and dependencies of those interactions (e.g. Does the deformation of an object that collides with another depend on the applied force? What kind of dependency does it have? Is the displacement of the particles of the object directly proportional to the force? Or is the velocity, or the acceleration the one that is proportional to the force? Could it be proportional to the square of the force? Does it depend on the temperature of the object? Or does it depend on combinations of the previous ones? What type of combination? linear? exponential? Are those terms enough? How accurate is the proposed model? Should it be further expanded or replaced? Are there other variables that could/should be taken into account? Would the model improve its predictions?).

Consequently, if the model is good – that is, the analogy between the proposed mathematical formula and the physical phenomenon is strong – the more knowledge we obtain about the mathematical terms, through the manipulation of the equations, the

more knowledge is gained about the overall physical phenomenon. To write an equation for a phenomenon, the first step is to identify the actors: Who takes part in it? What are the variables involved? Are those actors continuous or discrete? Is an actor being studied in isolation or is the study about interactions between actors? What are the relationships between them? How can we describe them?

In some cases, there are relationships that have been very well established and help in building the mathematical model. The equation of Newton $F = ma$, where F is the force, m the mass and a, the acceleration, is a basic example. However, when the phenomena are complex, the space of possible combinations that must be explored grows. Such is the case in the study of properties of thermodynamic materials [95].

5.1.1 Modelling of Deformable Objects

Deformable objects are highly complex in nature. To make a simple yet reliable model of them, it is necessary to identify the main aspects that characterise them. Those aspects should be simple to represent and easy to work with. They include representations for the shape of the object and how this one is transformed over time, as well as the processes and/or interactions that cause such changes. They will form the backbone of the model. Further detail can be added afterwards if the key elements were correctly identified. There is a "general to specific" approach in physics which helps us to generate these models. It consists in trying to represent the most general case and subsequently adding more and more constraints to simplify the problem. Once the scenario is simple enough, a model is built. Once the simplest model has been understood, the constraints are removed one by one, and the model is thus extended to

cover more cases. This has been done successfully in classical mechanics for solid objects and it provides a strong basis for studying deformations of those solids, through the addition of new elements.

Now, it is important to be aware of the reaches and limitations of a model. A model is typically defined by a set of elements, either discrete or continuous, that interact with each other. It is feasible to imagine the space of all the possible shapes, properties and behaviours that can emerge or be represented by the set of elements and interactions in a model. Does that space potentially cover all the shapes, properties or behaviours of the real objects the robot encounters? The answer to this question is determined by the elements and rules of interaction we have selected. As it was explained in Chapter 4 it would be necessary to use combinations of different models to cover the full spectrum.

The following sections of this chapter present some traditional elements for models of matter. The presentation includes small descriptions of their behaviour, in the hope that this will illustrate how and where they are likely to succeed or fail. The first four sections are based mainly on [34].

5.2 Particles

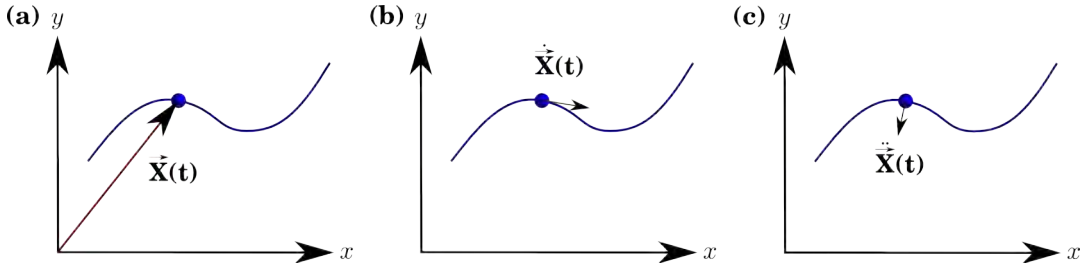

FIGURE 5.1: *(a) Position $\vec{X}$, (b) velocity $\dot{\vec{X}}(t)$ and (c) acceleration $\ddot{\vec{X}}(t)$ vectors for a 2D path in the plane xy. The particle moves along the path as time progresses.*

The simplest objects to model are particles, therefore a great number of models start with them. Particles are idealised as zero-dimensional dots, consequently they can not collide with anything. Each can have a position with respect to a frame of reference[i]. The coordinates of the particle can be given by a vector $\vec{X}$, of as many dimensions as the particle can move through. For example, if the particle moves along a line, its position is one-dimensional; if it moves in a plane, 2-dimensional, etc. This position can be continuously registered through time, forming a curve; given that, for each time, the particle is assumed to be in only one place[ii], this curve is a function parameterised by time $vecX(t)$ [Figure 5.1(a)]:

$$\vec{X}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} \tag{5.1}$$

The velocity is the rate at which the particle changes position in time, it can be associated with the derivative of the function of the position with respect to time $\dot{\vec{X}}(t)$ and is calculated as the vectorial difference between consecutive positions in the curve as the time interval between them tends to zero[iii] [Figure 5.1(b)]. We can continue defining the acceleration $\vec{a}$, as the change of velocity with respect to time $\ddot{\vec{X}}(t)$ [Figure 5.1(c)]. In principle, if it was useful/necessary in some case, depending on the shape of $\vec{X}(t)$[iv], it could be possible to continue defining changes, using derivatives, perhaps ad infinitum.

[i] The frame of reference is delimited by a set of elements, whose relative distances are fixed. One position is taken as the origin, and the position of all other elements is given by their vector distance to it.

[ii] This assumption does not necessarily hold for quantum mechanics

[iii] That is, in the limit $t \to 0$.

[iv] That is, depending on the number of derivatives that are well defined.

5.3 Multiparticle Systems

It is possible to have systems with many particles. These systems include gases, liquids and solids. Which system is described depends on the relations and interactions between the particles. For example, the particles may mutually ignore each other, to model an ideal gas; they may have a volume and interact with each other, like in the Van der Waals model of gases and liquids; they may form solids, if the distance between the particles is fixed; and so on. It is easy to think of atoms and molecules as the interacting particles that form a macroscopic object Therefore, multiparticle systems can have from two to thousands or millions of particles. Frequently, problems become intractable with such high numbers. Fortunately some systems can be reduced to simpler elements, for example:

(a) Spheres seen from a certain distance, and under certain circumstances, can be treated like particles. E.g. the Earth and the Sun when looked from far away, if the estimation of their movement does not require great precision[v].

(b) Some systems can be treated as continuous materials with continuous properties; consequently, differential and integral calculus is adequate to model their deformations. In this case, it can be assumed that the systems is made of small continuous homogeneous elements. E.g. water, rocks, plates, tubes or cylinders made of an homogeneous material, etc.

(c) Other solids can be approximated as having fewer but bigger particles than they actually have. E.g. sand or gas.

[v] As greater precision is required, becomes necessary to take into account their rotation, the relative displacement of their internal components, among many other things.

Rigid solids are the easiest to work with, because the geometrical relations between their components do not change; therefore it is possible to forget about the particles that constitute them and describe their movements by following a few key geometrical points in them. It is not so obvious how to do the same for objects whose shape can be modified without mayor restrictions, like in the case of putty.

5.4 Inertial Systems: Force, Momentum and Kinetic Energy

The elements in the previous section only describe the movement, but what makes a particle move or not move? What makes it change the way in which it moves? To address this fundamental issue, Isaac Newton paid attention to a special set of frames of reference, which are called *inertial*. In an inertial system all elements move at a constant velocity, unless a force is applied to them[vi]. According to Newton, we call *"force"* the element in an interaction that can change the state of movement of an object. This state can be characterised by the product of the mass of the particle m by its velocity $\vec{v}$ and is called the ***momentum*** $\vec{p} = m\ vecv$. The relationship between the force and the change in the state of movement is given by Newton's equation:

$$F = \frac{d\vec{p}}{dt} = m\vec{a} \tag{5.2}$$

[vi]That constant velocity can be zero. If the frame of reference is changed for another one that moves at constant velocity with respect to the previous one, the velocity of the particle with respect to the new frame will not be zero, but will still be constant. An accelerometer at rest in any of this frames will mark zero.

If the system is not inertial, objects will appear to accelerate for no reason. A classical example is an object in a rotating system, like a person in a funfair ride, experiencing the centrifugal force. If the frame of reference is placed on the wheel of the game[vii], the person experiences a *virtual force*. If the frame of reference is placed on the ground, it can be seen that the person experiences a force as a result of the game moving in an accelerated manner, the force producing the acceleration comes from the motor of the game and the surfaces in the game that are in contact with the person.

Another important quantifier of movement is the kinetic energy of the particle. It becomes apparent when considering the amount of work required to move the particle from one point to another, if its mass doesn't change. This is defined as:

$$W_{12} = \int_1^2 \vec{F} \cdot d\vec{s} = \int_1^2 m\frac{d\vec{v}}{dt} \cdot \vec{v}dt = \frac{1}{2}m(v_2^2 - v_1^2) \tag{5.3}$$

Where each of the terms $\frac{1}{2}mv^2$ corresponds to the ***kinetic energy*** K of the particle, either at the beginning or the end of the action.

5.5 Forms of Interaction: Contact vs. Fields

In general, there are two ways of thinking about interactions between objects. In the first one, objects interact with each other through contact. *"Two objects can not occupy the same space at the same time"*[viii], therefore whenever their state of movement pretends to take them to the same location, they, instead, exchange momentum and change their state of movement. That is, they exert a force on each other and thus get slowed down,

[vii] That is, the system rotates with the wheel and the game is considered to be at rest.
[viii] This is a macroscopic manifestation of Pauli's exclusion principle in quantum mechanics.

quickened, deviated, deformed, broken or combinations of these. These interactions can be described through geometry and some predictions about them can be made using the laws of Newton. Nevertheless, the outcome of a collision depends on the strength and type of the binding between the internal atoms or molecules that constitute each object and the strength and type of the collision. If the collision is strong enough, it will produce changes in the state of equilibrium of the atoms in the objects that collided, thus displacing them and producing deformations that could be noticed at the macroscopic scale; it might even fracture them.

Alternatively, the interaction may appear to take place *at distance*. To understand this interaction the concept of ***"fields"*** is introduced. A field describes the area of influence of an object, and how another object would be affected when placed at each point in this area. It consists of a vector field, where the vector at each point, corresponds to the force that a visiting particle would experience. For example, an alpha particle[ix] is surrounded by an *electric field*, and all other particles with electric charge will experience an attraction towards it or repulsion, while they remain within its area of influence, according to their distance from it and their own charge. In this case, the attraction or repulsion is manifested as the apparition of a force, inversely proportional to the square of the distance to the particle [Figure 5.2].

5.5.1 Potential Energy

When the force the particle experiences depends on the position of the particle, and not on time, that force can be expressed as the gradient of a scalar field [95]. Then, the value of each vector of the derived field, at a given position, comes from:

[ix] Helium nuclei, whose charge is positive.

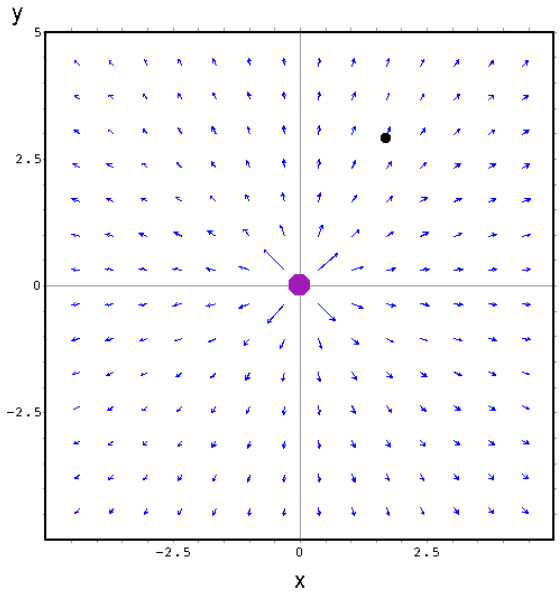

FIGURE 5.2: *The electric field of a positively charged particle will repel another small positive particle, located in the surroundings. The force this small particle will experience depends on its position with respect to the other particle.*

$$\vec{F} = -\nabla U(x, y, ...) = \begin{bmatrix} \frac{\partial U}{\partial x} \\ \frac{\partial U}{\partial y} \\ \cdots \end{bmatrix} \tag{5.4}$$

The effect on a particle placed in this field is that it experiences a force that modifies its state of movement. Besides, when only forces of this type are in action, the amount of change in the scalar field ΔU, corresponds with the amount of change in the kinetic energy ΔK. For this reason, it is said that U corresponds to a ***potential energy***.

Frequently, the concept of *potential energy* is used to express the tendency of a system to find its equilibrium under given constraints. Those constraints push the system in a favored direction, the final configuration is thus implicitly defined by its energy

equations. However, before solving the equations, issues like the existence itself or the stability of a solution must be cared for, since they depend on whether the proposed model is adequate or not. For example, energy based models have been proposed for linear deformable objects (like wires or strings) in [39].

5.6 A Simple Model of Interactions: Ideal Springs

Ideal springs are a very useful construction, inspired by real springs. Real springs have mass, can not be compressed beyond a certain point, get permanently deformed if pulled beyond a threshold distance and wear out with time. Ideal springs, on the other side, represent the tendency of an ideal object to recover its original length when deformed. The force with which the spring pushes or pulls is directly proportional to the amount of deformation [Figure 5.3]. Within certain limits, real springs do behave similarly to ideal springs. Furthermore, it is possible to find the exact solution to the problem of one mass hanging/pulling one spring against a fixed ceiling/wall, expressed as an analytic function of time. This solution is used to approximate and explain several phenomena. Due to its high relevance, the problem is presented below in some detail. A full account can be found in [63].

5.6.1 Simple Harmonic Oscillator

If $y(t)$ is the displacement of the mass from the position of equilibrium at a given time t [Figure 5.3(a)], downward is positive (the spring is extended) and upward is negative (the spring is compressed), the spring will pull the mass with a force:

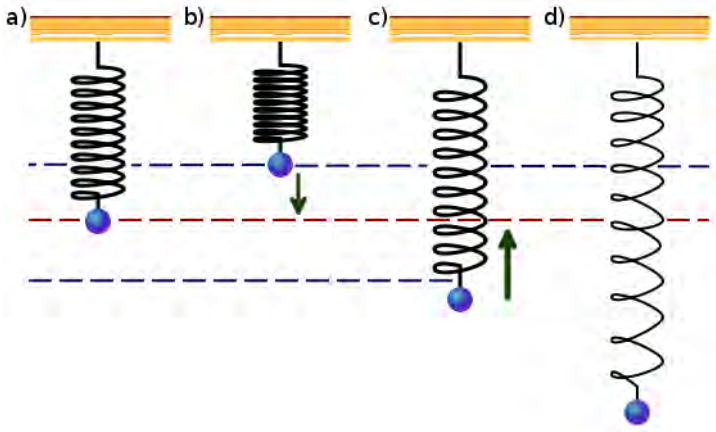

FIGURE 5.3: *a) Position of equilibrium. b) Maximum compression. c) The weight is pulled below the position of equilibrium. For an ideal spring it is irrelevant how far this position is, the equations are always well defined; in the most extreme case, the spring will be inverted. However, for a real spring, the behaviour changes drastically when the weight collides with the compressed spring, the movement also stops being unidimensional. d) A real spring also gets permanently deformed when pulled beyond its elastic limit.*

$$F = -ky(t). \tag{5.5}$$

Where k is the stiffness of the spring. By the second law of Newton, this force F is equal to the product of the mass hanging from the spring m, multiplied by its acceleration. Where the acceleration $\ddot{y}(t)$ is the second derivative of the position $y(t)$, with respect to time t:

$$\frac{d}{dt}[m\dot{y}(t)] = m\ddot{y}(t) = -ky(t). \tag{5.6}$$

Any solution to this equation, where $y \not\equiv 0$, has the form:

$$y(t) = A\cos\left(\sqrt{\frac{k}{m}}t\right) + B\sin\left(\sqrt{\frac{k}{m}}t\right). \tag{5.7}$$

Where A and B are constants that depend on the initial conditions of the problem. For example, if the position at time zero $y(0)$ is known, as well as the initial velocity $\dot{y}(0)$, the position of the mass at any time is:

$$y(t) = y(0)\cos\left(\sqrt{\frac{k}{m}}t\right) + \dot{y}(0)\sqrt{\frac{m}{k}}\sin\left(\sqrt{\frac{k}{m}}t\right). \tag{5.8}$$

This system is a *simple harmonic oscillator*, the resulting function $y(t)$ is a sinusoidal curve whose amplitude, phase and frequency depend on the stiffness of the spring, the mass of the object hanging from it and the initial conditions of the movement. Figure 5.4 shows the graph of this function for some given sets of values.

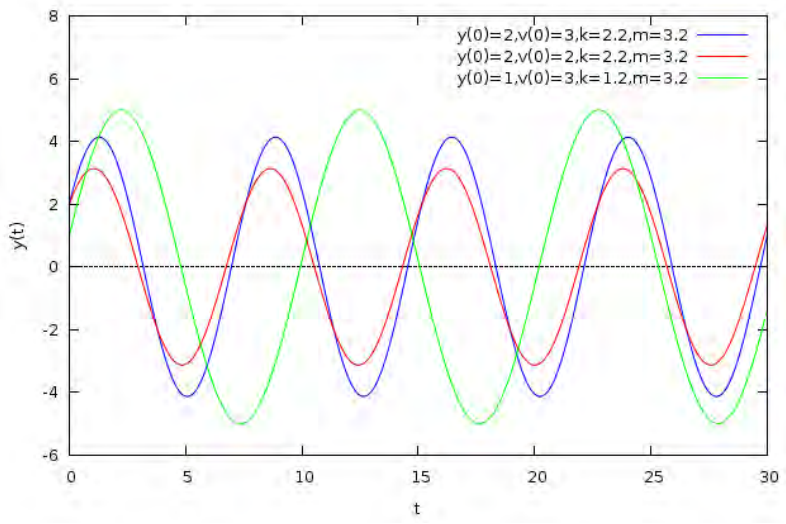

FIGURE 5.4: *Graphs of the analytic solutions for ideal springs determined by different sets of parameters. *Note that $v(0)$ stands for $\dot{y}(0)$.*

5.6.2 Interaction with Oscillatory Forces: Resonance

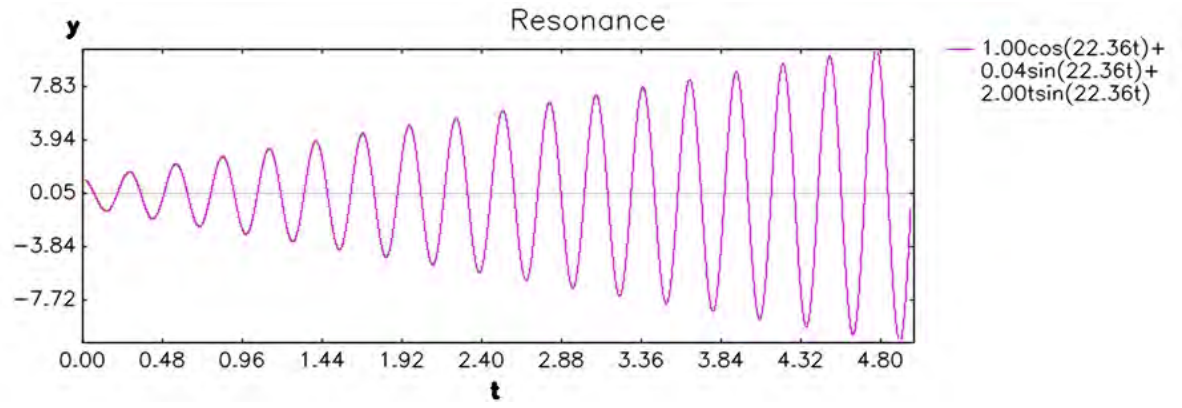

FIGURE 5.5: *Graph of the analytic solutions for and ideal spring when a periodic force is applied to produce resonance, with* $m = 1$, $k = 500$, $v(0) = 1$, $y(0) = 1$, $a = 4\sqrt{500}$.

In addition to the force of the spring, the mass can also be driven by an external force. Of particular interest is the case when the driving force oscillates with the same natural frequency as the spring. The equation that describes this movement is:

$$m\ddot{y}(t) + \omega_0^2 y = \frac{a}{m} cos(\omega_0 t). \tag{5.9}$$

where $\omega_0 = \sqrt{\frac{k}{m}}$. The general solution is illustrated in Figure 5.5 and its equation is [63]:

$$y(t) = Acos(\omega_0 t) + Bsin(\omega_0 t) + \frac{a}{2m\omega_0} tsin(w_0 t). \tag{5.10}$$

5.6.3 Numerical Integration

When solving differential equations, like Equation 5.6, it is not always possible to express $y(t)$ analytically. One approach then is to approximate the solution numerically. The most straightforward way to do it, is to directly approximate the state of the system for a time $t+\Delta t$, given the state at time t. If the state of the system is know at a time t_0, the state at time $t_0 + \Delta T$ can be approximated, with this approximation, the state at $t_0 + 2\Delta t$ can approximated from it, and so on. These type of approximations are know as **explicit**. There are other methods that improve their stability by estimating the state at $t + \Delta t$ in terms of itself. They are known as predictor-corrector methods [67]. However, they do require cycles of iterative approximations that consume too much time. They are not considered here because we are interested in producing simulations in real time.

5.6.3.1 Euler

The first step to obtain a numerical approximation is to rewrite the second law of Newton, which is a differential equation of second order:

$$F = m\ddot{y} = m\frac{d^2y}{dt}. \tag{5.11}$$

As two differential equations of first order:

$$\begin{aligned} \frac{d}{dt}y(t) &= v(t), \\ \frac{d}{dt}v(t) &= \frac{1}{m}F(y, v, t). \end{aligned} \tag{5.12}$$

In this way, for a given time t, the tangent of the function velocity $v(t)$ is known. If $v(t)$ is known at that time, it is possible to approximate $v(t+h)$ as $v(t) + h\frac{d}{dt}v(t)$. The same idea can be applied to estimate $y(t)$. This is the **Euler's method**. Figure 5.6 illustrates this idea. Summarising:

$$\begin{aligned} y(t+h) &= y(t) + hv(t), \\ v(t+h) &= v(t) + h\frac{1}{m}F(y,v,t). \end{aligned} \tag{5.13}$$

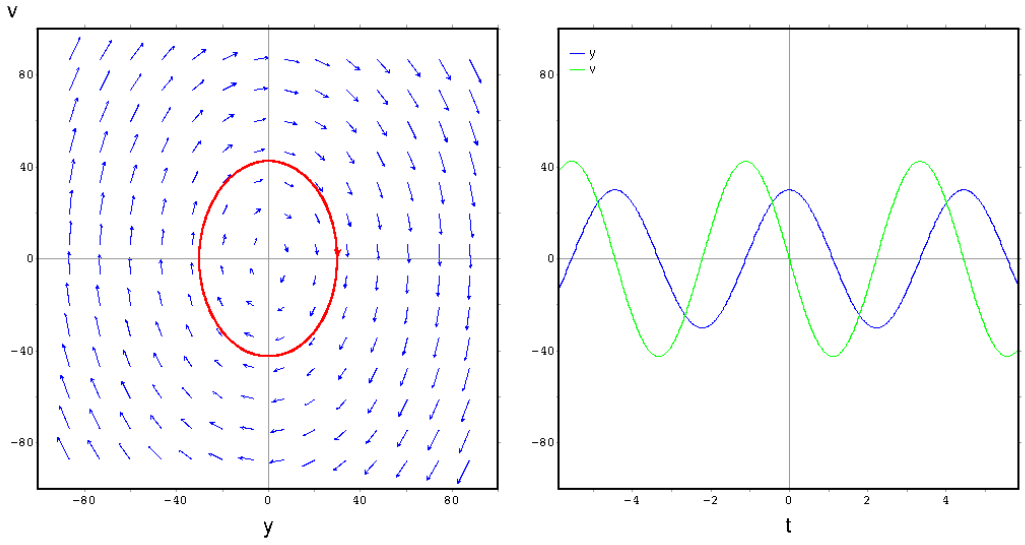

FIGURE 5.6: *Left: direction field of the harmonic oscillator. For each point* (y, v) *the vector tangent* $\left(\frac{dy}{dt}, \frac{dv}{dt}\right)$ *is ploted. Right: the integrated* y *and* v *as a function of time for* $m = 1$, $k = 2$, $v(0) = 30$, $y(0) = 0$.

This method works when the curves $y(t)$ and $v(t)$ can be correctly approximated by

their tangents, that is, they are ***differentiable***[x]. In this case, the approximation is more exact the smaller h is. However it is important to remember that an error gets accumulated at each step, as it can be seen in Figure 5.7.

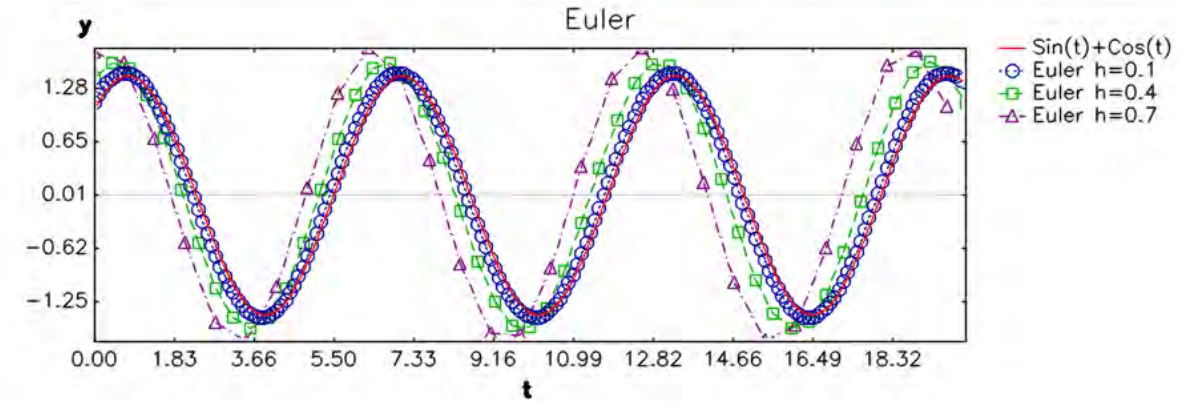

FIGURE 5.7: *The integrated y as a function of time for $m = 1$, $k = 1$, $v(0) = 1$, $y(0) = 1$ and different values of h. The red line shows the exact solution for comparison.*

Another important factor is that numeric methods do not always converge to the solution. For example, for the harmonic oscillator, for big values of k, it is necessary to have very small values of h or the approximation will rapidly diverge from the solution, as illustrated in Figure 5.8. This happens because, due to its high curvature, the function separates quickly from the tangent. Consequently, the tangent is not a good approximation for big values of h. To address this issue, better numeric methods have been devised.

5.6.3.2 Verlet

In the Verlet integration scheme, the new state of the system ($x(t + h)$ position at time $t + h$ and $v(t + h)$ velocity at time $t + h$) [30].

[x]Notice the difference between a function being *derivable*, which means that is has a well defined derivative, and a function being *differentiable*, which means that its tangent is a good approximation.

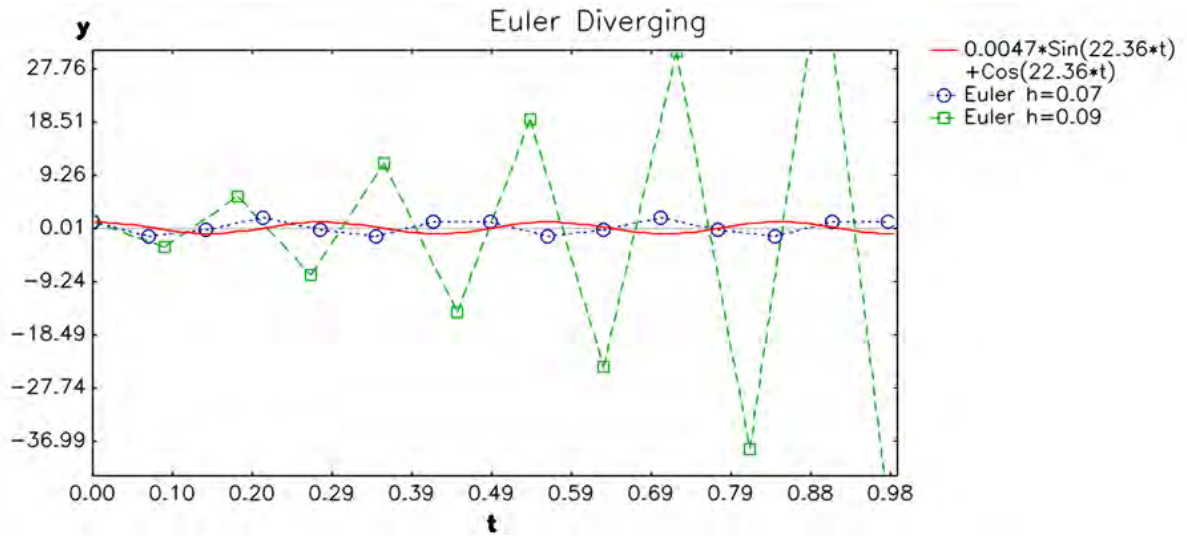

FIGURE 5.8: *The integrated y as a function of time for $m = 1$, $k = 500$, $v(0) = 1$, $y(0) = 1$ and different values of h. The red line shows the exact solution for comparison. For $h = 0.09$ the approximated solution quickly diverges from the real one. This is a know problem with stiff springs (those with a big value of k).*

$$x(t+h) = 2x(t) - x(t-h) + h^2 \frac{\vec{F(t)}}{m} + O(h^4) \tag{5.14}$$

$$v(t+h) = \frac{x(t+h) - x(t-h)}{2h} + O(h^2) \tag{5.15}$$

This produces a slight improvement over Euler.

5.6.3.3 RK4

The **fourth order Runge-Kutta method (RK4)** has proved to be both computationally efficient and accurate [63]. It can be thought of as a modified Euler method where the slope of the tangent is modified to make a cleverer approximation. Here:

$$\begin{aligned} y(t+h) &= y(t) + \frac{1}{6}h\left[W_{y1} + 2W_{y2} + 2W_{y3} + W_{y4}\right], \\ v(t+h) &= v(t) + \frac{1}{6}h\left[W_{v1} + 2W_{v2} + 2W_{v3} + W_{v4}\right], \end{aligned} \tag{5.16}$$

where:

$$\begin{aligned} W_{y1} &= v(t) & W_{v1} &= F\left(y, v, t\right), \\ W_{y2} &= v(t) + \frac{h}{2}W_{v1} & W_{v2} &= F\left(y(t) + \frac{h}{2}W_{y1}, v(t) + \frac{h}{2}W_{v1}, t + \frac{h}{2}\right), \\ W_{y3} &= v(t) + \frac{h}{2}W_{v2} & W_{v3} &= F\left(y(t) + \frac{h}{2}W_{y2}, v(t) + \frac{h}{2}W_{v2}, t + \frac{h}{2}\right), \\ W_{y4} &= v(t) + hW_{v3} & W_{v4} &= F\left(y(t) + hW_{y3}, v(t) + hW_{v3}, t + h\right). \end{aligned}$$

This means that:

- W_{y1} and W_{v1} are the slopes at t, the beginning of the interval, using $y(t)$ and $v(t)$, respectively (Euler's method).
- W_{y2} and W_{v2} are the slopes at $t + \frac{h}{2}$, the midpoint of the interval, using $y(t) + \frac{h}{2}W_{y1}$ and $v(t) + \frac{h}{2}W_{v1}$.
- W_{y3} and W_{v3} are again the slopes at $t + \frac{h}{2}$, the midpoint of the interval, using $y(t) + \frac{h}{2}W_{y2}$ and $v(t) + \frac{h}{2}W_{v2}$.
- W_{y4} and W_{v4} are the slopes at $t + h$, the end of the interval, using $y(t) + hW_{y3}$ and $v(t) + hW_{v3}$.

When the problem illustrated in Figure 5.8 is approached with RK4, the approximated solution remains stable for smaller values of h, as it can be seen in Figure 5.9.

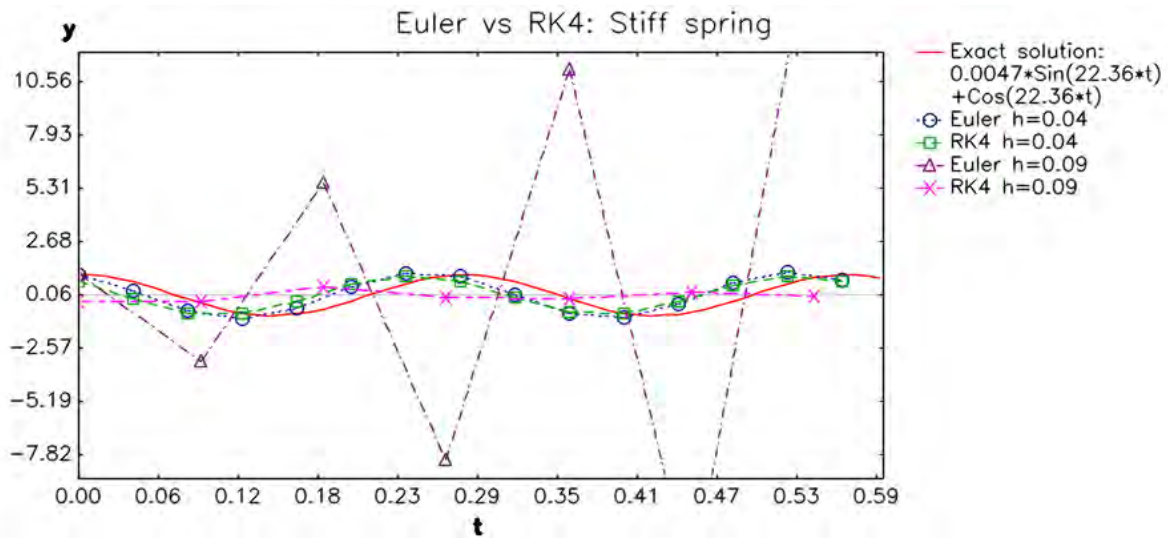

FIGURE 5.9: *The integrated y as a function of time for $m = 1$, $k = 500$, $v(0) = 1$, $y(0) = 1$ and different values of h. The red line shows the exact solution for comparison. For $h = 0.09$ the solution approximated with Euler quickly diverges from the real one, while Runge-Kutta remains stable.*

The solution to the simple harmonic oscillator is a periodic function. The mass oscillates indefinitely. Real springs, when left on their own, slow down with time due to friction. Friction can be modelled in several ways, depending on its causes.

5.7 A Model of Friction

When modelling real objects, the amount of kinetic energy that gets dissipated due to friction has a very visible effect: it slows movement down. Whether friction is caused by the interaction of rubbing surfaces, or internal viscosity of a material, or

other causes, it is possible to take this slowing down effect into considerations directly by including some simple terms in the equations of motion of the object. Instead of modelling the physical origin of friction, these terms intend to produce the adequate final effect: to slow the movement down. There are many terms that can achieve this effect, and different options may be required to reproduce the behaviour of different materials. A couple of the most widely used methods are described below.

5.7.1 Viscous Damping

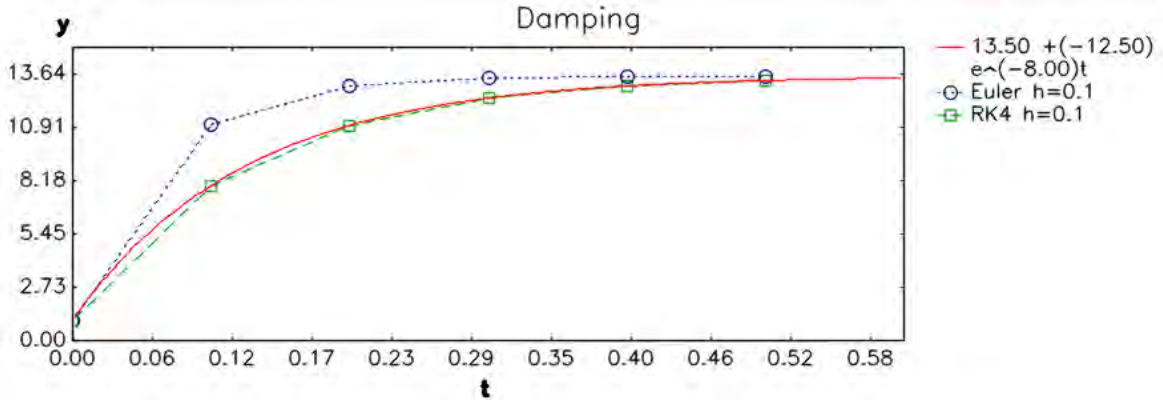

FIGURE 5.10: *Damping slows down movement. The integrated* $y(t)$ *for* $m = 1$, $c = 8$, $v(0) = 100$, $y(0) = 1$.

A very common model of friction is to consider a force that opposes movement. This force acts in the opposite direction of the velocity of the object and with magnitude directly proportional to such velocity. This model is known as *viscous damping*, because it is similar to the effect over an object that moves through a viscous fluid and thus gets slowed down. It converts the kinetic energy to heat, which is dissipated into a viscous liquid. The equation derived from Newton's second law, for an object in movement,

subject only to viscous damping is:

$$\frac{d}{dt}[m\dot{y}(t)] = -c\frac{d}{dt}y(t), \tag{5.17}$$

$$m\ddot{y}(t) = -c\dot{y}(t). \tag{5.18}$$

where c is the damping coefficient. The general solution is a decreasing exponential:

$$\begin{aligned} y(t) &= A + \dot{y}B(t_0)e^{-ct/m}, \\ A &= y(t_0) + \frac{m}{c}\dot{y}(t_0), \\ B &= -\frac{m}{c}e^{ct_0/m}. \end{aligned} \tag{5.19}$$

In this case, as time tends to infinity, the mass approaches the position given by A, as shown in Figure 5.10. Observe that, if the damping is the only force acting on the object and the velocity at time t_0 is zero, the object remains at the position $y(t_0)$. In this case, the net effect of this type of damping is to slow down movement until it stops.

5.7.2 Spring Damping: Dampers

FIGURE 5.11: *Hydraulic damper. [Taken from the web.]*

When the damping effect must be reduced to work only along the spring, it is necessary to specify that direction. This model can represent the natural damping of a real spring, or it can represent another parallel element called a *hydraulic damper*. A damper is a tube filled with viscous liquid and has the effect of slowing down motion along its length Figure 5.11. In this case, the position of both ends of the spring needs to be considered x_1 and x_2. In general:

$$\hat{d} = \frac{x_2 - x_1}{|x_2 - x_1|}, \tag{5.20}$$

$$F_1 = c(\dot{x}_2 \cdot \hat{d} - \dot{x}_1 \cdot \hat{d})\hat{d}, \tag{5.21}$$

$$F_2 = -c(\dot{x}_2 \cdot \hat{d} - \dot{x}_1 \cdot \hat{d})\hat{d}. \tag{5.22}$$

where c is again the damping coefficient, $\hat{d}$ gives the direction of the spring/damper, and F_1 and F_2 are the forces over x_1 and x_2 respectively [87]. Now, if x_1 is fixed as in the case of the hanging spring:

$$\begin{aligned} \dot{x}_1 &= 0, \\ F_2 &= -c(\dot{x}_2 \cdot \hat{d})\hat{d}. \end{aligned} \tag{5.23}$$

5.8 Damped Harmonic Oscillator

When viscous friction is added to the spring equation [Equation 5.6], the resulting equation is:

$$F = m\ddot{y}(t) = -ky(t) - cv(t). \tag{5.24}$$

The general solution for $y(t)$ depends on the nature of the roots of its characteristic equation[xi]:

$$\lambda = -\frac{c}{2m} \pm \frac{1}{2m}\sqrt{c^2 - 4km} \tag{5.25}$$

It is important to notice that the algorithm for the numerical approximations[xii] is exactly the same for all those cases, even if the analytic solutions are qualitatively different.

5.8.1 Case 1: Overdamping

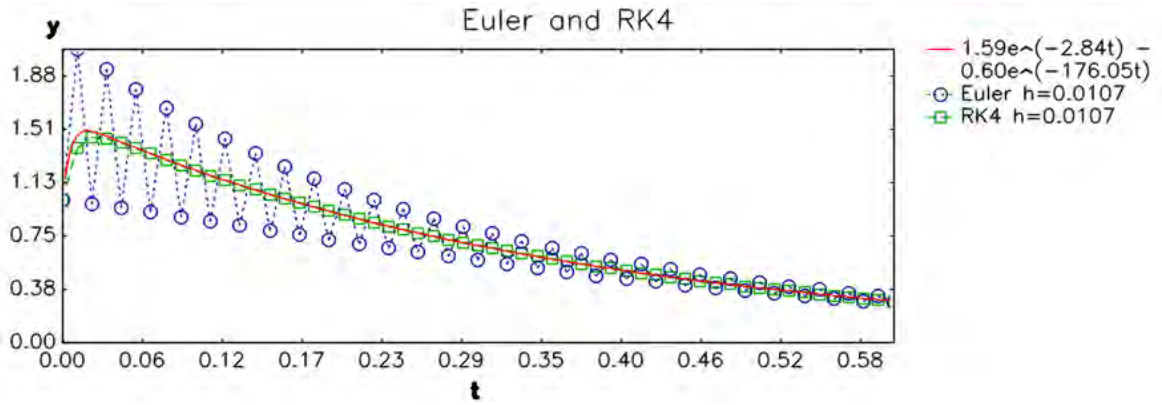

FIGURE 5.12: *Overdamping. The integrated $y(t)$ for $m = 1$, $k = 500$, $c = 8\sqrt{k}$, $v(0) = 100$, $y(0) = 1$. Euler shows an oscillatory behaviour due to numerical errors, this problem can be fixed by using smaller values of h.*

When $c^2 - 4km > 0$:

The general solution in this case is formed of two decreasing exponentials. The mass goes back to its position of equilibrium without oscillating. Figure 5.12 shows a graph

[xi] For details on the solution of the differential equation see [63]
[xii] That is, the same code is used.

of the analytic solution $y(t)$, and two approximations with Euler and RK4 for comparison.

$$\begin{aligned} y(t) &= Ae^{\lambda_1 t} + Be^{\lambda_2 t}, \\ \lambda_1 &= -\frac{c}{2m} + \frac{1}{2m}\sqrt{c^2 - 4km}, \\ \lambda_2 &= -\frac{c}{2m} - \frac{1}{2m}\sqrt{c^2 - 4km}. \end{aligned} \tag{5.26}$$

5.8.2 Case 2: Critical damping

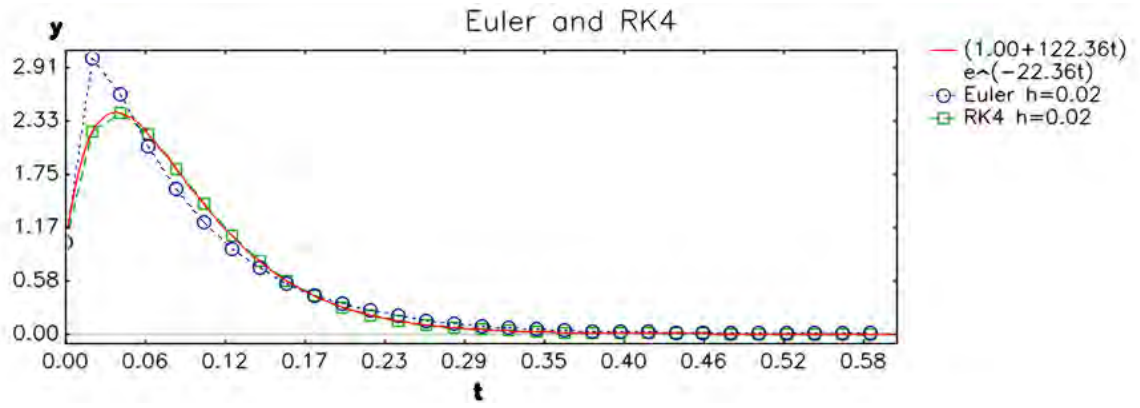

FIGURE 5.13: *Critical damping. The integrated $y(t)$ for $m = 1$, $k = 500$, $c = 2\sqrt{k}$, $v(0) = 100$, $y(0) = 1$.*

When $c^2 - 4km = 0$:

The characteristic equation has a single root. Figure 5.13 shows a graph of the analytic solution $y(t)$, and two approximations with Euler and RK4 for comparison.

$$y(t) = (A + Bt)e^{-ct/2m}. \tag{5.27}$$

5.8.3 Case 3: Underdamping

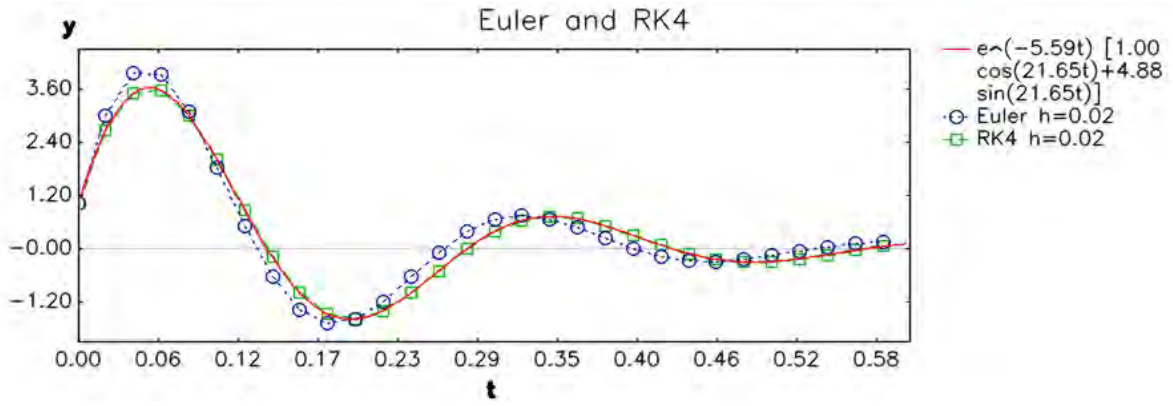

FIGURE 5.14: *Underdamping. The integrated* $y(t)$ *for* $m = 1$, $k = 500$, $c = \sqrt{k}/2$, $v(0) = 100$, $y(0) = 1$.

When $c^2 - 4km < 0$**:**

In this case the resulting motion is oscillatory, but the amplitude of the oscillations decreases with time, as is shown in Figure 5.14.

$$\begin{aligned} y(t) &= e^{-ct/2m}\left[Acos(\beta t) + Bsin(\beta t)\right], \\ \beta &= \frac{1}{2m}\sqrt{4km - c^2}. \end{aligned} \tag{5.28}$$

5.8.4 A One Dimensional Model

One mass hanging from one spring is not enough to model a complex object, but it can be useful to characterise its properties in one direction. For example, a sequence of a single damper and two damped springs connected in line was used to model a compliant robotic foot stepping on soft terrain [79]. The leg is modelled as a mass connected to a damper; while both, the ankle of the foot and the terrain, were modelled

as damped springs which got connected to each other when the foot stepped on the terrain. The terrain was characterised by the values of its spring and damper constants. Since the interactions between the springs are simple, an analytical solution was found in this case. It was then possible to adjust the value of the constants to the experimental data about the compression of the terrain and the deformation of the ankle of the robot. Further studies allowed for a more detailed characterisation of the terrain, by using simple physics based models of elasticity and friction citeSinha1993.

5.9 A Model for Complex Objects or Materials: Mass-spring Systems

However, to fully model the deformations of a complex object, the shape of the object is approximated by a mesh of masses and springs, with each spring governed by Hooke's law [See Figure 5.15].

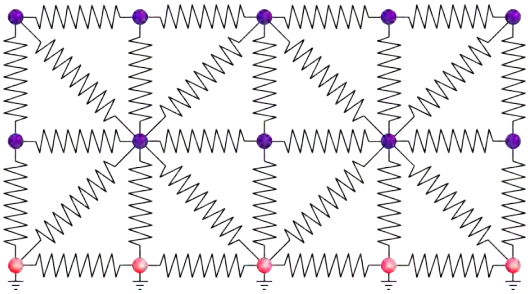

FIGURE 5.15: *Mass spring system. The mass is concentrated in the nodes of a mesh, while the edges represent springs. A common constraint on the system is to fix some of the nodes (red nodes).*

5.9.1 Representation of the Shape: Meshes

A mesh is a collection of vertices connected through edges forming a graph. 2D Mass spring systems commonly use planar graphs[xiii] known as *polygonal meshes*. Common shapes for the elements are triangles, quadrilaterals and hexagons. 2D approximations with other physics based models have been used successfully to support tracking small deformations of 3D objects like facial gestures [51]. Figure 5.16 shows a 2D mesh representing a deformed dish-washing sponge. It is also possible to approximate volumes with *volumetric meshes* whose elements can be tetrahedrons, for example. A 3D simulation of a 3D object would certainly be a better approximation than a 2D model of one of its surfaces, because they would consider volumetric deformations, that would not be accounted for in a 2D model. However a 2D approximation can be used to test the concept, before developing more complex implementations, if the main deformations happen in that dimension.

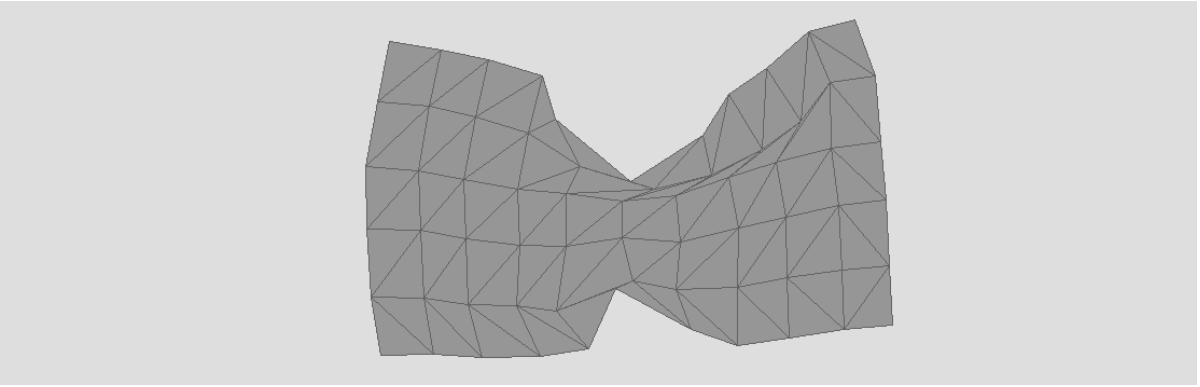

FIGURE 5.16: *Mesh for a deformed sponge, medium resolution.*

[xiii] In a planar graph the edges do not intersect

5.9.2 The Dynamics: Particles Connected with Springs

When the object is made of a deformable material, and a mesh model is used, the mass of the object is usually distributed among particles located at the nodes of the mesh. Ideally, this distribution should resemble the distribution of the mass of the original object, however, since the number of elements in the mesh is too small compared to the number of atoms in the object, this approximation is quite rough for most materials.

The consistency of the material is emulated by the springs, located at the edges of the mesh. Those springs bind the particles together. With this structure, every time a particle is pushed or pulled out of place, it will "attract" or "repel" its neighbours, and the neighbours will also push or pull it back to its position of equilibrium. Again, ideally, an atom or molecule experiences forces from all its neighbours, the discretisation here introduces an often unwanted anisotropy in the material, because the particles only experience a force where a spring has been placed. A clever positioning of the springs can help to alleviate this problem[xiv].

In a mass-spring system, the equation for Hooke's law considers the change in the length of each spring, with respect to its original length. The restorative force acts in the direction of the spring itself. Mathematically, if all the components are located in the Cartesian plane[xv], the equation for a spring between masses m_i and m_j is:

[xiv] A more dynamic allocation of the spring forces is used by Bourguignon to address isotropy and anisotropy issues in [8].
[xv] Or in the Cartesian space, for 3D.

$$\begin{aligned} \vec{F}_{ij} &= -\vec{F}_{ji} = -k_{ij}\left(|\vec{l}| - l_0\right)\frac{\vec{l}}{|\vec{l}|}, \\ k_{ij} &= k_{ji}, \\ \vec{l} &= \vec{X}_j - \vec{X}_i, \\ l_0 &= |\vec{X}_{j0} - \vec{X}_{i0}|. \end{aligned} \tag{5.29}$$

Where $\vec{F}_{ij}$ is the force that the particle i experiences from the spring that connects it to particle j, k_{ij} is the stiffness of this spring that connect them, $\vec{X}_i$ is the vector position of the particle i, and the subindex 0 indicates original lengths, positions, etc. Observe that the behaviour of the springs depends on the relative positions of the particles to one another and not on the exact coordinates.

Each particle in the system will be subject to *internal forces*, coming from all the springs linked to it, and can be subject to *external forces* like gravity, viscous damping, or from contact with other objects. The total force will be the result of the vectorial sum of all those forces. The Equation 5.30 is, in fact, a system of equations that describes a mass-spring system with viscous damping and where each node can be subjected to other external forces.

$$\vec{F}_i = \sum_{j \subset \text{neighbours of } i} \left(-k_{ij}\left(|\vec{l}| - l_0\right)\frac{\vec{l}}{|\vec{l}|}\right) - c\dot{\vec{X}}_i + \vec{F}_{ext} \tag{5.30}$$

Here the movement of each mass depends also on the the movement of its neighbours. It is possible to obtain analytic solutions for particular cases with meaningful symmetries, but in the general case it is necessary to calculate numerical solutions. Solving a

system of equations like these, involves inverting the matrix of its coefficients, a process that becomes highly time consuming as the number of elements increases. To avoid these operations it is possible to use numeric methods like Euler or RK4, as it was explained in Section 5.6.3. Given the current position and velocities of all nodes at time t, these methods estimate *explicitly* the next set of positions and velocities at time $t+1$, which can be used in turn to make prediction for the next time step. This process can be used to make a simulation of several steps, if the conditions are known at an initial time. The main advantage of this type of systems is that they can be used for virtual environments working in real time [29]. However, calibrating them properly for this use is done by hand and is highly time consuming.

5.9.3 An Extended use of Springs: Teschner's Model

Matthias Teschner uses the same model of springs, but extends the idea to areas and volumes as well [88]. Concretely, for a volumetric mesh, not only the edges tend to recover their lengths according to Hooke's law, the faces of the tetrahedrons tend to recover their original areas and the tetrahedrons themselves, try to recover their volumes. To make the equations clearer, he presented them in terms of another physics concept: *potential energy*. In practical terms, the system tries to minimise its potential energy, this causes the system to experience forces in the opposite direction to the gradient of this potential. Teschner expressed this potential energy in terms of constraints C, that increase the energy of the system when the shape of the mesh is deformed. The expressions for the forces where published by Morris in [55] and he gave a geometrical justification for them.

$$E(\vec{p}_0, ..., \vec{p}_{n-1}) = \frac{1}{2}C^2, \tag{5.31}$$

$$\vec{F}^i(\vec{p}_0, ..., \vec{p}_{n-1}) = -\frac{\partial}{\partial \vec{p}_i}E = -kC\frac{\partial C}{\partial \vec{p}_i}. \tag{5.32}$$

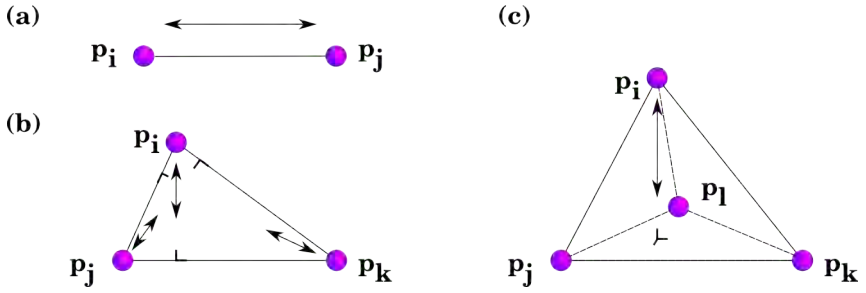

FIGURE 5.17: *(a) The linear force of the spring pulls inside if the spring was elongated, or outside if it was compressed. (b) The triangle tries to quickly recover its area by pulling all its vertices along its corresponding heights. (c) The tetrahedron tries to recover its volume by pulling all its vertices along its corresponding heights (only one is illustrated).*

5.9.3.1 Preservation of Length

Each spring will tend to recover its original length. Strictly this is the only term that will force the triangles to recover their original shape. See Figure 5.17(a)

Energy:

$$E_D(\vec{p}_i, \vec{p}_j) = \frac{1}{2}k_L\left(\frac{|\vec{p}_j - \vec{p}_i| - D_0}{D_0}\right)^2 \tag{5.33}$$

Force:

$$\vec{F}_D(\vec{p}_i) = k_L\left(|\vec{p}_j - \vec{p}_i| - D_0\right)\left(\frac{\vec{p}_j - \vec{p}_i}{|\vec{p}_j - \vec{p}_i|}\right) \tag{5.34}$$

where $\vec{p}_i, \vec{p}_j$ are the mass point positions, k_L the stiffness coefficient, E_D the potential energy based on the difference between the current distance of two points and the

initial or rest distance D_0, with $D_0 \neq 0$. $\vec{F}_D$ is the force resulting from this energy and it will pull or push the masses in the direction of the line that joins them.

5.9.3.2 Preservation of Area

Every triangle in the mesh tries to recover its area. See Figure 5.17(b). This term does not respect the original shape of the triangle, thus allowing the hole mesh to find a new equilibrium even if it remains greatly deformed from its original shape, but it warranties non empty/flattened triangles.

Energy:

$$E_A(\vec{p}_i, \vec{p}_j, \vec{p}_k) = \frac{1}{2} k_A \left(\frac{\frac{1}{2}|(\vec{p}_j - \vec{p}_i) \times (\vec{p}_k - \vec{p}_i)| - A_0}{A_0} \right)^2 \tag{5.35}$$

Force:

$$\begin{aligned} \vec{F}_A(\vec{p}_i) &= k_A \cdot forcemag_A(\vec{p}_i) \cdot forcedir_A(\vec{p}_i) \\ forcemag_A(\vec{p}_i) &= \frac{1}{2}|(\vec{p}_k - \vec{p}_i) \times (\vec{p}_j - \vec{p}_i)| - A_0 \\ forcedir_A(\vec{p}_i) &= \frac{\vec{F}_A(\vec{p}_i)}{|\vec{F}_A(\vec{p}_i)|} = \frac{areagradient(\vec{p}_i)}{|areagradient(\vec{p}_i)|} \\ areagradient(\vec{p}_i) &= (\vec{p}_i - \vec{p}_j) - \left((\vec{p}_k - \vec{p}_j) \cdot \frac{(\vec{p}_k - \vec{p}_j) \cdot (\vec{p}_i - \vec{p}_j)}{(\vec{p}_k - \vec{p}_j) \cdot (\vec{p}_k - \vec{p}_j)} \right) \end{aligned} \tag{5.36}$$

where the energy E_A considers triples of mass points that build surface triangles. E_A represents energy based on the difference of the current area of a surface triangle and its initial area A_0 with $A_0 \neq 0$. Here k_A is an area stiffness coefficient. Each mass will move in the direction of the height of the triangle that passes through it, to increase or decrease its area. This term will favour a type of recovery that emulates materials that can not be compressed, even if deformed, thus pouring substance around. If combined

with the preservation of length, it repairs degenerations like very small angles between edges quicker.

5.9.3.3 Preservation of Volume

Every triangle in the mesh tries to recover its volume. This term is the one that helps to prevent the tetrahedra from being inverted, when a vertex is pushed against the opposite face and goes through it, hence its importance. In addition it enforces the volumetric incompresibility of the material. See Figure 5.17(c).

Energy:

$$E_V(\vec{p}_i, \vec{p}_j, \vec{p}_k, \vec{p}_l) = \frac{1}{2} k_V \left(\frac{\frac{1}{6}(\vec{p}_j - \vec{p}_i) \cdot [(\vec{p}_k - \vec{p}_i) \times (\vec{p}_l - \vec{p}_i)] - V_0}{\tilde{V}_0} \right)^2 \tag{5.37}$$

Force:

$$\begin{aligned} \vec{F}_V(\vec{p}_i) &= k_V \cdot forcemag_V(\vec{p}_i) \cdot forcedir_V(\vec{p}_i) \\ forcemag_V(\vec{p}_i) &= \frac{\frac{1}{6}(\vec{p}_j - \vec{p}_i) \cdot [(\vec{p}_k - \vec{p}_i) \times (\vec{p}_l - \vec{p}_i)] - V_0}{V_0} \\ forcedir_V(\vec{p}_i) &= \frac{F_V(\vec{p}_i)}{|F_V(\vec{p}_i)|} = \frac{volumegradient(\vec{p}_i)}{|volumegradient(\vec{p}_i)|} \\ volumegradient(\vec{p}_i) &= (\vec{p}_k - \vec{p}_j) \times (\vec{p}_l - \vec{p}_j) \end{aligned} \tag{5.38}$$

where E_V is the energy associated with the volume of every tetrahedron, and $\vec{F}_V$ the force that minimises it. With V_0 being the rest volume, and $\tilde{V}_0 = V_0$ if $V_0 \neq 0$ otherwise $\tilde{V}_0 = \frac{\sqrt{2}}{12}\bar{l}$ with $\bar{l}$ denoting the average edge length of a tetrahedron, $\vec{p}_i$, $\vec{p}_j$,$\vec{p}_k$ and $\vec{p}_l$ are the coordinates of the four vertices. k_V is the volumetric stiffness coefficient. The force

will move every vertex in the direction of the height of the tetrahedron to increase or decrease its volume.

5.9.3.4 Damping

Using Teschner's approach, the damping element is added to give stability to the numerical integration process. The formula to calculate the force, for a damper along the spring, is:

$$\vec{F^i}(p_0, ..., p_{n-1}, v_0, ..., v_{n-1}) = \left(-kC - k_d \sum_{0 \leq j < n} \frac{\partial C}{\partial p_j} \vec{v}_j \right) \frac{\partial C}{\partial p_i} \tag{5.39}$$

5.9.4 Plastic Deformation

The simplest method to model plastic deformation consists in changing the rest length l_0 of the springs, when they have been deformed beyond a certain threshold. The typical parameters are:

Yield The minimum strain value from which plastic deformation takes place.

Creep The proportion of strain that becomes permanent.

This is expressed as:

$$l_0 = \begin{cases} l_0 & \text{if } l - l_0 <= yield, \\ creep * (l - l_0) & \text{if } l - l_0 > yield. \end{cases} \tag{5.40}$$

where l is the length of the spring after the resultant force is applied to it.

5.10 Summary

In this chapter the main elements and concepts used to model matter and its interactions system were introduced. Particle and multiparticle systems were considered. Solid materials were presented as multiparticle systems where a number of constraints allow for a simplified representation. A very quick review of the concepts of energy and momentum was included, with the additional intention of presenting a conservative potential energy as the gradient of a function from which a force field is derived. This requirement is tightly related to the validity of the theorems of conservation of energy that are frequently mentioned in the literature to be presented in the following chapter.

The second half of the chapter focused on models of ideal springs, which can be used to model the interactions between the particles that constitute the system. Detailed diagrams were included to illustrate the most relevant behaviours of an ideal spring, so that the reader can have an intuition of what can be expected of a mass-spring system with more springs. Finally the theory of mass-spring systems over a mesh was presented as a way to model the behaviour of deformable objects. Particularly, the extension made by Teschner was included here, since our model builds on it.

Chapter 6

Related Work I: Modelling of the Shape

Previous chapters have covered the topic of learning to characterise and predict the behaviour of deformable objects in a very broad sense, starting with Chapter 3. This was done with the purpose of presenting clearly the wide contexts in which a robot of general purpose may need to labour and the difficulties derived from them. There was a special emphasis on the mechanical aspects of the behaviour of deformable solid materials [Chapter 4]. Given these difficulties, it is necessary to adopt a strategy that will allow the robot to learn about new materials, as it encounters them, instead of trying to program it in advance to know everything about them. In order to make use of a systematic scientific frame of reference for further discussion, Chapter 5 introduced the use of mechanics (from physics). Now it becomes necessary to centre the discussion around the interdisciplinary field of robotics and current research in related areas.

This whole chapter is concerned with the great variety of ways to represent deformable shapes in a computer. Any of these, or combinations of them, could be used as internal representations of objects for robots. Each has advantages and disadvantages that may

favour some representations over others, depending on individual tasks. Additionally, any of these representations may appear in literature about simulation of the mechanical behaviour of deformable objects. Therefore, the intention has been to provide a simple introduction to these techniques. Consequently, this chapter can also serve as a reference for experts in robotics that need a simple introduction to representations that may have their origins in other fields of computer science.

To work with shapes in a digital computer, it is necessary to assign them a finite **discrete** representation. Discrete, in the sense that only a few parameters should be used to determine every desired detail. Figure 6.1 categorises the most used approaches. The following subsections present them in more detail.

6.1 Continuous vs Discrete Models

Even though the representation of curves, surfaces, volumes or even hyper-volumes[i] in the computer must be finite, it is possible to represent several continuous curves, surfaces, etc., with very few parameters, thanks to the use of mathematical equations capable of defining which points belong and which do not belong to them. Such representations are divided into implicit [Section 6.2] and explicit (or parameterised) [Section 6.3] [10]. A second option is to approximate them with listings of discrete simple elements [Section 6.4]. A very good review of methods to represent surfaces is the one by Montagnat et al. [54].

[i] The prefix *hyper* is used to refer to entities in spaces of more than 3D.

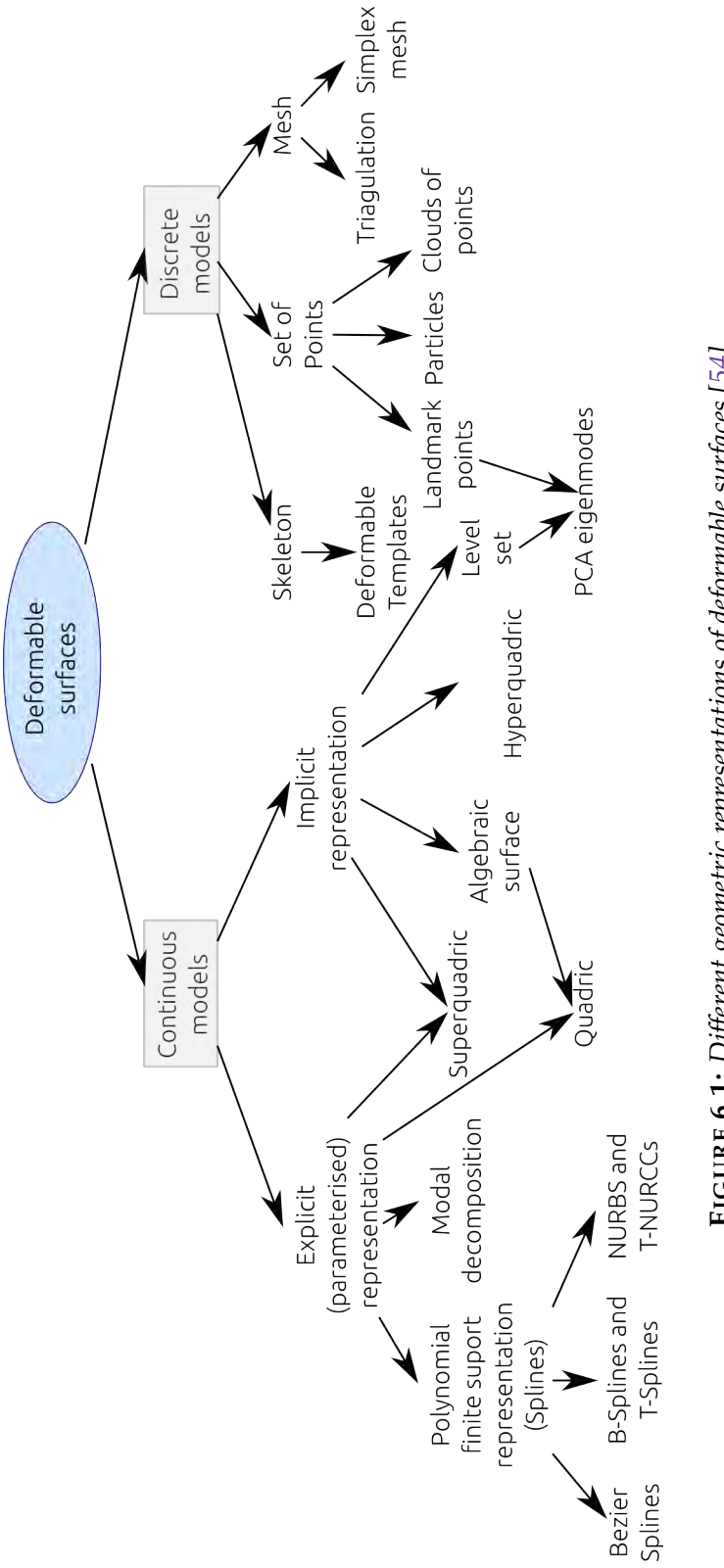

FIGURE 6.1: *Different geometric representations of deformable surfaces [54].*

6.2 Implicit Representations

An implicit curve or surface is generally defined as the zero set of a function f whose range is $\mathbb{R}$[ii]. For example, in the three dimensional space this is written:

$$\mathbf{S}_f = \{\mathbf{P} \in \mathbb{R}^3 | f(\mathbf{P}) = 0\}. \tag{6.1}$$

Where f is a function that goes from the 3D space of real numbers $\mathbb{R}^3$, to the real numbers $\mathbb{R}$. This is written $f : \mathbb{R}^3 \to \mathbb{R}$. Therefore, S_f is the surface formed by all points $\mathbf{P}$[iii] in $\mathbb{R}^3$ such that the function f, evaluated at $\mathbf{P}$, is equal to zero.

Because of their implicit form, many characteristics of a problem can be used for their definition, like it happens with the level set curves explained below; thus making these curves good candidates to represent shapes for known contexts. In general, it is not straightforward to plot a geometric figure defined in an implicit way in a computer, but there are important exceptions, like the algebraic curves. Computing the distance from a point to an algebraic surface, which is required by surface matching algorithms, can be difficult depending on the form of f. The value $f(\mathbf{P})$ is often used as an approximation.

6.2.1 Algebraic Curves and Surfaces

Algebraic curves and surfaces satisfy Equation 6.1 with f being a polynomial. They are specially good for standard curves and surfaces, like planes, circles and solenoids. that can be easily translated or deformed by modifying a simple parameter. For example:

[ii] This is, the set of all points where the value of the function f is zero.

[iii] The bold font is used because the point coordinates can be given in many dimensions, for computer graphics the most usual being 2D and 3D. E.g. $\mathbf{P} = \{4.1, 3.4, 2.5\}^T$.

Plane: $ax + by + cz = d$. By changing d the plane shifts without changing its normal.

Conics: These are circles, ellipses, parabolas and hyperbolas. E.g. for the circle: $(x - x_c)^2 + (y - y_c)^2 = r^2$, modifying r varies the radius of the circle.

Quadrics: They are algebraic surfaces in the 3D space where f is a polynomial of degree 2. Ellipsoids, paraboloids, hyperboloids, toroids, cones and cylinders are quadrics. They can also have parameterised representations.

Their D-dimensional extension are surfaces in a $D + 1$ dimensional space that satisfy the equation:

$$x^T Q x + P x + R = 0. \tag{6.2}$$

Where $x = \{x_1, x_2, ..., x_{D+1}\}^T$ is a column vector, x^T is the transpose of x (a row vector), Q is a $(D+1) \times (D+1)$ matrix and P is a $(D+1)$-dimensional row vector and R a scalar constant.

6.2.1.1 Superquadrics

They are a generalisation of quadrics and satisfy the equation:

$$\left|\frac{x}{A}\right|^r + \left|\frac{y}{B}\right|^s + \left|\frac{z}{C}\right|^t = 1 \tag{6.3}$$

where r, s and t are positive real numbers[iv]. They can also have parameterised representations. [Figure 6.2, for more details see [5]]. It is also very straightforward to determine whether a point is inside or outside the superquadric through the use of Equation 6.4: if $f(x, y, z) < 1$ the point is inside, if $f(x, y, z) > 1$ it is outside and if

[iv] Negative values give raise to super-hyperboloids.

$f(x, y, z) = 1$ the point is on the surface[v]. This method makes it easier to detect collisions with other shapes [66].

$$f(x, y, z) = \left|\frac{x}{A}\right|^r + \left|\frac{y}{B}\right|^s + \left|\frac{z}{C}\right|^t \tag{6.4}$$

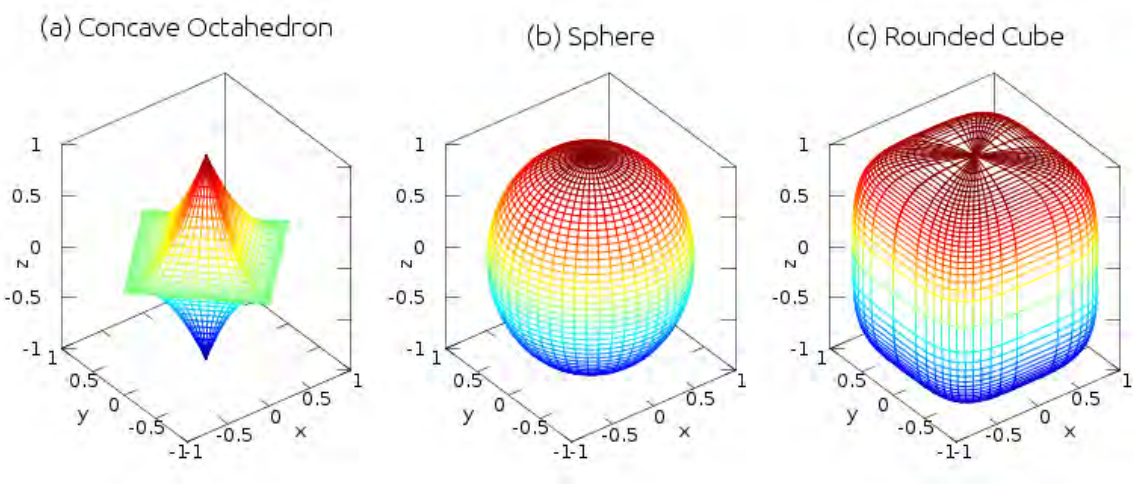

FIGURE 6.2: *Superquadrics with scaling factors* $A = B = C = 1$*, and exponents: (a)* $r = s = 1, t = 0.5$ *(b)* $r = s = t = 2$ *and (c)* $r = s = t = 4$*.*

Because of the absolute values in their equation, without external deformations, superquadrics are intrinsically symmetric. With so few degrees of freedom (DOF), the deformations they can represent are not useful for complex shapes. However they can be used in combination with local surface deformations to represent a wider set of surfaces [54]. In this case, the superquadric works as a rough backbone of the shape, helping to avoid too much sensitivity in the deformation process.

[v]Sometimes times this function is called the *inside-outside function* because of this property.

6.2.1.2 Hyperquadrics

They are volumetric shape models that include superquadrics as a special case and allow for the representation of non-symmetric shapes. They are defined by equations of the following form:

$$\sum_{i=1}^{N} |A_i x + B_i y + C_i z + D_i|^{\gamma_i} = 1. \tag{6.5}$$

where N is any arbitrary number and $\gamma_i \geq 0 \; \forall i$ [47].

Algebraic curves, surfaces and volumes can be used as 1D, 2D or 3D skeletons, or to represent objects of similar shapes and deformations [31]. They are easy to deform, but the types of deformations that are straightforward to apply are very limited. For this reason, they are more suited to represent articulated or semi-articulated objects; or objects composed of several algebraic curves. They can be combined with other skinning techniques to emulate soft deformable objects like the dolphin in Figure 7.5. However, it is not evident how they could be used to model arbitrarily deformable materials, like plasticine.

6.2.2 Level Set Methods

The main idea of level sets is to embed the deformable model in a higher dimension space. For example in $\mathbb{R}^3$, a 2D surface is represented as the zero level set of a function[vi]:

$$\Psi : (\mathbb{R}^3 \times \mathbb{R}^+) \rightarrow \mathbb{R} : S = \{\mathbf{P} \in \mathbb{R}^3 | \Psi(\mathbf{P}, t) = 0\}. \tag{6.6}$$

[vi] It is called *zero level set* because it is the collection of all points at height zero.

Where $\mathbf{P} = \{x, y, z\}^T$, $\mathbb{R}^+$ are the positive real numbers and t is time. With this form of equation, S can represent a closed surface or a set of closed surfaces that are deformed through time.

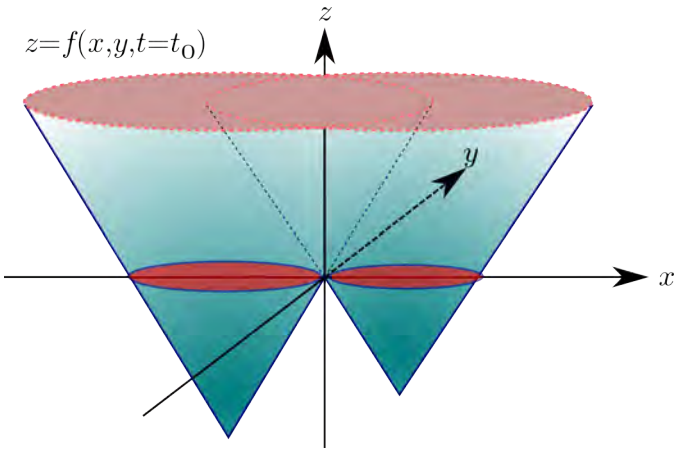

FIGURE 6.3: *Level Set Graph. The two intersecting cones correspond to $\Psi(\mathbf{P}, t_0)$. The two circles on the plane $z = 0$ correspond to the zero level set, the interface of interest. Displacing the cones up and down transforms the interface by changing the radius of the circles and separating them or merging them, respectively.*

Given an initial surface S_0, the hypersurface Ψ is defined as $\Psi(\mathbf{P}, 0) = dist(\mathbf{P}, S_0)$, where $dist$ can be the signed Euclidean distance between a point and the surface. The distance is positive if the point lies outside the surface and negative otherwise. The evolution of the surface S is guided by a partial differential equation involving the function $\Psi(\mathbf{P}, t)$ and a speed function F, that tells the speed at which each point of the surface must be moved [54, 77].

In this approach, instead of deforming the surface S directly, the function Ψ is the one that evolves in time. The final effect is that the surface of interest varies its shape as

a propagating "wave" front, whose current shape always corresponds to the surface given by $\Psi(\mathbf{P}, t) = 0$[vii] [See Figure 6.3].

Level sets have been applied to medical image segmentation, modelling of silicon chips manufacture and tracking of physical phenomena that highly depends on the surface curvature of the substance of interest [77]. They have also been used in combination with statistical information to improve tracking of deformable objects [18, 83].

Because of their general definition, they could be used to model arbitrary deformations of closed curves, surfaces or even volumes. The difficulty lies in learning Ψ, since it might not have an algebraic expression. Also, even though they can represent curves that get joined and split, the underlying equation is the same. Therefore, objects that would be perceived as different, would have the same representation internally. This would be counter-intuitive for scenarios like a piece of plasticine that is broken in two. However, they would be appropriate to describe deformations like drops of water over a surface, the propagation of lines of fire [77] or temporal deformations of sponges and plasticine.

6.2.3 Gaussian PCA eigenmodes

In general, the representation of a deformable shape can have any number of elements. In this sense, it can be said that it is *infinite-dimensional*. For example, if a contour is given by a list of pixels in an image, the number of pixels required to describe it is undetermined and variable. A way around this problem can be applied when the type of deformations belongs to a single class. Given a representative set of the types

[vii] It is important to notice that the function Ψ can be written in a parameterised form. For example: $\Psi = \Psi(x(u,v), y(u,v), z(u,v), t)$. However the surface is still defined in its implicit form: $\Psi(x(u,v), y(u,v), z(u,v), t) = 0$.

of deformation that the objects can undergo, it is possible to use *Principal Component Analysis* to detect the main modes of deformation, and thus re-represent the shapes as a linear combination of those modes.

The first requirement is to be able to represent the shape on each image as a column vector in a matrix. Several options are available: to use a fixed set of landmark points [15], the parameters of deformable templates [6] or level set functions defined over the whole image [18, 49, 89]. Taking the last case as an example, Cremers defines a level

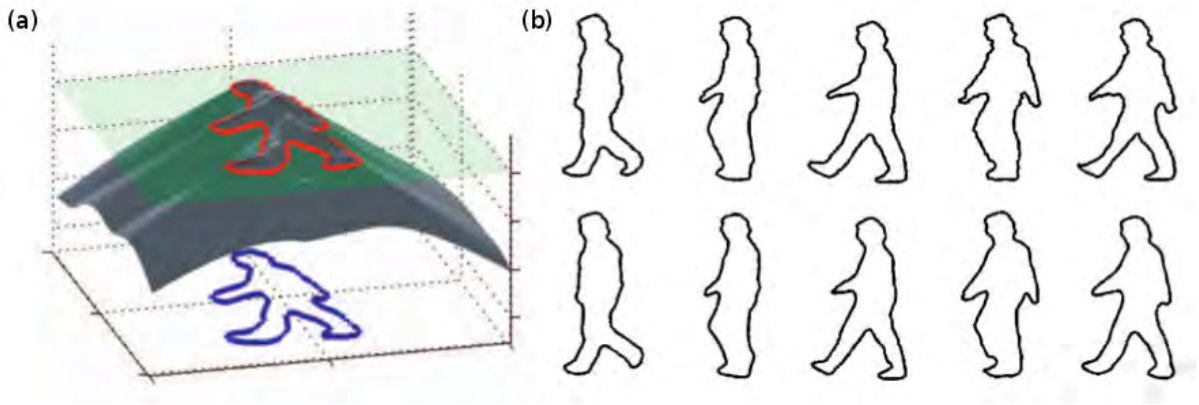

FIGURE 6.4: *(a) Level set representation of a human silhouette. (b) The original silhouettes (above) are approximated by the first six principal components of their embedding functions (below). Taken from [18], with permission.*

set function over the whole domain Ω [Figure 6.4(a)], such that:

$$\Psi = \begin{cases} distance < 0 & \text{for pixels inside the shape,} \\ 0 & \text{for pixels on the border,} \\ distance > 0 & \text{for pixels outside.} \end{cases} \tag{6.7}$$

However, if each pixel is a *dimension*[viii], there are as many dimensions as pixels in the image, and those may be too many. For some machine learning techniques, it is desirable to derive another representation that requires only a finite and reduced fixed amount of parameters to describe the shape, this is to make it *finite-dimensional*.

If there are n sample images for training, with N pixels each, there are n embedding level set functions ψ_i that can each be written as a column vectors of size N. Each value in the column is the value of the distance function for the corresponding pixel. The first step is to obtain the mean shape function:

$$\bar{\phi} = \frac{1}{n} \sum \psi_i \tag{6.8}$$

The variability of the shapes with respect to the mean shape will be encoded in the set of mean-offset functions:

$$\tilde{\psi}_i = \psi_i - \bar{\phi}. \tag{6.9}$$

To obtain the principal modes of the set of shapes, each of the $\tilde{\psi}_i$s is written as a column of the shape-variability matrix $S = \{\tilde{\psi}_1, \tilde{\psi}_2, ..., \tilde{\psi}_n\}$[ix]. Using Singular Value Decomposition (SVD), the covariance matrix[x] can be written in terms of its orthogonal modes of variation and its eigenvalues:

$$\frac{1}{n} S S^T = U \Sigma U^T. \tag{6.10}$$

The columns of $U = \{\phi_1, \phi_2, ...\phi_{j \leq n}\}$ correspond to the orthogonal modes of variation

[viii] The use of italics is to distinguish from spacial dimensions like 2D or 3D

[ix] This procedure is described in detail in [49] for surfaces in a d dimensional space.

[x] The covariance between two jointly distributed real-valued random variables x and y is defined as $\sigma(x, y) = E[(x - [x])(y - E[y])]$ where $E[x]$ is the expected value of x, also known as the mean (average value) of x. Therefore the covariance matrix in this case is equal to $\frac{1}{n} S S^T$ *Oxford Dictionary of Statistics, Oxford University Press, 2002, p. 104.*

(the eigenvectors), and Σ is an $n \times n$ diagonal matrix whose diagonal elements are the corresponding eigenvalues σ_i. The square of each eigenvalue is the variance for the corresponding eigenvector. Let U_k be the matrix formed by the k columns of U with the largest corresponding eigenvalues; they will be enough to obtain a good compressed representations of the shapes in the problem. An approximation ϕ of the original shape ψ can now be written as a linear combination of the columns of U_k, with coefficients α [49]:

$$\alpha = U_k^T(\psi - \bar{\phi}), \tag{6.11}$$

$$\phi = U_k\alpha + \bar{\phi}. \tag{6.12}$$

[18] used this method for predictive tracking of deformable shapes and applied it to human silhouettes [Figure 6.4(b)]. However, a representation with a few eigenmodes is essentially limited to the shapes that can be constructed by combining those eigenmodes. Again, an arbitrarily deformable piece of plasticine or a sponge, could acquire shapes that can not be correctly represented by them.

6.3 Explicit/Parameterised Representations

The parameterised representation is used for *free-form surfaces*, and in 3D it is of the form:

Curve: $C(u) = \{x(u), y(u), v(u)\}^T$

Surface: $S(u, v) = \{x(u, v), y(u, v), z(u, v)\}^T$

Where u and v are the parameters. The curve or surface is traced as the values of the parameters are varied. It is common practise to parametrise a curve by time, if it will represent a trajectory, or by length of arc[xi].

6.3.1 Splines

Splines are particularly good to approximate a wide variety of curves and surfaces. They are extremely useful for signal and image processing [90] as well as computer animation [52]. A mathematical spline is a piecewise-defined polynomial real function, used to represent curves or surfaces [16]. For curves this is written:

$$S : [a, b] \to \mathbb{R} \tag{6.13}$$

where $[a, b]$ is an interval composed of k ordered disjoint subintervals $[t_{i-1}, t_i)$ with $a = t_0 < t_1 < ... < t_{k-1} < t_k = b$. Inside each subinterval the function S is a polynomial. Thus $S(t)$ is defined by:

$$S(t) = \begin{cases} s_1(t) & \text{for } t \in [t_0, t_1), \\ s_2(t) & \text{for } t \in [t_1, t_2), \\ \dots & \\ s_k(t) & \text{for } t \in [t_{k_1}, t_k]. \end{cases} \tag{6.14}$$

[xi]The length of arc is the distance between a starting point in the curve and the current point. A common choice is to use the Euclidean distance $d = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}$.

The order of the spline corresponds to the highest order of the polynomials $s_i(t)$. The points $t_0, t_1, ..., t_{k-1}, t_k$, where the polynomial pieces connect, are called *knots*. Frequently, for a spline of order n, S is required to be differentiable to order $n-1$, that is, to be C^{n-1}. This means that at all the knot points t_i, the derivatives of adjacent polynomials up to the $n-1$ degree, must have the same value[xii]:

$$s_i^{(j)}(t_i) = s_{i+1}^{(j)}(t_i) \text{ for } 0 \leq j \leq n-1. \tag{6.15}$$

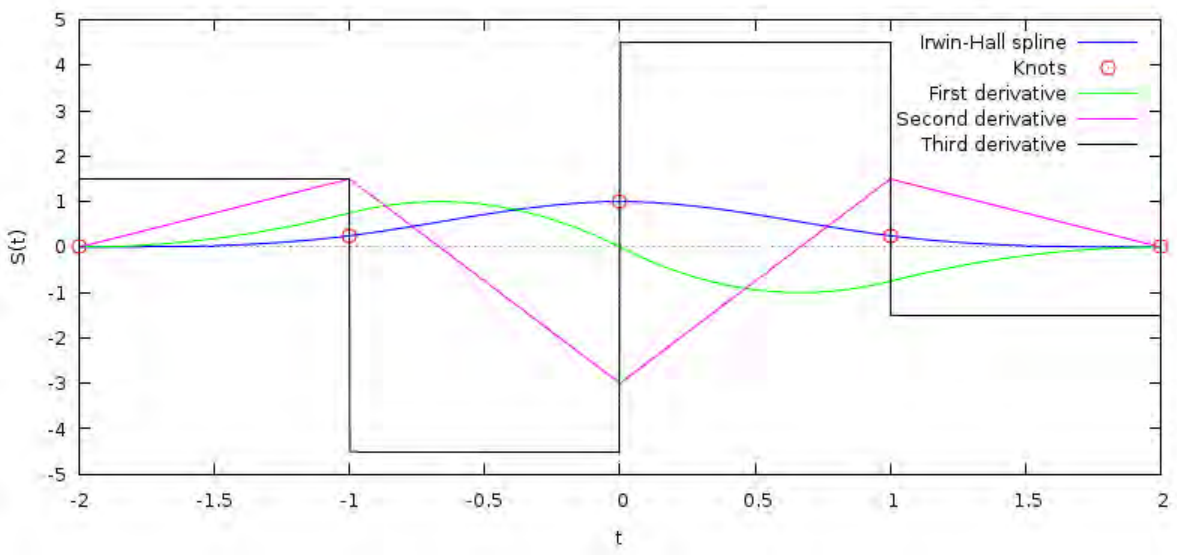

FIGURE 6.5: *Cubic spline in the shape of a bell (blue line). Each segment is a cubic polynomial. The spline is C^2 at the knots and C^∞ everywhere else. The derivatives of the whole curve are splines of smaller orders.*

[xii]The function $s_i^{(0)}$ is the polynomial s_i itself.

Figure 6.5 is an example with the spline equation being:

$$S(t) = \begin{cases} \frac{1}{4}(x+2)^3 & \text{if} -2 \leq x < -1, \\ \frac{1}{4}(3|x|^3 - 6x^2 + 4) & \text{if} -1 \leq x < 1, \\ \frac{1}{4}(2-x)^3 & \text{if} 1 \leq x \leq 2. \end{cases} \tag{6.16}$$

For some types of splines it is possible to add several knots at the same position to reduce its differentiability, thus allowing for discontinuities to be represented as well.

In general, any spline function $S(t)$ of order n with knots $t_0, ..., t_k$ can be expressed as:

$$S(t) = \sum_{j=1}^{k+n+1} \beta_j s_j(t), \tag{6.17}$$

where the β_js are the *control points* that determine the shape of the spline, and

$$\begin{aligned} s_j(t) &= (t - t_j)^n \quad j = 1, ..., k, \\ s_{k+j}(t) &= t^{j-1} \quad j = 1, ..., (n+1). \end{aligned} \tag{6.18}$$

constitute a basis for the space of all spline functions with knots $t_0, ..., t_k$, called the *power basis*. This space of functions is a $k+(n+1)$-dimensional linear space. The power basis has the advantage that deleting a term $s_j(t)(j \leq k)$ is the same thing a deleting a knot [25, p. 247-248]. However other basis can be used, each with different advantages, disadvantages and geometrical interpretations, thus giving rise to a big family of spline variations[xiii]. Some of the most remarkable are [32]:

[xiii] See [74] for a very large list.

Bezier Splines Each segment of the spline is a Bezier curve. A Bezier curve of degree n is a polynomial of degree n, written as the linear combination of the $n+1$ Bernstein basis polynomials of degree n, weighted by $n+1$ control points $\mathbf{P}_i$[xiv]. Therefore, each Bezier curve is given by:

$$\mathbf{B}(t) = \sum_{i=0}^{n} \mathbf{B}_i^n(t)\mathbf{P}_i, \tag{6.19}$$

$$\mathbf{B}_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i, \tag{6.20}$$

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}. \tag{6.21}$$

where $\mathbf{B}_i^n$ are the Bernstein polynomials of degree n, $\binom{n}{i}$ are the binomial coefficients, and $t \in [0,1]$.

Two beneficial properties of Bezier curves are:

1. The curve passes through its first and last control points (P_0 and P_n), and remains close to the polygonal of control, obtained by joining all the control points, in order, with straight lines.
2. Its recursive definition in terms of Bezier curves of lesser degree, which allows for a stable method of evaluation, know as the *de Casteljau's algorithm*. Let $\mathbf{B}_{\mathbf{P}_0\mathbf{P}_1\ldots\mathbf{P}_n}$ denote the Bezier curve determined by the points $\mathbf{P}_0\mathbf{P}_1\ldots\mathbf{P}_n$. Then:

$$\begin{aligned} &\mathbf{B}_{\mathbf{P}_i}(t) = \mathbf{P}_i, \\ &\mathbf{B}(t) = \mathbf{B}_{\mathbf{P}_0\mathbf{P}_1\ldots\mathbf{P}_n}(t) = (1-t)\mathbf{B}_{\mathbf{P}_0\mathbf{P}_1\ldots\mathbf{P}_{n-1}}(t) + t\mathbf{B}_{\mathbf{P}_1\mathbf{P}_1\ldots\mathbf{P}_n} \end{aligned} \tag{6.22}$$

[xiv]The bold font is used because the point coordinates can be given in many dimensions, for computer graphics the most usual being 2D and 3D. E.g. $\mathbf{P} = \{4.1, 3.4, 2.5\}^T$. Consequently, $\mathbf{B}(t)$ denotes the coordinates of each point of the curve in the corresponding space.

This property also makes it easy to add and remove control points [16].

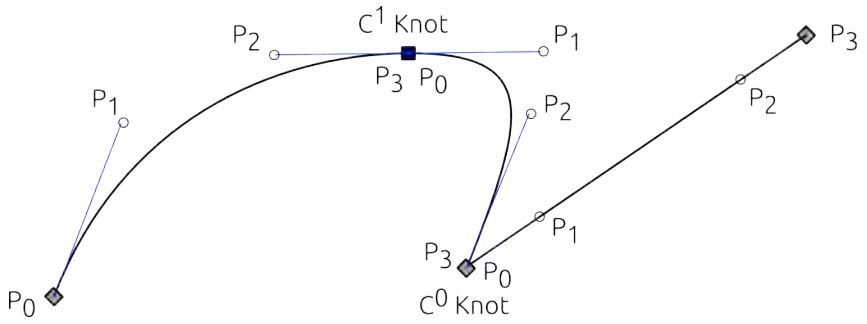

FIGURE 6.6: *Cubic bezier curve. Each segment has four control points. C^0 continuity can be achieved by making the first and last points of adjacent curves have the same value. C^1 continuity requires the tangents of adjacent curves to be the same.*

Cubic Bezier splines are particularly useful for graphic software. Of the four control points required by each spline, the $\mathbf{P}_0$ and $\mathbf{P}_3$ are contained as the extremes of the curve, while the linear segments $\overline{\mathbf{P}_0\mathbf{P}_1}$ and $\overline{\mathbf{P}_2\mathbf{P}_3}$ correspond to the tangents at the respective ends. Therefore a graphic interface that allows for the manipulation of these control points is highly intuitive [Figure 6.6].

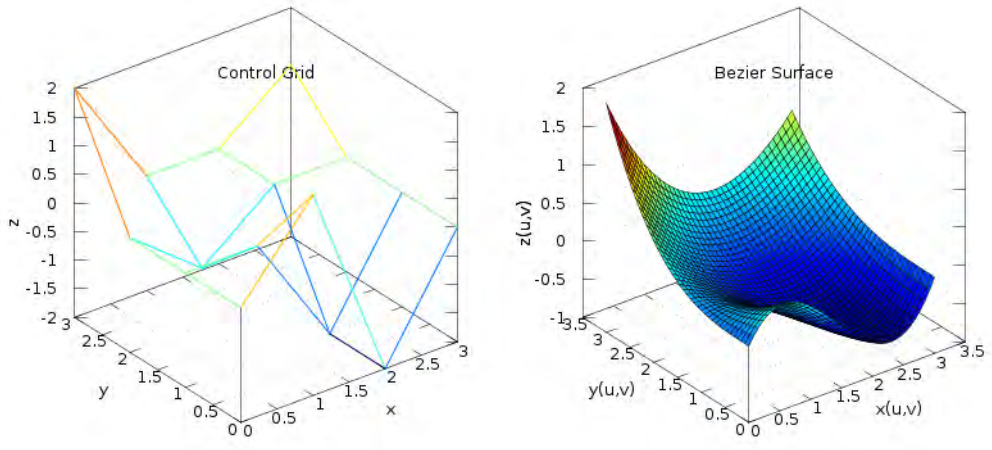

FIGURE 6.7: *Cubic bezier surface. [left] The grid of 4×4 control points, as specified by the designer. [right] The 2D bezier surface patch in 3D space.*

A two-dimensional Bezier surface is obtained as the tensor product of two bezier curves [Figure 6.7]:

$$\mathbf{B}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{B}_i^n(u)\mathbf{B}_j^m(v)\mathbf{P}_{i,j}. \tag{6.23}$$

B-Splines B-Splines are a more stable version of splines than Bezier Splines. A very detailed review of their properties is found in [23] and [22]. For example, they are particularly adequate to solve variational problems. Here, the work by Song and Bai is of particular interest [81]. They used B-Splines to model 3D deformable objects from data obtained with 3D scanners. By representing the data with splines, they managed to compress the space of storage by 90%, with respect to the original scanned data. Although some detail was lost, noise was reduced as well. The resulting shape is easy to edit, 3D metamorphosis and real time multi-resolution rendering is straightforward [Figure 6.8].

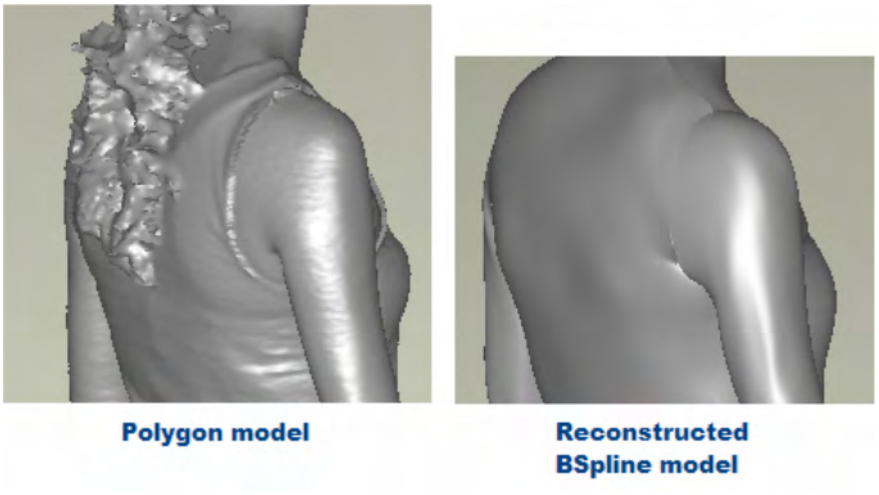

FIGURE 6.8: *Reconstruction of a 3D scaned image with B-Splines. Song's approach also manages to fill holes in the original data. (Reproduced with permission from [81].)*

Catmull-Clark Surfaces In 1978 Catmull and Clark proposed a method for recursively generating surfaces, that approximate points lying on a mesh of arbitrary topology. Their method includes bicubic B-Splines as the particular case when the control-point mesh is rectangular [11, 42].

Non-uniform Rational B-Splines (NURBS) Useful as they are, Bezier and B-Splines can not represent circles, ellipses, spheres and other curves. NURBS address this issue by introducing quotients of polynomials in the basis [78].

T-Splines & T-NURCCs T-Splines [Figure 6.9(a)] and Non-uniform Rational Catmull-Clack Surfaces with T-junctions (T-NURCCs) [Figure 6.9(c)] are generalisations of B-Splines and NURBS respectively, that allow for the presence of T-junctions in the control grid [Figure 6.9(b)]. This allows for high resolution representations of 3D deformable objects with a highly reduced number of faces and improved representations of joints between surface patches [76].

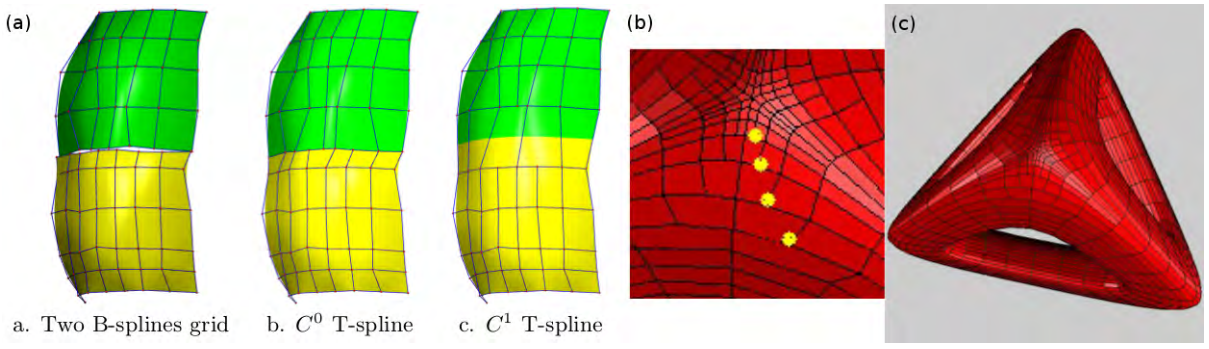

FIGURE 6.9: *(a) Merging of two B-splines into a C^1 T-spline. (b) T-junctions. (c) T-NURCC representation of a shape. [Taken with permission from [76].]*

Splines are an extremely flexible tool to represent all sorts of deformable shapes. However, there is still no general way to produce spline representations, or even more to learn deformations, in an autonomous way. The main problem that is avoided is that,

when new corners or points or high curvature appear, more control points are required for an adequate representation. Furthermore, the deformation of a spline can be quite sensitive to the displacement of the control points. Thus, most animations are produced by hand. In spite of that, splines have been used in combination with energy optimization techniques for tracking of deformable objects, mainly in 2D images. This approach is known as active contours.

6.3.2 Active Contours (Snakes)

Active Contours were proposed by Kass and Terzopoulos for the purpose of identifying contours in digital images [43]. Their main objective is to incorporate global criteria to a segmentation process that used to rely only on local information. An active contour was designed so that it would store information of all possible edges that could be considered within its vicinity. For this purpose, an energy functional was defined, whose local minima comprise the set of alternative solutions available to higher-level contour finding processes. This energy functional depends on the intensity of the image (through E_{image}) and the shape of the contour, which is represented with a spline. The internal spline forces impose a piecewise smoothness constraint E_{int}, which is increased as the snake bends. External constraint forces E_{con} can be used to locate the snake close to the desired minimum. If the snake is represented parametrically by $\mathbf{p}(s) = \{x(s), y(s), z(s)\}^T$ with $s \in [0, 1]$ the energy functional is written:

$$E_{snake} = \int_0^1 [E_{int}(\mathbf{p}(s)) + E_{image}(\mathbf{p}(s)) + E_{con}(\mathbf{p}(s))]\, ds \tag{6.24}$$

The numerical minimization of E_{snake} determines the position of the snake in the image. Since the minimization methods are iterative, the spline seems to slither towards the minimum, therefore the name *snakes*.

There are many works that use snakes for tracking in 2D images, but their use for 3D has not spread so widely. The main disadvantage is that the energy minimization process is time consuming, which becomes even worst for 3D, if there are many control points. There is a representative work in the area of medicine, where 3D snakes are used to track the deformation of the left ventricle [68]. The method is good in this case because they require a very simple model with only a few control points. However, the problem of working with splines whose number of control points change remains open.

6.3.3 Modal Decomposition

In modal decomposition, again, curves or surfaces are expressed as the sum of terms in a basis. However, their special characteristic is that the elements in the basis correspond to frequency harmonics. The sum of the first modes composing the surface gives a good rough approximation of its shape, which becomes more detailed as more modes are considered [54]. Fourier decompositions are of widespread use. A curve may be represented as a sum of sinusoidal terms; while a surface, as a combination of spherical harmonics. It is also possible to use other bases that can be better suited for other shapes. They have been used in model based segmentation and recognition in 2D and 3D medical images [84]. However it is easy for them to loose details in objects, like small dents, because shapes are approximated by a limited number of terms.

6.4 Discrete Models

The representations catalogued as discrete, contain only a finite fixed number of elements describing them, mainly points and lines. Therefore their resolution and descriptive power is much more limited.

6.4.1 Meshes

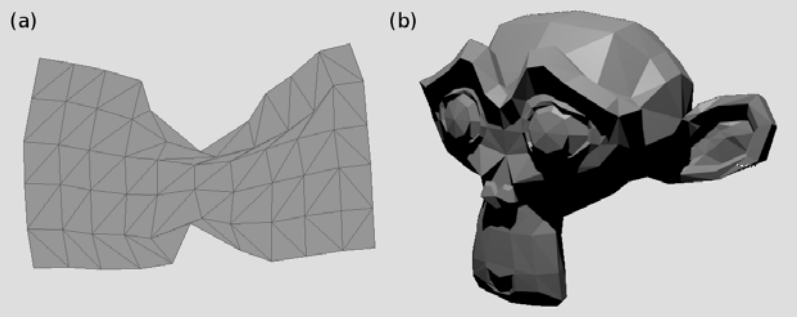

FIGURE 6.10: *(a) 2D mesh representing a deformed dish-washing sponge. (b) Default monkey mesh, rendered with Blender.*

A mesh is a collection of vertices connected through edges forming a graph. These surfaces can be in a two or three dimensional space[xv]. Other common shapes for the elements are quadrilaterals and hexagons. The left side of Figure 6.10 (a) shows a 2D mesh representing a deformed dish-washing sponge. An example of mesh being used to approximate a more complex shape can be seen in Figure 6.10 (b). It is also possible to approximate volumes with *volumetric meshes* whose elements can be tetrahedrons, for example.

[xv] In this last case, the surface may be enclosing a volume.

To have a good approximation of complex shapes with meshes many elements may be required. To reduce the amount of them, it is possible to define meshes with elements of different sizes, putting more elements where more detail is required. However, big irregularities in the size may not be adequate for all applications.

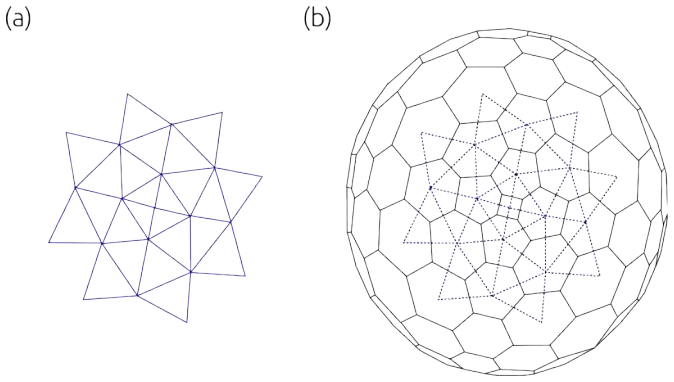

FIGURE 6.11: *(a) Triangulation. (b) 2-simplex mesh of a sphere, with part of its dual triangulation in blue [24].*

6.4.1.1 Triangulations

Triangulations are the particular case where all the elements are triangles. See Figure 6.11(a). This is the most popular one because of the simplicity and rigidity of triangles; that is, if the lengths of the three sides of two triangles are equal, their angles are the same as well. It is also quite straightforward to approximate a wide variety of shapes with meshes of triangles, while preserving relevant corners and edges. There is public software with algorithms that generate good triangulations for any shape, also to generate meshes of tetrahedrons for volumetric shapes, which makes them accessible to a wide variety of users.

6.4.1.2 Simplex Meshes

Simplex meshes have a constant vertex connectivity. For example, each vertex of a 2-simplex mesh is connected to three, and only three, neighbours [Figure 6.11(b)]. 2-Simplex meshes are topologically dual to triangulations: there exists a dual triangle for each mesh vertex and a dual triangulation vertex for each mesh face. Because of this properties, they permit smooth deformations in a simple and efficient manner. They were introduced by Delingette [24]. Further information about their applications to deformable objects is cited in [54].

6.4.2 Skeletons

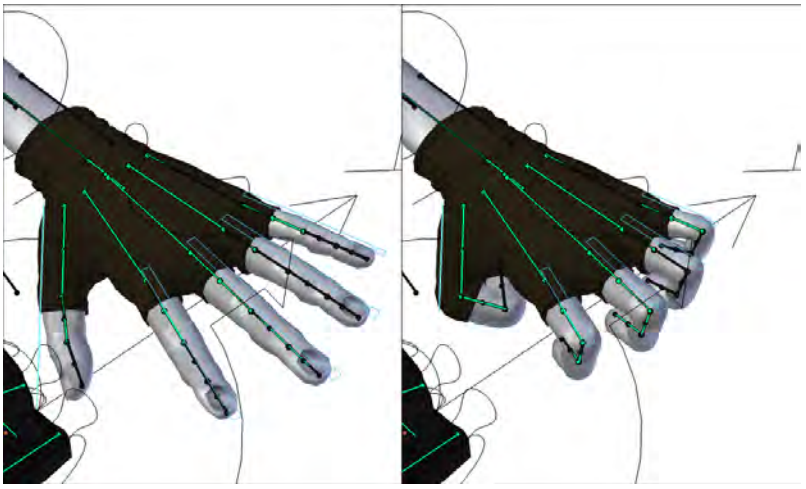

FIGURE 6.12: *Hand modelled using bones (in green). The shape of the whole hand is controlled in first instance by the position of the bones. [Model by Angela Guenette, simplified by Ben Dansie, for the Open Source Project Sintel. ©Blender Foundation]*

Skeletons were of the first approaches to represent deformable objects. They are made of rigid edges connected by joints. The joints grant extra degrees of freedom by allowing bending. Their most traditional exponent is the simulation of human movements.

The process of deformation focuses on the animation of the skeleton. The position or deformation of other elements, like muscles, are defined with respect to their assigned bone [Figure 6.12]. By nature they are designed to model only articulated deformations.

6.4.3 Deformable Templates

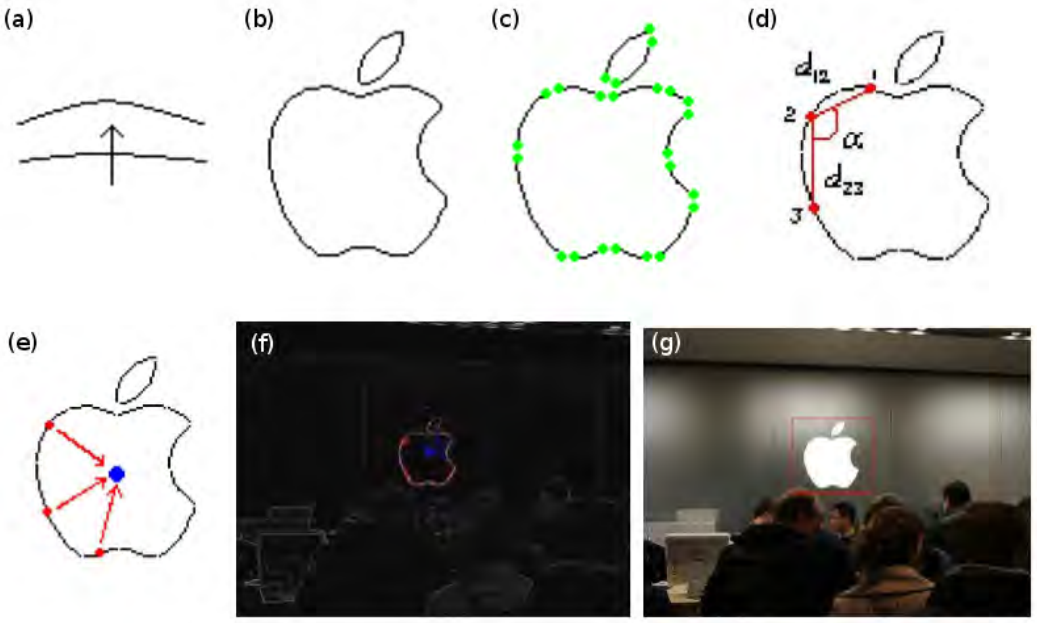

FIGURE 6.13: *Segment break up of a deformable model for detection [69]. (a) Segments with small curvature are scaled along the direction perpendicular to its curvature. (b) Apple logo from the ETHZ dataset. (c) High curvature points are the end points of the segments. (d) More bending is allowed at the points of high curvature, adjacent segments can rotate with respect to each other. (e) Groups of matched segment contributes to a local object centroid. (f) Matched groups in red with the centroids in blue. (g) Object localization with bounding box.*

Deformable Templates were introduced to include a priori knowledge about the expected shape of features to be detected, when searching for specific deformable objects

in images, like facial features (e.g. eyes, nose, mouth). The template is a parameterised representation that specifies key values of salient features in simple terms. These features may be peaks and valleys in the image intensity, edges and the intensity itself. An energy function is defined which contains terms attracting the template to the salient features. The minimum of the energy function corresponds to the best fit with the image. Changing these parameters corresponds to altering the position, orientation, size, and other properties of the template. The method has been used for image recognition and tracking [94].

Another form of deformable template was proposed by Ravishankar, et al [69]. This type of template resembles the more traditional templates used for image recognition, however it incorporates the use of extra flexibility around points of high curvature, because they observed that deformation typically happens around these points [Figure 6.13]. In fact, inspired by this observation, [2] presents some results regarding the importance that corners and points of high curvature can have, for internal representations of deformable objects, for natural and artificial agents.

The mayor constraint to the deformability of a template lies in the method used to represent its elements. If the representation of the template could be modified to add/remove elements when the deformation of a shape requires it, they would become a more flexible representation for elasto-plastic materials, which tend to deform in favoured directions and do not undergo big qualitative changes most of the time.

6.4.4 Landmark Points

When a domain is restricted to a particular class of deformable objects, it is possible to identify landmark points that would remain stable across deformations. These points

can be corners, t-junctions or points of high curvature [14, 69]. In particular, landmark points can correspond to control points of splines [6]. Methods for further processing can work more efficiently if they focus only (or mainly) on these points [14].

6.4.5 Particles

These particle systems could be thought of as extensions of the particle systems that were described in Section 5.2. Each particle in the system can store a set of attributes (e.g. position, velocity, temperature, shape, age, lifetime, etc.). These attributes influence the dynamical behaviour of the particles over time and are subject to change due to procedural stochastic processes. These particles can pass through three different phases during their lifetime: generation, dynamics and death. However, manipulating them and maintaining constraints, like a boundary between them can become non-trivial. For this reason they are used mainly to represent gases or visual effects in animations where the interaction between them is very limited.

6.4.6 Clouds of Points

Clouds of points are a very primitive representation, compared with particles. They are frequently obtained from 3D scanned data. Sometimes they may include the colour of each point, in addition to its 3D coordinates. A typical problem consists in reconstructing 3D surfaces from these clouds [59]. But, in principle, they could be processed one by one and used as a representation themselves if it were not because this is extremely time consuming if they are too many. [19] presents a comparative review of several

methods to process clouds of points and introduces the use of self organising neural gas networks for this purpose.

6.5 Free-Form Deformation

Free-form deformation is a method where the space, in which a figure is embedded, is deformed according to a set of control points [54, 75]. In a deformed space, the new positions of the nodes or control points of the original shape are estimated with respect to the new positions of the control points of the deformed space.

The simplest example is a shape inside a cubic mesh. At the beginning of the process the position of the control points of the shape is measured with respect to the nodes of the cubic mesh. These nodes of the cubic mesh are used to deform the space. When the position of the nodes is modified, the positions of the control points of the shape have to be re-interpolated in such a way that their relative positions with respect to the control grid remains. As a consequence, the shape gets distorted proportionally to the distortion of the space.

This technique can be applied to both continuous and discrete representations of shape. It can also be useful to extend the shapes than can be represented with a certain group of primitives or to control dynamic processes of deformation (e.g. for the synthesis of video sequences).

Therefore, with Free-Form Deformations, the representation of a deformed shape is the combination of the base and the deformed space. Since the shape of the space can be arbitrary, practically any type of deformation can be produced in this way, except for

fracture. However, the problem of representing a deformed shape is now split in two: to select the base and to calculate the deformation of the embedding space.

6.6 Summary

Among the ways to represent the shape, the main ones are: clouds of points or particles, skeletons, meshes, implicit algebraic curves (e.g circles, spheres) and level sets, and parametric splines. It is also common to find modal simplifications of some of these, which capture the main features of the figure. A complementary aspect to the representation of the shape is the method to deform it. This can be accomplished through the direct displacement of key/control points (e.g. nodes of meshes or control points of splines), or through the warping of the surrounding space (free-form deformation).

Clouds of points are basically clusters of coordinates of occupied space. They are primitive data that could be processed as swarms of components of an object, but such treatment would be extremely time consuming.

Particles are used in reduced numbers, in comparison with points in a cloud. They can store attributes and interact with each other. However, it is hard to constrain them to remain within predefined boundaries, which makes it harder to used them to model solid objects. Even though it has been done.

Skeletons are designed to represent articulated objects. However, used in combination with other techniques they are useful model bending around favoured points.

Meshes are discrete approximations of shapes. In principle they can represent any shape, but they require enough enough elements. Curved surfaces are harder to approximate. There are many standard algorithms to generate them and they can be easily manipulated by displacing their vertices as desired.

Algebraic curves are well known and easy to manipulate. Their topological properties have been widely studied in mathematics which allow knowledge about complex manipulations to be available. However their degrees of freedom are few, which restricts the shapes and deformations they can represent. They are good to represent objects that look alike or to be used as backbones or components of complex objects.

Level sets are extremely flexible implicit curves. However they are not always easy to plot. Since they can be determined by differential equations, information about the dynamics of a deformation can be incorporated into the representation itself.

Splines are highly promising. They can represent a wide variety of shapes in great detail, using little memory. However, there are not many methods to generate and animate them autonomously to produce simulations. Most of their applications are in image analysis, tracking or hand tailored animations. Research in this area is open.

When selecting the representation that would be used for this research, it would have been desirable to use splines. Unfortunately it was not evident how to incorporate physics based knowledge into their evolution. Because of the method we chose to model the evolution of the objects, which was physics based, the natural choice became to use meshes.

Chapter 7

Related Work II: Methods for the Simulation of Deformation Dynamics

Chapter 4 presented the difficulties that come when trying to learn to characterise and predict the behaviour of deformable objects. A theoretical framework where the deformable material is described in part by its equation of state was introduced as a point of reference for many of the models that will be explained in this chapter. Chapter 5 went deeper and explained the basic mathematical elements required to build physics based models, putting special emphasis in mass-spring models.

The term *simulation* as used here encompasses a wide variety of options. Apart from some work in industry, which is mainly restricted to very particular domains, the research groups that have actually addressed simulation of the behaviour of deformable objects have been interested in identification, tracking or computer animation, but less effort has been concentrated on actual prediction for robotic interaction. However, all the techniques present in this chapter and classified in Figure 7.1 have the potential

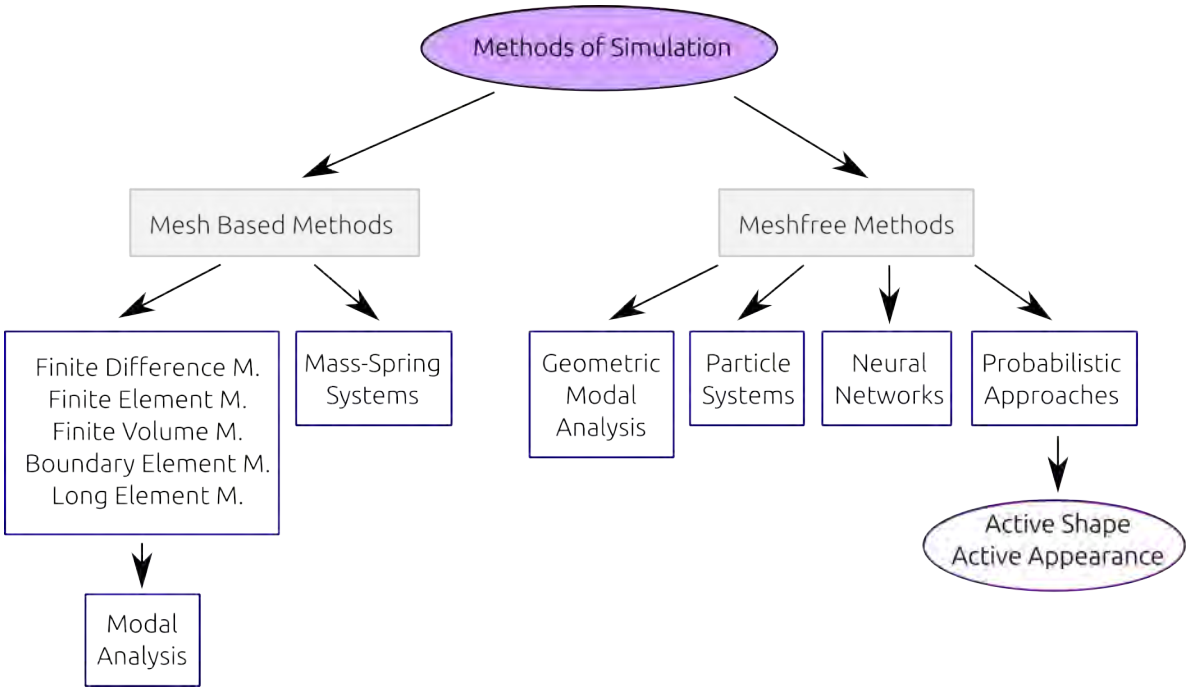

FIGURE 7.1: *Families of Methods for Simulating Deformable Objects.*

or have actually been used for prediction. Each has advantages and disadvantages, as will be explained below. It is also frequent that works in robotics use combinations of them, as will be shown by some of the examples in the final section.

7.1 Dynamics of Elasto-Plastic Continuous Media

A common model for deformable materials consists in assuming that they are made of continuous matter. It is then possible to express their deformation processes with differential equations. Deformations would be the product of the flowing of material or its compression/expansion. If we think of the object as a continuous subset M of $\mathbb{R}^3$, then the coordinates $\vec{m} \in M$ correspond to points occupied by the material.

These types of physics based methods have been used as a common approach to non-rigid image registration. The method consists in computing elastic transformations which simulate deformations of a solid body under the impact of hypothetical applied forces. This evolution would be ruled by the Navier-Stokes partial differential equations, which describe the evolution of fluids in movement [33].

7.1.1 Stress Tensor

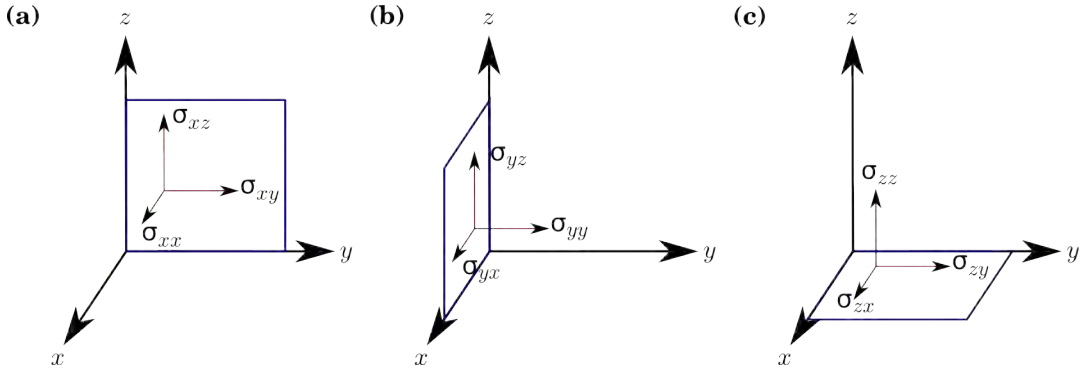

FIGURE 7.2: *Stress vectors act on surfaces characterised by their norms. Surface perpendicular to axe (a) x, (b) y and (c) z.*

In general, for each point $\vec{m}$, which can be visualized as an infinitesimally small cube, it is possible to define the relationship between a stress applied on this point, and the strain being produced, as was seen in Chapter 4. For more than one dimension, this relationship can vary depending on the direction in which the stress is applied and the orientation of the surface the stress is acting on (e.g. this reflects whether the applied stress is compression or shear). According to Cauchy, the stress at any point in an object, assumed to behave as a continuum, is completely defined by nine component stresses: three orthogonal normal stresses and six orthogonal shear stresses [93]. Therefore, a second rank tensor is used to specify stress or strain. The i,j, component of

a tensor **A** is written A_{ij}. Thus, the component σ_{ij} equals the stress in direction j acting on a surface that is oriented perpendicular to direction i [Figure 7.2]. Matrix notation can be used for tensors of rank two:

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{pmatrix} \tag{7.1}$$

7.2 Finite Element Methods (FEM)

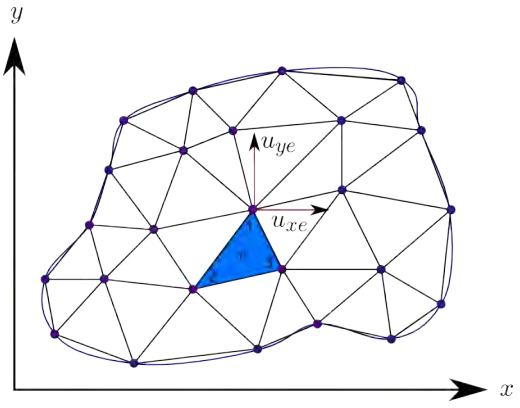

FIGURE 7.3: *Stress vectors act on surfaces characterised by their norms. Surface perpendicular to axe (a) x, (b) y and (c) z.*

The finite element method solves partial differential equations of continuous media, based on the method used to solve the algebraic equations governing a discrete structural system [98]. The first requirement to apply this technique is to discretise the domain through the following steps:

1. Separate the continuum domain (the space occupied by the object) by imaginary lines or surfaces into a number of *finite elements* [Figure 7.3]. (E.g. triangulate a surface, obtain a mesh of tetrahedrons for a volume).

2. The elements are assumed to be interconnected at a discrete number of nodal points situated on their boundaries and occasionally in their interior. The most frequent practise is to use the vertices of the mesh as the nodal points. The method will focus on studying the displacements of these nodal points.

3. Choose a set of functions to define uniquely the state of displacement of any other point within each finite element and on its boundaries in terms of its nodal displacements.

4. Determine a system of *equivalent forces* concentrated at the nodes, equilibrating the boundary stresses and any distributed loads. From this point onwards, it is assumed that all forces are applied on nodal points.

This set of steps implies that it is not exactly the original problem that is being solved, but only an approximation, consequently some errors are introduced [98].

If, additionally, the masses of the elements are lumped to the vertices, the *explicit Finite Element* approach is obtained [57]. In general, the relationship between nodal forces (stress) and nodal positions (strain) is nonlinear. When linearised, the relationship for an element e connecting n_e nodes can be expressed as:

$$\vec{f_e} = \mathbf{K}_e \vec{u}_e, \tag{7.2}$$

where $\vec{f_e} \in \mathbb{R}^{3n_e}$ contains the n_e nodal forces and $\vec{u}_e \in \mathbb{R}^{3n_e}$ the n_e nodal displacements of an element. The matrix $\mathbf{K}_e \in \mathbb{R}^{3n_e \times 3n_e}$ is the *stiffness matrix* of the element.

To take into account the contributions of adjacent elements to a node, a global stiffness matrix $\mathbf{K} \in \mathbb{R}^{3n \times 3n}$ is defined:

$$\mathbf{K} = \sum_e \mathbf{K}_e. \tag{7.3}$$

Let $\vec{u} \in \mathbb{R}^{3n}$ be the vector of the $\{x, y, z\}^T$ displacements of the n nodal points relative to the object's centre of mass. Then the linear algebraic equation of motion for an entire mesh is:

$$\mathbf{M}\ddot{\vec{u}} + \mathbf{D}\dot{\vec{u}} + \mathbf{K}\vec{u} = \vec{f}_{ext}, \tag{7.4}$$

where $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the mass matrix, $\mathbf{D} \in \mathbb{R}^{n \times n}$ the damping matrix and $\vec{f}_{ext} \in \mathbb{R}^n$ externally applied forces. To calculate the result of applying some force $\vec{f}_{ext}$ to the object, the equation is discretised in time and solved using numerical methods [66]. This process may be quite expensive depending on the number of elements to be generated.

Malassiotis et al, use FEM and modal analysis on FEM to track textured deformable objects and also provide a very good review of other related developments [51]. The work by O'Brien et al. making simulations of fracture of brittle and plastic materials for computer animation is quite well known [61, 62]. However these works were not designed to be used at interactive rates. A more complete review of the Finite Element Method can be found in [57, 82].

7.2.1 Automatic Calibration of FEM models

In [26], Frank et al. assume a homogeneous, isotropic and linear material which can be modelled with a 3D-FEM. They used gradient descent to find the Young Modulus and Poisson ratio required as parameters for the model. To evaluate the simulation, 3D point clouds obtained from real objects are used as ground truth and compared with

the FEM-mesh. The applied forces are measured with a sensor located in the robotic manipulator. The behaviour of the model departs from the real object when deformations are large. However, the obtained model is good enough to be used for planning of motion around deformable objects, which does not require too much precision from the simulation [27].

7.2.2 FEM + Snakes = Visual Force Sensors

FEMs have been used in combination with Snakes to relate visual information with force sensing. Since snakes are directed by the intensity of the image, but can be constrained by external forces as well, it is possible to use FEM to calculate the guiding external forces. Luo and Nelson [50] used this technique for predictive tracking, but they also managed to use an already calibrated FEM model, to infer the external forces and internal stresses involved, from the captured images. They used a non-linear hyperelastic model for large deformations and applied it to a rubber-like material called Buna-N. The sample has the shape of a ring and the external force is applied along one of the diameters. They used the experimentally determined stress-strain curve of the material to calculate the parameters for the FEM. For the first problem, the system would predict the new shape of the object (the shape of the snake) given the applied loads. For the second problem they used the snake recognized in the image to infer the forces being applied. However, the theoretical model had to be carefully selected and tuned for this particular type of material (almost incompressible[i] and highly elastic).

[i] Incompressible means that its volume is preserved.

7.2.3 Finite Difference Method

It could be considered a particular case of the finite element method, where the object is sampled using a regular spacial grid (e.g. rectangles or rectangular prisms) [54, 57]. It is easier to implement but the approximation is not as good. In [85, 86] Terzopoulos et al. used the method to simulate elasticity, viscoelasticity, plasticity and fracture. It is also possible to apply the regular discretisation only to the time intervals with a FEM style discretisation for the spacial domain.

7.2.4 Finite Volume Method

In the Finite Volume Method the stress tensor is used directly to compute the internal force $\vec{f_A}$. If the stress tensor is taken to be constant within an element and the faces are planar, the force over the face of the element is:

$$\vec{f_A} = A\sigma\hat{n}, \tag{7.5}$$

where the scalar A is the area of the face and $\hat{n}$ its unitary normal vector. To obtain the nodal forces, the forces $\vec{f_{A_i}}$ of each face are distributed among the nodes adjacent to it. [57].

7.2.5 Boundary Element Methods (BEM)

When the material is homogeneous, it is possible to apply the Green-Gauss theorem, that transforms an integral over a volume to an integral over a surface, to an integral form of the equation of motion. This change allows for a substantial speed up

[57]. Gladilin makes a comparison of the performance of BEM vs FEM when used for medical image registration in [33]. Greminger uses BEM on deformable templates to improve the tracking of deformable objects against occlusions and spurious edges [36].

7.2.6 Long Element Method

The Long Element Method was introduced by Balaniuk et al [4, 17]. It is based on the decomposition of the object by projection of the object into three orthogonal spaces. Each point inside the object is determined by its distance to three mutually perpendicular planes that cross the object. The lines that join the point to each plane can be seen as threads. When the cross sections defined by the reference planes are divided into polygons the threads become long rods crossing the object. These are called *long elements* (LE). Each long element is assumed to behave like an incompressible fluid.

7.3 Mass-Spring Methods

Since the basics of mass-spring models have already been presented in Chapter 5, this section focuses on presenting other relevant extensions.

7.3.1 Handling Anisotropic Behaviour of Mass-Spring Systems

Mass-spring systems have a very bad weakness: since particles interact with each other only when there is a spring connecting them, their behaviour is highly dependant on the location of those springs. The simulations are more stable when the symmetry of the mesh is aligned with the applied force. To address this issue, Bourguignon and

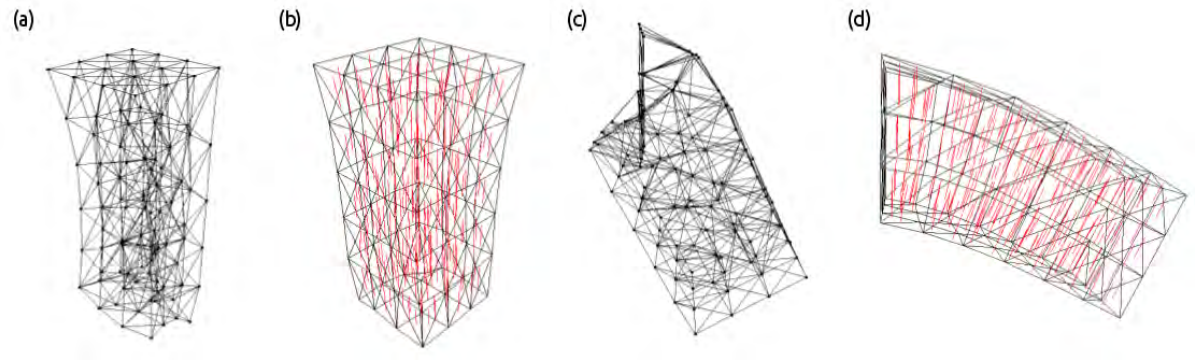

FIGURE 7.4: *For the first two examples a tetrahedral mesh undergoes a downward pull at their bottom end while their top end is fixed. (a) Undesired anisotropy is observed. (b) Same mesh and materials, but using the axes of interest defined. The behaviour is now isotropic. The next two examples correspond to attempts to model flexion. (c) State of equilibrium of a cantilever beam, whose left end is fixed, under force of gravity obtained with a standard mass-spring model. (d) State of equilibrium obtained using axes of interest. Images taken from [8], courtesy of David Bourguignon.*

Cani [8] allow the user to define three orthogonal axes along orientations of interest, per mesh element. The mechanical characteristics of the material along each of these axes can be specified as well at the barycentre of the corresponding element. All internal forces act along these axes instead of acting along the mesh edges. During deformations of the material, the three axes of interest, evolve with the volume element to which they belong. As a result, the model has better control of isotropic and anisotropic behaviour for forces being applied in different directions [Figure 7.4]. The addition of this work to our mass-spring system could improve the quality of our simulations, however we left this option for future research.

7.3.2 Use of Spheres for Volume Preservation in Mass-Spring Systems

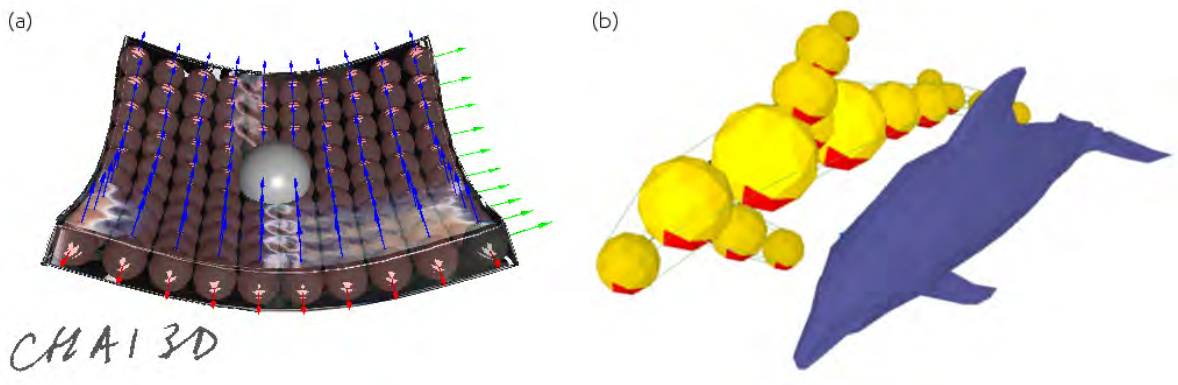

FIGURE 7.5: *(a) Snapshot of Chai, a deformable membrane is filled with spheres and connected with springs that model elongation, flexion and torsion [13]. (b) Dolphin with skeleton taken from [9].* Courtesy of Francois Conti.

Conti et al. 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 [Figure 7.5(a)]. They are placed along the skeleton of the object and are connected together with elastic links which model elongation, flexion and torsion properties [Figure 7.5(b)]. Each vertex of the mesh is attached to the nearest sphere or link [9, 13]. This system favours bending around favoured axes/regions and tends to prevent compression of the spheres. Because of this, it would not always be adequate to model arbitrarily deformable materials, like plasticine.

7.3.3 Automatic Calibration of Mass-Spring Models

There have been several attempts to automatically calibrate the parameters of a mass-spring model. Usually, a FEM simulation is used as ground truth, but no real object has been used that we know of. The idea is to search the space of parameters for the mass-spring model proposing different sets of parameters that are selected using genetic based algorithms, neural networks, gradient descent or particles [9, 55, 96]. The behaviour of the mesh controlled by the mass-spring model is compared node by node with the behaviour of the one controlled by the FEM, when both reach a steady state. The type of comparison can be an overall distance function, or differences in the forces being produced. Several simulations are performed until a set of parameters is obtained, that produces similar steady states to those produced by the FEM.

7.4 Particle Systems

The dynamical behaviour of particle systems in computer science is ruled by a computer algorithm that may include the evaluation of numeric equations as well. The individual attributes of each particle influence the dynamical behaviour of the particles over time, just as the values of variables in an equation, and are subject to change due to procedural stochastic processes. The behaviour of the particles can depend on their individual attributes only, on those of groups of particles or hierarchies. Particles can pass through three different phases during their lifetime: generation, dynamics and death. During the dynamics phase, particle attributes might change as functions of both time and attributes of other particles.

In their simplest form, when they do not interact with each other, they are quite adequate to simulate non-structured substances, like fire or sand. But particles can also interact with each other depending on their spatial relationship. A way to do this, is by associating a potential energy to each particle. Given the potential, the interactions between the particles can be ruled by the equations of traditional mechanics.

When simulating continuous materials, one problem of particle systems is that the surface is not explicitly defined. Thus, an implicit coating needs to be evaluated, for which various methods have been proposed. Some particles, specially designed for rendering, can have an orientation [57]. So far we have no knowledge of any work calibrating simulations with this type of particles, having real objects as ground truth.

7.5 Neural Networks

The two methods covered in this section make use of very different types of neural networks and are therefore relevant as examples.

7.5.1 Cellular Neural Networks

Zhong et al. formulated the problem of soft object deformation in terms of a dynamic cellular neural network (CNN) by drawing an analogy between the CNN and the elastic deformation from the energy propagation point of view. "A CNN is a dynamic nonlinear circuit composed by logically coupled, spatially recurrent circuit units called cells, which contain linear capacitors, linear resistors, and linear/nonlinear current sources." [97]. These cells are arranged on a grid in such way that the position of

the cell corresponds to a mass particle of the deformable object and the applied stress is the source of the current. Any cell is connected only to its neighbouring cells. As time evolves, the activity of a cell is propagated to other cells through the local connectivity of cells. The individual cells are nonlinear dynamical systems, but the coupling between them, i.e., the local connectivity of cells is linear. This allows the system to reproduce some of the nonlinear behaviours of deformable materials.

To emulate the effect of a deformation potential that gets propagated from the point of application of the force through the rest of the material, the equation for heat conduction is used. If the grid of cells is rectangular, finite differences can be used to discretise this equation and thus define the functions that will rule the activation of neighbouring cells. For other meshes, including triangular grids, the finite volume method can be used. A linear model of elasticity is used to vinculate applied stresses and resulting deformations. This model can lead to problems during the numerical integration similar to those of mass-spring systems. Zhong et al. proposed to use a piecewise continuous polynomial for the potential, to avoid singularities. They succeeded in developing a system for haptic and visual simulation of deformable objects.

7.5.2 Neural Gas Networks

In [20] Cretu et al. associate force and visual information of deformable objects, manipulated with a robotic hand, using purely gas and growing gas neural networks. Growing neural gas is a network that builds itself by incrementally adding node by node to a neural map, thus eliminating the constraint imposed by the fixed map size of standard unsupervised networks. The mathematical theory is exposed in [28]. In Cretu's work, measurements of the interaction force at the level of the fingertips (F_1, F_2, F_3) and of

the position of the fingertips (P_1, P_2, P_3) of a three-finger robotic hand, while grasping, are enriched with information about the contours of the deformed object tracked in a series of images.

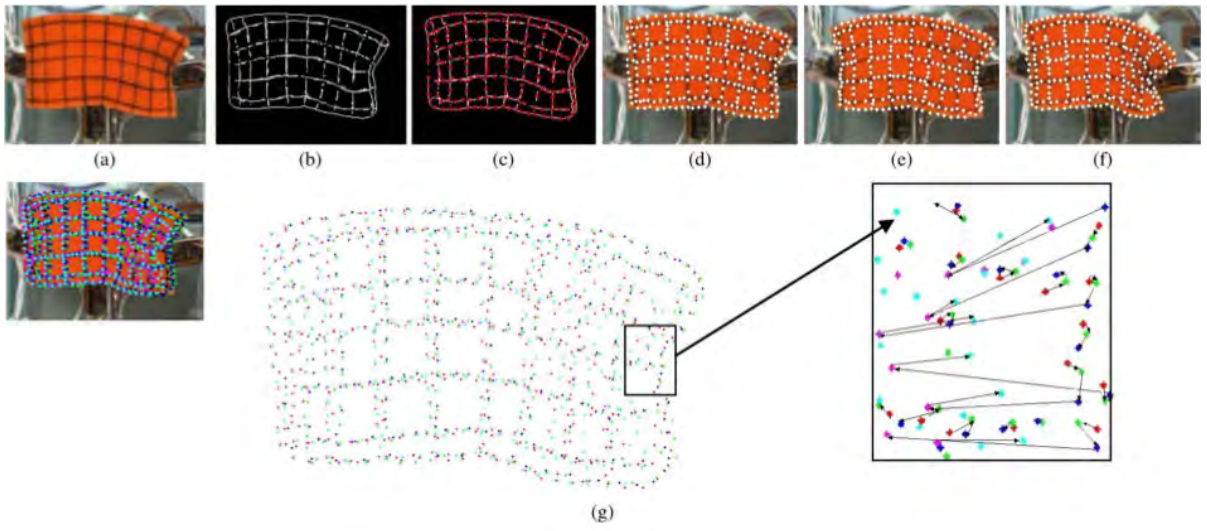

FIGURE 7.6: *"Contour and grid point tracking for an orange sponge: (a) Initial frame, (b) grid and contour points, (c) initial growing neural gas, (d)-(f) tracking using a sequence of neural gas networks, and (g) trajectory of tracked points and detail of the one-to-one correspondence of tracked points". Taken from [20], permission for reproduction pending.*

The input to the first unsupervised growing neural gas network is a set of given HSV color-coded pixels, together with their spatial coordinates (x, y) in an image. The input vector is of the form $\mathbf{P} = p_1, ..., p_n$, with p_i being a 5D vector defined as $[H_i, S_i, V_i, X_i, Y_i]$, with $i = 1, ..., n$, where n represents the number of pixels in the image. This network classifies the pixels in foreground and background. The contour of the object is identified based on the filtered image with the aid of the Sobel edge detector.

A second growing neural gas is employed to represent the position of each point over the contour. An adequate number of points is achieved due to the inherent property

of neural gas and growing neural gas networks to find compact data representations based on feature vectors, while preserving the topology of the input space.

The compact growing neural gas description of the contour is then used as the initial configuration for a sequence of neural gas networks that monitor the contour deformation in the image sequence. The number of nodes in this network is fixed and it is equal to the number of points used to represent the contour in the first frame. Each neural gas network monitors the contour of an object by predicting and readjusting the position of the neurons to follow the contour. Finally, a feedforward neural network is trained, using a backpropagation algorithm, to take the input parameters $(P_1, P_2, P_3, F_1, F_2, F_3)$ and output the corresponding object contour.

In order to test the method also on the whole surface of the objects, grids were drawn on them to track the internal deformation. Some results are show in Figure 7.6.

7.6 Probabilistic Approaches

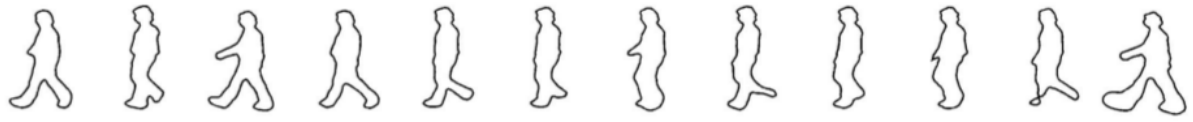

FIGURE 7.7: *Synthetically generated walking sequence, as generated by a statistically learned second order Markov model on level set embedding functions. The oscillatory behaviour of a walking person is well captured, but not all samples correspond to permissible shapes (e.g. last two silhouettes). Taken from [18], with permission.*

The main aim of probabilistic approaches is to incorporate prior information about the process of deformation of shapes, as obtained from a sequence of training samples, into a model that can predict future positions and shapes. It has been applied mainly

to segmentation and predictive tracking. However, the work by Blake [6, 7] and Cremers [18] is specially relevant to this research, because they developed systems that learn a dynamical statistical model that captures the temporal coherence of consecutive deformable shapes in a video, which therefore allows them to make predictions for the following elements of a sequence under observation [Figure 7.7].

Cremers used an *autoregressive model of order* k [58] simultaneously on a description of deformable shapes based on the principal modes of deformation of level set functions [see Section 6.2.3] and on the translation and rotation transformations, relative to the position in the preceding video frame. This allowed him to separate the deformations of the shape, from the global movement of the object. The best results were obtained with a second order Markov chain. In order to learn the parameters, he looked for the probability distribution that maximizes $\mathcal{P}(\alpha_t, \theta_t | I_t, \hat{\alpha}_{1:t-1}, \hat{\theta}_{1:t-1})$, where I_t is the image at time t, α are the shape parameters, and $\hat{\alpha}$ and $\hat{\theta}$ are the best estimates of shape and transformation obtained for the past frames. He did this by taking the negative logarithm of the Bayes formula applied to this probability, calling that an energy, and applying gradient descent to minimize it. The details of how to estimate each of the terms can be found in [18]. The predictions capture correctly the characteristics of the oscillatory movement of the object, but may still produce unlikely shapes [Figure 7.7]. Cremers also points that this "second order autoregressive model can be interpreted as a stochastic version of a time-discrete damped harmonic oscillator" [18]. In [60] North and Blake use expectation maximization to calculate a dynamical model as well.

7.7 Modal Analysis

The objective of modal analysis is to break non-rigid dynamics down into the sum of independent vibration modes [66].

7.7.1 Modal Dynamics

In [66], Pentland extracts the main modes of vibration of a deformable object, represented with superquadrics, from the original FEM system of equations. If Equation 7.4 is taken, the *whitening transform* can be used to diagonalise M, D and K[ii] simultaneously. This transform is the solution to the eigenvalue problem:

$$\lambda\phi = \mathbf{M}^{-1}K\phi, \tag{7.6}$$

where λ and ϕ are the eigenvalues and eigenvectors of $M^{-1}K$. Therefore, by using the transformation $u = \phi\ddot{\vec{u}}$ it is possible to rewrite Equation 7.4 as $3n$ independent equations:

$$\begin{aligned}
\tilde{\mathbf{M}} &= \phi^T\mathbf{M}\phi, \\
\tilde{\mathbf{D}} &= \phi^T\mathbf{D}\phi, \\
\tilde{\mathbf{K}} &= \phi^T\mathbf{K}\phi, \\
\tilde{\vec{f}} &= \phi^T\vec{f}, \\
\tilde{\mathbf{M}}_i\ddot{\vec{u}}_i + \tilde{\mathbf{D}}_i\dot{\vec{u}}_i + \tilde{\mathbf{K}}_i\vec{u}_i &= \tilde{\vec{f}}_i,
\end{aligned} \tag{7.7}$$

[ii] Remember that M is matrix of masses cumulated at the nodal points, D contains the damping terms and K describes the material stiffness

where $\tilde{\mathbf{M}}_i$ is the i^{th} diagonal element of $\tilde{\mathbf{M}}$ and so forth. Each i^th-equation describes the time course of one of the object's vibration modes. The i^{th} row of ϕ describes the deformation the object experiences as a consequence of the force $\tilde{\vec{f}}_i$, and the eigenvalue $lambda_i$ is proportional to the natural resonance frequency of that vibration mode. To obtain an accurate simulation of the dynamics of an object a linear superposition of the main modes is used.

The advantage of this modal representation is that the dynamic equations of movement are now untangled and can be solved analytically for the main modes. Each mode will have a solution from among the solutions for the damped harmonic oscillator [Section 5.8]. Pentland and Williams used the reduced amount of required parameters, that define the movement, to produce dynamics of deformable objects for animations. They also managed to find the right forces to produce the desired motion for simple cases[iii].

7.7.2 Geometric Modal Analysis

FIGURE 7.8: *Shape matching with geometrically inspired modes. The linear shear and stretch modes and the quadratic bend and twist modes are shown. Taken from [56], with permission.*

In [56], Müller et al. present a method for very efficient animation of deformable objects, based on modal geometrical transformations, geometric constraints (in analogy

[iii] This problem is know as inverse kinematics

to energy) and distances of current positions to goal positions (in analogy to forces). They do not require the connectivity information of the members of the object, but they do require to know in advance which point in the current frame corresponds to which point in the target frame. They can then interpolate appropriate transformations between them, through the use of an ad hoc integration scheme that does not overshoot the goal position and allows also for the incorporation of external forces acting on all points (e.g. gravity).

The generic starting point is the minimization of the following quantity, which is analogous to an energy: Given two sets of points $\mathbf{x}_i^0$ and $\mathbf{x}_i$, find the linear transformation matrix $\mathbf{A}$ and the translation vectors $\mathbf{t}$ and $\mathbf{t}_0$ which minimize

$$\sum_i w_i(\mathbf{A}(\mathbf{x}_i^0 - \mathbf{t}_0) + \mathbf{t} - \mathbf{x}_i)^2, \tag{7.8}$$

where the w_i are the weights of individual points (e.g their masses). The optimal translation vectors $\mathbf{t}$ and $\mathbf{t}_0$ take the initial points towards the centre of mass of the initial shape, then the transformation $\mathbf{A}$ is applied and finally the points are sent back away from the centre of mass of the final shape, i.e.

$$\mathbf{t}_0 = \mathbf{x}_{cm}^0 = \frac{\sum_i m_i \mathbf{x}_i^0}{\sum_i m_i}, \mathbf{t} = \mathbf{x}_{cm} = \frac{\sum_i m_i \mathbf{x}_i}{\sum_i m_i} \tag{7.9}$$

A can be decomposed in rotational and scaling parts and can represent shear and stretch deformations. The goal positions for each point are calculated by applying the translations and rotation, or the translations and a linear combinations of the rotational part and the full A. A further extension that uses generalized coordinates $\tilde{\mathbf{q}} = [q_x, q_y, q_z, q_x^2, q_y^2, q_z^2, q_x q_y, q_y q_z, q_z q_x]$ includes quadratic transformations that allow bending and twisting to be performed. See Figure 7.8.

They can apply the same sets of transformations to clusters of points, instead of doing the calculations per point, thus making the whole process much more efficient.

7.8 Summary

To simulate the dynamics of deformable objects the most common approach is to make use of physics based methods that involve differential equations. Since these equations must be solved numerically, there is a great variety of ways to approximate them. The most accurate, but resource consuming, are Finite Element Methods (FEM); while the most intuitive and easy to implement are Mass-spring Systems. Both these methods have been used for simulations, but either they require a careful process of calibration or the approximations are not good, or they are not very stable. However, if the calibration is adequate, they can be used for visual and haptic simulations in virtual environments, working at interactive rates (e.g. [92]). There have been attempts to calibrate them automatically like Morris [55], who used simulated annealing and Howard [40], who used artificial neural networks, both over a damped mass-spring system; and Frank [26], who estimated the average value of the constitutive parameters for a simple FEM system.

It is also possible to apply a model of interactions among individual particles. This method has been used mainly for computer animation. A final option is to try to capture directly the dynamics with neural networks or probabilistic methods like Markov processes, sometimes using principal modes of deformation as basis to represent possible shapes [18]. However these methods have been used only to predict the shape of an object for the next frame [20] or their behaviour is analogous to that of physics based systems [97].

For the present research we are interested in finding a combination of techniques that will allow a robot to make predictions, through the use of a method that the robot should calibrate autonomously. The following chapter presents in detail our concrete research scenario, the methodology we chose and the reasons for this choice. However, it is important to recognize that some of the methods presented here show great potential and we keep them as reference for future research.

Chapter 8

Scenario

From the literature review it can be seen that the most developed methodology to predict the behaviour of deformable objects is the use of physics based models. Of these, mass-spring models have been successfully implemented in virtual reality systems to simulate some deformable objects at real time rates; even though some issues regarding stability and difficulties in choosing good parameters for the simulations still remain. Additionally, mass-spring systems have proved to be a very good mathematical basis for modelling complex materials and interactions. The likelihood that they can be used to approximate the behaviour of a wide variety of materials is high, particularly if the model is extended to incorporated permanent deformations (plasticity). Therefore, we considered mass-spring models the most adequate candidates.

Regarding the learning of the parameters for the model, previous works calibrate mass-spring models using FEM simulations as ground truth, instead of real objects, and they compare only the final stable configurations. Furthermore, we are interested in evaluating the similarity between the simulated behaviour and the behaviour of the

real object on a frame by frame basis, in real time, since this is the kind of interactions a robot would be subject to in real life environments. For this purpose, we designed the experiments explained in this chapter.

8.1 Experimental Set-up

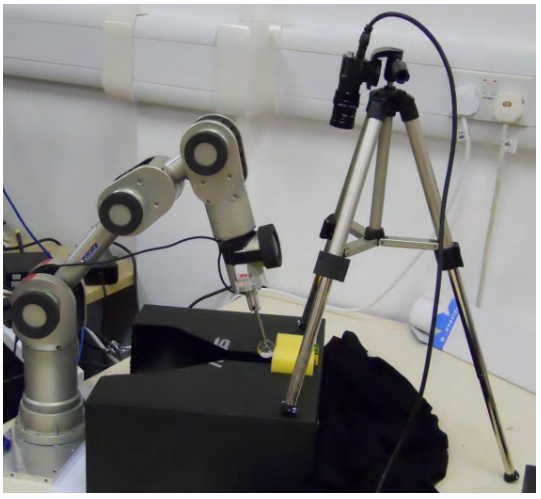

FIGURE 8.1: *Experimental setup. Viewing from a side, a one fingered Katana arm with a force sensor pushes a sponge. The transparent finger pushes the sponge away from it. On the other end, a pencil is used as an obstacle always opposed to the finger. A camera observes and records the action from the top.*

The central idea is to allow the robot to interact with the material in the most simple and informative way possible. This was achieved by allowing it to push a block of material of $8.5\,\mathrm{cm}$ length, $5\,\mathrm{cm}$ width and $1.8\,\mathrm{cm}$ depth, against an obstacle. Data was captured in two sets.

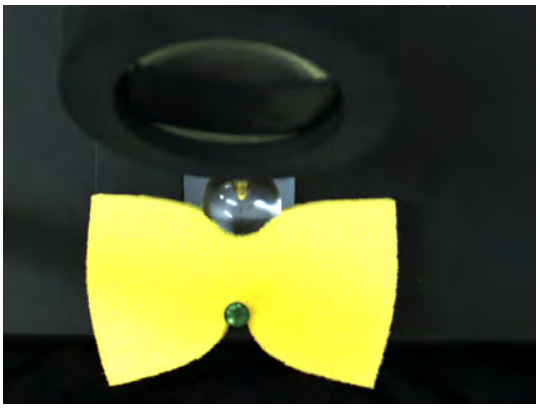

FIGURE 8.2: *First experimental setup. Top/camera view. A one fingered Katana arm with a force sensor pushes a sponge. The transparent finger [top] pushes the sponge away from it. On the other end, a pencil cap is used as an obstacle [green circle below].*

8.1.1 First Set of Experiments: Pressing a Sponge

For the first set of experiments a spheric finger mounted on a Katana arm pressed a dish-washing sponge against the cap of a pencil. A camera recorded the action from a perpendicular view [Figure 8.2]. This first set was used to test our implementation of the mass-spring method and the learning algorithm.

8.1.2 Second Set of Experiments: Pressing and Releasing a Sponge and Plasticine

For the second set of experiments a cylindrical finger, mounted on a drill press, pressed and released a block of target material against the floor. The materials analysed were a dish-washing sponge and a block of putty. A camera recorded the action from a perpendicular view [Figure 8.3]. The scale between the real world measurements and

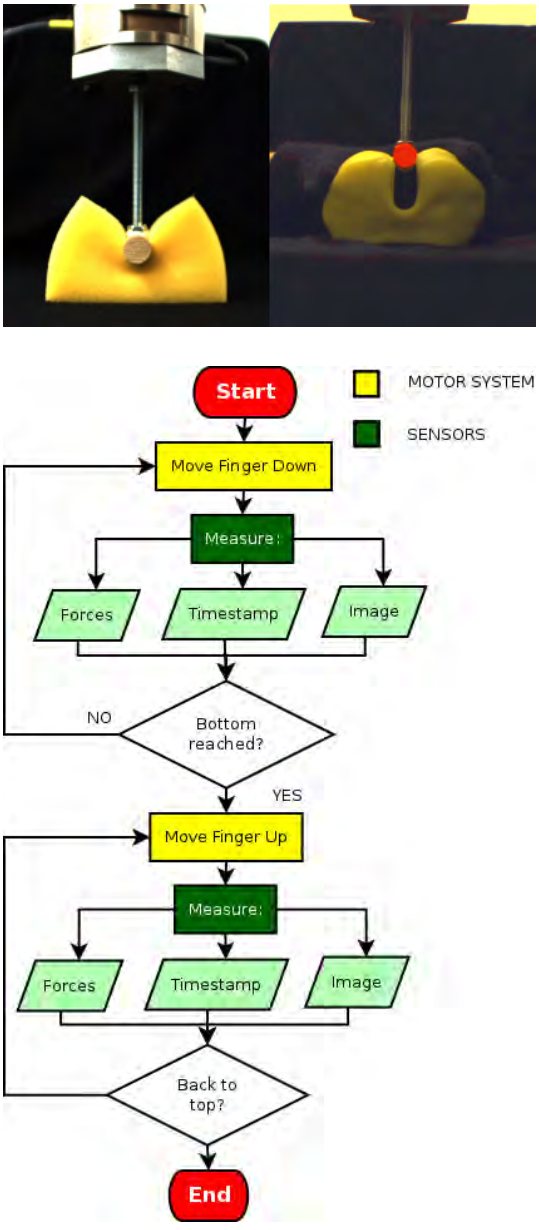

FIGURE 8.3: *Second experimental setup. Top: Front/camera view. A cylindrical finger mounted on a drill press with a force sensor pushes and releases a sponge [left] and plasticine [right] against the floor. Bottom: Flux diagram of the actions of the robot for the gathering of data.*

the pixels in the image were: $4600\frac{\text{pixels}}{\text{m}}$[i] 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). An image and a force measurement were taken in synchrony every $0.17 \pm 0.015s$. The movement was slow enough for the images to capture small transitions between frames. The change of area between frames is of the order of 2600 pixels, of a total area of around 1,240,00, this represents the $0.001 \pm 0.0004\%$ of the total area. To test integration steps that were smaller than this interval, it was possible to use linear interpolation to approximate force values between readings [Figure 10.19 and Figure 10.20]. However, the quality of the simulation was evaluated only on those steps where an image was available. This was important because the evaluation function was the slowest step. This second set was used to test our implementation of the mass-spring method, the additions for plastic behaviour and the learning algorithm, because we had data available for elastic and plastic materials, and about the behaviour of the material while the finger was retrieved, where the difference between both behaviours becomes more noticeable.

8.1.3 Training and Testing

For both sets of experiments, each of the objects 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.

[i] $\text{m} = metre$

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 was selected. The other two videos were used to test whether the predictions made by the model for the new interactions also resembled the behaviour of the real object.

8.2 Input

The robot received information about the object in three ways: through a 2D camera, with a force sensor mounted on its effector and with the computer clock. A timestamp was used to associate each photograph with its corresponding reading of the force and to know the time length of the interaction between readings.

8.2.1 Image

The information about the image of the object was obtained through a colour firewire camera. It consists of a series of photographs with a resolution of 800×600 pixels, every $0.17 \pm 0.015s$. For practical reasons some were compressed as videos.

8.2.2 Force Sensor

Two different sensors were used for the different sets of experiments: the DAQ-FT-Nano17, that can measure forces up to 25 Newtons and torques up to 250 Newton-millimetre, for the first set of experiments; and the 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, for the second set. It was important to use the gamma sensor for the

second set, because the forces involved in pushing plasticine are much bigger than the ones required to bend the sponge.

8.2.3 Timer

For each photograph taken, there was a corresponding force measurement. The internal clock of the computer was used to measure the time between readings. These intervals are required by the simulator to produce the predictions of the physics based model, by applying the corresponding force at the right time, during the appropriate length. It is important to notice that, from here, the problem has already been discretised. The force function $F(t)$ or the shape are not obtained exactly, but only sampled versions.

8.3 Internal Representation of the Shape

There are two aspects to be considered for the internal representation:

1. The representation of the ground truth as extracted from the sensors.

2. The representation used and given by the prediction module.

Both representations can be different. In our implementation they are different but it is also necessary to have a way to compare them. The following sections specify the details.

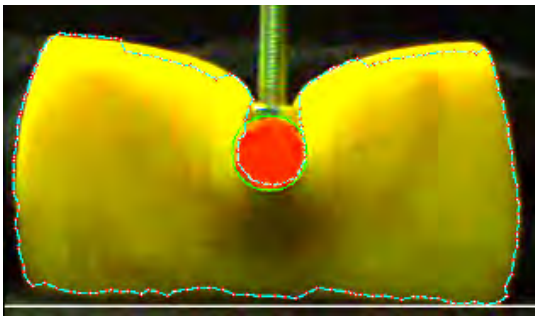

FIGURE 8.4: *The contour of the target is approximated by a linear snake, a kind of flexible polygon, that can have hundreds of sides.*

8.3.1 Contour: Linear Splines

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* [Figure 8.4]. **Note:** Solid obstacles are represented in a very simple manner: the finger and the pencil cap are represented as circles, while the floor is represented as a rectangle.

8.3.2 Surface: Meshes

From among the several representations for 3D deformable objects, here it was chosen to simplify the representation to a 2D mesh for the following reasons:

1. The representation is deformable, like the original material.
2. Given that the robot receives visual information in 2D, having an internal representation also in 2D avoids, temporarily, the problem of reconstructing the 3D structure.

FIGURE 8.5: *The deformable target is represented internally as a mesh. Here the mesh is overlapped with the ground truth.*

3. It is possible to represent interactions between solids (rigid and deformable). Unfortunately, as is the case with any computer representation, it poses some problems when required to detect contact with obstacles.

4. The model provides information beyond the points where data was collected [interpolation and extrapolation] e.g. beyond the points where testing forces were applied and deformations or displacements were registered. Observe that, in particular, the force sensor in this experiments only provides readings in one point, but the shape of the robotic finger is actually a sphere (or cylinder). Also, the initial shape and position of the mesh is given for the first frame, but the model is be able to deform the initial representation accordingly.

For mass-spring models, the shape of the mesh can lead to undesired anisotropic behaviours, since forces are applied only through the edges (which become springs) [8]. To address this, to a certain extent, we opted for generating a triangular mesh whose symmetries tend to compensate the anisotropy [Figure 8.5]. The mesh was implemented as a map of faces, edges and vertices. Information about the original and

current length and areas of the elements are calculated and stored once per simulation step.

The main inconvenient of this approximation is that we are not modelling the 3D deformation of the object, but the 2D deformation of the image of one of its faces, which does not take into account displacements of matter in perpendicular directions. The 2D recovery terms will press to recover the area between the triangles closest to the finger, when 3D tetrahedrons could compensate their volume by displacing matter perpendicularly to the camera. This will be reflected in loss of stability in the region due to extreme compression of the triangles. This effect is present specially with the plasticine. While conscious about this, we tested the general suitability of the method to determine whether it is worth making a new implementation with a 3D model, and remained tolerant with this issue.

8.4 Deformation Process

Because of the dynamic nature of the problem, it is necessary to adapt the representation of the target object for each frame.

8.4.1 Tracking: Linear Snakes

The tracking of the deformable target in the ground truth has two stages:

1. The target object is segmented from the environment.

2. The internal representation of its contour is adapted to its new shape.

8.4.1.1 Identification

With the intention of simplifying the identification of the different objects, the target was bright yellow, the finger got a bright orange sticker and the background was black. However, due to the 3D nature of the problem, which produces deformations perpendicular to the camera, which produce shadows, some reflections on the metal of the robotic finger and some problems with the illumination, colour segmentation was not enough, specially for the second set of experiments. For this reason, a combination of a Canny edge detector (described below) and colour segmentation was used to guide a linear snake. The parameters for the hue-saturation-value (HSV) segmentation and Canny can vary from video to video. The open libraries ffmpeg[ii] and OpenCV[iii] were used to make the program that processes the visual information.

Applying Canny is composed of three steps: first, the image is turned grey; second, it is blurred with a 9×9 Gaussian kernel to reduce noise; finally, Canny is called with parameters $lower\_threshold = 6$, $upper\_threshold = 3 \times first\_threshold$ and $kernel\_size = 3$. This process returns a sparse matrix that indicates which pixels of the image are more likely to be edges, but does not know which belong to the deformable target.

The hue space is divided in H_N slices. The Hue component, preserved from the original image, is used to filter the edges in the image and keep only those edges that are likely to belong to the target. These are the edges whose hue falls within the slice corresponding to the dominant colour of the target. This prevents the snake from getting distracted with the wrong edges. The next stage is to pull the snake towards the target. This is accomplished with the tracking algorithm.

[ii] http://ffmpeg.org/
[iii] http://opencv.org/

8.4.1.2 The Tracking Algorithm

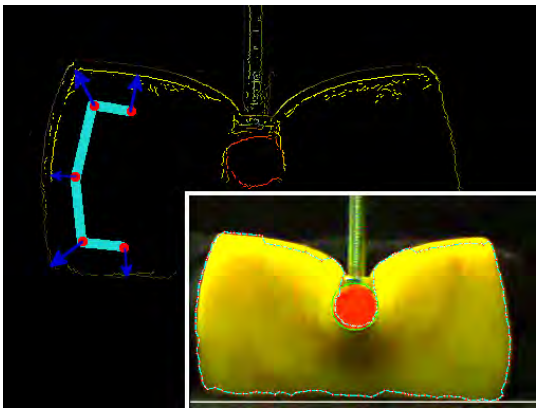

FIGURE 8.6: *The control points of the linear snake adhere themselves to the closest edge. They reassign their positions for each frame, and delete or add neighbours as they are required. The image in the back shows an amplified illustration of this concept. It also shows the edges as they are detected by the segmentation algorithm. Frequently these edges are not continuous and pieces of them may disappear for some frames.*

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, up to a certain threshold distance [Figure 8.6]. A distance of 12 pixels has worked for our experiments. If there is no edge before that, the control point remains were it is. This criterion was selected to make the representation more tolerant to edges that are easier to detect on some frames than in others. Once the edge reappears, the control point adapts itself again. Given the fact that the target does not move drastically from one frame to another, the likelihood of a control point being left behind because of this assumption is small. In our videos it only swaps to parallel edges detected by Canny. However, if this aspect had to be addressed, it is important to notice that this implementation of snakes still does not make use of any energy function or continuity constraint, which leaves a margin for improvement.

The reason why the energy function has not been used, is because it is more time consuming than our simple version. Furthermore, our algorithm is good at tracking new corners (as long as Canny can detect them), while the original one tends to over-smooth them. Since the energy terms for smoothness and hardness of traditional snakes are global, segments that should bend, do not; or segments that should not, do. This feature is important for our scenario, specially in the experiments with the plasticine, where the finger of the robot creates new corners and concavities. Additionally, since our main focus was not in tracking, we left further improvements for future work.

Algorithm 1 Regularise Linear Spline

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

Traditional snake tracking algorithms use a fixed number of control points. Our choice of using plasticine defies algorithms with a fixed number of elements to represent the shape, because it is always possible to deform it so much that it will require more elements, or simplify it so much, that most elements will become spurious. Therefore an innovation in our algorithm for snakes, specially relevant to our scenario, is that the

snake is regularized for each frame in such a way that the distance between two consecutive control points remains within the interval $[min\_distance, max\_distance]$ and two adjacent edges do not form angles smaller than min_angle [Algorithm 1]. This allows the snake to adapt very quickly to the complexity of the contour it represents.

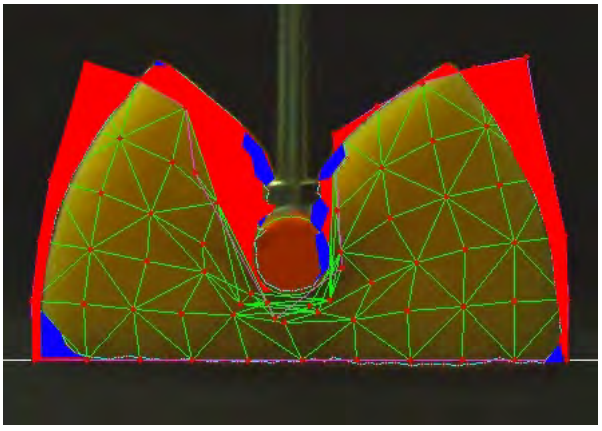

FIGURE 8.7: *This frame illustrates the typical errors of the tracking algorithm (blue) and the prediction system (red) after some frames. Since the failures of the prediction system are much bigger, we did not invest more effort in improving the tracking system.*

Some slight errors are introduced by glosses and shadows that make Canny fail to detect segments of the edges in some frames, but those errors are small in comparison with the errors observed during the simulations, so we have not made an effort to improve this [Figure 8.7]. The ground truth is considered to be the area enclosed by the tracked contour. Therefore this method considers only deformations without occlusions.

8.4.2 Prediction

The approach of this research is to code the basics of a mass-spring like model in the program of the robot, and allow the robot to automatically calibrate a model for each

new material it interacts with. To calibrate the model means to find a proper set of elastic and plastic parameters.

During the simulation, the mesh corresponding to the initial shape of the target is generated from a rectangle that approximates the block of material. From then on, once the program has:

- The right values for the elastic and plastic parameters of the model.
- The force being applied at each frame.
- The position of the obstacles at each frame.

The robot can approximate the behaviour of the target for the next frames[iv] for pressures applied in different places, by using the mass-spring model and the geometric constraints (collision detection and keeping the planarity of the 2D mesh Section 9.4).

8.4.2.1 Physics Based Model

This work is mainly based on the mass-spring like model proposed by Teschner et al. [88], as described in Section 5.9.3. We added some extensions to take into account the 2D nature of the mesh (even though the real object is three dimesional) and the shape of the effector, which is not a point mass, but has physical extent. They are explained in detail in Chapter 9.

[iv] 50 frames = 11 seconds approximately, for the first set of experiments. During this time, the robotic finger pressed the sponge. 100 frames = 19 seconds approximately for the second experiment. During this time the finger compressed the material followed by the retrieval of the finger.

8.5 Learning: Automatic Calibration

With the exception of Cretu [20], other works in the literature that focus on predicting the behaviour of the deformable object (rather than only on the control strategy for the manipulator), have used machine learning algorithms only to calibrate variants of mass-spring and FEM models. Therefore we expected that we would obtain a better intuition of the working of the models and the deformable objects, by exploring the physics based models first. A natural consequence was that our approach consisted as well in calibrating a physics based model, using a search strategy. However, we placed a stronger emphasis on evaluating the quality of the simulation against real objects, than previous works.

It was know that the value of the spring constants is highly dependent on the shape of the mesh of springs, and it was not evident how to determine those values directly from the behaviour of the material [55]. With such uncertain beginning it was more adequate to begin the calibration with a random sampling of a large space of possible values. Those values that produced the best simulations could be used as seeds to explore a region around them, in search of better sets. This structure clearly suggested the use of an evolutionary algorithm [3]. The best parameter set after a fixed number of generations is adopted as a predictor. The details can be seen in Section 9.6.1.

8.6 The System

All the components explained in the previous sections are integrated into a system. This system works in two stages: training and runtime. During runtime, the system receives information about the initial shape of the object to be modelled, the obstacles

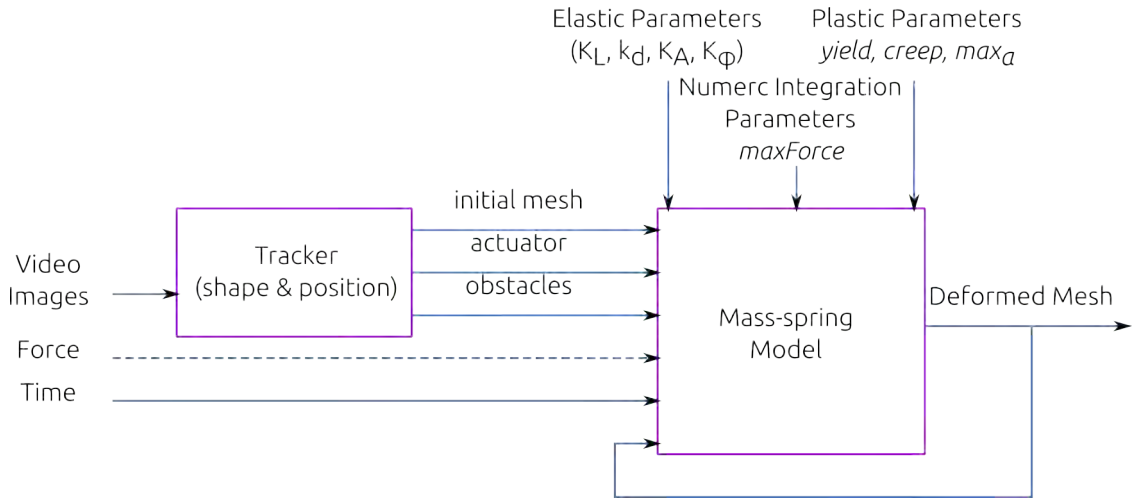

FIGURE 8.8: *Runtime components of the system. Visual and haptic information about the environment of the target object are fed into a calibrated mass-spring model. The calibration yields the appropriate values for the elasticity parameters (EP), plasticity parameters(PP) and parameters for the numeric integration method (NP). The model predicts the shape of the target throughout those frames.*

in the environment, the actuator and, optionally, the reaction forces detected by a force sensor mounted in the actuator. Given that information, the calibrated mass-spring model predicts the shape of the target for the subsequent frames [Figure 8.8].

The training part of the system makes use of the runtime part. A component generates randomised values for the parameters of the system. The shape of the mesh predicted by the mass-spring runtime simulator is compared with the area enclosed by the contour of the real object, as detected by a tracker. The set of parameters that obtained the best marks after a number of trials is returned as an answer [Figure 8.9]. This set of parameters can be used later on the run-time simulator for a variety of applications.

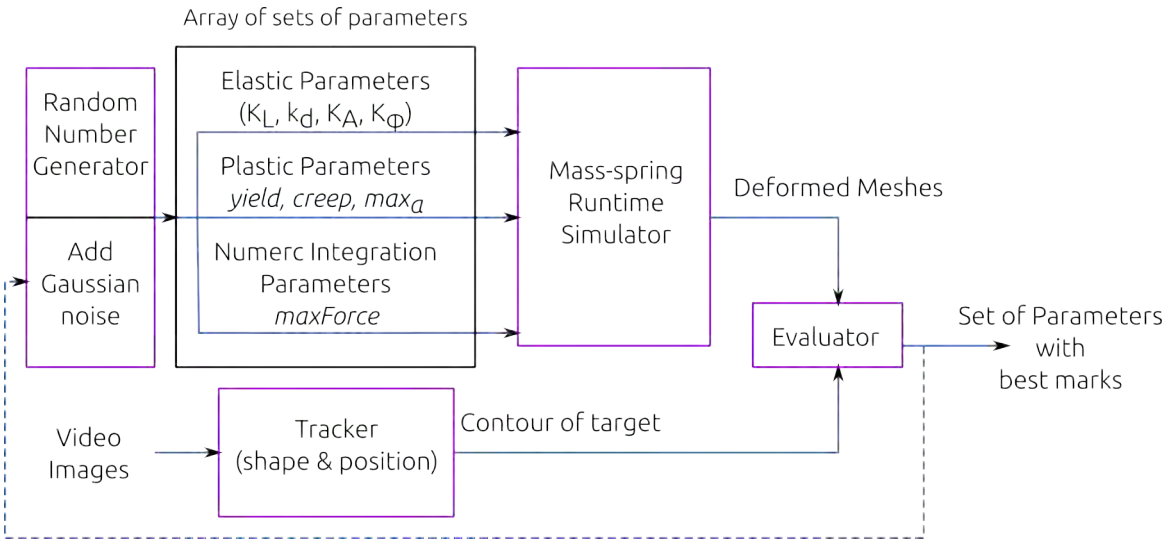

FIGURE 8.9: *Components of the system during learning time that implement the Evolutionary Algorithm. A random generator proposes sets of values for the elasticity parameters (EP), plasticity parameters(PP) and parameters for the numeric integration method (NP) of a mass-spring model. The simulations produced with them are compared with the detected shape of the ground truth. A group of the best candidates can also be used to generate new proposals by adding Gaussian noise to them. The set of parameters that produces the best simulation is returned as answer.*

8.7 Summary

This chapter presented the concrete choices made for this research, for each stage of the robotic interaction with the deformable object: the experimental set ups, the use of a linear snake and a 2D mesh for the internal representations of the shape, the use of a mass-spring like method for the simulation of the deformation of the object, which is calibrated autonomously by using a genetic algorithm. Finally, the different components have been integrated into a system. The following chapters explain our concrete extensions to the original mass-spring model, the details of the implementation and the analysis of the results.

Chapter 9

Extensions to the Physics Based Model of Deformable Objects for Robotics

This chapter presents the main contributions to the physics based model used for the problem of allowing a robot to learn to predict the behaviour of deformable objects, through the automatic calibration of a mass-spring like model of elasto-plastic deformation, from the information it gathers while pushing the object. The main contributions, in order of appearance are:

1. The use of a physics based model to predict the behaviour of a deformable object that a robot pushes.

2. The addition to the theoretical model of an energy term that enforces the preservation of angles in the mesh.

3. The use of sensed reaction forces and graphic constraints that incorporate information about the geometry of obstacles in the environment and the actuator.

4. The use of an evolutionary search strategy to calibrate the model making use of a real object as ground truth.

5. The use of the calibrated models to classify the learned materials, when the robot interacts with them in a different way.

These contributions were achieved on the basis of our main research hypotheses, stated below.

9.1 Research Hypothesis

When working with materials that present a solid behaviour at rest, but that allow for deformations of their shape when forces are applied to them (e.g. sponge and plasticine) [Section 1.3]:

- Hypothesis 1:

  Can a Mass-spring model be learned from position & force data of the deformations of real solid deformable objects?

  - ⋆ Can it be calibrated automatically from real data?
  - ⋆ Can it make predictions of shape deformations given novel actions?
  - ⋆ Can it model both elastic & plastic deformations?
  - ⋆ Can it be used for classification of materials?

- Hypothesis 2:

  Can a regression model be learned from the stress-strain data generated by an action on a material?

- ⋆ Can it predict the stress-strain data for a new action?
- ⋆ Can it serve as input to the mass-spring model?
- ⋆ Can this two-stage model be tuned for elastic and plastic objects?
- ⋆ Can this two-stage model be used for classification of materials?

9.2 Extensions to Teschner's Model

The original mass-spring model proposed by Teschner is based on constraints that take the form of energies that must be minimized [Section 5.9.3]. These energies depend on the 3D deformation of the object of interest. To actually implement this idea, but using 2D information coming from real 3D objects it was necessary to add a term that would help to preserve the planarity of the original triangulation, just as the original term for the preservation of volume did the work for 3D tetrahedra. Also, to minimise the energy, the gradients were derived explicitly and used to calculate the forces that should act on each vertex of the mesh. These forces, as estimated by Morris [55], were presented in Section 5.9.3. The following sections introduce minor corrections to the formulas, that arise when calculus formulas are applied, instead of only geometric arguments.

9.2.1 Preservation of Length with Damping Along the Spring

From the generic formula of Equation 5.39, that is repeated below, it is possible to obtain an expression for the damping of a spring in a 2D mesh. Let p_i be the position of the i^{th} mass particle, C the deformability constraint, k the elasticity constant, k_d the

damping constant and v_j the velocity of the j^{th} mass point.

$$\vec{F}^i(p_0, ..., p_{n-1}, v_0, ..., v_{n-1}) = \left(-kC - k_d \sum_{0 \leq j < n} \frac{\partial C}{\partial p_j} \vec{v}_j \right) \frac{\partial C}{\partial p_i}, \tag{5.39}$$

with the constraint for preservation of length being:

$$C = \frac{|\vec{p}_j - \vec{p}_i| - D_0}{D_0}, \tag{9.1}$$

the derivation of the explicit formula for the damped force is as follows:

$$\begin{aligned} \vec{F}^i(p_i, p_j, v_i, v_j) = & \left(-k \left(\frac{|\vec{p}_j - \vec{p}_i| - D_0}{D_0} \right) \right. \\ & \left. -k_d \sum_{0 \leq j < n} \frac{\partial}{\partial p_j} \left(\frac{|\vec{p}_j - \vec{p}_i| - D_0}{D_0} \right) \vec{v}_j \right) \frac{\partial}{\partial p_i} \left(\frac{|\vec{p}_j - \vec{p}_i| - D_0}{D_0} \right) \end{aligned} \tag{9.2}$$

$$\begin{aligned} = & \left(-k \left(\frac{|\vec{p}_j - \vec{p}_i| - D_0}{D_0} \right) \right. \\ & \left. -k_d \left(\frac{\partial}{\partial p_j} \left(\frac{|\vec{p}_j - \vec{p}_i| - D_0}{D_0} \right) \cdot \vec{v}_j + \frac{\partial}{\partial p_i} \left(\frac{|\vec{p}_j - \vec{p}_i| - D_0}{D_0} \right) \cdot \vec{v}_i \right) \right) \\ & \frac{\partial}{\partial p_i} \left(\frac{|\vec{p}_j - \vec{p}_i| - D_0}{D_0} \right) \end{aligned} \tag{9.3}$$

$$\begin{aligned} = & \left(-k \left(\frac{|\vec{p}_j - \vec{p}_i| - D_0}{D_0} \right) \right. \\ & \left. -k_d \left(\frac{1}{D_0} \left(\frac{\vec{p}_j - \vec{p}_i}{|\vec{p}_j - \vec{p}_i|} \right) \cdot \vec{v}_j - \frac{1}{D_0} \left(\frac{\vec{p}_j - \vec{p}_i}{|\vec{p}_j - \vec{p}_i|} \right) \cdot \vec{v}_i \right) \right) \frac{-1}{D_0} \left(\frac{\vec{p}_j - \vec{p}_i}{|\vec{p}_j - \vec{p}_i|} \right) \end{aligned} \tag{9.4}$$

$$\vec{F}^i(p_i, p_j, v_i, v_j) = \frac{k}{D_0^2} (|\vec{p}_j - \vec{p}_i| - D_0) \frac{\vec{p}_j - \vec{p}_i}{|\vec{p}_j - \vec{p}_i|} + \frac{k_d}{D_0^2} \left(\frac{\vec{p}_j - \vec{p}_i}{|\vec{p}_j - \vec{p}_i|} \cdot (\vec{v}_j - \vec{v}_i) \right) \frac{\vec{p}_j - \vec{p}_i}{|\vec{p}_j - \vec{p}_i|} \tag{9.5}$$

It is common practice to incorporate all constant values into one. In this way Equation 5.34 corresponds with the first term of Equation 9.5. However, Teschner's intention, when dividing the constraint by D_0, was to obtain constants that would be scale independent. Morris's geometric arguments, while correct, do lose that invariability. Here we introduce it again. It has the consequent effect that, if we obtain a good set of parameters for a mesh of 20×10 elements, the same set can be used for a mesh of 10×5. It is not as straightforward to do the same derivation for the other cases, but a similar method will be used to estimate the normalization constant for the preservation of areas without damping.

9.2.2 Preservation of Areas

Since Teschner [88] and Morris [55] found it not particularly helpful to add damping to the preservation of areas, it will be avoided here. Accordingly, the derivation of the force depends on the terms:

$$\vec{F} = -kC\frac{\partial C}{\partial \vec{p}_i}, \tag{9.6}$$

$$C = \frac{\frac{1}{2}|(\vec{p}_j - \vec{p}_i) \times (\vec{p}_k - \vec{p}_i)| - A_0}{A_0} \tag{9.7}$$

Where F is the force, k the spring constant, C the constraint that enforces the recovery of the original area A_0 and p_i, the coordinates of the i^{th} mass. Applying the rules of

calculus:

$$\frac{\partial C}{\partial p_i} = \frac{1}{2A_0}\frac{(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})}{|(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})|} \left[\frac{\partial}{\partial p_i}(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i}) + (\vec{p_j}-\vec{p_i})\times\frac{\partial}{\partial p_i}(\vec{p_k}-\vec{p_i})\right] \tag{9.8}$$

$$= \frac{1}{2A_0}\frac{(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})}{|(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})|} \left[-\vec{1}\times(\vec{p_k}-\vec{p_i}) + (\vec{p_j}-\vec{p_i})\times-\vec{1}\right] \tag{9.9}$$

$$= \frac{1}{2A_0}\frac{(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})}{|(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})|} \left[\vec{p_k}\times\vec{1} + \vec{1}\times\vec{p_i} + \vec{1}\times\vec{p_j} + \vec{p_i}\times\vec{1}\right] \tag{9.10}$$

$$= \frac{1}{2A_0}\frac{(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})}{|(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})|}[\vec{1}\times(\vec{p_j}-\vec{p_k})] \tag{9.11}$$

$$\vec{F}^i(p_i,p_j,p_k) = -\frac{k}{2A_0}\frac{\frac{1}{2}|(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})| - A_0}{A_0}\frac{(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})}{|(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})|}[\vec{1}\times(\vec{p_j}-\vec{p_k})] \tag{9.12}$$

$$Area = \frac{1}{2}|(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})|, \tag{9.13}$$

$$\vec{F}^i(p_i,p_j,p_k) = -\frac{k}{2A_0^2}(Area - A_0)\frac{(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})}{|(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})|}[\vec{1}\times(\vec{p_j}-\vec{p_k})] \tag{9.14}$$

Even though it is not evident how to establish the equivalence between Equation 9.14 and Equation 5.36, except for their geometrical similarity[i], it is possible to recover the scale independence if the magnitude of the force is given by:

$$forcemag_A(\vec{p_i}) = \frac{\frac{1}{2}|(\vec{p_j}-\vec{p_i})\times(\vec{p_k}-\vec{p_i})| - A_0}{A_0^2} \tag{9.15}$$

[i] Both are the gradients of the area of the triangle formed by the three verticies.

While the rest remains:

$$\vec{F}_A(\vec{p}_i) = k_A \cdot forcemag_A(\vec{p}_i) \cdot forcedir_A(\vec{p}_i) \tag{5.36}$$
$$forcedir_A(\vec{p}_i) = \frac{\vec{F}_A(\vec{p}_i)}{|\vec{F}_A(\vec{p}_i)|} = \frac{areagradient(\vec{p}_i)}{|areagradient(\vec{p}_i)|}$$
$$areagradient(\vec{p}_i) = (\vec{p}_i - \vec{p}_j) - \left((\vec{p}_k - \vec{p}_j) \cdot \frac{(\vec{p}_k - \vec{p}_j) \cdot (\vec{p}_i - \vec{p}_j)}{(\vec{p}_k - \vec{p}_j) \cdot (\vec{p}_k - \vec{p}_j)} \right)$$

9.2.3 Preservation of Angles

Given that the original system proposed by Teschner was designed for 3D meshes, it became necessary to modify it to prevent the triangles of the planar mesh from flipping. Adding a term that enforces the preservation of angles alleviates this problem. For this term the energy depends on the difference of the angles between adjacent edges.

Energy[ii]:

$$E_\varphi(\varphi) = \frac{1}{2} k_\varphi (\varphi - \varphi_0)^2 \tag{9.16}$$
$$\varphi(p_i, p_j, p_k) = \arccos \left(\frac{(p_j - p_i) \cdot (p_k - p_i)}{\|p_j - p_i\| \, \|p_k - p_i\|} \right)^2$$

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. Contrary to the previous cases, it is not so evident in which direction the force will act. Given that the force acts in the direction of the gradient, it will tend to restore the angles in the most efficient way, but it may produce

[ii]It was also considered to multiply E_φ by the lengths of the edges, but it hasn't improved the performance of the model.

tiny or very big triangles if it is not accompanied by some of the other terms proposed by Teschner, that tend to restore the original dimensions, apart from the angles.

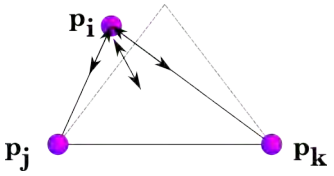

FIGURE 9.1: *The angular force displaces the vertex of interest in the direction of maximum change of the angle in order to recover its original value. This direction is a linear combination of the vectors that emerge from both edges forming the angle, and does not respect the original size of the triangle.*

The force emerging from this term is a linear combination of the vectors along the edges that form the angle of interest, it pretends to restore the original angle, but does not take the original size into account. See Figure 9.1. Therefore, it helps to recover a similar triangle, but if used alone can collapse or explode the triangle. The derivation of this force is as follows:

$$F_\varphi(p_i) = k_\varphi(\varphi - \varphi_0)\frac{\partial \varphi}{\partial p_i} \tag{9.17}$$

$$\frac{\partial \varphi}{\partial p_i} = \frac{\partial}{\partial p_i} arccos\left(\frac{(\vec{p}_j - \vec{p}_i)\cdot(\vec{p}_k - \vec{p}_i)}{\|\vec{p}_j - \vec{p}_i\|\,\|\vec{p}_k - \vec{p}_i\|}\right) \tag{9.18}$$

$$\begin{aligned}\frac{\partial \varphi}{\partial p_i} = &-\frac{1}{\sqrt{1-\left(\frac{(\vec{p}_j - \vec{p}_i)\cdot(\vec{p}_k - \vec{p}_i)}{\|\vec{p}_j - \vec{p}_i\|\,\|\vec{p}_k - \vec{p}_i\|}\right)^2}} \\ &\left\{\frac{[-(\vec{p}_k - \vec{p}_i) - (\vec{p}_j - \vec{p}_i)]\,\|\vec{p}_j - \vec{p}_i\|\cdot\|\vec{p}_k - \vec{p}_i\|}{\|\vec{p}_j - \vec{p}_i\|^2\,\|\vec{p}_k - \vec{p}_i\|^2} - \right. \\ &\left.\frac{(\vec{p}_j - \vec{p}_i)\cdot(\vec{p}_k - \vec{p}_i)\left[-\frac{(\vec{p}_j - \vec{p}_i)}{\|\vec{p}_j - \vec{p}_i\|}\|\vec{p}_k - \vec{p}_i\| - \frac{(\vec{p}_k - \vec{p}_i)}{\|\vec{p}_k - \vec{p}_i\|}\|\vec{p}_j - \vec{p}_i\|\right]}{\|\vec{p}_j - \vec{p}_i\|^2\,\|\vec{p}_k - \vec{p}_i\|^2}\right\}\end{aligned} \tag{9.19}$$

$$
\begin{aligned}
&= -\frac{1}{\sqrt{1-\left(\dfrac{(\vec{p}_j-\vec{p}_i)\cdot(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|\,\|\vec{p}_k-\vec{p}_i\|}\right)^2}} \\
&\left\{-\frac{(\vec{p}_k-\vec{p}_i)+(\vec{p}_j-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|\,\|\vec{p}_k-\vec{p}_i\|}+\right. \\
&\left.\frac{(\vec{p}_j-\vec{p}_i)\cdot(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|^2\,\|\vec{p}_k-\vec{p}_i\|^2}\left[\frac{(\vec{p}_j-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|}\,\|\vec{p}_k-\vec{p}_i\|+\frac{(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_k-\vec{p}_i\|}\,\|\vec{p}_j-\vec{p}_i\|\right]\right\} \qquad (9.20)
\end{aligned}
$$

$$
\begin{aligned}
&= -\frac{1}{\sqrt{1-\left(\dfrac{(\vec{p}_j-\vec{p}_i)\cdot(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|\,\|\vec{p}_k-\vec{p}_i\|}\right)^2}} \\
&\left\{-\frac{1}{\|\vec{p}_j-\vec{p}_i\|}\frac{(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_k-\vec{p}_i\|}-\frac{1}{\|\vec{p}_k-\vec{p}_i\|}\frac{(\vec{p}_j-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|}+\right. \\
&\left.\frac{(\vec{p}_j-\vec{p}_i)\cdot(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|^2\,\|\vec{p}_k-\vec{p}_i\|}\frac{(\vec{p}_j-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|}+\frac{(\vec{p}_j-\vec{p}_i)\cdot(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|\,\|\vec{p}_k-\vec{p}_i\|^2}\frac{(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_k-\vec{p}_i\|}\right\} \qquad (9.21)
\end{aligned}
$$

$$
\begin{aligned}
&= -\frac{1}{\sqrt{1-\left(\dfrac{(\vec{p}_j-\vec{p}_i)\cdot(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|\,\|\vec{p}_k-\vec{p}_i\|}\right)^2}} \\
&\left\{\left[\frac{(\vec{p}_j-\vec{p}_i)\cdot(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|\,\|\vec{p}_k-\vec{p}_i\|^2}-\frac{1}{\|\vec{p}_j-\vec{p}_i\|}\right]\frac{(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_k-\vec{p}_i\|}+\right. \\
&\left.\left[\frac{(\vec{p}_j-\vec{p}_i)\cdot(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|^2\,\|\vec{p}_k-\vec{p}_i\|}-\frac{1}{\|\vec{p}_k-\vec{p}_i\|}\right]\frac{(\vec{p}_j-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|}\right\} \qquad (9.22)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial\varphi}{\partial p_i} &= -\frac{1}{\sqrt{1-\left(\dfrac{(\vec{p}_j-\vec{p}_i)\cdot(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|\,\|\vec{p}_k-\vec{p}_i\|}\right)^2}}\frac{1}{\|\vec{p}_j-\vec{p}_i\|\,\|\vec{p}_k-\vec{p}_i\|} \\
&\left\{\left[\frac{(\vec{p}_j-\vec{p}_i)\cdot(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_k-\vec{p}_i\|^2}-1\right](\vec{p}_k-\vec{p}_i)+\right. \\
&\left.\left[\frac{(\vec{p}_j-\vec{p}_i)\cdot(\vec{p}_k-\vec{p}_i)}{\|\vec{p}_j-\vec{p}_i\|^2}-1\right](\vec{p}_j-\vec{p}_i)\right\} \qquad (9.23)
\end{aligned}
$$

Summarising, the force is:

$$F_\varphi(p_i) = k_\varphi(\varphi - \varphi_0)\frac{\partial \varphi}{\partial p_i} \tag{9.24}$$

$$\begin{aligned}
\frac{\partial \varphi}{\partial p_i}(p_i) &= \frac{1}{d_{ji}d_{ki}\sqrt{1-\left[\frac{pp}{(d_{ji})(d_{ki})}\right]^2}}\left\{\left[1-\frac{pp}{d_{ki}^2}\right](p_k-p_i)+\left[1-\frac{pp}{d_{ji}^2}\right](p_j-p_i)\right\} \\
pp(p_i) &= (p_j-p_i)\cdot(p_k-p_i) \\
d_{ji}(p_i) &= \|p_j-p_i\| \\
d_{ki}(p_i) &= \|p_k-p_i\|
\end{aligned} \tag{9.25}$$

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.

9.3 Plastic Deformation

When an object is deformed and later the deforming force is retrieved, the degree to which the object recovers its shape can vary. With a traditional mass spring model the object always tends to recover the shape, even though it may experience oscillations around it. In the case of the energy terms that reinforce the preservation of area or angles, each by itself would not produce perfectly elastic deformation, since each of these terms can find equilibrium points with values different from those of the original shape. The preservation of area helps to model the incompressibility of the material, that is, that the total amount and density of mater is preserved, but the shape may

remain permanently deformed. The preservation of angles will tend to preserve the shape, but can be stabilized at a different size. When both terms are used in combination, all the characteristics of the shape tend to be preserved. Therefore, extra criteria are required to model permanent deformations.

The model of plastic deformation that is the most straightforward to implement considers the critical value *yield* and the proportionality constant *creep*. If the material is deformed beyond a given *yield* stress point, part of the deformation is incorporated to the model by decreasing the length of the spring at rest D_0. A second adaptation that we have left as future work is to incorporate this permanent modification also to the areas of rest of the adjacent triangles A_0 and the angles between them φ_0.

9.3.1 Compression and Dilation

The effect of the permanent deformation can be either to permanently compress the length of the spring, or to permanently expand it (up to the breaking point). Here, this permanent modification is expressed in terms of a constant of compression α, whose value 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 ($0.2D_0$), the length at rest suffers no further modifications. See Algorithm 2.

Algorithm 2 Permanent Deformation of an Edge

1: Be l the new length of the edge, after an integration step. And max_α the maximum proportional permanent deformation being allowed.
2: **if** $\alpha > 0.8$ **then**
3: $base\_length \leftarrow l$
4: **else**
5: $base\_length \leftarrow (1.0 - \alpha)D_0$
6: **end if**
7: $elastic\_deformation \leftarrow \frac{[base\_length]-l}{D_0}$
8: **if** $elastic\_deformation > yield$ **then**
9: $\alpha \leftarrow \alpha + creep * elastic\_deformation$
10: **if** $\alpha > max\_\alpha$ **then**
11: $\alpha \leftarrow max\_\alpha$
12: **end if**
13: **end if**

9.4 Graphical Constraints

Given that the main focus of this work is on the deformable object, the treatment of the rigid objects involved has been simplified as much as possible. Rigid objects are treated as geometric obstacles, this implies that any element of the deformable mesh will not be allowed to enter or cross over the enclosed region. This constraint is enforced at every frame during the simulation through the set of collision detection and repair subroutines that will be explained in the following sections. Also, the triangulation must cross over each other, or flip. Whenever the displacement of a node distorts the

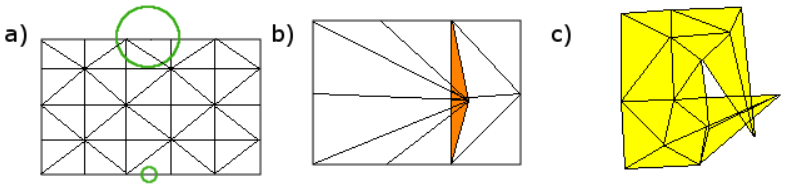

FIGURE 9.2: *Different scenarios where graphical constraints can help: (a) Collision detection prevents the representations of rigid objects from penetrating the deformable mesh. (b) When triangles are deformed too much and get inverted, they can be flipped back. (c) Triangles in the border that overlap each other cause the simulation to be discarded.*

planarity of the mesh a recover routine can try to undo the error. It is not always possible to recover from unwanted scenarios so, if a set of parameters can not respect these constraints, it is eliminated. Summarising, at each step of the simulation the triangles of the mesh must not:

1. Penetrate inside an obstacle. If they do, they are pushed away. [Figure 9.2(a)]
2. Flip. If one does, the mesh will not be planar. It must be flipped back to its original direction. [Figure 9.2(b)]
3. Collide with other triangles in the mesh. If these constraints can not be held, the simulation is discarded. [Figure 9.2(c)]

9.4.1 Collision Detection

To make the interaction with the obstacles (finger, pencil in the experiments) more realistic, they are represented as circles with area instead of as punctual forces, as happens in naive implementations of mass-spring systems for simulations. For the floor the

treatment was easier: it behaved as a lower line that can not be crossed. In all cases, the vertices and edges of the mesh are prevented from entering the obstacles.

For circular obstacles, in the case of vertices, it is enough to make sure that the distance between them and the centre of the circle is greater then the radius; if it is smaller, the vertex gets pushed out in the direction of the vector obtained by subtracting the point minus the centre of the circle, until it is outside the radius plus a small value ϵ. Since the resolution of the image is given in pixels, we used pixels as our unit of distance; therefore any value greater than zero and smaller than one should be enough for the algorithm to work. Since neighbouring edges are likely to accumulate in the region around the obstacle, before they transmit the movement to their neighbours, it would be better if the value is closer to zero than to one. In our implementation $\epsilon = 0.01$ worked well.

Algorithm 3 Closest point on the 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: $\quad closest \leftarrow a$
6: **else**
7: $\quad$ **if** $\|x\| > \|\vec{AB}\|$ **then**
8: $\quad\quad closest \leftarrow b$
9: $\quad$ **else**
10: $\quad\quad closest \leftarrow a + \|x\| \cdot \dfrac{\vec{AB}}{\|\vec{AB}\|}$
11: $\quad$ **end if**
12: **end if**

For edges, there is a known algorithm to estimate the shortest distance between a point and a line segment [Algorithm 3]; if the shortest distance between the centre of the

Algorithm 4 Push Edge Out

1: Be a, b the ends of the edge $\overline{AB}$, c the centre of the circle. Use algorithm 3 to find the closest point to the centre in the edge.
2: **if** $\|(closest - c)\| < radius$ (There is an intersection) **then**
3: $\quad displacement = (radius - \|(closest - c)\| + \epsilon) * \dfrac{(closest - c)}{\|(closest - c)\|}$
4: $\quad a \leftarrow a + displacement$
5: $\quad b \leftarrow b + displacement$
6: **end if**

circle and the edge is smaller than the radius, both ends of the edge get pushed in the direction of the vector obtained by subtracting the nearest point minus the centre of the circle, until it is outside the radius plus the same value ϵ [Algorithm 4].

9.4.2 Mesh Shape Preservation

For the preservation of the planarity of the mesh, one criterion has been used: if a triangle has been reversed (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[iii], 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.

There is a small subroutine to make sure that triangles recover some area if they got flattened at some point. This is obtained by pushing the longest edge and its opposite

[iii] $3/4$ this distance was used in our implementation. This choice produces a triangle of half the height of the inverted one, but with the correct orientation. That is, its area got reduced by half in the process, estimating that this would reduce the possibility of creating new conflicts with neighbouring triangles, or degenerating the triangle by making its area too close to zero.

vertex along the perpendicular to the edge, in opposite directions, as to form a tiny triangle[iv].

9.5 Integration Scheme

The differential equations that rule the behaviour of the system are integrated with a numerical approach [For a review see Section 5.6.3]. Another variation we introduced for the first set of experiments, was on the numeric integration scheme. Originally, the acceleration of the masses is proportional to the force according to Newton's law, the equation must be integrated twice to obtain the positions as a function of time.

Traditional mass spring methods have succeeded using the Verlet integration scheme, even though Euler can be successful as well with smaller integration steps. This yields the typical oscillatory behaviour of springs. However, the graphic constraints added to this model provoke undesirable effects: since vertices are displaced to comply with the constraints, the estimation of their accelerations and velocities are significantly altered, thus inducing oscillations and instabilities for the integration scheme that move the vertices outside from the object, or even outside from the visible area in a single integration step.

For this reason, to simulate a heavily damped sponge we made the velocities proportional to the forces. Now, instead of having a second order set of equations, the set is first order. The simulations we obtained were closer to the observed behaviour, even though, in a mathematical sense, they change the original nature of the equations of motion, as proposed by Newton.

[iv]Given that the areas are measured in $pixels^2$, an area of 2.0 units was good enough to keep values away enough from zero as to cause forces that would move the pixels away from the visible area.

$$x(t+h) = x(t) + h\frac{\vec{F(t)}}{m} \tag{9.26}$$

with $F(t) = F_D(t) + F_A(t) + F_\varphi(t)$ being the sum of the forces for the preservation of length, area and angle respectively and $x(t+h)$, the position of the vertex at time $t+h$.

For every frame of captured data[v], there were several steps of numerical integration that have to use the same measured forces. A second option was to interpolate the values of the force in between readings.

9.6 Learning: Automatic Calibration

From the previous sections it is possible to make a summary of all the parameters that must be set for the system to work.

For the tracking system they are:

- The number of intervals within hue space,
- The maximum distance a control point of the linear snake will look for a point on an edge,
- For the linear snake: the minimum and maximum length per linear segment and the minimum angle allowed between adjacent segments,
- The size of the kernel used to blur the edges,
- The second threshold for Canny. In this implementation it is calculated with the variable $ratio$ [$upper\_threshold = ratio \times first\_threshold$].

[v] Every $0.17 \pm 0.015s$.

For the moment, this values are being set up by hand and they may vary slightly from video to video. The main focus has been on the simulation, but even there, there are more parameters than originally considered, since we had focused on the theoretical model which requires the mass m, the elasticity constants k_L, k_A, k_φ and k_d, and the plasticity constants $yield$ and $creep$. However this model does not take account of numeric approximations due to being implemented in a discreet system, such as a computer. Additionally, the propagation of forces through the springs was not enough to translate the whole object, as happened with the sponge in the first experimental set up. This required the addition of a term to estimate translation f_t. The complete list is:

For the simulation:

- 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.
- The force that produces the translation of the whole object, while affected by dynamic friction, useful for the first experimental set up f_t.
- The plastic parameters: $yield$, $creep$ and the maximum amount of permanent deformation max_α.
- The integration time step Δt. After a series of experiments, using values of Δt between 0.01 and 0.5 it was determined that the best compromise between running time, number of stable simulations and fitness value of the best simulation obtained during training, was obtained with a value of $0.1 \pm 0.05s$[vi].

[vi] A single run of the simulation with $\Delta t = 0.01$ takes $156\,\mathrm{s}$ in average, while using $\Delta t = 0.1$ it takes $32\,\mathrm{s}$. This implies that the learning algorithm is almost five times slower. Regardless of the effort, the mark $\mu(F)$ did not improve over the other results. Because the quality of the prediction is evaluated

- An upper limit for the magnitude of the total forces per vertex max_force.
- The minimum area before a triangle is considered *flat*. It has been set to 2.0 units, after considering that the minimum resolution of the image is given in pixels. A unit corresponds to the area represented by a pixel^2.
- The maximum number of times a constraint, such obstacles going inside the mesh, or triangles overlapping each other, can be violated before the simulation is aborted.

The parameters that we not included within the automatic calibration are listed in Table 9.1.

9.6.1 Searching in the Space of Parameters with an Evolutionary Algorithm

From the previous lists, this work focuses on searching for appropriated values of k_L, k_A, k_φ, k_d, f_t[vii], max_force, $yield$, $creep$ and max_α. Table 9.2 shows the minimum and maximum possible values. These ranges were selected after a series of experiments where some simulations were tuned by hand. The values for $yield$ and $creep$ are limited by their definition. max_α was chosen to be smaller than 1.0 to prevent edges from collapsing permanently due to plastic deformation.

The evolutionary algorithm selects the best α individuals. It mutates the best β by adding a Gaussian noise whose standard deviation decreases from 1000.0 to 10.0 at

only on those times where there is visual information, and that is the most time consuming action, there is a small relative speed up when Δt is smaller. Nevertheless, a set of 15 runs of the learning algorithm with $\Delta t = 0.1$ took a night, while $\Delta t = 0.01$ would have taken some days.

[vii] Only for the first scenario.

TABLE 9.1: *Parameters calibrated by hand.*

Tracking System		
Parameter	Value(s)/Interval	Meaning
$\#hue\_slices$	9	The number of intervals within hue space.
$max\_distance$	$[12, 21]$ pixels	Maximun distance a control point can be moved per frame.
max_length	14 pixels	Maximum length of a spline element.
min_length	$\{3, 6\}$ pixels	Minimum lenght of a spline element.
min_angle	$\{33°, 90°*\}$	Minimum angle between consecutive linear spline elements. *The value 90 was used for a video with bad illumination that produced fictitious sharp peaks.
$BlurKernelSize$	9	Size of the kernel used to blur the edges.
$CannyRatio$	$\{3, 5\}$	Used to calculate the second threshold for Canny: $upper\_threshold = CannyRatio \times first\_threshold$.
Mass-spring System		
Parameter	Value(s)/Interval	Meaning
m	1	Mass per vertex.
Δt	$[0.01, 0.5]$	Integration step. After some series of experiments it was set to 0.1.
$Min\Delta Area$	2 units	Minimum area before a triangle is considered *flat*.

TABLE 9.2: *Parameters calibrated autonomously.*

	Min value	Max value	Meaning
k_L	0.001	30,000	Length elasticity constant.
k_d	0.001	2.0	Length damping constant.
k_A	0.001	30,000	Area elasticity constant.
k_φ	0.001	1000	Angle elasticity constant.
max_force	5	100,000	Maximum force on each vertex.
f_t	0	3000	Translation force applied evenly on all vertices.
$yield$	0	1.0	Critical value of proportional deformation after which permanent deformation occurs.
$creep$	0	1.0	Proportion of the deformation that becomes permanent.
max_α	0	0.9	Maximum permanent proportional deformation allowed.

equal intervals each generation; if the value is outside the valid range for the variable, another value is calculated until it falls within the appropriate interval. There is no recombination, even though would could have tried combining the values of some variables of one parent with the values of the other variables of the other parent. However, since after analysing our first results we noticed that the maximum fitness value converges to the same average value independently of the initial individuals, and the convergence towards this value happens quite quickly (15 generations), it seems that modifying the search strategy will not yield better results. The function for the fitness value is explained in the following section. Algorithm 5 is used with $N = 15$,

$M = 50$[viii] and $\alpha = \beta = \gamma = 1/3N$.

Algorithm 5 Evolutive Search for Calibration

1: Be N the number of sets of parameters to be evaluated per generation and M the number of generations to try.
2: Let $\alpha + \beta + \gamma = N$ denote three portions of the N elements.
3: **for** $i = 1 \rightarrow M$ **do**
4: Run the N simulations and compare prediction with ground truth (real object) [Figure 9.3(a)], by comparing the area occupied by the real object [Figure 9.3(b)] with the area occupied by the mesh of the simulation [Figure 9.3(c)].
5: Keep the best α of all simulations.
6: Generate β by adding random gaussian noise to the best candidates.
7: Generate γ completely new.
8: $i \leftarrow i + 1$
9: **end for**

Once a suitable set of parameters has been selected, they can be used in the mass-spring model to make predictions of how the shape of the material will change, as the robot applies forces in other points, where it had not tried before. Another application is to collect a group of good sets, by running this algorithm several times, and use them to distinguish between the materials that have been trained. The classification takes place by using a voting system, where the models that make the best predictions vote for the material they were trained on. The following section gives the details of how the performance of the predictions is evaluated.

[viii] Even though, the analysis presented in the following chapters showed that the improvements after 15 generations are small. After these analysis we used $N = 10$ and $M = 25$, but the fitness of the best element did not improve, so we do not include more graphs.

9.6.2 Evaluation Function: Difference of Areas

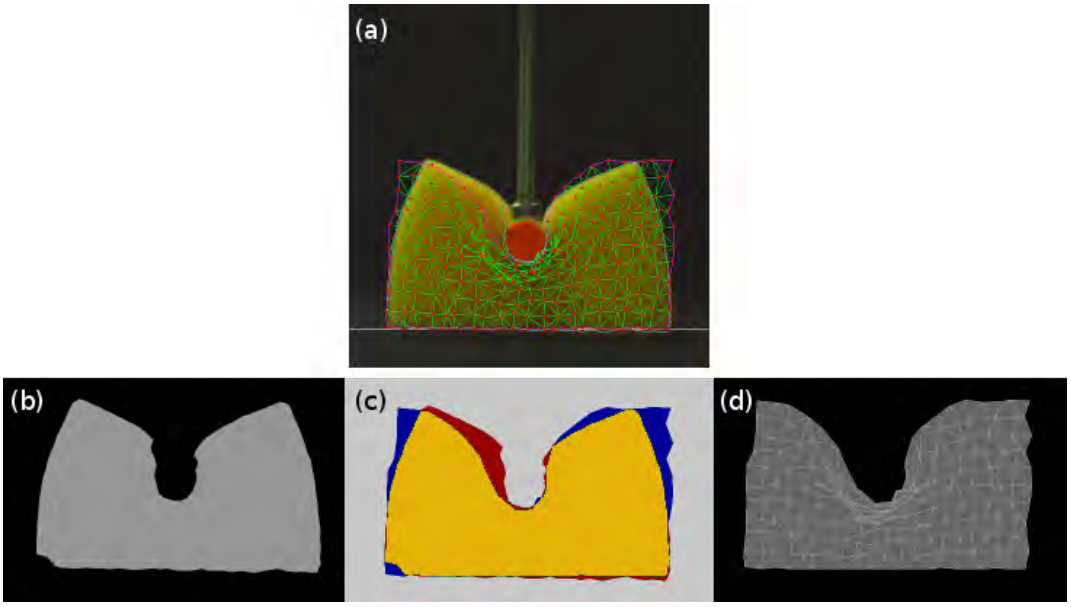

FIGURE 9.3: *(a) Mesh over real sponge. (b) Tracked area occupied by the real sponge, used as ground truth. (c) Difference between ground truth and prediction: [yellow] true positives, [red] false negatives, [blue] false positives, [gray] true negatives. (d) Area occupied by the mesh.*

Given the contour indicated by the linear snake, an OpenCV function is used to determine the pixels inside of it. 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*, where the material is present, and the simulation predicted it would be present; the *true negatives*, where the absence of material is in accordance with the prediction; the *false positives*, where the prediction said there would be material, but there is not; and the *false negatives*, where there is material, but the prediction said there wouldn't [Figure 9.3]. These measures are arranged in what is called a confusion matrix:

		Predicted	
		True	False
Actual	True	TP (true positives)	FN (false negatives)
	False	FP (false positives)	TN (true negatives)

Given that maintaining a record of the true negatives (the empty/background space around the material) is mostly irrelevant and highly inefficient, the mark that was used as fitness value for a whole video was derived from the following quantities:

$$Precision = \frac{TP}{TP + FP}, \tag{9.27}$$

$$Recall = \frac{TP}{TP + FN}, \tag{9.28}$$

$$F = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{1}{1 + \frac{FN+FP}{2TP}} \tag{9.29}$$

where F is the harmonic average of the precision [Figure 9.4 above] and the recall [Figure 9.5 below]. This measurement is used in the computer vision community to evaluate segmentation algorithms. $F = 1$ means that there were no false predictions ($FN + FP = 0$). $F \to 0$[ix] 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 [Figure 9.5 above] and the mark for the video is the mean over all the frames so far, $\mu(F)$ [Figure 9.5 below]. 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 model with the highest final value of $\mu(F)$ is considered the best.

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

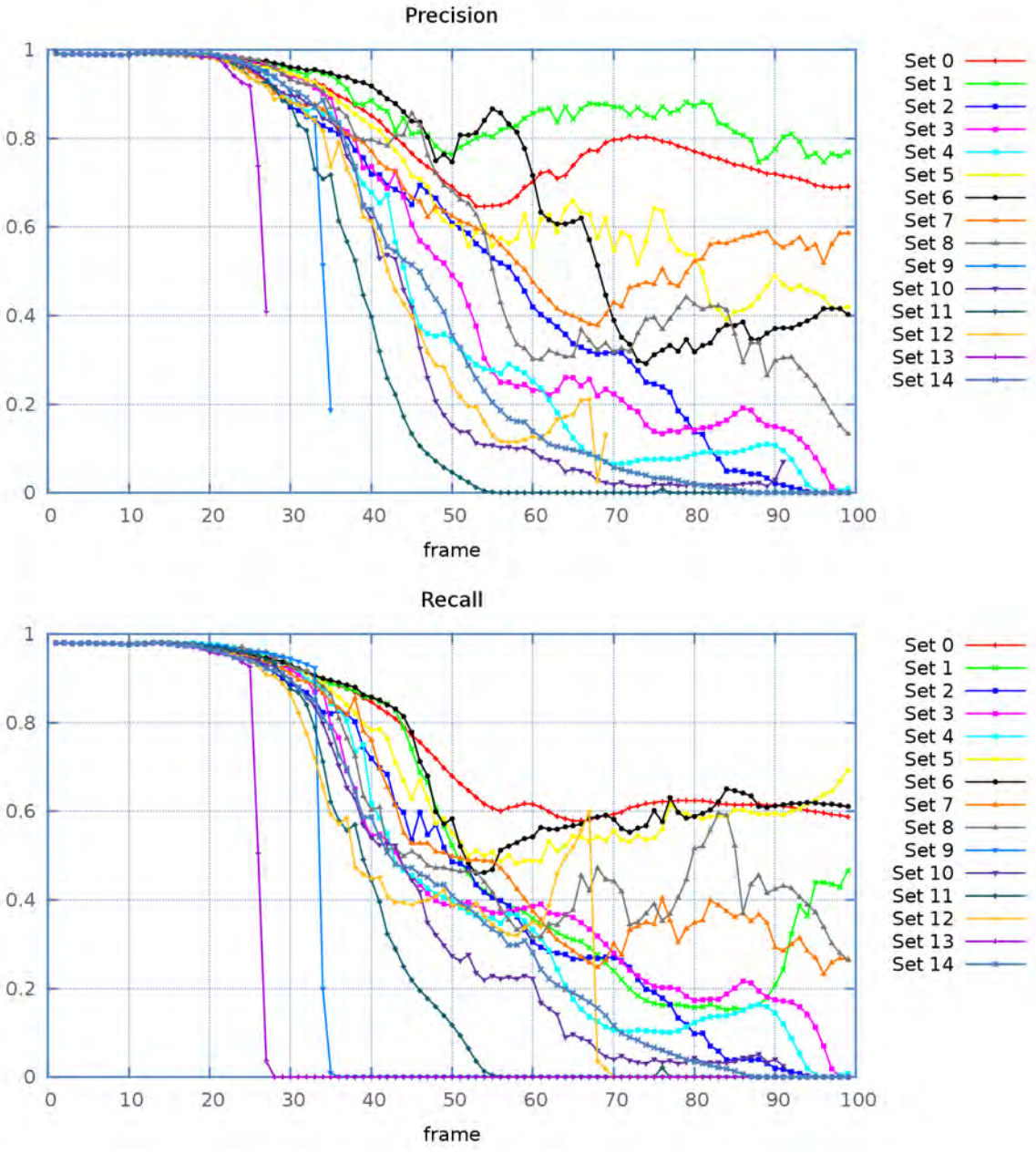

FIGURE 9.4: *Precision and recall for each frame of the video, for each of the 15 sets in a first generation of the learning algorithm. They illustrate the behaviour of the simulations as the video advances. The values are good (close to* 1.0*) at the beginning when the initial mesh and the ground truth match. The worst sets of parameters diverge soon, producing a quick drop.*

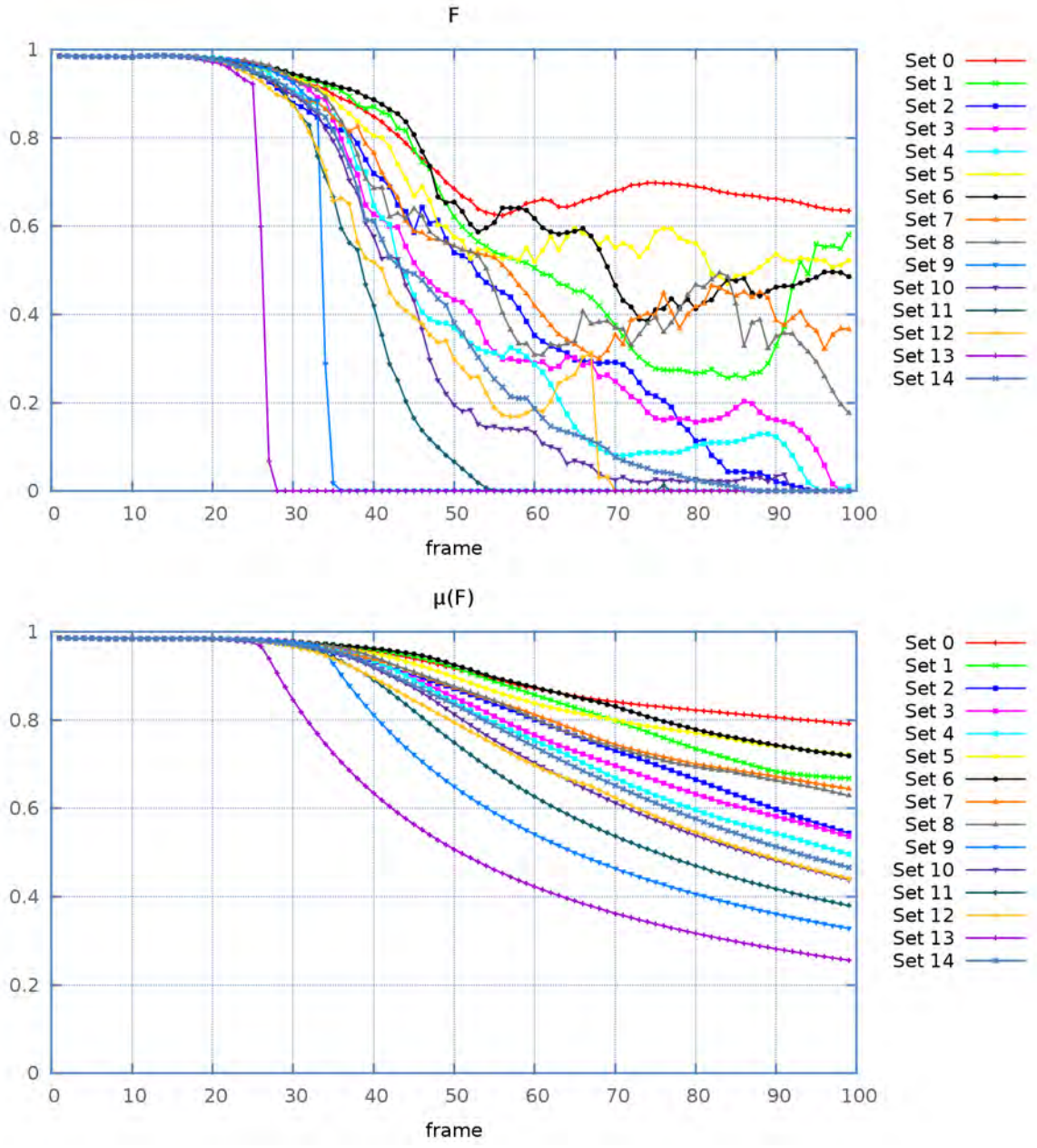

FIGURE 9.5: *F and $\mu(F)$ for each frame of the video, for each of the 15 sets in a first generation of the learning algorithm. A value of 1.0 would correspond to a perfect prediction, while 0.0 is completely erroneous. Since $\mu(F)$ is a cumulative average, it decreases slowly even after simulations fail, but it retains its value if the ground truth and the simulation remain similar.*

Additionally, these values provide useful information about the behaviour of the simulations as the video progresses. For example, the curve of the set 6 illustrates a simulation that fails to deform as much as the real sponge, but catches up after the sponge recovers its shape [Figure 9.5 below].

If the geometric constraints are used, the simulation stops sometimes, when the model has been unable to comply with these constraints beyond a threshold value. From that frame on, the value of F is zero. This happened to the sets 13 and 9 in the graphs, whose precision became undefined, the recall and F dropped to zero, while the value of $\mu(F)$ decreased monotonously.

9.7 Prediction of the Force

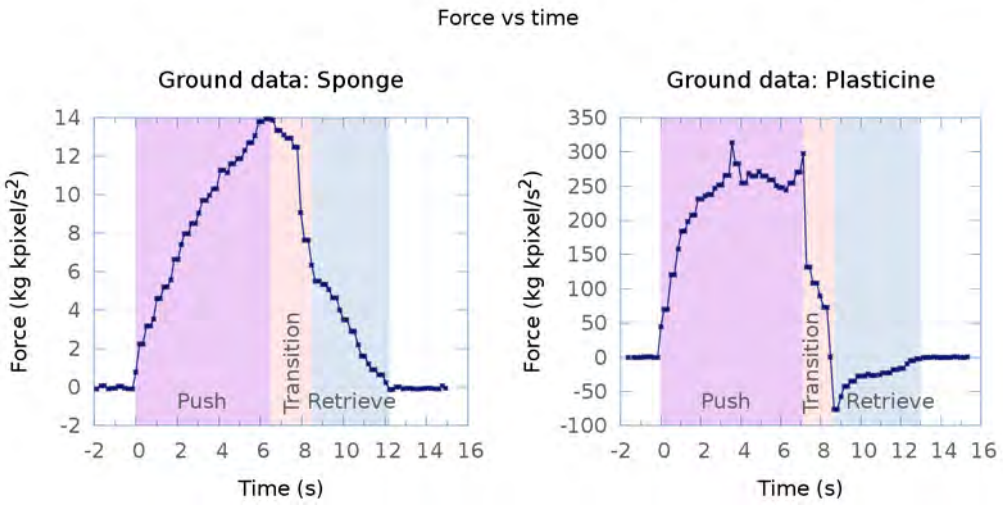

FIGURE 9.6: *The response force during the interaction can be characterised in three phases: pushing, transition and retrieval of the finger. A different regression curve can be used to approximate the force during each phase.*

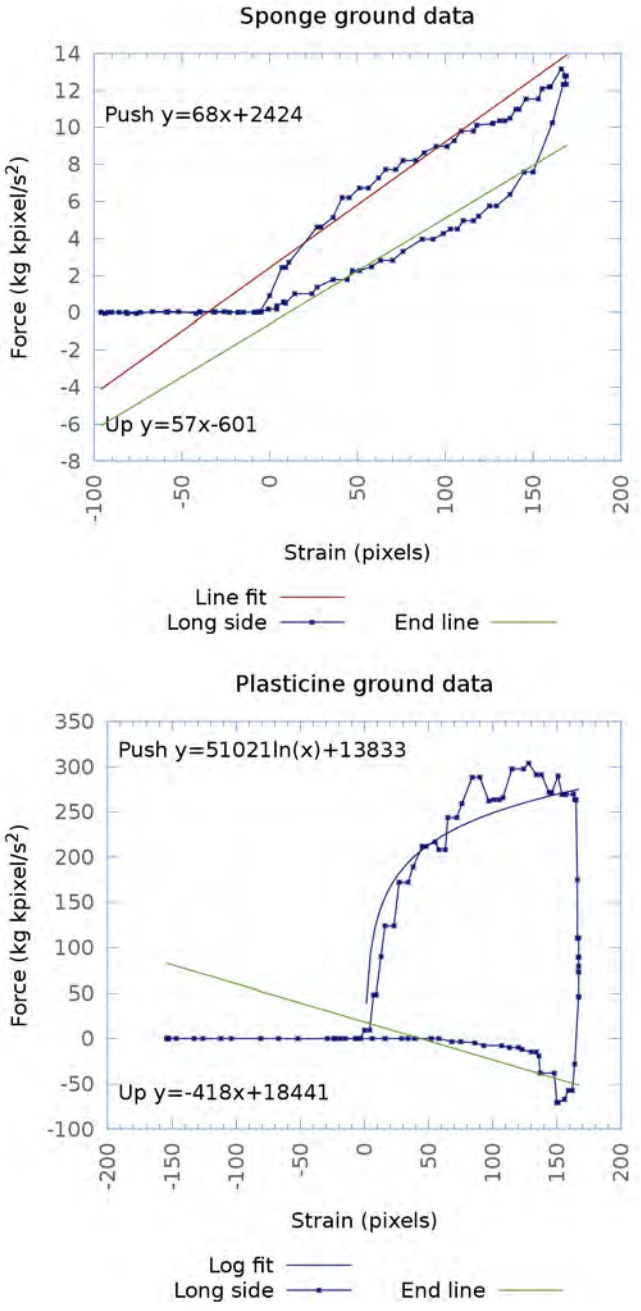

FIGURE 9.7: *A different regression curve can be used to approximate the force during the pushing and retrieving sub-actions. The sponge is modelled with lines, while the plasticine is better modelled by a logarithmic curve. The retrieval of the finger was grossly approximated by a line.*

By making use of a stress-strain diagram, that can be obtained from the collaboration of the visual and force information during the training phase, it is possible to make simple predictions about the reaction force that the robot will perceive in future interactions. To illustrate the idea, Figure 9.6 shows the graphs of the response force vs time for the training set of the sponge and the plasticine. Each sub-action, (push, transition and retrieve) of the whole interaction, can be modelled by a curve. To obtain an approximation that would be independent of the exact movement of the arm, the curves are not adjusted to the force vs. time graph obtained with the force sensor alone, but rather to the stress-strain curve obtained by measuring the displacement of the finger between readings of the force sensor [Figure 9.7]. This approximation will be valid for homogeneous materials.

During the training phase the following events take place:

1. Reaching. The robot pushes the finger downwards. At this stage no force is detected by the force sensor and no collision is detected by the vision system.
2. Pushing. The finger touches the material. The first collision is detected. From this point on, the system can generate a plot of the distance that the finger travels vs. the measured force. Notice that this distance corresponds to the amount of strain as well. A regression curve is fitted for this part of the action.
3. Transition. The robot stops pushing and begins to retrieve the finger. The stress-strain curve almost has a discontinuity, the response forces change from the maximum value to less than half of it in less than 2 seconds. The points corresponding to this phase can be ignored and be modelled as a discontinuity, or a straight line that joins the last point of the previous phase and the first point of the following phase could be used instead.

4. Retrieving. The material may respond if it tries to recover its shape or other collisions take place. A second regression curve is adjusted to this segment of the curve, as described below [End lines in Figure 9.7].
5. Moving off. The finger goes back to the initial position. No more data is generated.

To adjust the regression curves the least squares technique is used. Since there is only one independent variable the equations for adjusting a line are quite straightforward:

$$y = mx + y_0 + Error \tag{9.30}$$

$$m = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \tag{9.31}$$

$$y_0 = \frac{\sum y_i}{n} - y_0 \frac{\sum x_i}{n} \tag{9.32}$$

$$Error = \sum (y_i - (mx_i + y0))^2 \tag{9.33}$$

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 a logarithmic curve a change of variable is used. Before computing the line, the natural logarithm of the independent variable is calculated, that is: $x_i' = \ln(x_i)$. 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 moment.

During simulation time the regression curves can be used to predict the reaction force if the system provides the amount of deformation, which is obtained from the position of the finger starting from the first collision. By introducing this technique, the system will only need to know the model for the material, the position of the finger, the obstacles in the environment and the initial position/shape of the material, to make

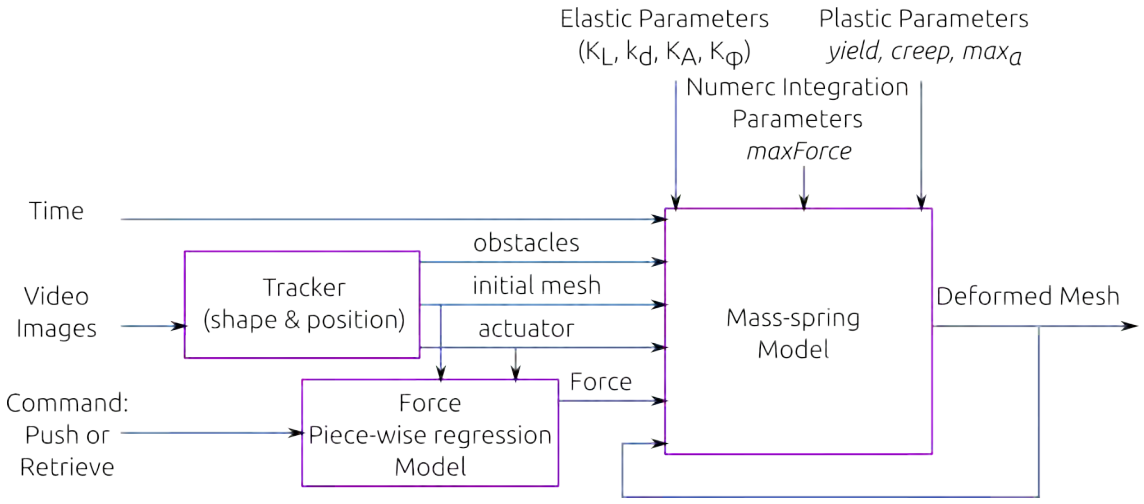

FIGURE 9.8: *When the predictor for the force is added, the system no longer requires the force sensor as it did in Figure 8.8. Instead, it is enough to indicate whether the robot is pushing down or retrieving the finger. The value of the strain is extracted from the position of the finger and the initial shape of the target object by using the curves adjusted to the stress-strain diagram obtained during training.*

predictions about both: the shape and the force that the robot will detect for the rest of the action [Figure 9.8].

9.8 Classification

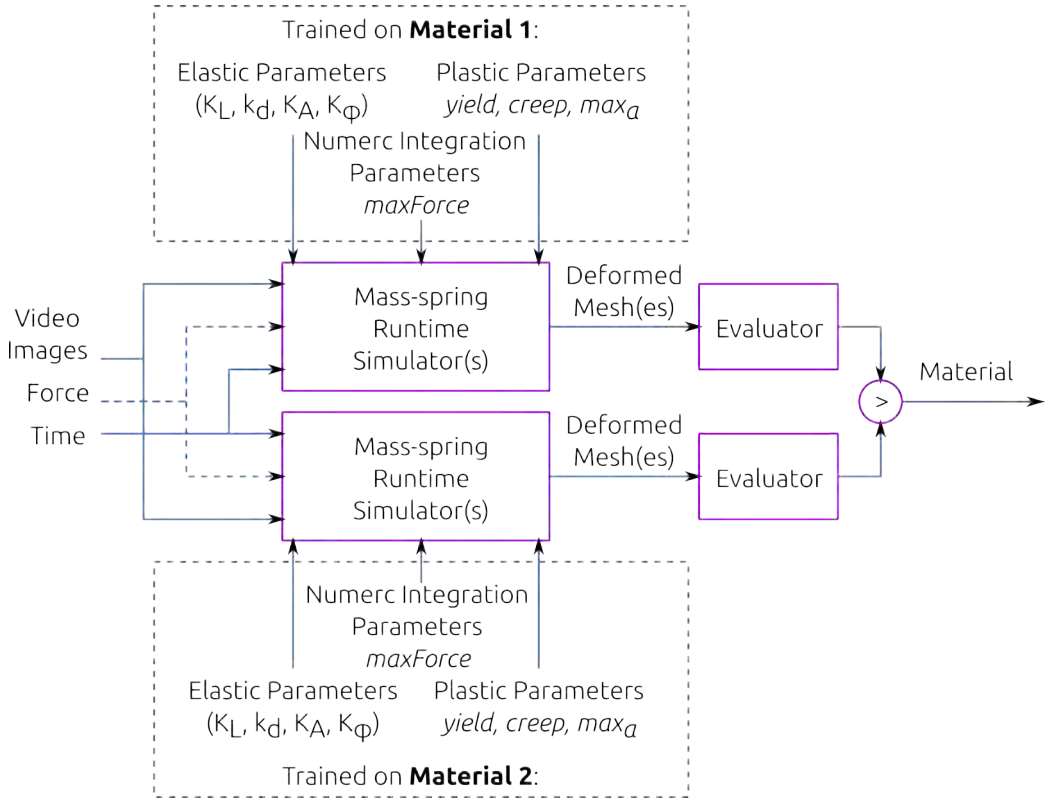

FIGURE 9.9: *Simulations are run using visual and haptic information coming from an unknown material. The simulations that produce the best predictions for the behaviour propose the material they were trained on as the class of the unknown material.*

The calibrated models were applied to the problem of identifying the material the robot is interacting with. This classification problem was tested with the trained materials: the sponge and the plasticine, but on interactions different from those that were used for the training [Figure 9.9]. The importance of this test lies in the fact that it consists in recognising previously seen materials, by comparing the quality of the its expected behaviour, even if the interaction is different (in this case, the object is pushed in a different location). To accomplish this, the algorithm does not use explicit information about

the behaviour of each class. If we had used knowledge about the typical behaviour of elastic and plastic materials, it would only be capable of distinguishing those two classes. Instead, the mechanism should be useful, in principle, to identify any previously seen material at any point within the spectrum of elastic and plastic materials, since the underlying physics based model is the same for all of them. However, further improvements would be required before we achieve such level of precision, like making 3D models or using more advanced mass-spring models (e.g. [8]).

There are two algorithms for classification. One classifies the material on a frame by frame basis, by comparing the F values obtained by each predictor. The material, whose predictor obtains the highest F value, is returned as the proposed class for the unknown material. An overall evaluation can be done on the basis of in how many frames the classification was correct. A second algorithm gives an overall classification after the whole interaction is over, and it takes into account that the material didn't change throughout the interaction. It compares the cumulative difference of the F marks of each model's predictions [See Algorithm 6].

The classification was attempted in the training set and both test data sets. To give greater statistical significance to the results of the learning method, rather than using only one model for each material, an ensemble of the 15 best models obtained after a number of executions of the learning routine were used. Each model within the ensemble made a prediction and the mean of the F mark, for each frame, obtained by all the predictions was used in the classification algorithm [Algorithm 6].

Algorithm 6 Classification

Let TP be the true positives and FN the false negatives.
$TP \leftarrow 0$
$FN \leftarrow 0$
for each frame **do**
  $vote = F(correctmaterial) - F(incorrectmaterial)$
  if $vote > 0$ **then**
    $TP \leftarrow TP + vote$
  else
    $FN \leftarrow FN - vote$
  end if
end for
if $TP > FN$ **then**
  The material is classified correctly
else
  The classification is incorrect
end if

9.9 Summary

This chapter presented the main extensions and implementation details of the mass-spring like model that the robot uses to model the behaviour of the deformable objects it pushes. It also explains how good sets of parameters can be searched for and how to evaluate them. Those same sets were used to solve a classification problem after training.

Chapter 10

Experiments on Prediction

Chapter 8 presented the scenario where the hypothesis of this work has been tested. Chapter 9 introduced the theoretical contributions and this chapter shows the results and analysis of our tests.

10.1 Training of the Mass-spring Model

As was explained in Section 9.6, there are many possible variations for the configuration of the simulator. For a first round of experiments, only the visual information was used, while the force sensor was neglected, since most robotic systems we are familiar with do not rely on force feedback. The possible variations to be considered were grouped as follows:

L Only damped springs on the edges are considered (preservation of the length). Values for K_L and k_d are searched for.

AA Only area and angle preservation are considered. Values for K_A and K_φ are - searched for.

ALL Considers all the previous ones: damped length, area and angle preservation.

Only Elastic The deformation of the springs over the edges of the mesh is only temporal.

Plastic The rest length of the springs can be permanently deformed. $Yield$, $creep$ and the limit for the maximum deformation max_α are searched for, in addition to the other parameters. This does not have an effect on area or angle preservation.

Three integration schemes were considered:

- Euler
- Verlet
- Our customised style [See Section 9.5].

It was also possible to make use of the extra geometric constraints (denoted below by 1 or +) or not to use them (denoted by 0 or -). These include: flipping back distorted triangles and recovering area for degenerated triangles. For this first series of experiments, collision detection was active for all variations, since it allows the mesh to respond to the shape of the finger and avoids crossing the floor. The different combinations of these configurations that were tried can be seen in Table 10.1.

A single run of the evolutionary algorithm for the calibration of the model was executed for each variation, on the data gathered from the sponge and the plasticine. A mesh of $20 \times 10 \times 2$ triangles [Figure 10.1] was used. We estimated that a mesh of this

TABLE 10.1: *Different ways to configure the mass-spring simulator*

Energy terms ⇒ Integration Scheme ⇓	Length Preservation + Damping (L)		Area + Angle (AA) Preservation		All Previous (ALL)	
EULER	1	0	1	0	1	0
CUSTOMISED	1	0	1	0	1	0
VERLET	1	0	1	0	1	0

1: with Graphic Constraints, 0: without Graphic Constraints.

resolution would be good enough to approximate the shape of the deformed materials, by comparing the size of the triangles with the size of the finger that produces the deformations. However, as will be seen later, the numerical methods had some minor problems working with such tiny triangles, that prevented them from being useful for classification. These problems could be fixed by reviewing the parameters of the implementation that were not explored with the evolutionary algorithm[i] or by creating geometric recovery routines that affect more neighbours, rather than only the immediate ones, because the displacements take place so quickly that more than one overlapping takes place. However these changes would have a negative impact on the run time performance of the system.

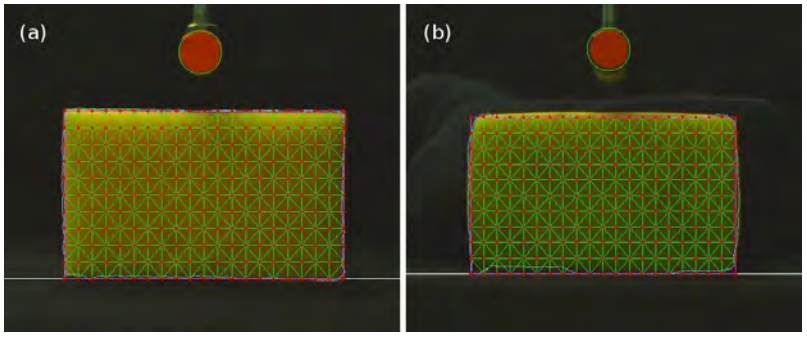

FIGURE 10.1: *Initial mesh for the (a) sponge and (b) plasticine.*

[i]Some changes to consider would be the integration time step, since after it was selected it was not modified again; or the distance the elements of the triangles are displaced by the geometric recovery routines Section 9.4

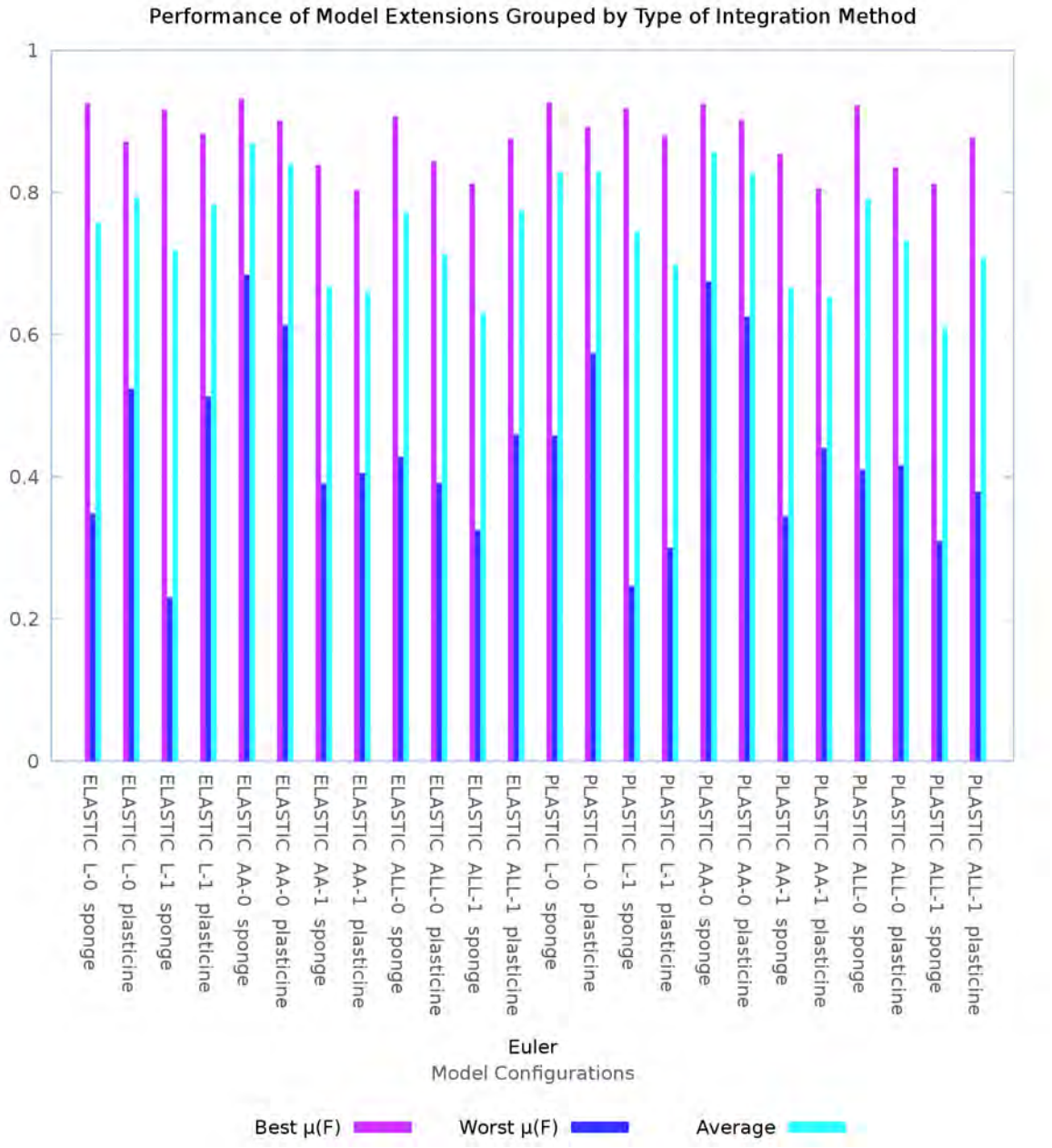

FIGURE 10.2: *Best, worst and average mark for each variation of the mass-spring model as explained in Table 10.1, when trained for the sponge and plasticine (putty), with the* ***Euler*** *integration scheme.*

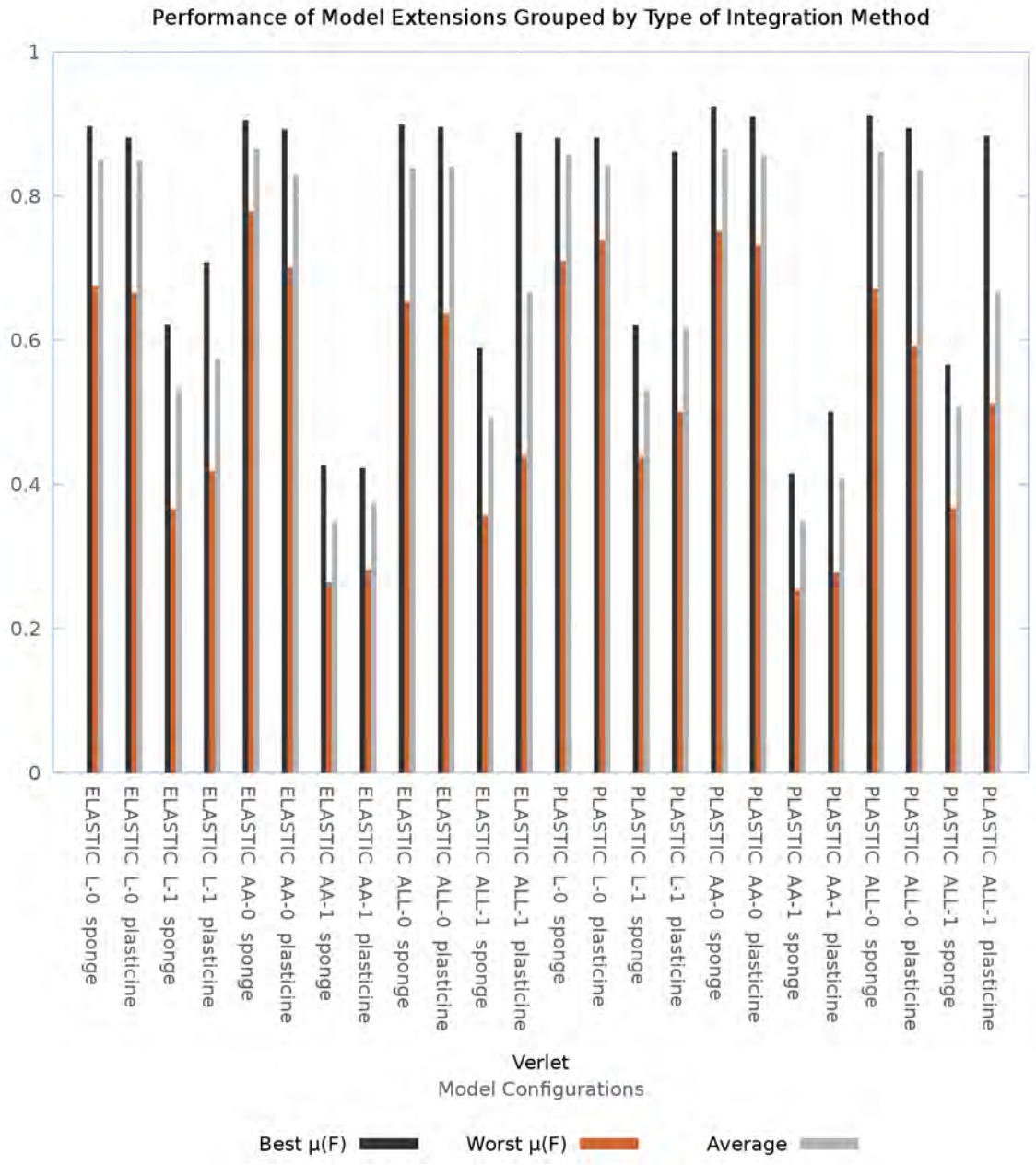

FIGURE 10.3: *Best, worst and average mark for each variation of the mass-spring model as explained in Table 10.1, when trained for the sponge and plasticine (putty), with the* ***Verlet*** *integration scheme. All fitness values are consistently lower when geometric recovery routines, like flipping back inverted triangles, are used.*

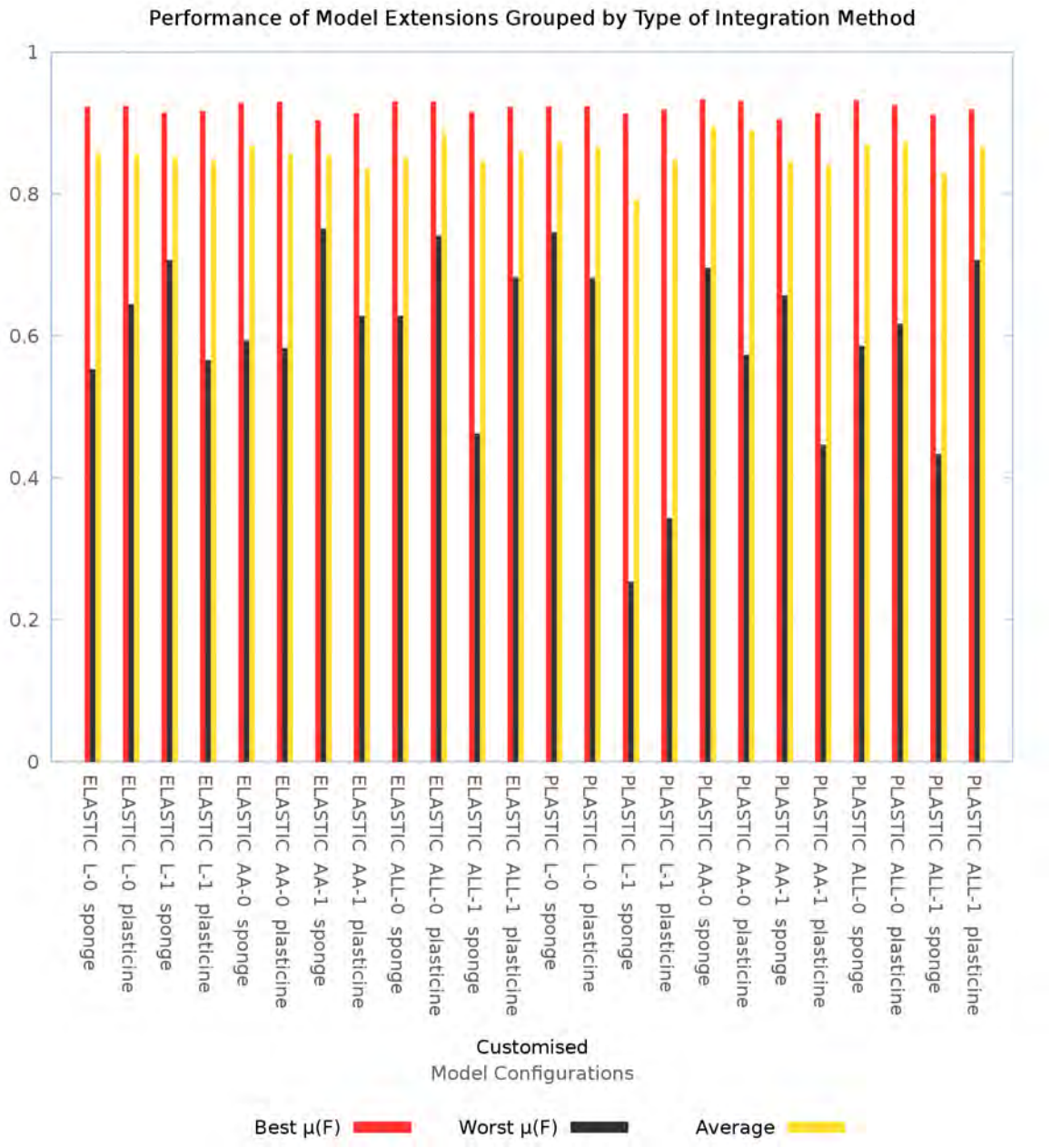

FIGURE 10.4: *Best, worst and average mark for each variation of the mass-spring model as explained in Table 10.1, when trained for the sponge and plasticine (putty), with our* ***customised*** *equations.*

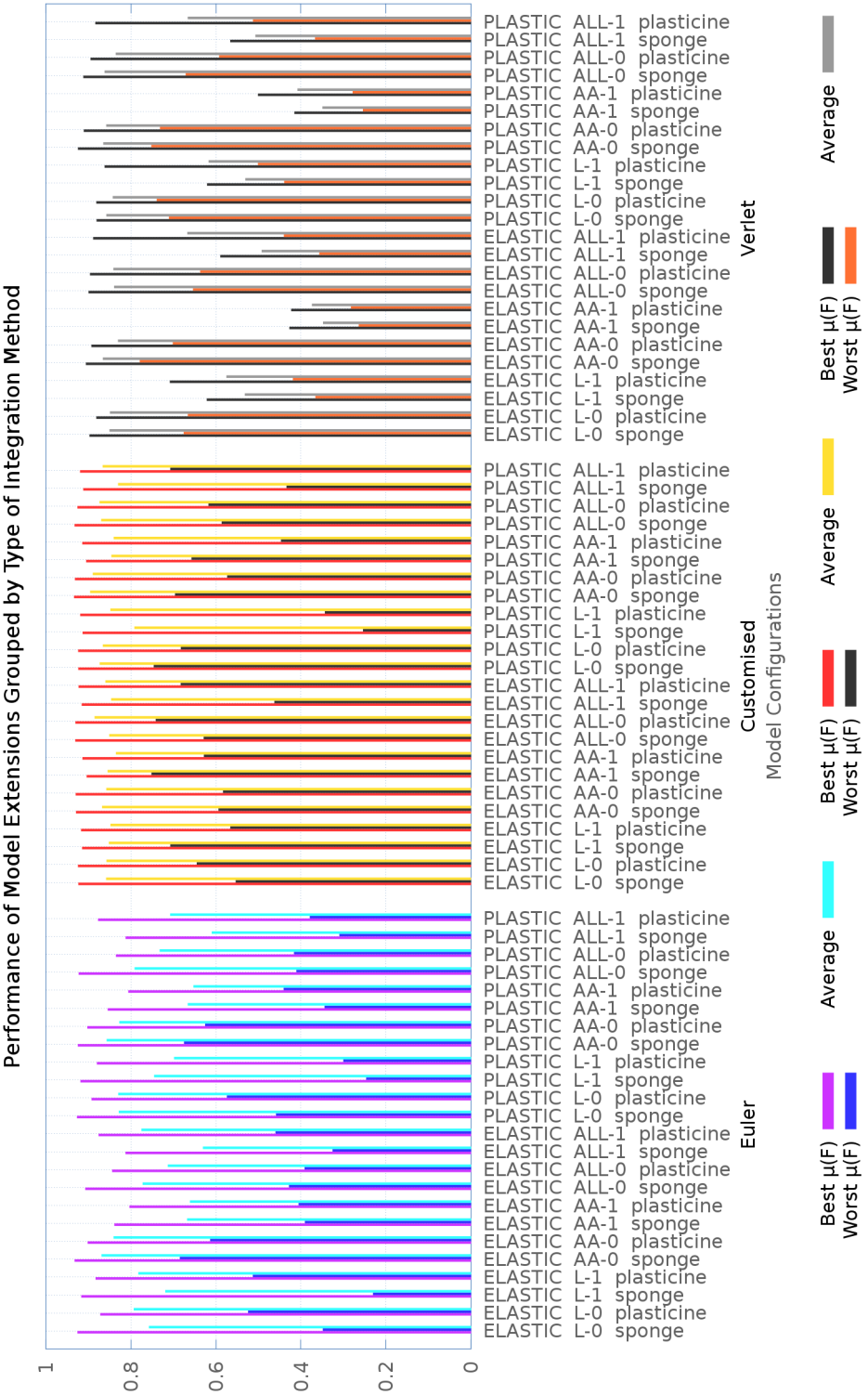

FIGURE 10.5: *Side by side presentation of the previous three graphs: best, worst and average mark for each variation of the mass-spring model, when trained for the sponge and plasticine (putty). The best marks for our customised scheme are consistently better for all the tested combinations.*

The graphs in Figures 10.2, 10.3 and 10.4 illustrate the best, worst and average marks obtained after running the 50 generations of the learning algorithm, with the three integration methods: Euler, Verlet and our customised method.

According to Figure 10.3, Verlet tended to lose stability, especially when the geometric constraints were activated. This happened because Verlet is better at calculating the effect of the acceleration than Euler; when a vertex is displaced to satisfy a geometric constraint, its velocity and acceleration for the next integration step are altered, which makes it seem as if a bigger force was acting on them, this effect is transmitted to subsequent steps, producing bigger oscillations of the masses and affecting the stability of the simulatio Euler was not affected so badly Figure 10.2.

The results in Figure 10.5, where all the marks are placed side by side, show that the customised integration scheme got better marks, consistently across all tested variations illustrated in table Table 10.1. It was useful as well to notice that activating the use of the parameters for plastic deformation did not have a negative impact on the mark of the best set of parameters found after the fifty generations. Following the results of these graphs, the simulations obtained with all the elasticity and plasticity parameters, and the customised integration scheme were analysed in more detail.

10.2 Performance of the Training Phase of the Genetic Algorithm

To test the performance of the evolutionary algorithm, the learning routine was repeated 15 times, for each selected variation of the configuration and for each material. Figure 10.6 and Figure 10.7 show the average of the best, worst and average mark for

the sets of parameters tested during each generation. The shaded areas correspond to the standard deviation from the average of $\mu(F)$.

These graphs show that, even though the geometric constraints were present to maintain a planar mesh, they do cause problems with the overall behaviour of the simulation and actually reduce the similarity between the ground truth and the area covered by the mesh of the simulation, according to the criterion of evaluation. When the triangles are small the effect of the geometric constraints is difficult to visualise, but it is still present. Figure 10.8 and Figure 10.9 illustrate this effect on the simulation of the sponge. To make it more evident, the training was repeated with meshes of $10 \times 5 \times 2$ triangles; again, the models without the use of geometric constraints consistently obtained better fitness values. For more details see Appendix B.1.

10.3 Performance of the Predictions on the Test Data

The best models of the 15 runs of the learning algorithm, which were trained on videos where the robot pushes the material in the centre of its longest side, were tested against the data gathered when the robotic finger presses the materials closer to the corner on the longer side, and on the centre of its shortest side. The statistics are presented in Figure 10.10.

Following, a series of snapshots illustrate the behaviour of the mesh produced by the set of parameters that obtained the highest mark among all 15 runs of the learning algorithm. These were the results obtained with a mesh of $20 \times 10 \times 2$ triangles.

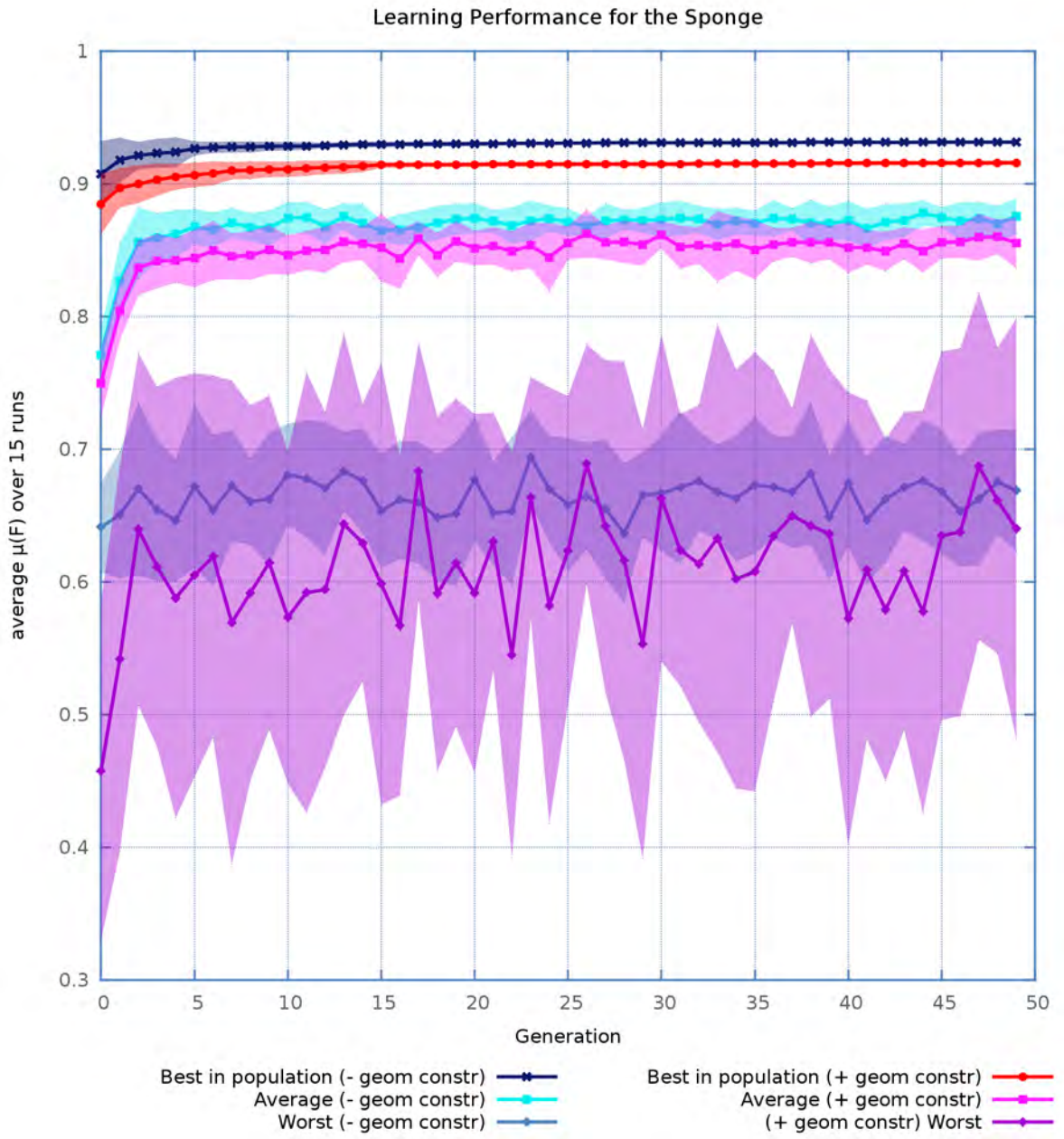

FIGURE 10.6: *Average across 15 runs of the learning algorithm of the best, worst and average mark for each generation of the learning algorithm. Using the customised integration scheme, with (gC=1) and without (gC=0) graphic constraints, when trained with the sponge. The shaded areas correspond to the standard deviation of the average of $\mu(F)$. The mesh had $20 \times 10 \times 2$ triangles.*

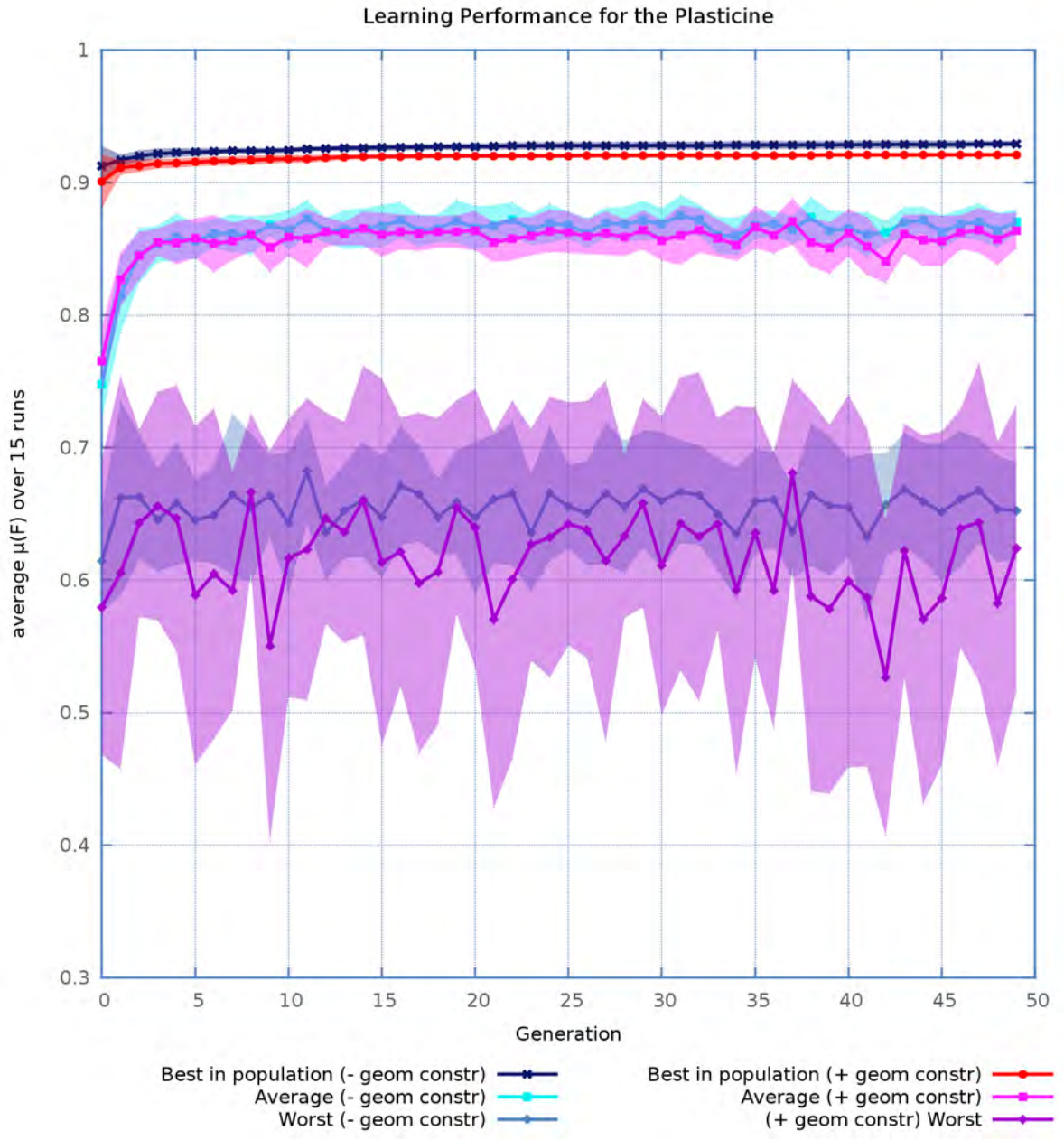

FIGURE 10.7: *Same as Figure 10.6 but trained with the plasticine. Notice that the values of $\mu(F)$ tend to be high because the match between the ground truth and the mesh of the simulation is good at least at the beginning of the video.*

Simulation of the Sponge with and without Geometric Constraints

Mesh: $20 \times 10 \times 2$ triangles

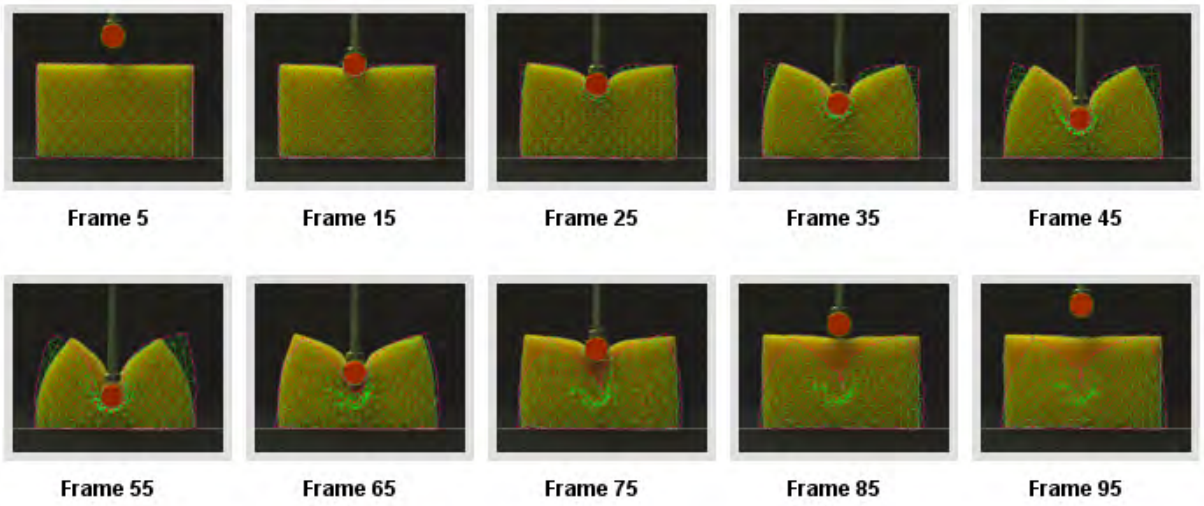

FIGURE 10.8: *Without geometric constraints.* $KL = 7296.91$, $kd = 0.0467218$, $KA = 21981.3$, $KFI = 89.4586$, $yield = 0.896704$, $creep = 0.440973$, $maxAlpha = 0.146345$, $maxForce = 3531.32$, $mark = 0.931993$.

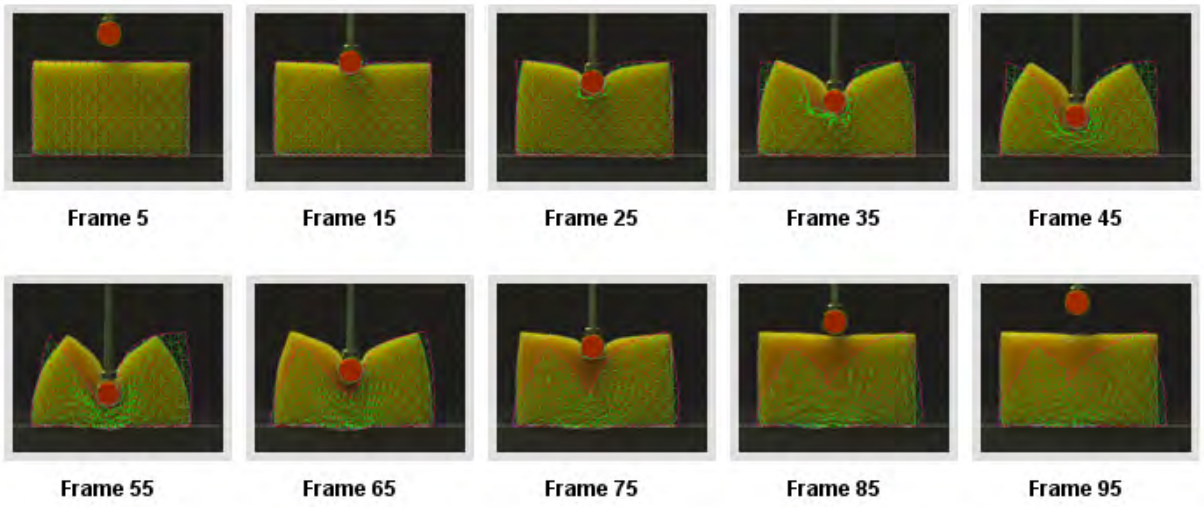

FIGURE 10.9: *With geometric constraints.* $KL = 16877.5$, $kd = 1.29051$, $KA = 28702.2$, $KFI = 331.45$, $yield = 0.610312$, $creep = 0.79242$, $maxAlpha = 0.815037$, $maxForce = 959.348$, $mark = 0.905101$

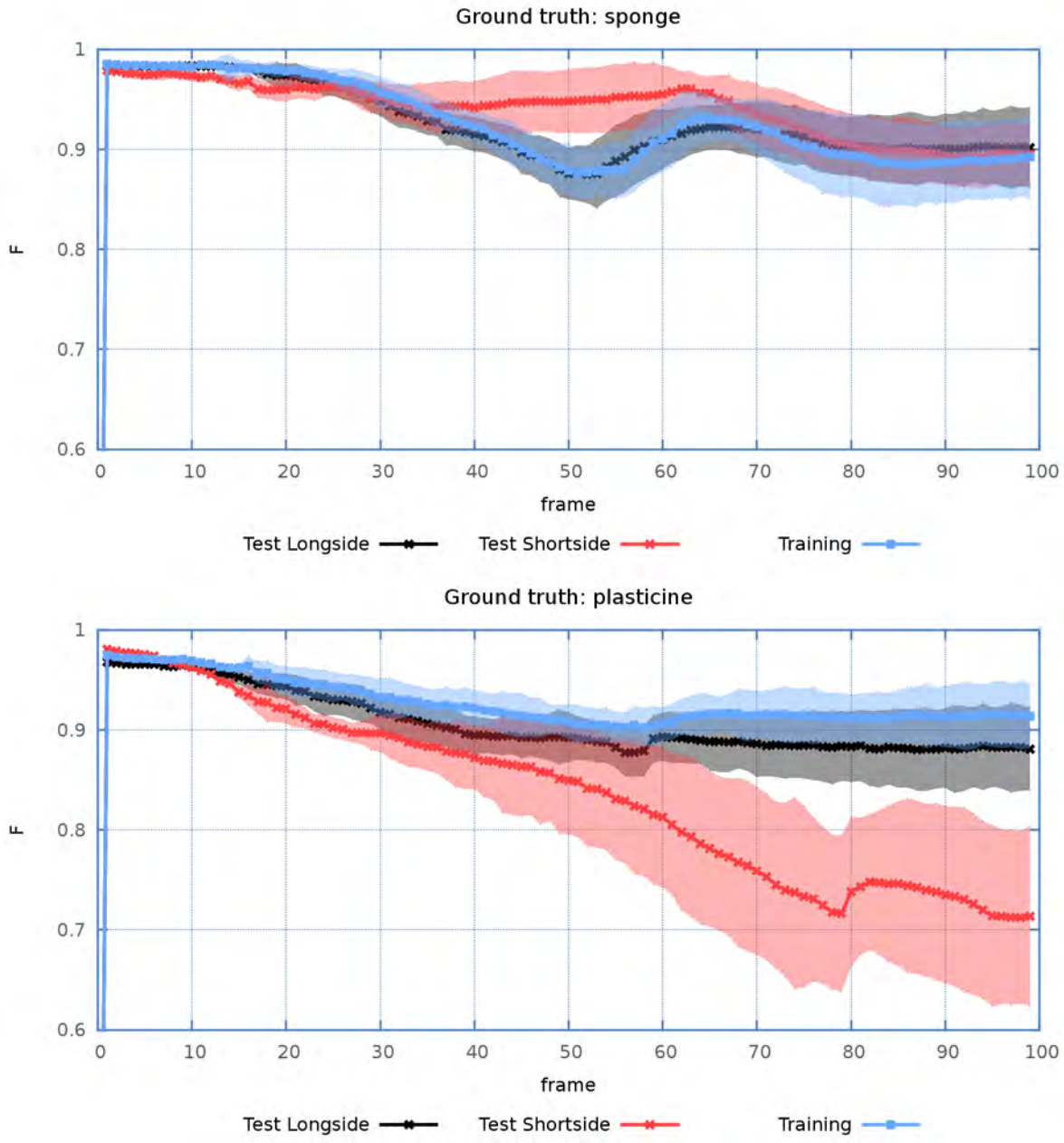

FIGURE 10.10: *Average of the marks for the best sets of parameters, from 15 runs of the learning algorithm, over the training and test data sets, when all types of springs are considered and without geometric constraints(gC=0). The value of F is plotted for each frame. This value indicates how different was the shape of the prediction with respect to the ground truth. The shaded areas correspond to the standard deviation from the average of F across the 15 sets.*

Simulation of the sponge with the best set of parameters:

$KL = 7296.91$, $kd = 0.0467218$, $KA = 21981.3$, $KFI = 89.4586$, $yield = 0.896704$, $creep = 0.440973$, $maxAlpha = 0.146345$, $maxForce = 3531.32$, $mark = 0.931993$.

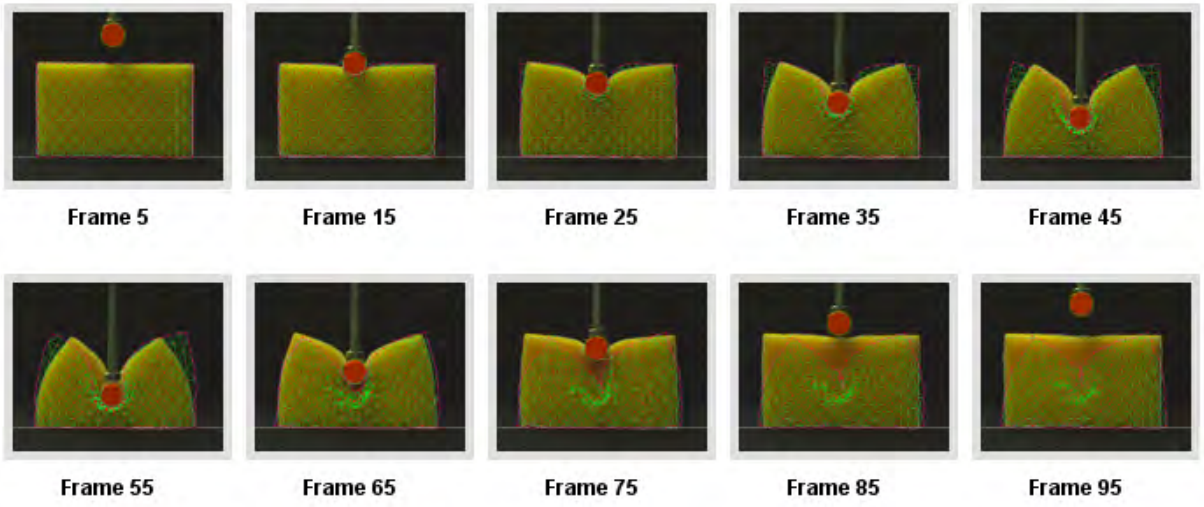

FIGURE 10.11: *Training data.*

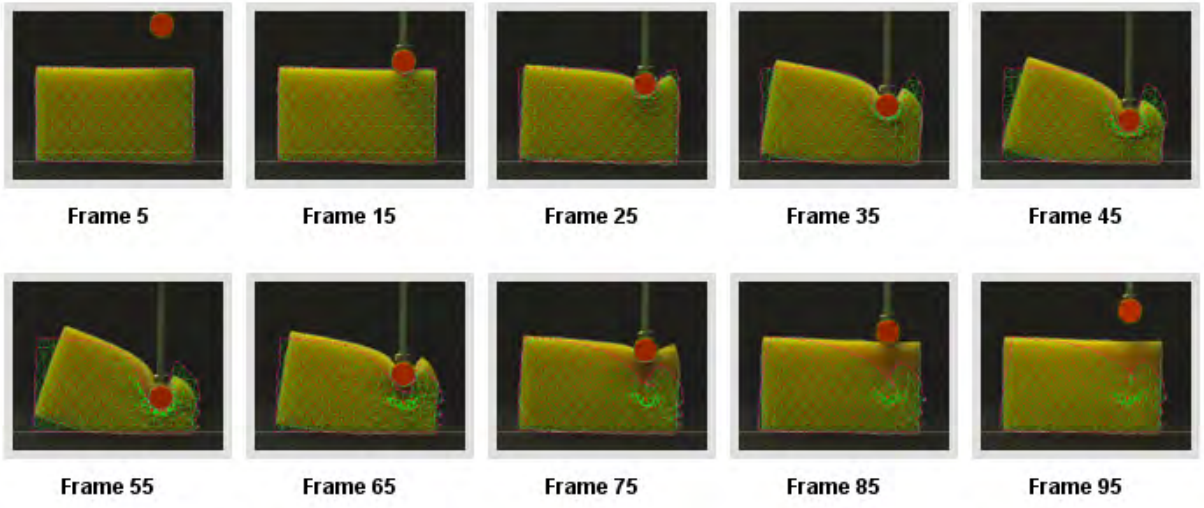

FIGURE 10.12: *Test data.*

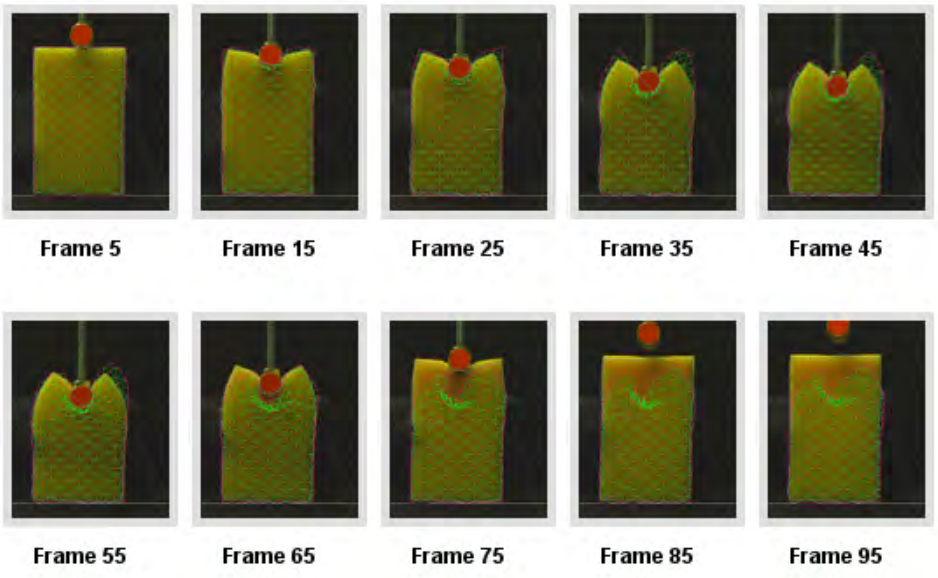

FIGURE 10.13: *Second test data.*

The images in Figure 10.11 show the behaviour of the simulation of the sponge on the training data. Figure 10.12 shows how it worked on the test data. The same is shown for the plasticine in Figure 10.14 and Figure 10.15. It can be seen that the use of the mesh and the physics based model allows naturally for the predictions to work with similar performance under similar conditions.

However the predictions are not equally good when the stresses are greater and the triangles of the mesh overlap, like when the material is deformed along its longest edge [Figure 10.13 for the sponge and Figure 10.16 for the plasticine]. In this last case it can also be seen that the permanent contraction of the mesh' edges produced an unwanted reduction of the volume of the mesh, that prevented it from expanding to the sides, as the ground truth did. What this shows is that our simple model of plastic deformation, that allowed for permanent contraction of the springs, did not allow for

a proper conservation of the volume (which can symbolise conservation of the density of matter).

In both cases, with the sponge and the plasticine, the springs close to the finger were compressed quickly and failed to propagate the deformation as far as the edges of the mesh, quickly enough as to replicate the flux of matter of the real object. Instead, the triangles overlapped, provoking a slower recovery after the finger was retrieved, because the springs were entangled. The movement of the real objects finished before the simulation could catch up. Since the model is a 2D mesh, the part of the material that was displaced in the direction perpendicular to the camera, should have been replaced by a more pronounced deformation in the borders but, as it was said, the effect did not travel quickly enough.

Therefore, at this point it is important to remember that a group of millions of atoms that interact with each other at the speed of light and with all their neighbours at the same time, are being modelled with a mesh of a few masses that interact only with the neighbours that are connected with them with a spring. This means that, at each integration step, only the closer neighbours feel the effect of the displacement of a mass, and only then can they propagate it to the next. This discreet and layered behaviour has proved not to be quick and smooth enough to replicate the almost instantaneous response of the real atoms in the object to be modelled. Another consequence of the same effect is that it was easier to calibrate models with fewer nodes (as few as 9), than models with many nodes; because the number of layers between the finger and the border were less. Unfortunately few nodes mean very little resolution. Morris [55] and Frank [26] avoid measuring this because they do not modify the applied force, or compare with the ground truth, until the whole mesh stops moving under the influence

Simulation of the plasticine with the best set of parameters:

$KL = 1637.18$, $kd = 0.320376$, $KA = 26875$, $KFI = 115.099$, $yield = 0.13826$, $creep = 0.13826$, $maxAlpha = 0.364691$, $maxForce = 5597.45$, $mark = 0.931282$

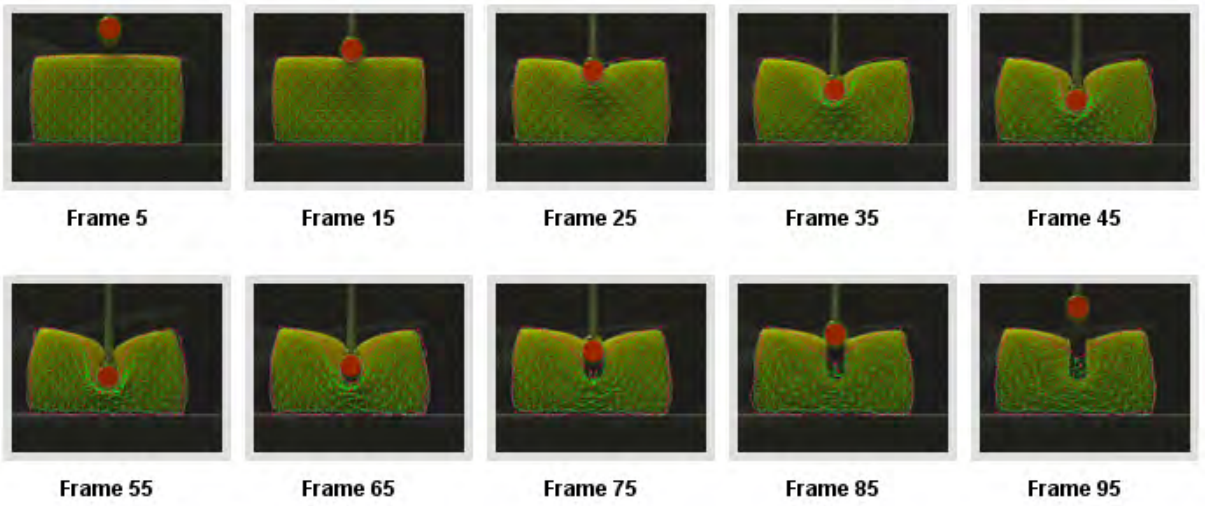

FIGURE 10.14: *Training data.*

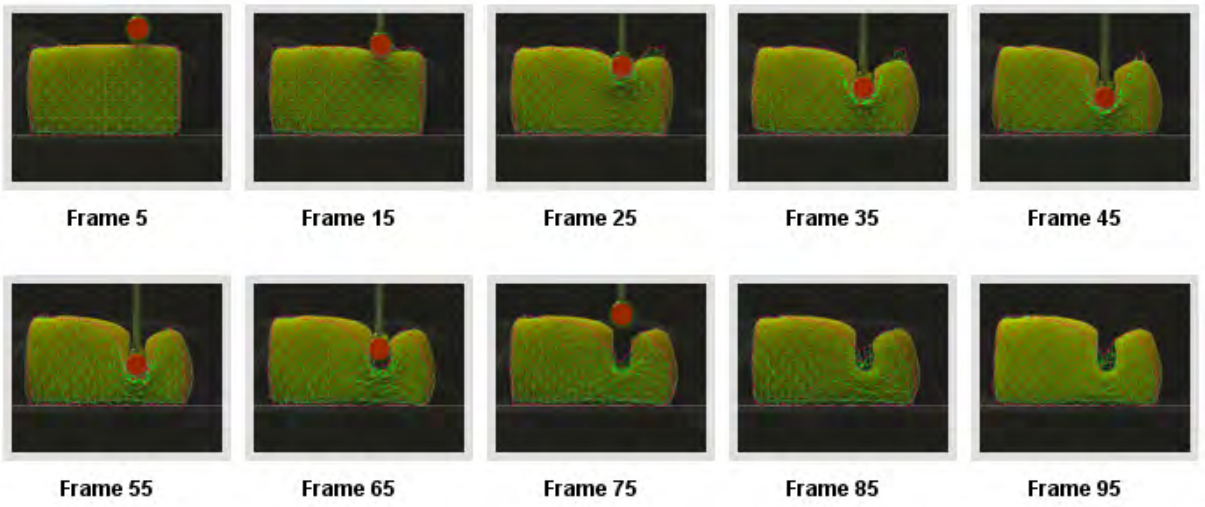

FIGURE 10.15: *Test data.*

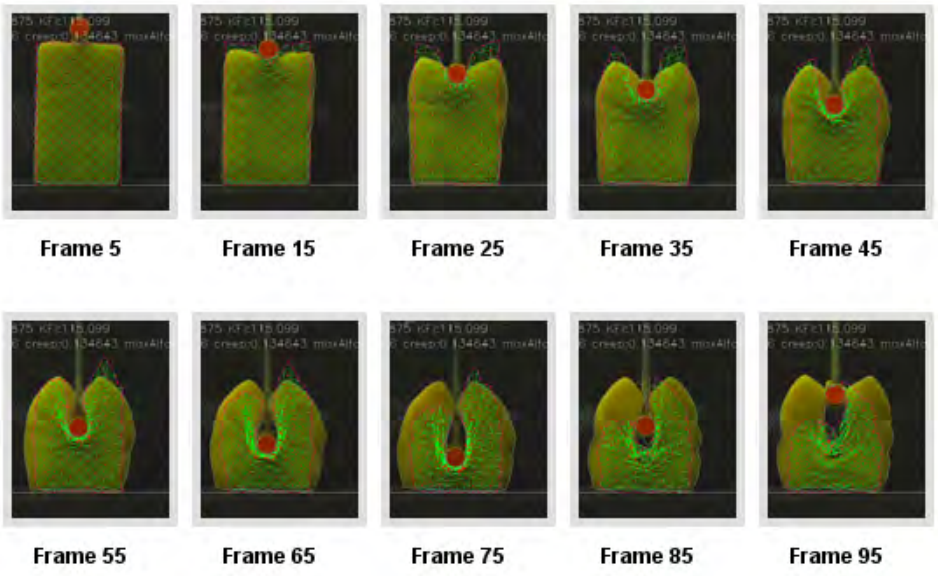

FIGURE 10.16: *Second test data.*

of the forces, that is, until it reaches a state of equilibrium. However, the material under the influence of a moving finger is not in equilibrium.

10.4 Adding Prediction of the Force

A physical property of homogeneous materials is that, if a stress-strain curve or surface is obtained, it is possible to use it to interpolate the stress for unknown values of the strain. In the strict sense, these measurements should be taken in a controlled environment, as was explained in Chapter 4. However, a general purpose robot will not have the opportunity to take such rigorous measurements. For this reason, the experiments presented here explore the effect of using a single set of measurements to train a predictor for the force, the very same set of measurements that was used to train the mass-spring model. It is important to make it explicit that this approach can

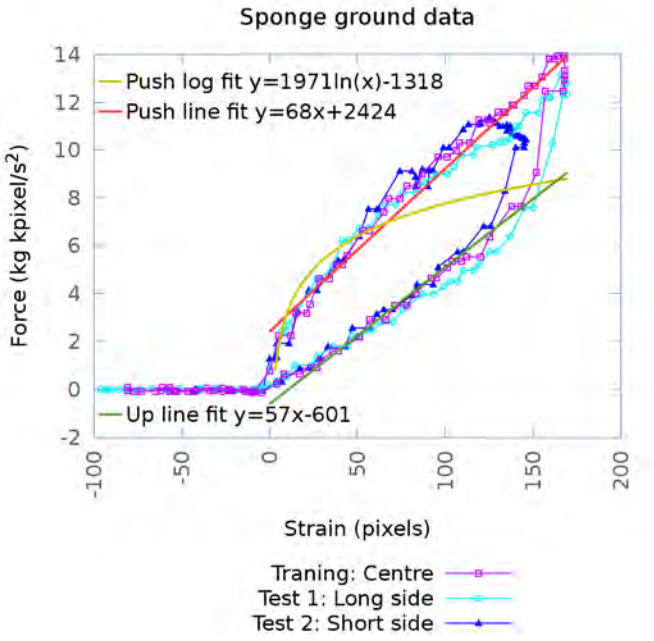

FIGURE 10.17: *Regression curves to model the force (sponge) [See text].*

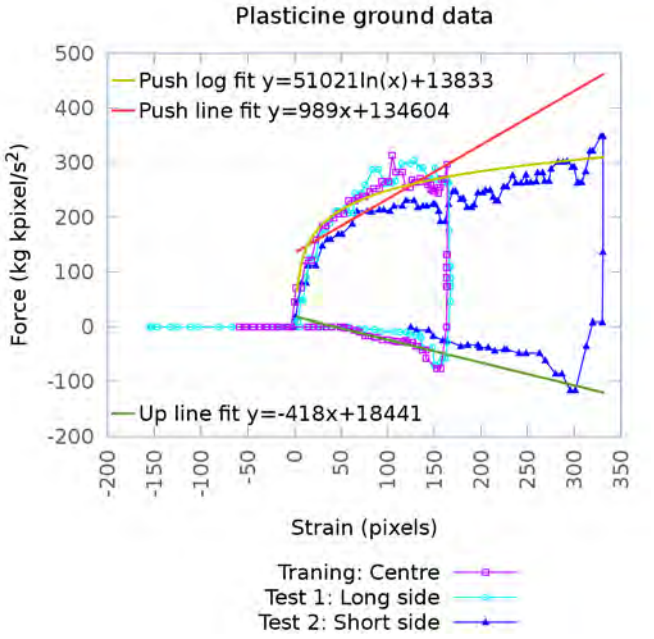

FIGURE 10.18: *Regression curves to model the force (plasticine) [See text].*

work because of the type of interactions that have been studied, that is: no hysteresis is being taken into account and the materials are homogeneous. In other words: the behaviour of the material does not change, more measurements under the same conditions would only yield similar data. However, there are other aspects that would produce variations, for example: a more rigorous model would allow the robot to play with the plasticine until it softens, that is, the softening process would be modelled as well. Even the mass-spring model studied here does not take that into account.

Summarising, if the material remains approximately homogeneous during the experiments, one measurement of the stress-strain curve will provide sufficient information for the robot. These values will not be as good when the conditions change, for example, when the robot pushes in a different position, as in the data used for testing. What the experimental data shows is that the errors can still be tolerated and a good approximation can be obtained [Figure 10.17 and Figure 10.18]. The measurements taken when the finger pushes the centre of the target, on its longest side, are used for training; the other two sets, for testing [Figure 10.19 and Figure 10.20]. Notice that the *Force vs Time* curves look more dissimilar because the independent variable in the graph is the time, here the function depends on how the robot moved the arm. When the force is plotted against the strain [Stress-strain diagram on the right corner] the curves look even more similar, because the constitutive dependence between the stress and the strain is correctly captured.

Additionally, when the right type of curve is adjusted to the data, it is possible to predict expected values even beyond the original range of data gathered during the training phase. For this reason, a type of piece-wise curve regression was used. One curve was fitted to the points corresponding to the pushing action and another one to the points corresponding to the retrieval of the finger. The rapidly changing points

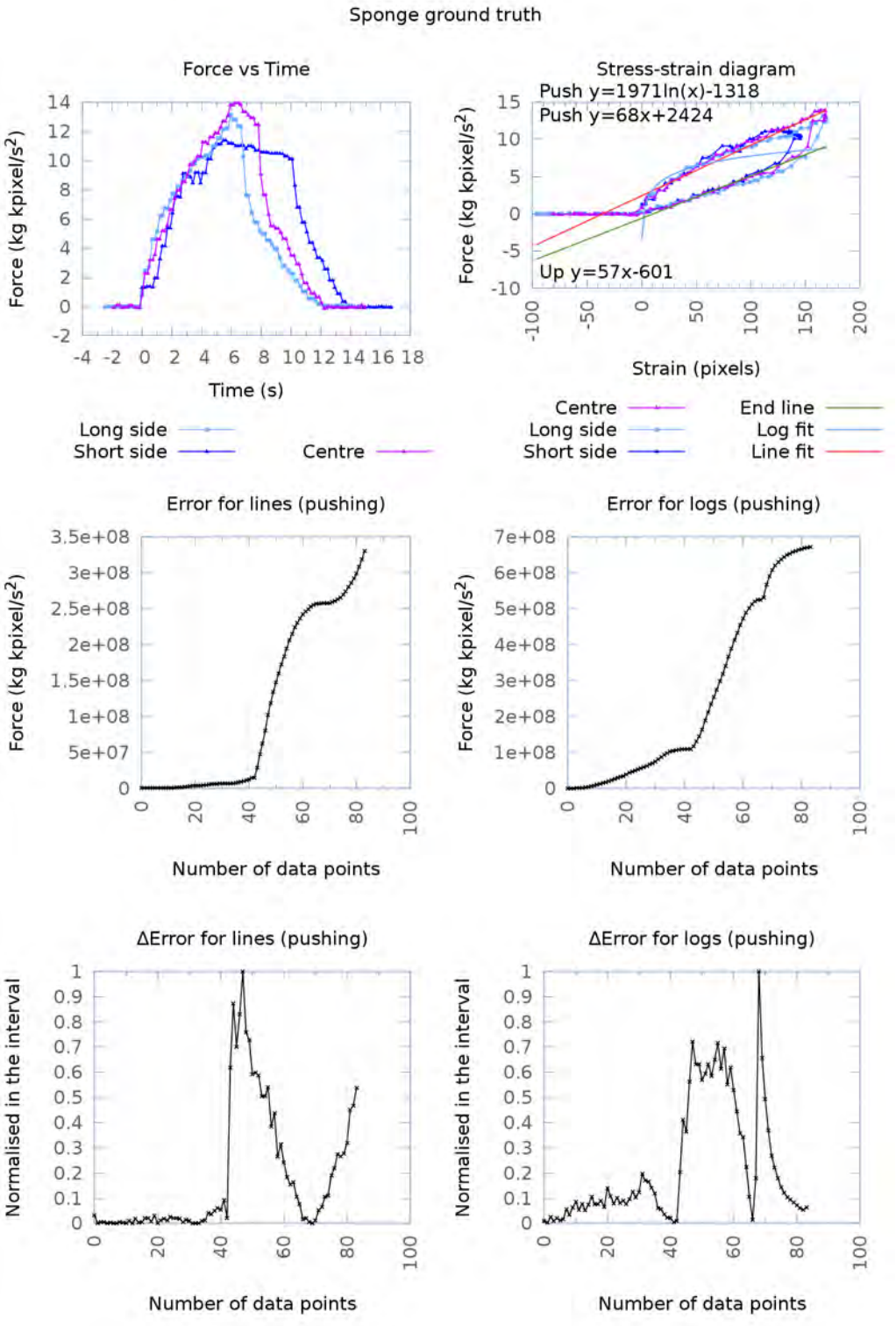

FIGURE 10.19: *Selecting regression curves to model the force (sponge) [See text].*

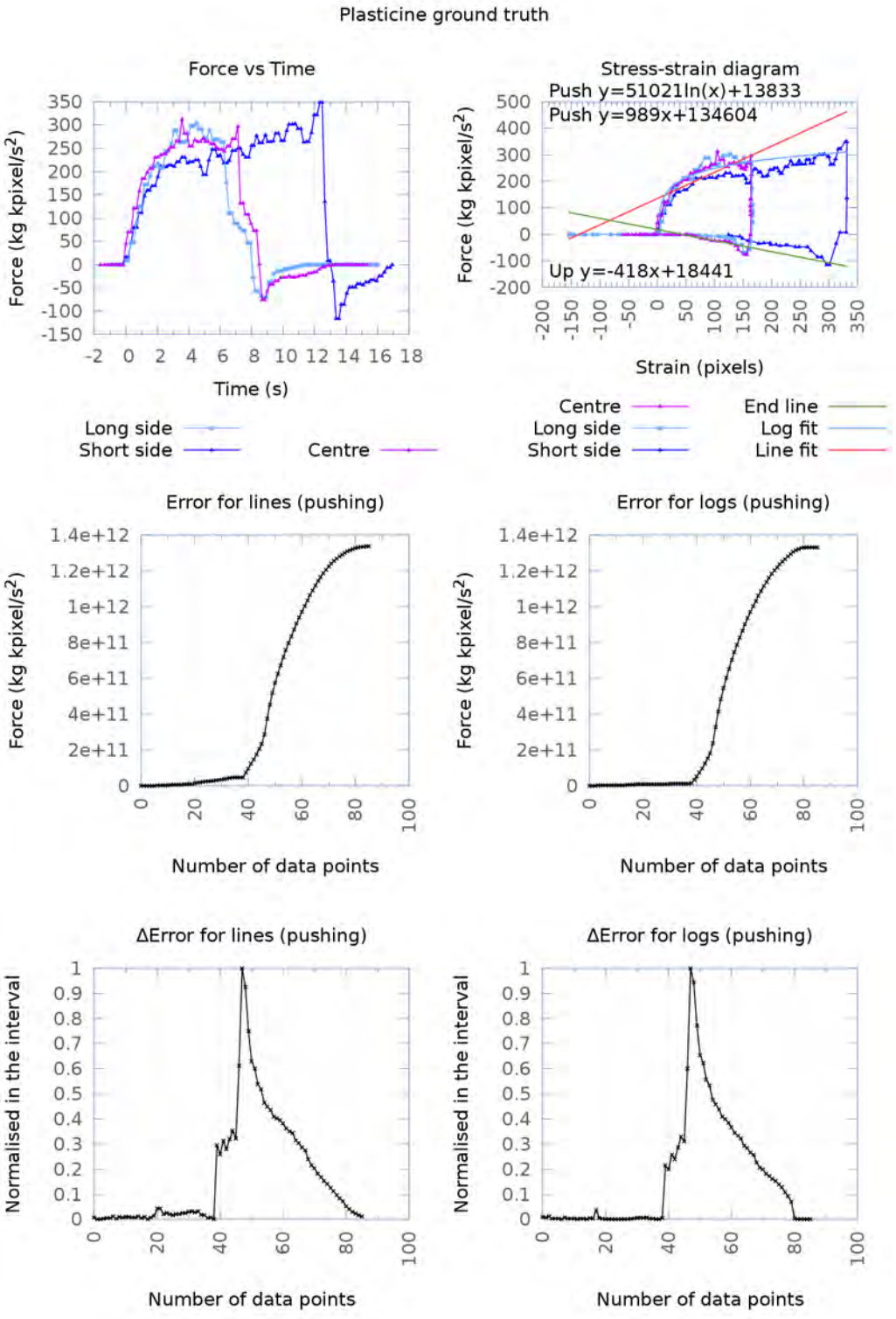

FIGURE 10.20: *Selecting regression curves to model the force (plasticine) [See text].*

in between were ignored. Depending on the material, the curve that better suits the points' series can be different. For this reason, the system proposed a set of candidates for each piece (lines and logarithms sufficed here). The best candidates for each type of curve for the pushing and retrieving stages are included in the stress-strain diagrams at the top right of Figure 10.19 and Figure 10.20.

The process of adjusting a curve to each stage of the interaction addresses some difficulties. The first difficulty consists in knowing which points should be modelled by each segment, with criteria that the robot can apply autonomously. For the first part of the movement, the pushing action, it was quite straightforward to solve this issue. There are two cues that indicate the start of the action: the visual system detects the fist collision between the finger and the target, and the force signal begins to increment its value in a noticeable way. Of these, the first is the most reliable. From this point on, the system will try adjust a curve using least squares. The graphs in the middle of Figure 10.19 and Figure 10.20 show how, as a curve is adjusted to more points, the errors between the real values and the values predicted by the curve change. When all the points considered are correctly approximated by the curve, this error remains low; as the point of discontinuity is found the error increases rapidly (this happens close to 40 points). When the rate of change in the error is considered [bottom line of the images], it is quite straightforward to identify when no more points must be added. A simple criterion like, accepting a maximum change in the error of 10% of the maximum error, is good to produce the desired result. Finally, the best curves of each proposed type are compared. The type of curve with the smallest error is adopted as a model for that part of the action.

Modelling the second part of the movement, the retrieval of the finger, can be trickier. To begin with, the moment when this segment applies, depends on when the robot

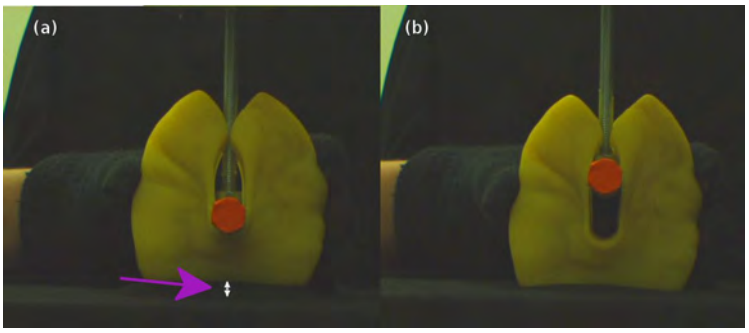

FIGURE 10.21: *Additional issues during the retrieval of the finger, in the case of the plasticine. (a) The plasticine gets stuck to the finger, which lifts it above the table, and it has to be pulled down. (b) The finger must push the new borders of the plasticine on its way back.*

decides to retrieve the finger, not on a particular value of the stress. Also, the points during the transition phase are not easy to classify. Furthermore, in the case of the plasticine, many of the forces come from collisions with the new borders of the object or from the plasticine being stuck on the finger [Figure 10.21]. Whatever relationship can be found here between the movement of the finger and the forces experienced, will not be linked so directly to the stress-strain relationship as before. An approximation can be made, but the physical significance is not as clear. To adjust the curve, it is easier to work backwards: from the moment when the finger goes back to the point of the first collision, adding data points, towards the farthest previous point that can be approximated with the same curve without increasing the error beyond the threshold point. The lines adjusted in this case can be seen in the stress-strain diagrams of Figure 10.19 and Figure 10.20 [top right].

In spite off all the restrictions and approximations, it was possible to use the adjusted curved to feed predicted forces to the mass-spring model during simulation time. When the prediction of the force is used, and the collision recovery routines are used at simulation time, there is no detriment to the overall performance of the simulations.

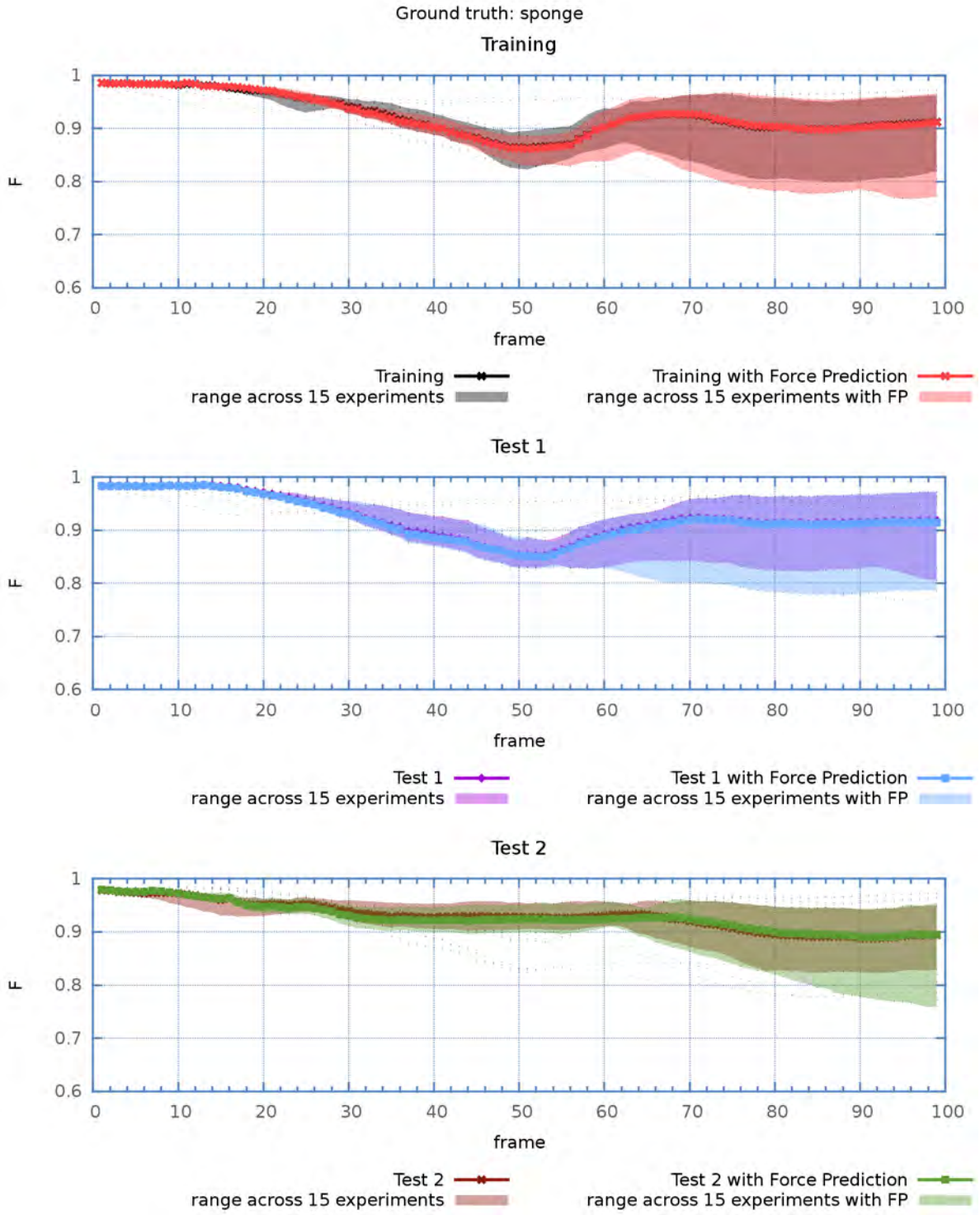

FIGURE 10.22: *Comparison of the performance of the simulation when the force is read at runtime vs when it is predicted for the sponge.*

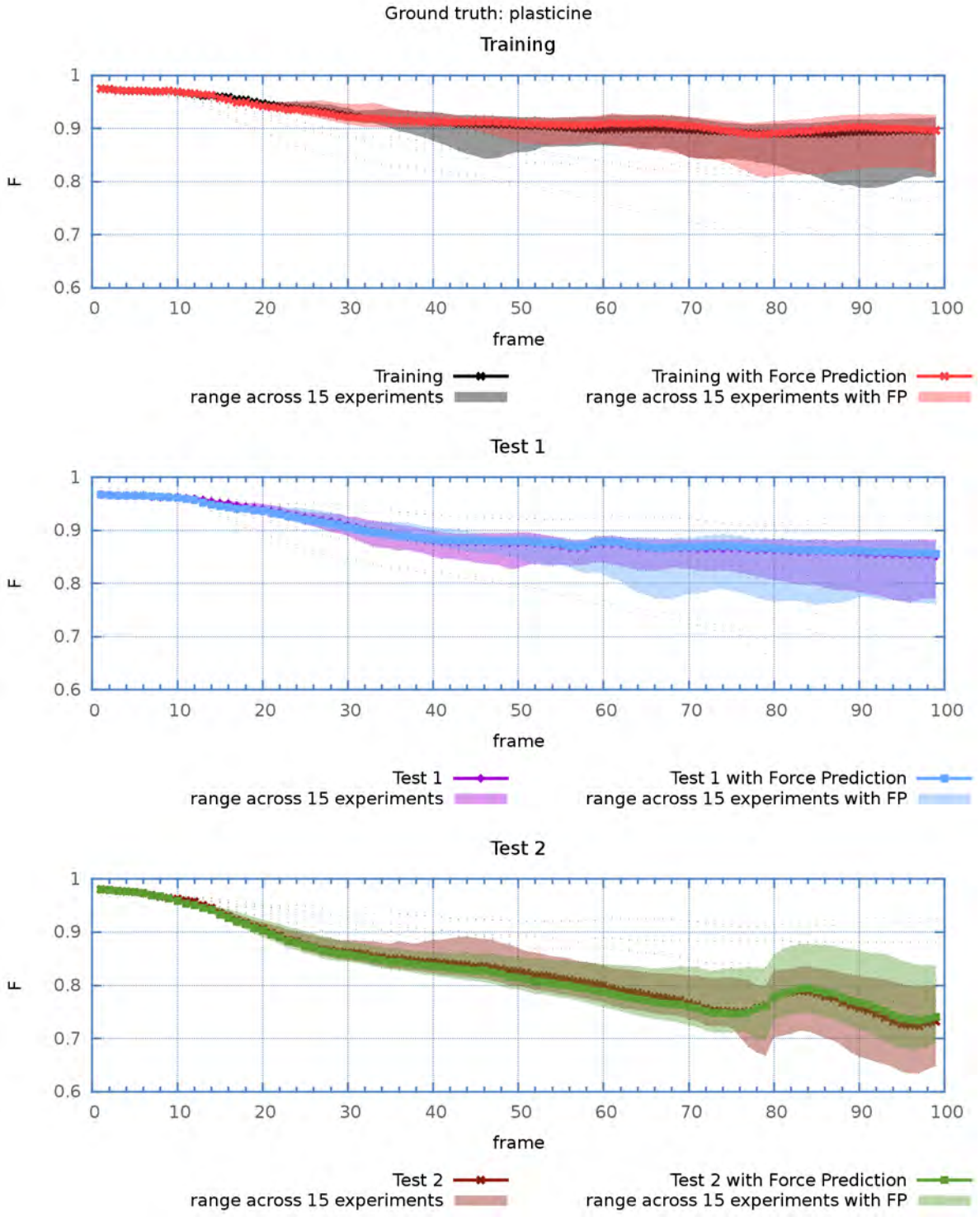

FIGURE 10.23: *Comparison of the performance of the simulation when the force is read at runtime vs when it is predicted for the plasticine.*

Figure 10.22 and Figure 10.23 show a comparison of the performance of the 15 best models obtained when the training made use only of force information and no graphic recovery routines were active, since this makes the parameters more sensitive to the type of material; however, at runtime the system is allowed to use the geometric recovery routines, since this produces better simulations. The results for the simulation of the sponge and for the simulation of the plasticine are quite similar [Figure 10.24, Figure 10.25].

To have a more sensitive evaluation of the performance of the prediction of the force, the same simulation was run but without the use of collision recovery routines (i.e. without correcting the mesh when the finger trespasses it). This type of simulation is much more sensitive to errors in the prediction of the force because it relies purely on it to deform the mesh. The simulation of the sponge is practically identical [Figure 10.26]. The simulation of the plasticine works well for the first part of the simulation but, not surprisingly, it fails badly on the way back when the finger collides with the new borders of the object [Figure 10.27]. This happened because the finger had already trespassed the border of the mesh and the predicted force affected the surrounding vertices. The force predictor, as it stands right now, does not have any way to adapt to changes in the configuration of the mesh, beyond what had been "felt" at training time.

There is still much work to do if this method is going to be used for more types of interactions. Particularly, this method depends largely on the type of collisions taking place (strains produced). On the contrary, the mass-spring model reacts as it has to, as the shape of the target changes. However, a fist step has been taken that shows how the haptic information acquired during training time can be used as well to generate a model that predicts expected forces for novel interactions. If these forces are used as

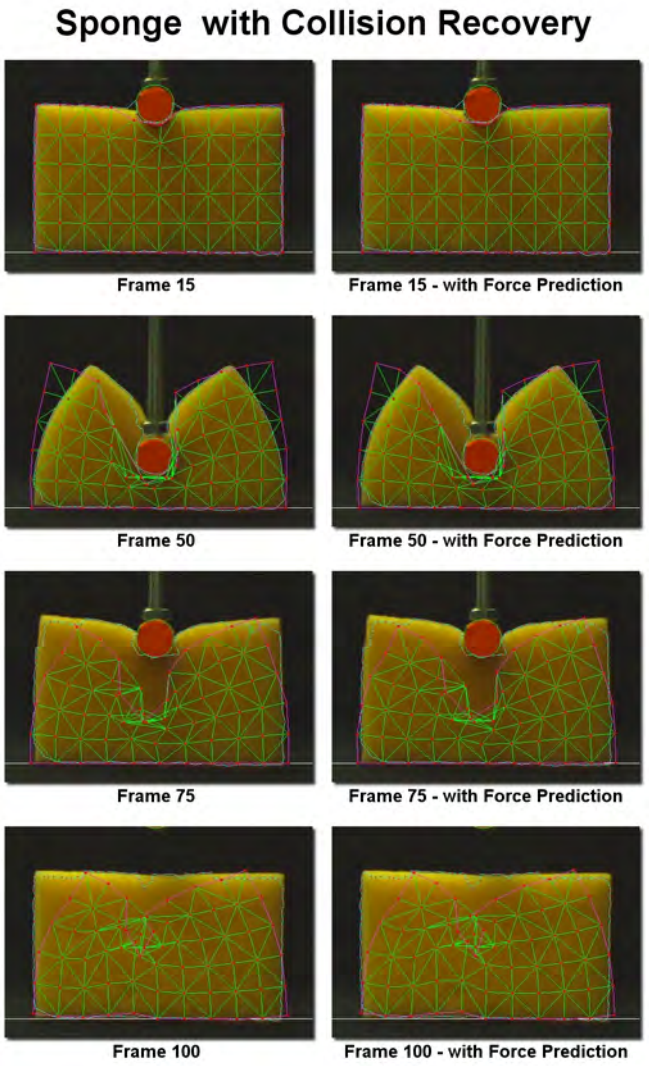

FIGURE 10.24: *Simulation of the sponge. Comparison of the images generated without and with force prediction, simulating with geometric collision recovery routines.*

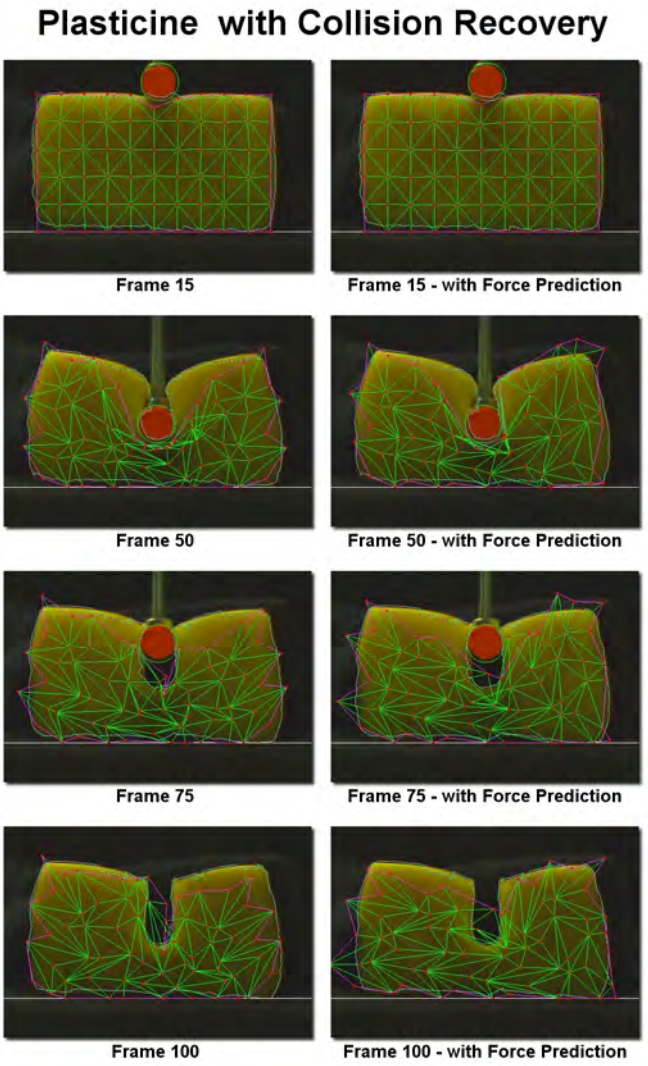

FIGURE 10.25: *Simulation of the plasticine. Comparison of the images generated without and with force prediction, simulating with geometric collision recovery routines.*

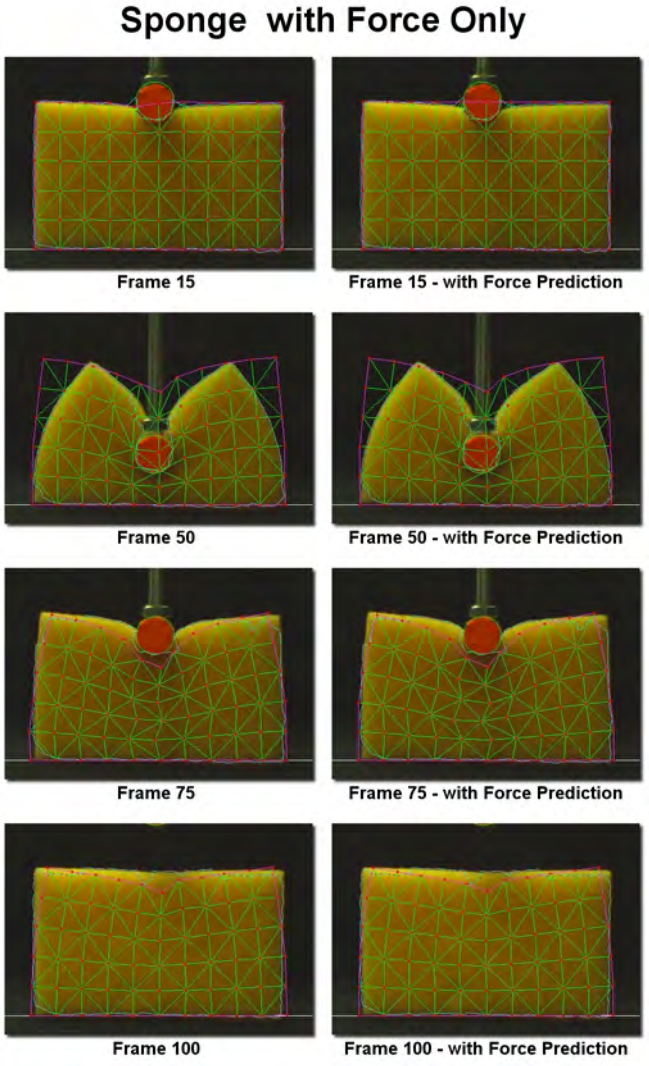

FIGURE 10.26: *Simulation of the sponge. Comparison of the images generated without and with force prediction when the simulation uses only forces.*

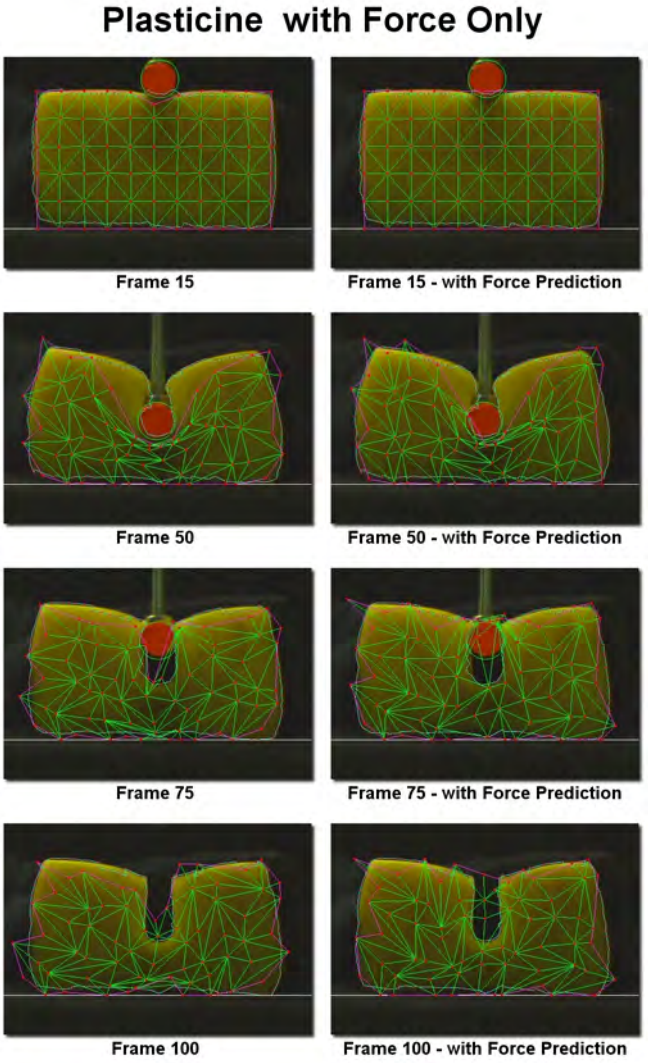

FIGURE 10.27: *Simulation of the plasticine. Comparison of the images generated without and with force prediction when the simulation uses only forces.*

input to the mass-spring model, the amount of information that the robot will need to make predictions is significantly reduced.

10.5 Summary

These sections presented the evaluation of the performance of various ways to configure the mass-spring model, in terms of the efficiency of the learning algorithm and the quality of the simulations obtained. It was found that, even though the use of additional graphic recovery routines helps to prevent the distortion of the mesh, the area enclosed by this mesh is not as similar to the ground truth as the area covered by the meshes produced by models that do not use these routines. Furthermore, the predictions made for interactions that had not been seen before remained acceptable, even though they began to deviate more from the ground truth when the conditions changes considerably, as happened with the plasticine when it was heavily pushed on its shorter side [Figure 10.16].

Additionally, the stress strain diagram obtained during the training phase was approximated by a piece-wise regression curve. This curve allows the system to make predictions about the force during simulation time. The forces predicted can then be used in the mass-spring system to predict the changes in the shape, instead of depending on a force sensor.

Now that the models are calibrated, and some predictions can be made, it is possible to use them to solve another problem: the classification of the material the robot encounters. However, some additional requirements were observed. The details are described in the following chapter.

Chapter 11

Experiments on Classification of the Materials by using the Learned Models

The previous chapter showed how the mass-spring models can be automatically calibrated to predict the reaction force and deformation of the shape of different materials. In this chapter the calibrated mass-spring models are used to find the class of the material the robot interacts with, if the material belongs to one of those already trained. Furthermore, it explores whether the system could differentiate between the two key opposing behaviours it is trying to model: elasticity and plasticity.

11.1 Parameters for Sponge vs. Parameters for Plasticine

In this section it is analysed when the parameters obtained by the learning algorithm reveal some information about the class of the material they come from. The best models, capable of modelling elastic and plastic behaviour, as obtained without the use of graphic constraints, with the customised integration scheme, were used, since these obtained the best marks consistently. The force predictor is not used here.

Three different possibilities for the type of training were explored. It has been mentioned that the system has the option to use or not to use information about the reactions forces of the material, through the force sensor mounted on the actuator. These forces are applied on the mass-vertices near the finger, thus producing the deformation. However, these forces can be ignored when producing the simulation because of the graphic collision resolution routines. These routines warranty that any vertex or edge near the actuator gets pushed away, so that they do not overlap with the actuator. The effect of this action gets propagated to the neighbouring elements due to the various spring forces explained before. Even though both options produce good visual predictions, differences appear when trying to classify materials. If the system is trained using the collision resolution routines it can not distinguish the materials reliably. This first way to observe this is by plotting the values for the elasticity, plasticity and numeric integration parameters K_L, k_d, K_A, K_φ, $yield$, $creep$, Max_α and Max_Force. Initially, meshes of $20 \times 10 \times 2$ triangles had been used [See Appendix B], but it was found that the effects of the different parameters were more noticeable with bigger triangles. Therefore, the same training was repeated with a coarser mesh of $10 \times 5 \times 2$ triangles.

To produce the graphs, the evolutionary algorithm was run 15 times. The height of the bars correspond with the magnitude assigned to the parameter. Notice that the parameters had to be grouped according to the order of their typical magnitudes.

When a simulation is run, there are two factors that produce displacements of the nodes close to the robotic finger: the forces being applied on them and the collision resolution routines, that push the vertices away of the area occupied by the finger, if the forces did not move them enough [Section 9.4.1]. Both, the force sensed and the collision resolution routines were used in the simulation when its final performance was evaluated for the previous chapter. Even though the quality of the simulation improves when collision resolution routines are used, whether they are used or not during training time does not produce any noticeable difference on the quality of the simulation at run time, if collision resolution is added again, according to the fitness function that we used. However, this fitness function evaluated only the visual appearance of the simulation. For the classification problem we found that, whether collision resolution is used or not during the training phase, did have an impact.

The first set of experiments uses the natural combination of force and geometric/-graphic information. That is, the forces are applied on the vertices of the mass-spring system and collisions are resolved using geometric routines. The histogram in Figure 11.1 shows the values that were assigned to each parameter during each of the executions of the learning algorithm, for the models of the sponge and the plasticine. The first bar of each cluster corresponds to the first execution, the second to the second and so on. The red and cyan bars correspond to the average of all the values of the parameters for sponge and plasticine, respectively; their error bars show the standard deviation. These bars and error bars show no noticeable difference between the average values of the parameters for both types of materials!

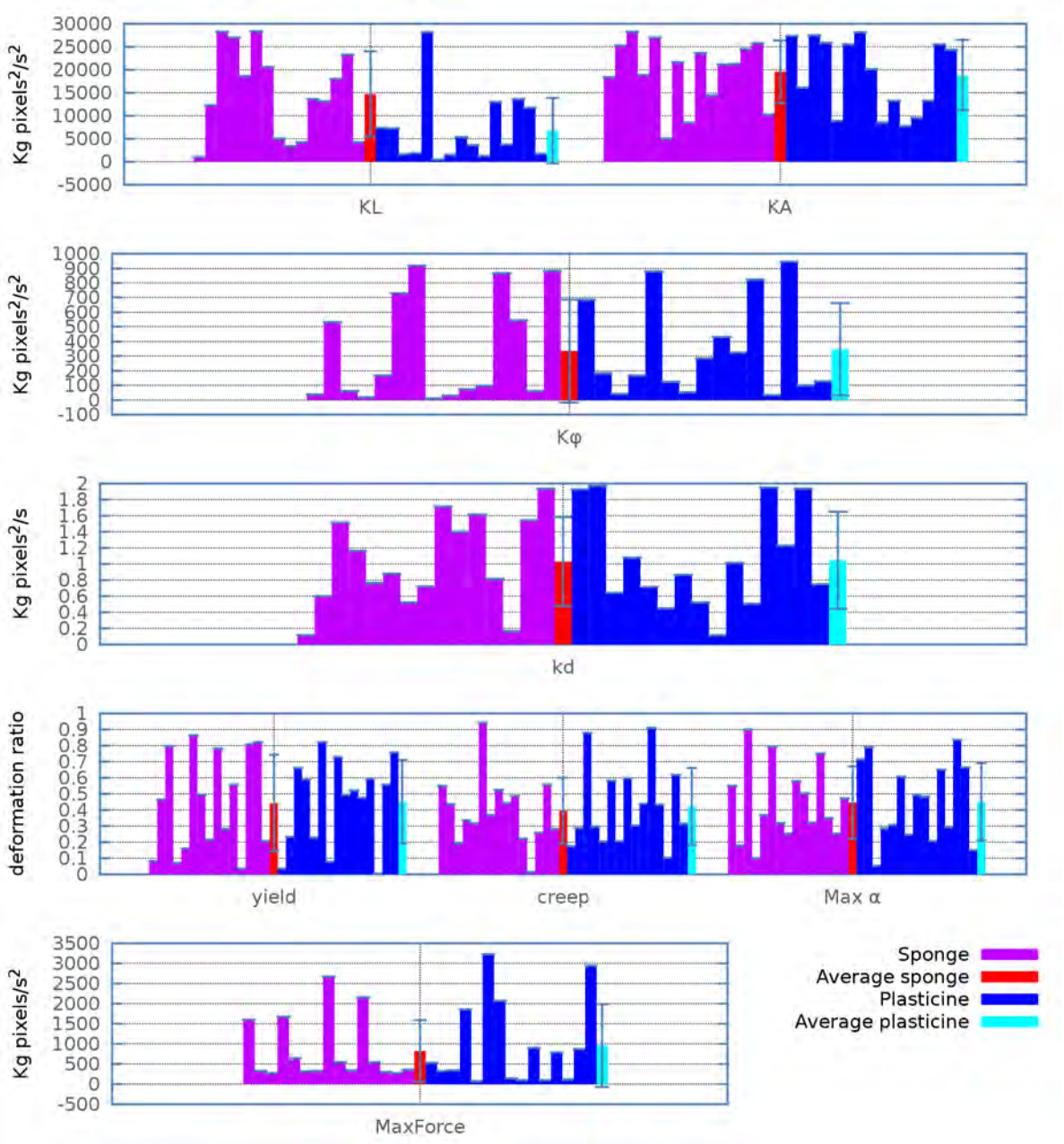

FIGURE 11.1: *Training with forces and geometric collision resolution. Values of the spring constants and plasticity parameters for the best models of 15 runs of the learning algorithm, with meshes of* $10 \times 5 \times 2$ *triangles. The last columns of each set correspond to the average value of all the previous ones and the bars, to their standard deviation. There is no visible difference between both sets.*

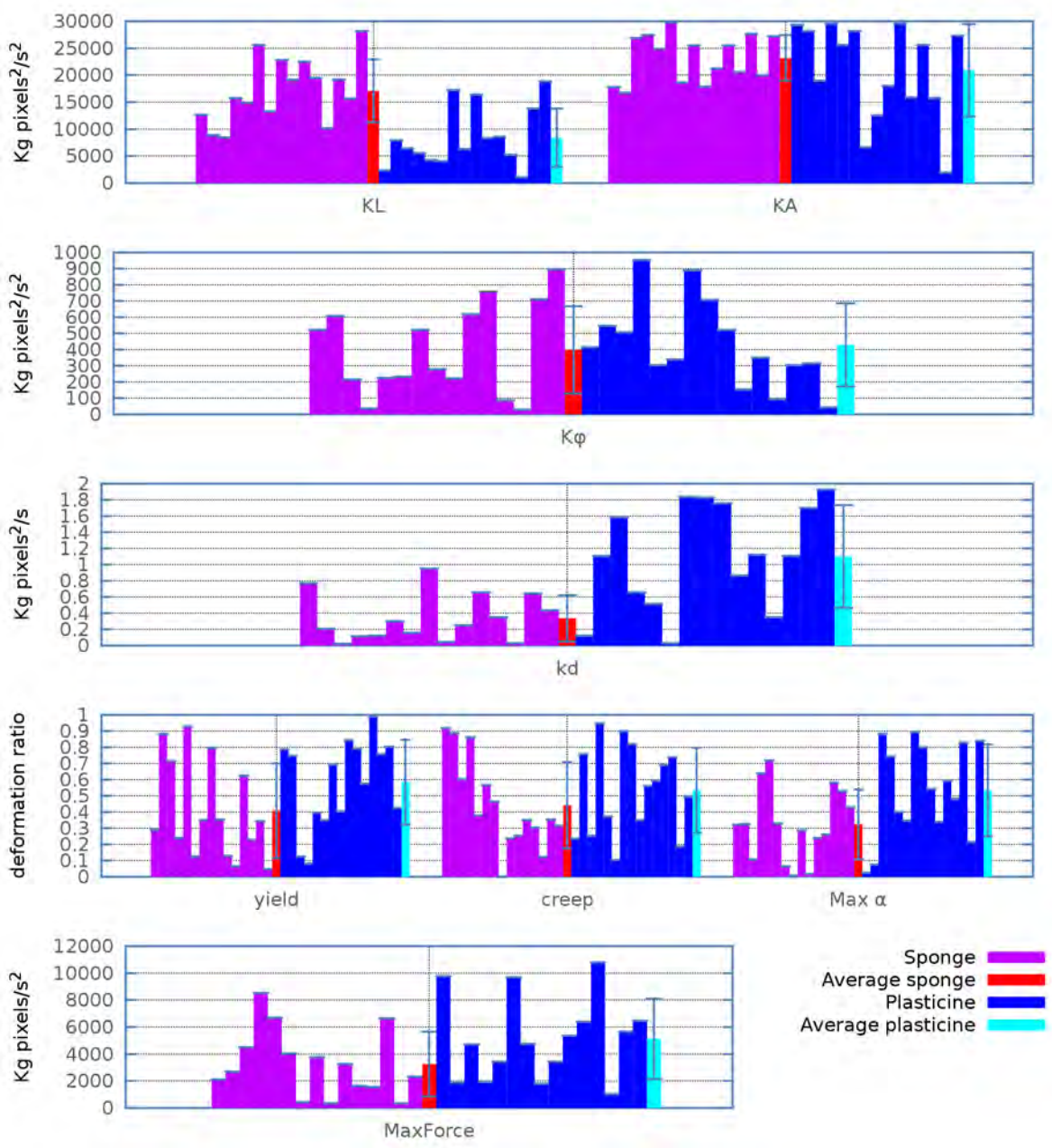

FIGURE 11.2: *Training with forces and geometric collision resolution. Values of the spring constants and plasticity parameters for the best models of 15 runs of the learning algorithm, with meshes of* $10 \times 5 \times 2$ *triangles. The last columns of each set correspond to the average value of all the previous ones and the bars, to their standard deviation. There is no visible difference between both sets.*

For a second set of experiments, only the geometric information was used [Figure 11.2]. This time, even though the standard deviation is big, a difference begins to appear for the values of $yield$[i] and Max_α[ii], in addition to smaller values of K_L and greater damping k_d for the plasticine.

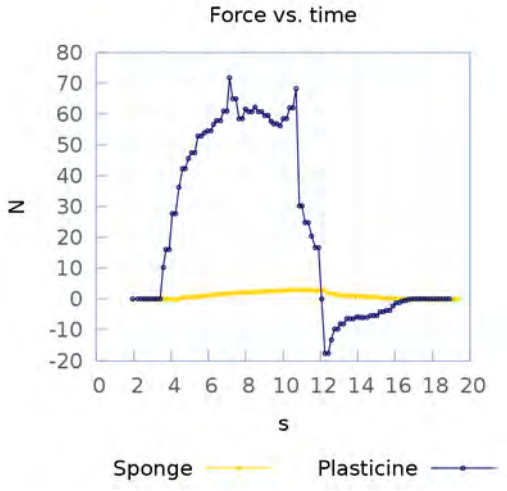

FIGURE 11.3: *Comparison of the force measured as the sponge is pushed vs. the force when the plasticine is pushed.*

However, if the original data obtained from the force sensor is observed, the difference between both materials becomes evident [Figure 11.3]. This suggests that use of only visual information to train the models, with the current marking system, is not as good for classification purposes, as the direct information from the force sensor. For classification purposes between these two materials, it would have been more than enough. However, this would not be enough still to distinguish a really hard sponge from softened plasticine, their visual behaviour would be highly relevant as well. Therefore, exploring the integration of visual and tactile information is important. Unfortunately, a great amount of research makes use of only visual information. Here we call for

[i] $Yield$ is the maximum amount of deformation before permanent plastic deformation takes place.
[ii] Max_α delimits the maximum amount of allowed permanent deformation in the model.

attention on the need to incorporate tactile information as Henrich and Wörn did in [37].

A third training set was run and evaluated, but this time, the training took place using mainly the information coming from the force sensor. The mass spring model only knew which force should be applied on vertices close to the finger, but it didn't enforce any deformation through the geometric resolution of collisions with the finger. In other words, the finger could go inside the mesh if the springs resisted the force. The results in Figure 11.4 show more interesting differences among the sets of parameters. The plastic deformation coefficients $yield$ and $creep$ are the most different. Also, some effect can still be noticed in K_L and k_d. Since both, the sponge and the plasticine are incompressible materials[iii], the lack of difference in K_A is not surprising.

11.2 Using the Quality of the Predictions

A second set of classification experiments consisted in using the best sets of parameters for the mass-spring model, learned during the 15 runs of the evolutionary algorithm, for both materials with opposing behaviours (elastic and plastic), to model: a) the material they were trained to model, b) the opposite material. The performance of both predictions could be compared on a frame by frame basis using the mark F, which gives information about the quality of the prediction for the current frame. The precision of the classification can be evaluated by considering in how many frames the classification was correct and incorrect. Alternatively, a final classification is decided

[iii] Their density does not change easily with pressure.

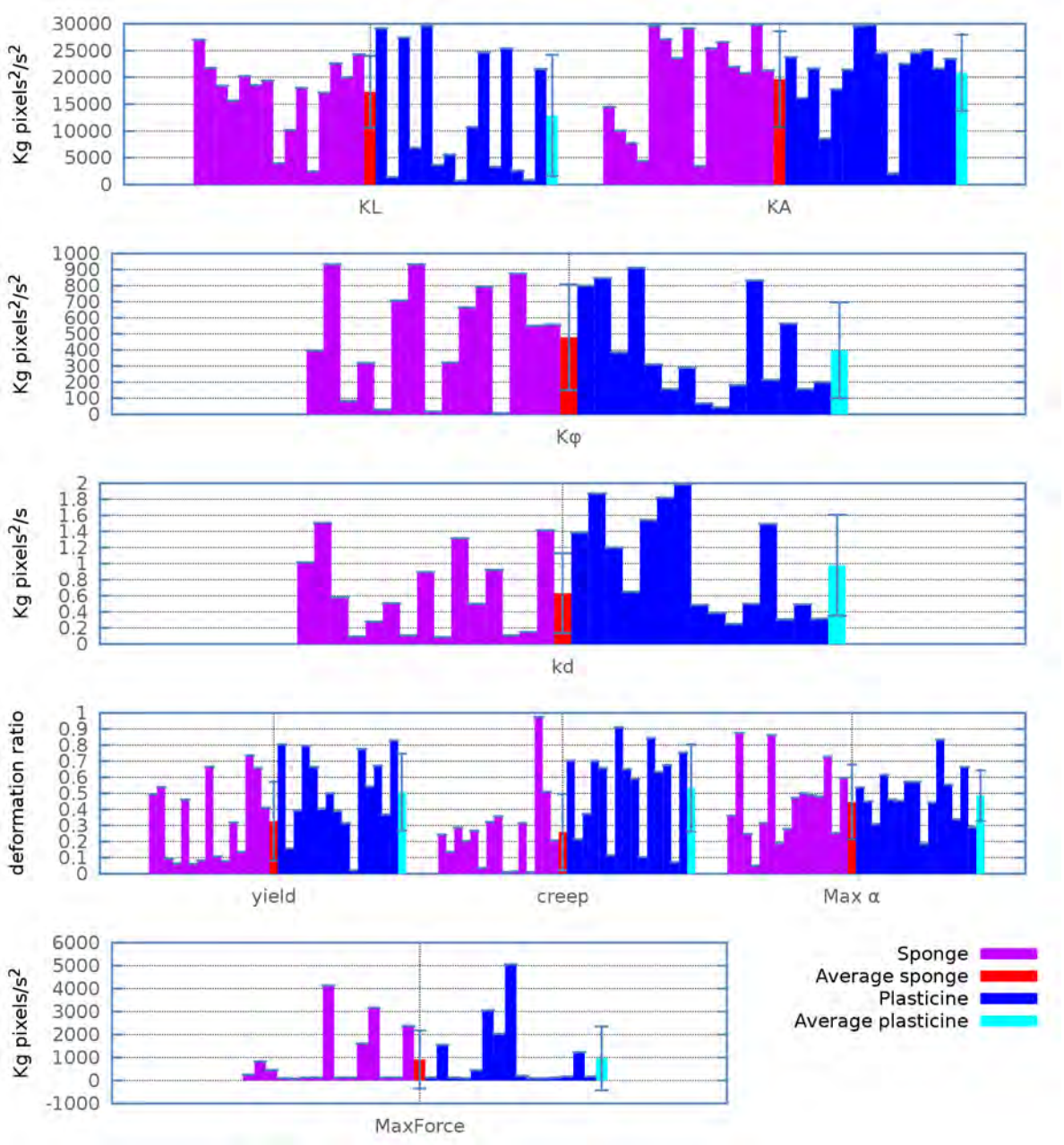

FIGURE 11.4: *Values of the spring constants and plasticity parameters for the best models of 15 runs of the learning algorithm, with meshes of* $10 \times 5 \times 2$ *triangles. For the training phase, information from the force sensor and no collision resolution with the finger were used. The last columns of each set correspond to the average value of all the previous ones and the bars, to the standard deviation.*

on the basis of which predictor performed better over the whole sequence, with Algorithm 6. The following data was obtained when using the 15 best sets of parameters, with meshes of $10 \times 5 \times 2$ triangles.

There were three groups of experiments:

1. Force and geometric collision recovery routines were in use during the training phase.
2. Only geometric collision recovery routines were in use during the training phase (visual information).
3. Geometric collision recovery routines were disabled, but force information was used during the training phase.

TABLE 11.1: *Classifications when models were trained using vision and force.*

Ground truth: Sponge				
Data Set	TP	FN	Precision	Class Assigned
training	85	15	1.00	sponge
test1	85	15	1.00	sponge
test2	57	43	1.00	plasticine
Ground truth: Plasticine				
Data Set	TP	FN	Precision	Class Assigned
training	14	86	0.12	sponge
test1	14	53	0.23	sponge
test2	35	65	0.93	plasticine

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

The results of these experiments are particularly interesting. Figures 11.5, 11.6 and 11.7 show the results on a frame by frame basis, when the training used both visual and force information. In this case the overall classification failed often [See Table 11.1].

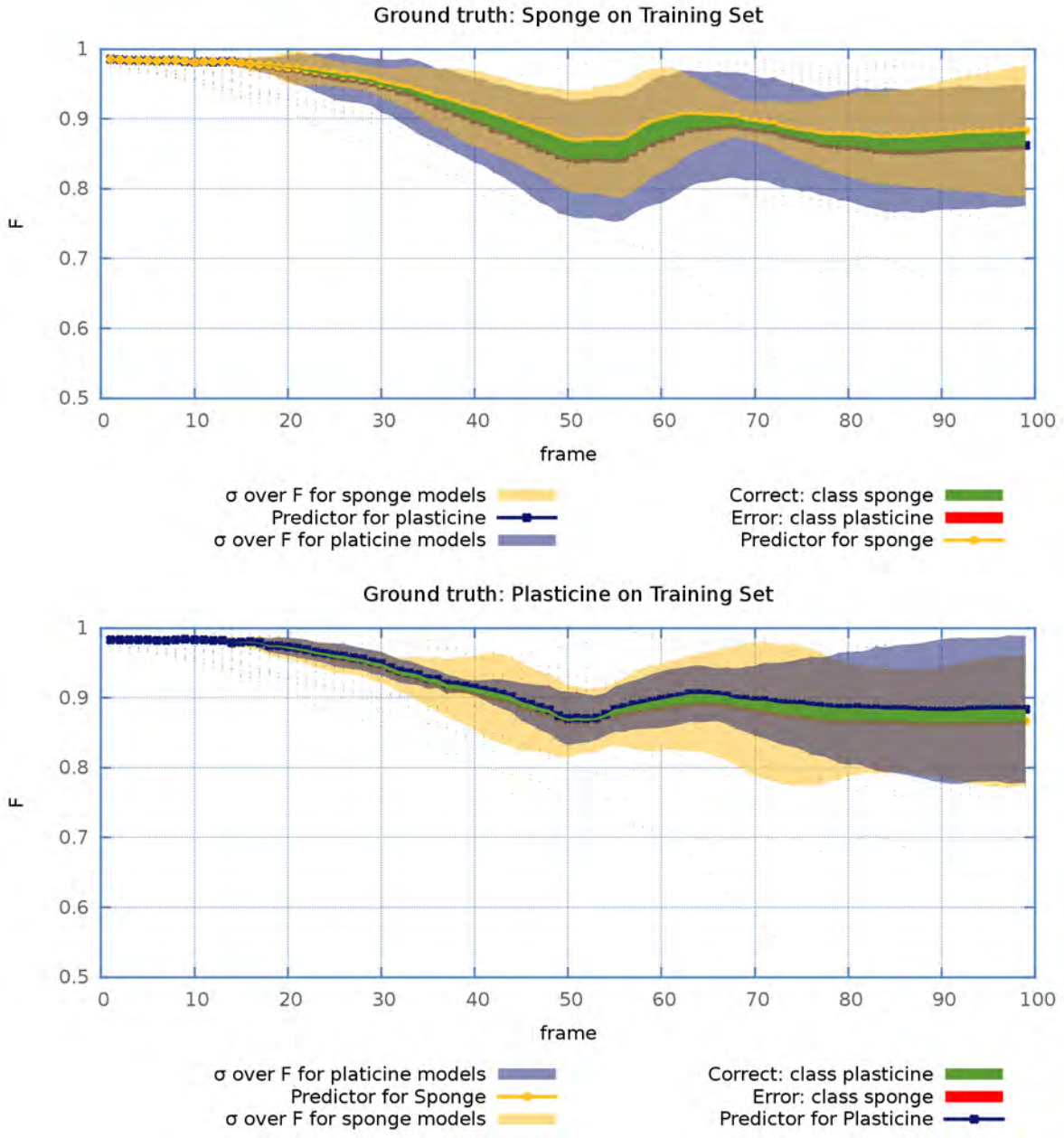

FIGURE 11.5: *Classification frame by frame. Training data sets. Vision and Force.*

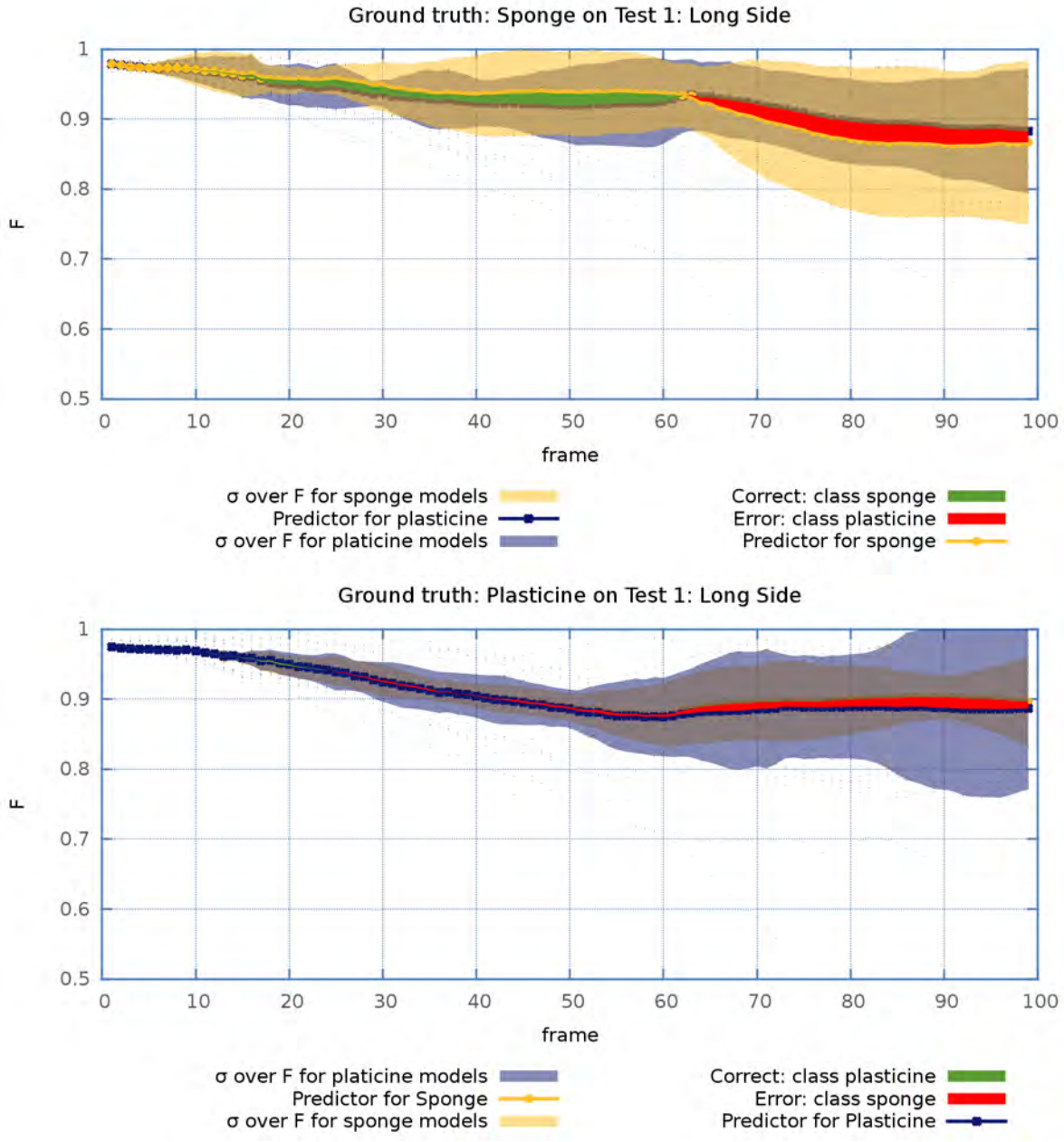

FIGURE 11.6: *Classification frame by frame. Test 1: Long side. Vision and Force.*

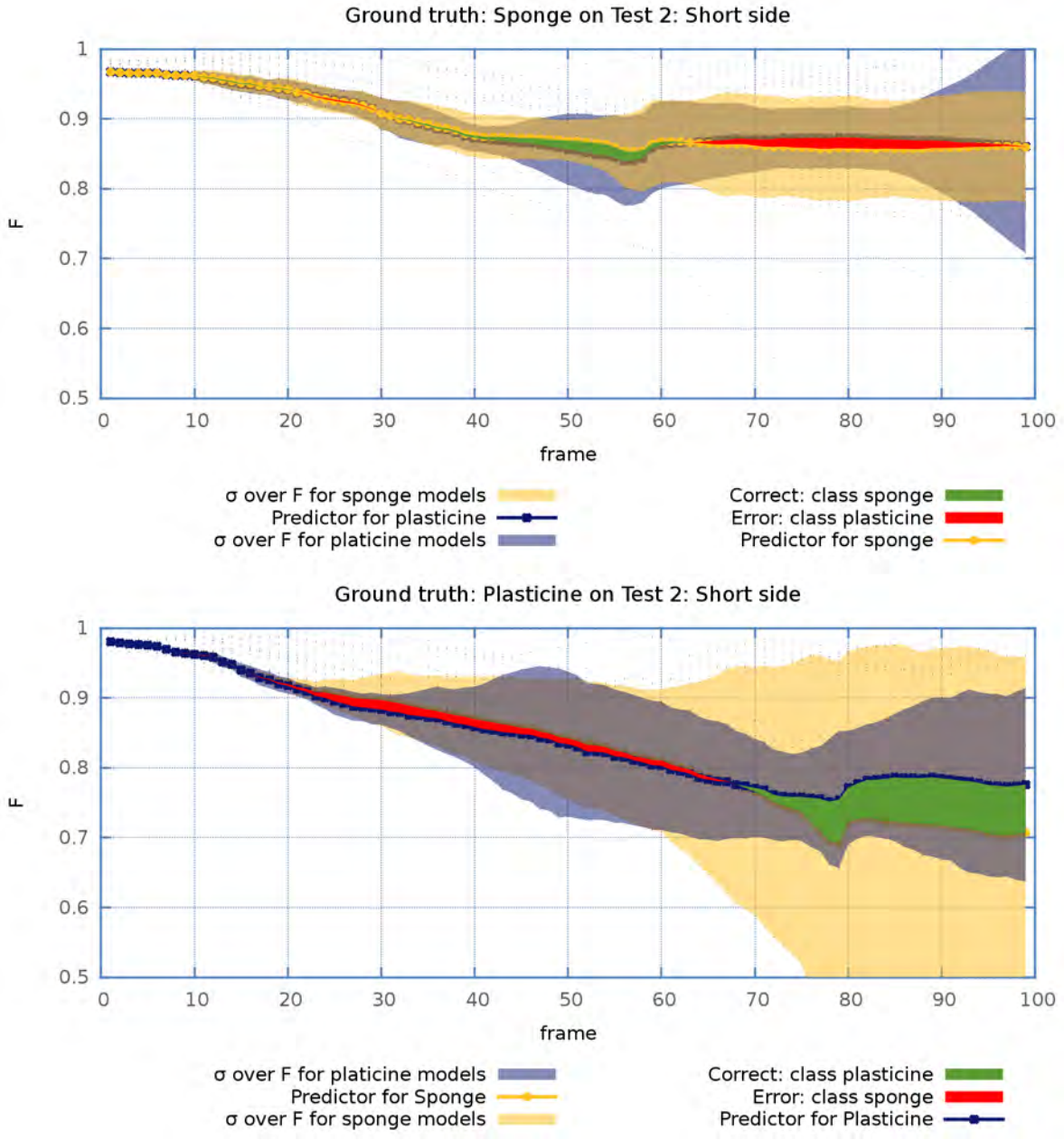

FIGURE 11.7: *Classification frame by frame. Test 2: Short side. Vision and Force.*

TABLE 11.2: *Classifications when models were trained using only vision.*

Ground truth: Sponge				
Data Set	TP	FN	Precision	Class Assigned
training	90	10	0.9	sponge
test1	86	14	0.86	sponge
test2	98	2	0.98	sponge
Ground truth: Plasticine				
Data Set	TP	FN	Precision	Class Assigned
training	31	69	0.31	sponge
test1	33	67	0.33	sponge
test2	64	36	0.64	plasticine

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

TABLE 11.3: *Classifications when models were trained using force information.*

Sponge				
Data Set	TP	FN	Precision	Class Assigned
training	87	13	0.87	sponge
test1	74	26	0.74	sponge
test2	63	37	0.63	sponge
Plasticine				
Data Set	TP	FN	Precision	Class Assigned
training	59	41	0.59	plasticine
test1	41	59	0.41	plasticine
test2	45	55	0.45	plasticine

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

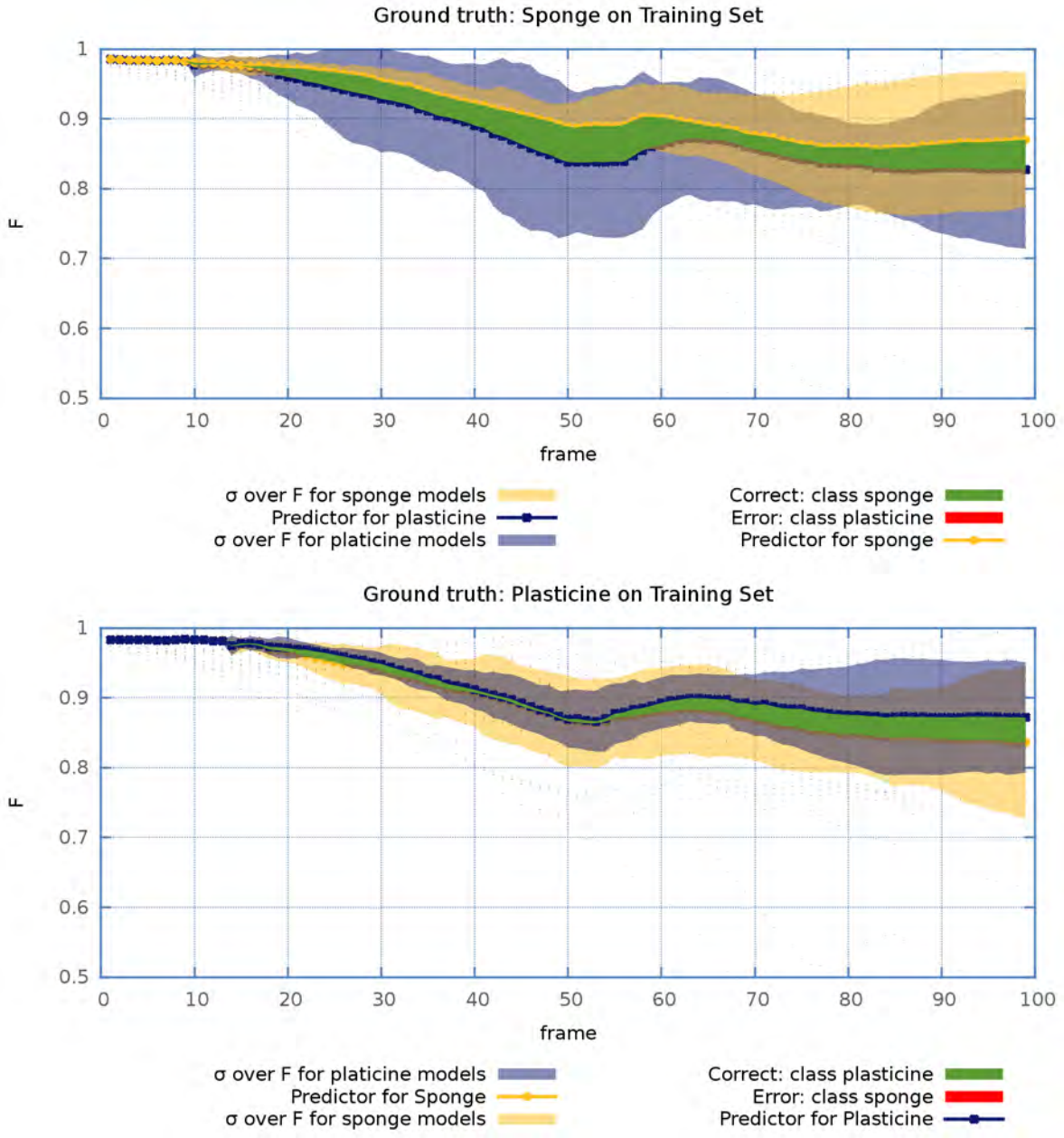

FIGURE 11.8: *Classification frame by frame. Training data sets. Vision Only.*

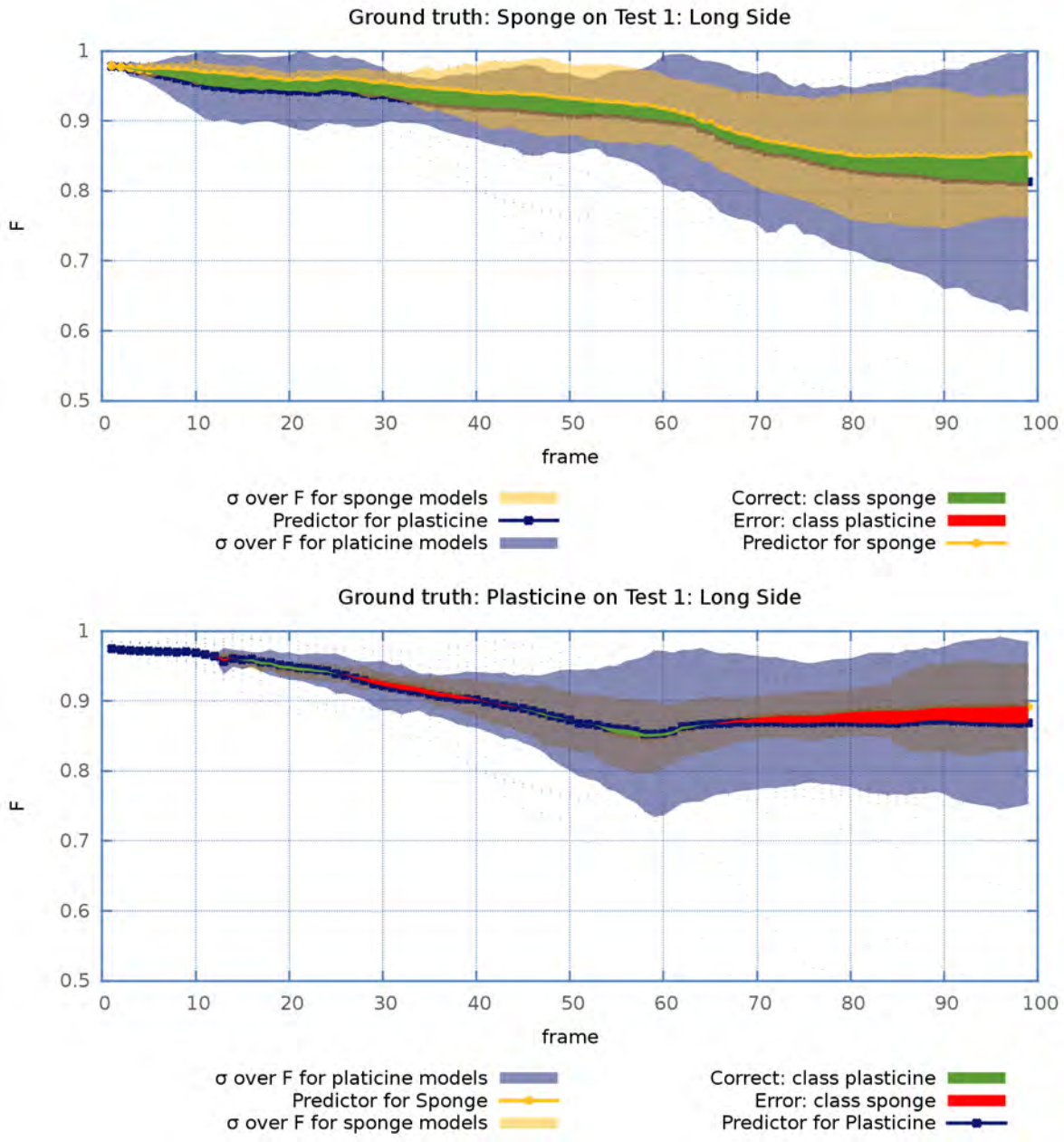

FIGURE 11.9: *Classification frame by frame. Test 1: Long side. Vision Only.*

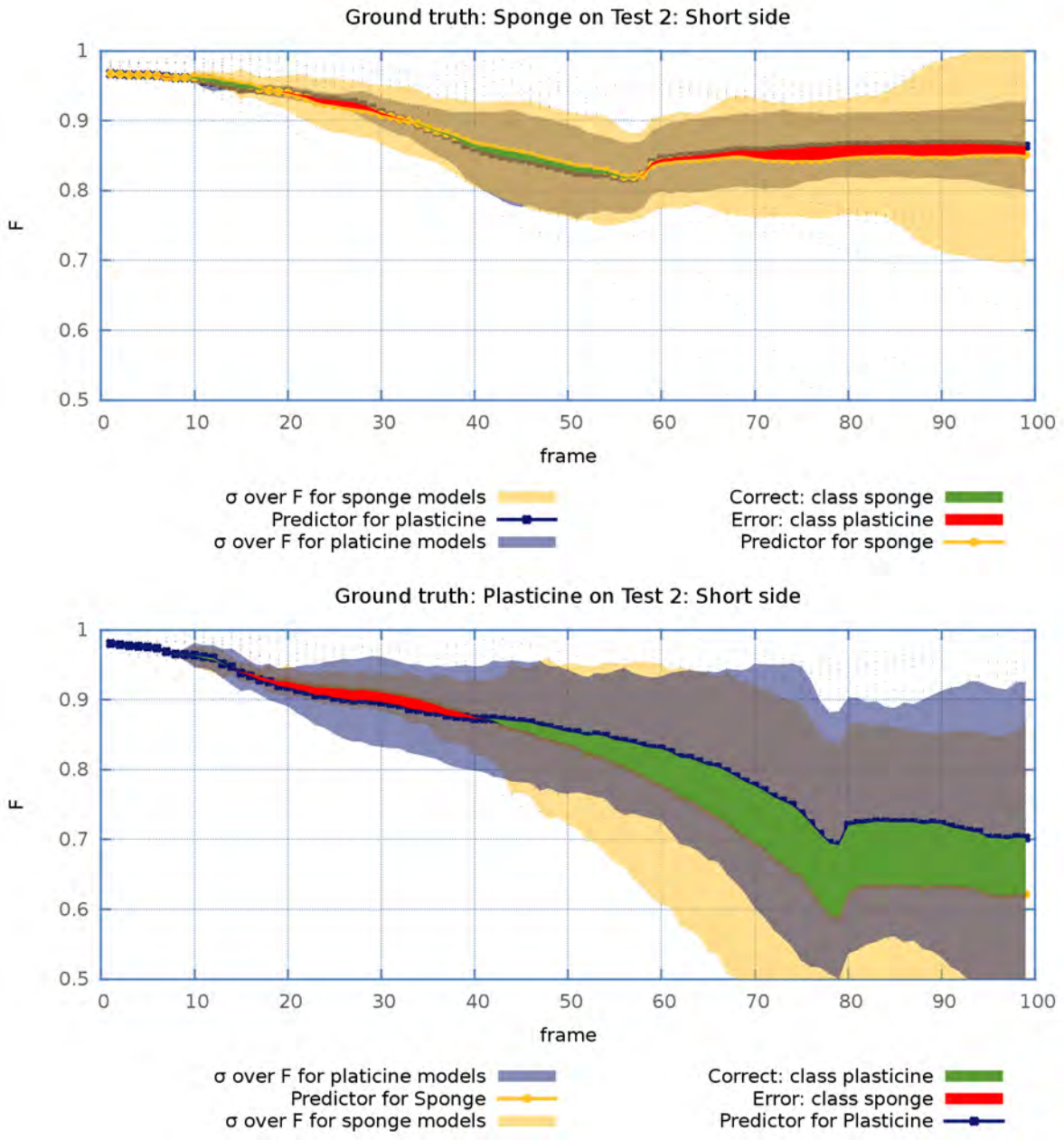

FIGURE 11.10: *Classification frame by frame. Test 2: Short side. Vision Only.*

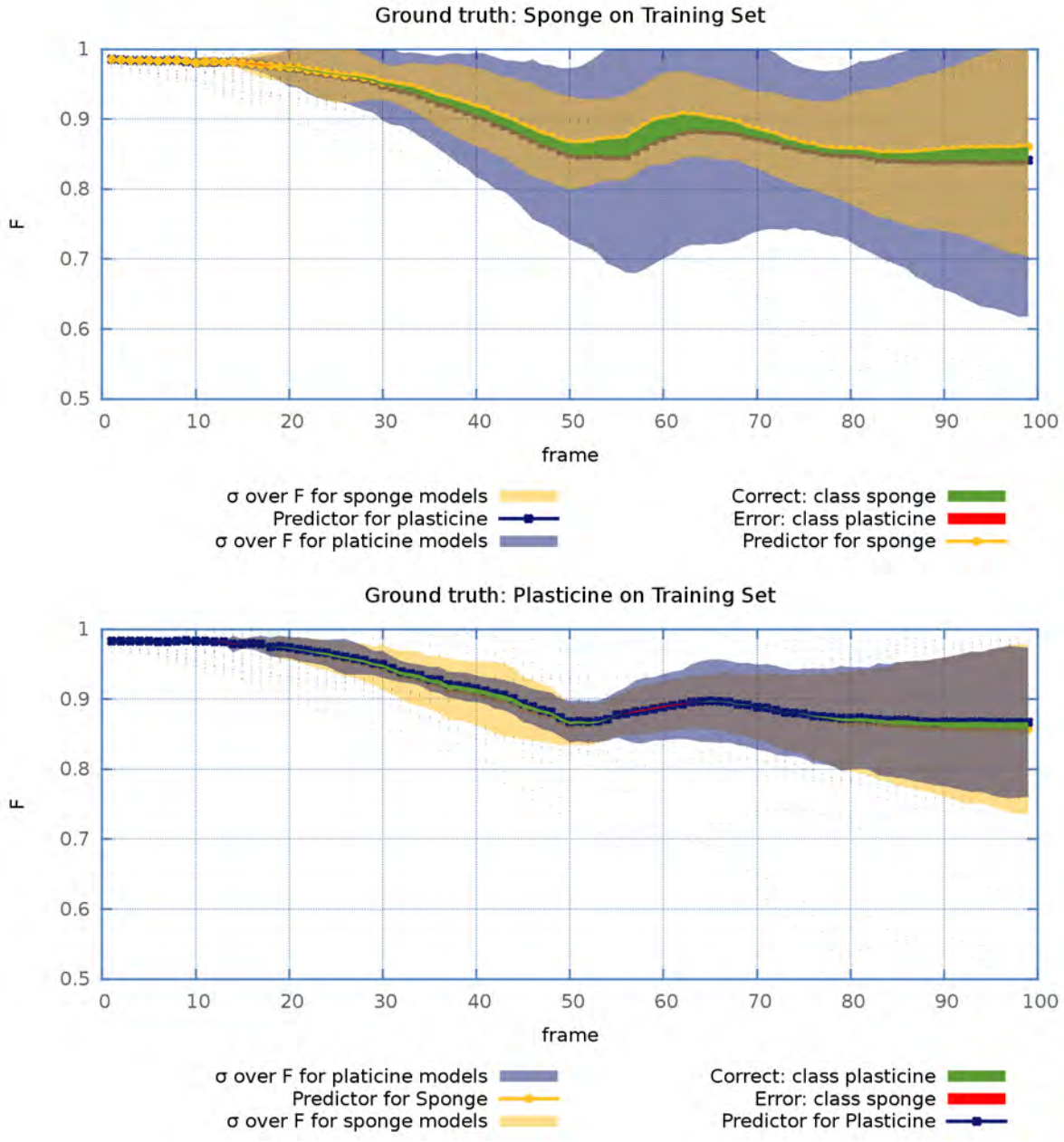

FIGURE 11.11: *Classification frame by frame. Training data sets. With Force Information*

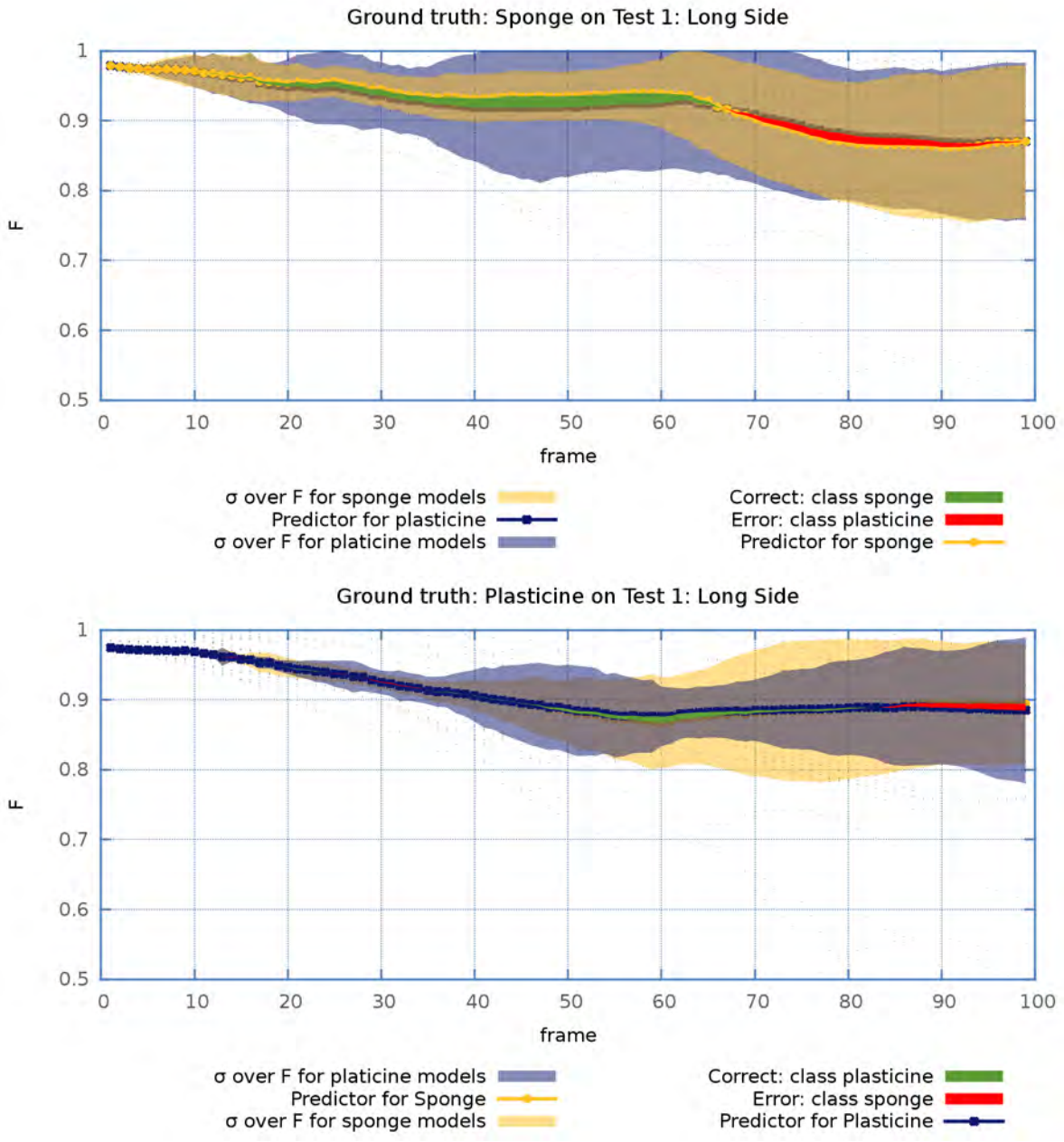

FIGURE 11.12: *Classification frame by frame. Test 1: Long side. With Force Information*

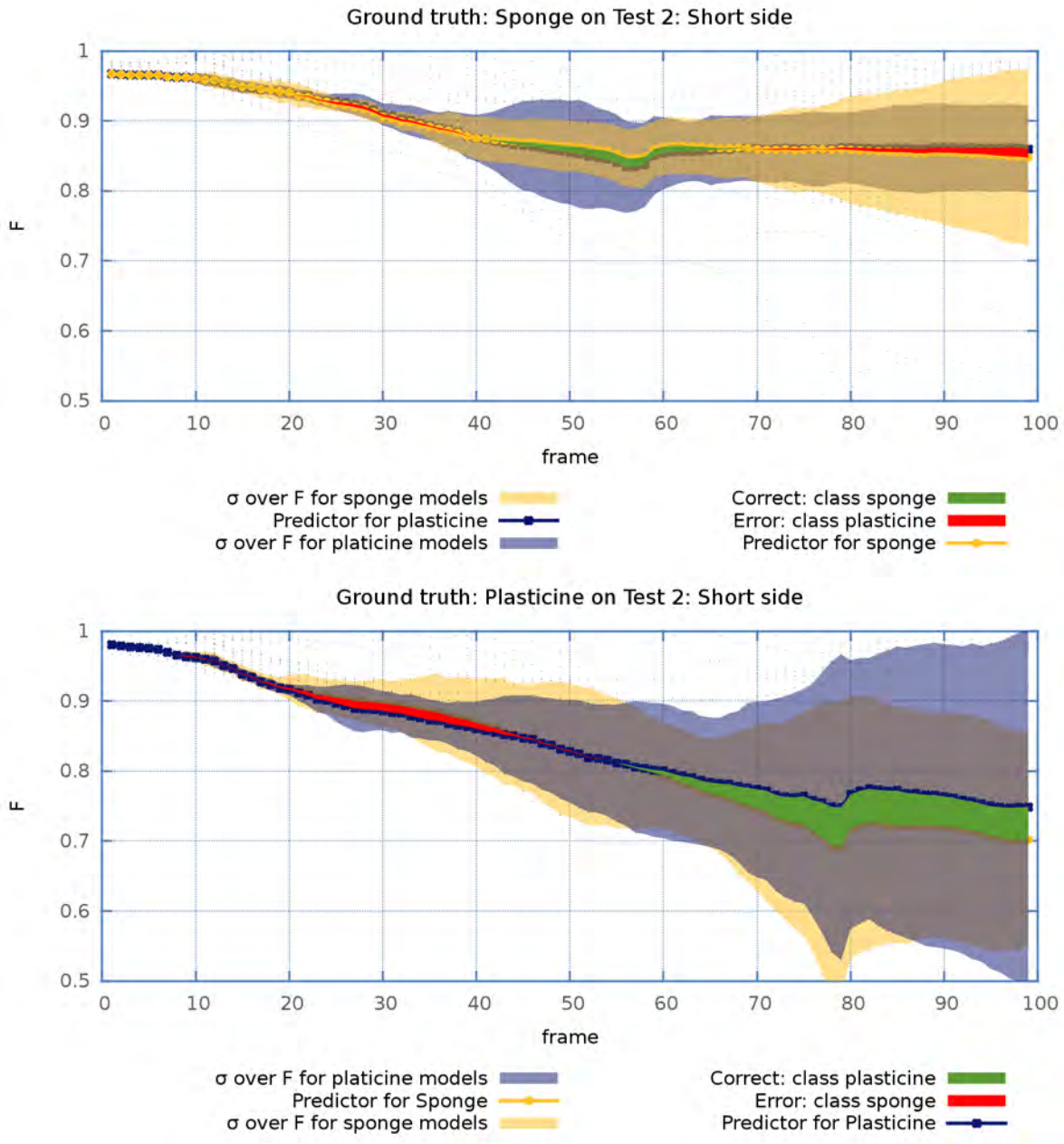

FIGURE 11.13: *Classification frame by frame. Test 2: Short side. With Force Information*

There is not much improvement when only the visual information is used during the training, as Figures 11.8, 11.9 and 11.10 show. On the contrary, when the training was run using the force information and avoiding the use of the collision resolution routine, the overall classification was always successful [See Table 11.2], and the frame by frame classification improves considerably [Table 11.3]. Figures 11.11, 11.12 and 11.13 show the results on a frame by frame basis.

The results of this section suggest that, even though the visual information is good to evaluate the accuracy of the predictions against a real ground truth, during the learning phase, the force information contributes in a more substantial way to the problem of distinguishing among materials. Once this is taken into account, the same models that were used for making predictions, can be used to suggest classes.

Chapter 12

Conclusions

12.1 Summary

The main aim of this thesis was to give robots the ability to autonomously generate models of deformable objects, as it encounters them, that would allow it to predict the behaviour of those objects when it interacts with them in different ways. This is one of the first works to address this problem in the field of robotics. For this reason, an extensive review of the literature in other fields was a prerequisite to proposing a suitable methodology. As a result, a physics based method was selected to initiate the research. Also, the requirements for real time performance and generalisability led us to choose a mass-spring model. Several simplifications were adopted for the research scenario, such as the use of 2D information and interactions whose greatest effects would conform to this restriction. Also, only a limited number of materials have been tested. The aim of this study was to evaluate the extent of the usefulness of the

selected method, and these materials provided a suitable scenario. However, future research should focus on the production of more detailed models.

Furthermore, it was found that a single run of the learning algorithm can take between 25 and 45 minutes in a laptop with an Intel Centrino 2 processor, but once the learning is done, the set of parameters can be used to run simulations almost in real time. It was found as well that the learning is more efficient for problems like classification, if force information is fed to the mass-spring model, but no geometric collision resolution routines are used during the learning stage.

12.2 Contributions

The main contributions of these research, as they had already been mentioned in the introduction and across the chapters of original work are:

1. The presentation of a scheme for addressing the problem of robotic modelling of the behaviour of deformable objects.
2. The transfer of knowledge about modelling of deformable objects from other fields, like physics, computer graphics and virtual reality. Such as the use of physics based models in addition to graphic routines, like collision detection and resolution.
3. The use of a physics based mass-spring model by the system to make predictions about the changes in the shape of deformable objects that a robot pushes.
4. The autonomous calibration of this model by the system using real objects as ground truth, while most previous work did not use real objects or was not autonomous.

5. The application of the models to the problem of context estimation (or classification).
6. Showing the benefits of using a force sensor for this type of research, following the results for the classification problem.
7. The sketching and use of a colour based tracking algorithm with snakes that adjusts the number of elements as required by the shape of the tracked object.
8. The addition to the mass-spring model of a term that enforces the preservation of the angles of the elements of the mesh.
9. The inclusion of an automatic procedure for prediction of forces, using a piecewise regression model, constructed from the stress-strain diagram obtained during the training phase.
10. The use of this model of the forces as input to the mass-spring deformation predictors developed earlier in the thesis.
11. The presentation of a detailed analysis of the dynamic behaviour of the mass-spring model, as it runs through and after the forces are removed. This analysis reveals a few problems of mass-spring models for robotic modelling.

12.3 Future Work

Mass-spring models proved to be useful models that can be calibrated and can produce accurate real-time simulations up to a certain extent. However, even though the current evaluation method, that makes use of the harmonic average of the precision and the recall, provides a standard way to select suitable simulations, when the images are observed by a human, essential errors that had gone unnoticed can be found. Notably, the fact that the plasticine does not bounce back at all when the actuator is retrieved

from the material, is not properly penalised when the simulation fails. A topic of first relevance for future research is the definition of an evaluation criterion that can take into account the actual geometric features of the shape.

Additionally, a more detailed sensitivity analysis of the approach to various parameters could be helpful. However, given the small standard deviation around the best models after the 50 generations, for different experiments which sampled very different regions of the space of parameters, it seems that a more sensitive evaluation method may be required for these tests to be truly informative.

Another field for improvement is the searching algorithm used for the training phase. So far it does not make explicit use of the characteristics imprinted in the simulation by each parameter. However, such improvement requires a more methodological analysis of these effects, and a way to automate the evaluation of which one must be tuned and by which amount.

There are also more variations to the configuration of the models that could be explored, like the shape of the meshes. Furthermore, the extension to 3D is required for more complex objects.

Finally, there are still many other systems, amongst those mentioned in the related work sections, that can be used and deserve exploration, for example particle systems.

Appendix A

Architecture

Figure A.1 summarises the stages of the robotic program that runs a simulation once the proper parameters for the model are given: the actions of the robot, the capturing of information from the environment, the identification of the objects of interest in the scene, and the execution of the simulation by using the information about the force detected and current positions of the actuator (finger) and obstacles, as tracked by the vision system.

The actual implementation of the code can be split in two main packages: Perception and Simulation. The first one is concerned with the reading of information from the sensors, and the generation of internal representations for the objects of interest in the scene [Figure A.2]. For practical reasons, the experiments were performed with the actual sensors only once. The forces and times measured were stored in a text file, while the images were stored as photographs, but this classes would work in the same way if they had to capture the data online in a synchronised way.

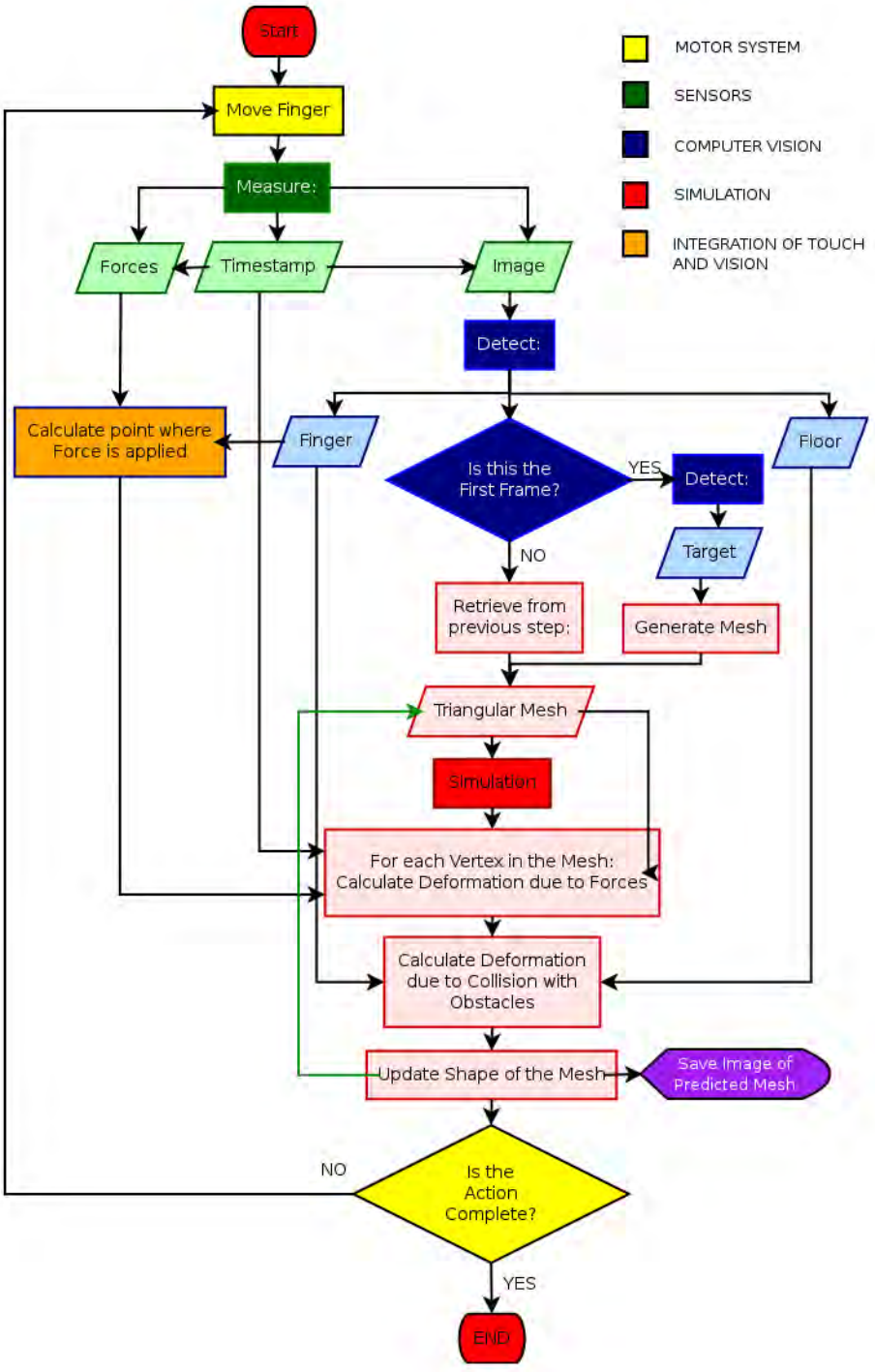

FIGURE A.1: *Flux diagram of the simulation of the deformable object, once the model has been calibrated. The green arrow indicates flow of information, but not of the control of the program.*

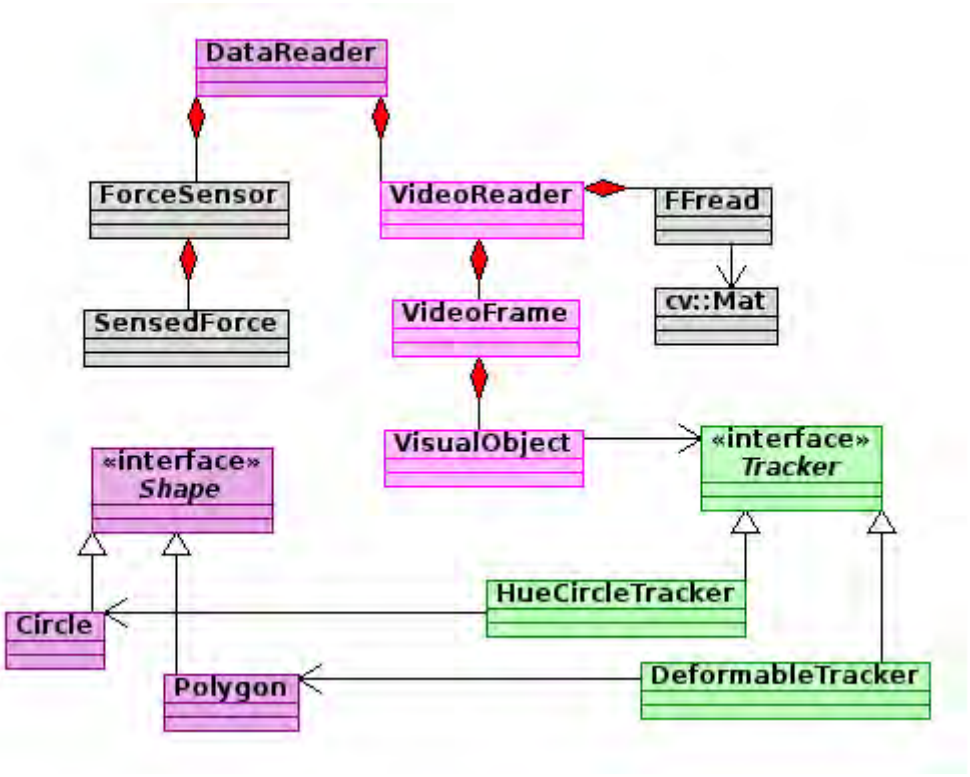

FIGURE A.2: *UML diagram of the package in charge of reading the information from the sensors and interpret it as forces and shapes.*

The second package takes the information from the first one, but reinterprets the observed shapes as obstacles and actuator. It feeds the force and time readings into the mass-spring model, alongside with the obstacles, so that the simulation can run [Figure A.3]. Each generated frame can be viewed on the screen and saved in a file.

The class called $GeneticAnalyser$ can run series of simulations, for as many generations as requested, of a given number of models, with different generated sets of parameters, evaluate them and return the results.

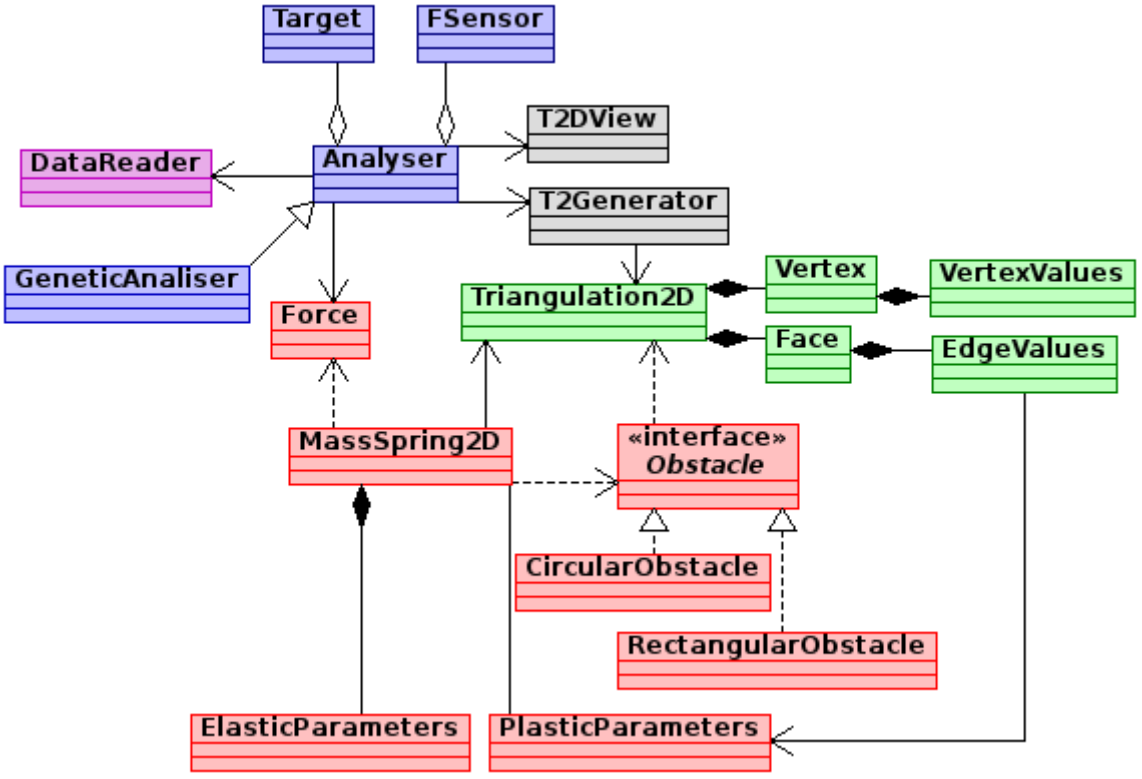

FIGURE A.3: *UML diagram of the package in charge of running simulations.*

Appendix B

Additional Experimental Data

B.1 Rougher Meshes for Predictions on the Sponge

In order to better investigate the impact of the use of graphical constraints on the predictions of the simulation, the training was repeated with meshes with bigger triangles. The results didn't vary from what was observed with the more refined mesh, as the graphs in Figures B.1, B.2 and B.3 illustrate.

B.2 Histogram of Parameters with the Finer Mesh

Initially, Figure B.4 shows the values of each parameter for the best models obtained from the 15 runs of the genetic algorithm, when no geometric constraints have been used and no force information was used during the training phase, on meshes of $20 \times 10 \times 2$ triangles. Both, the force sensed and the collision resolution routines where used to evaluated the final performance of the model.

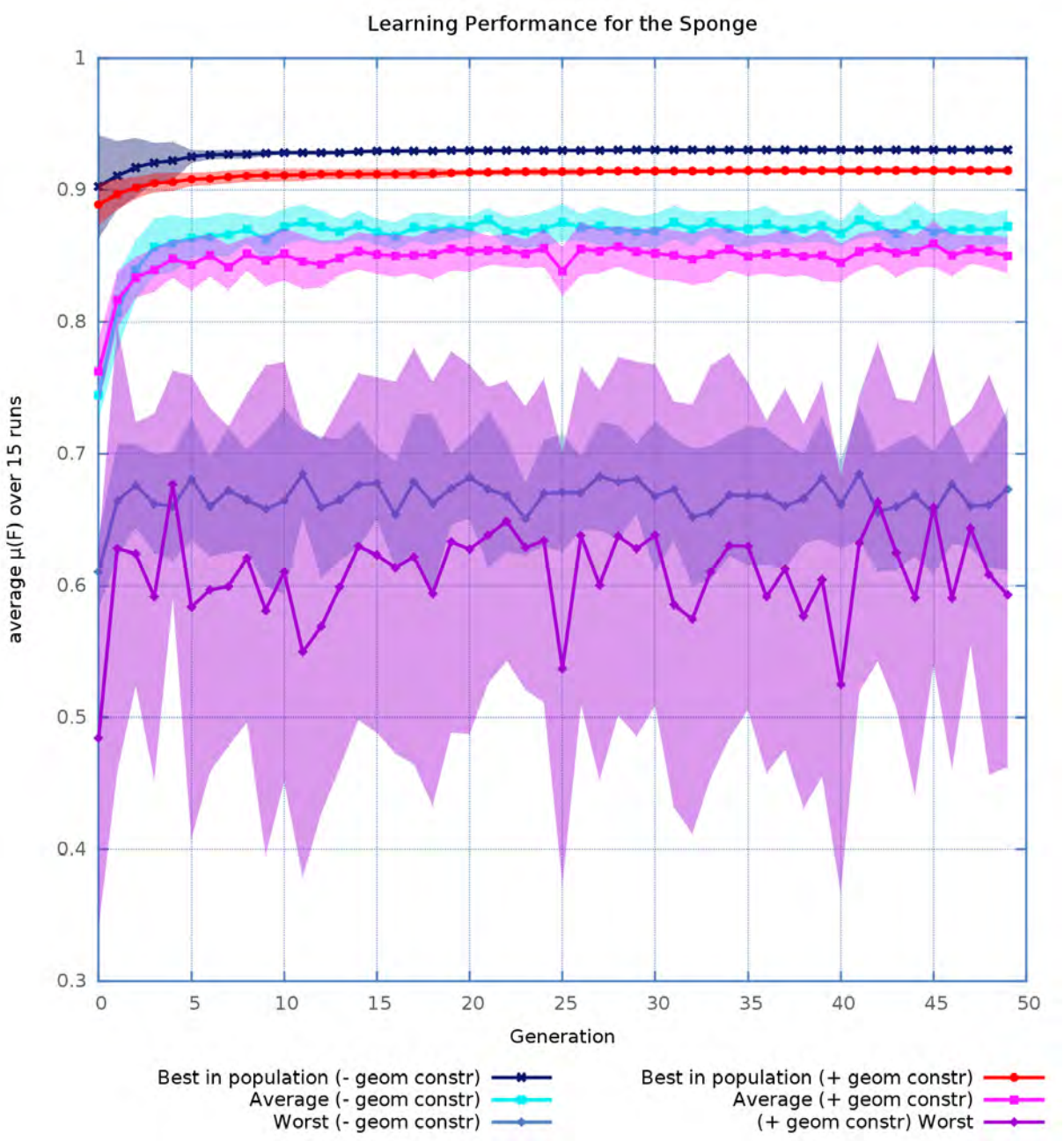

FIGURE B.1: *Average of the best, worst and average mark for each generation of the learning algorithm, for the customised integration scheme, with and without graphic constraints, when trained with the sponge. The mesh had* $10 \times 5 \times 2$ *triangles. The shaded areas correspond to the standard deviation over 15 runs of the learning algorithm.*

Simulation of the Sponge with and without Geometric Constraints

Mesh: $10 \times 5 \times 2$ triangles

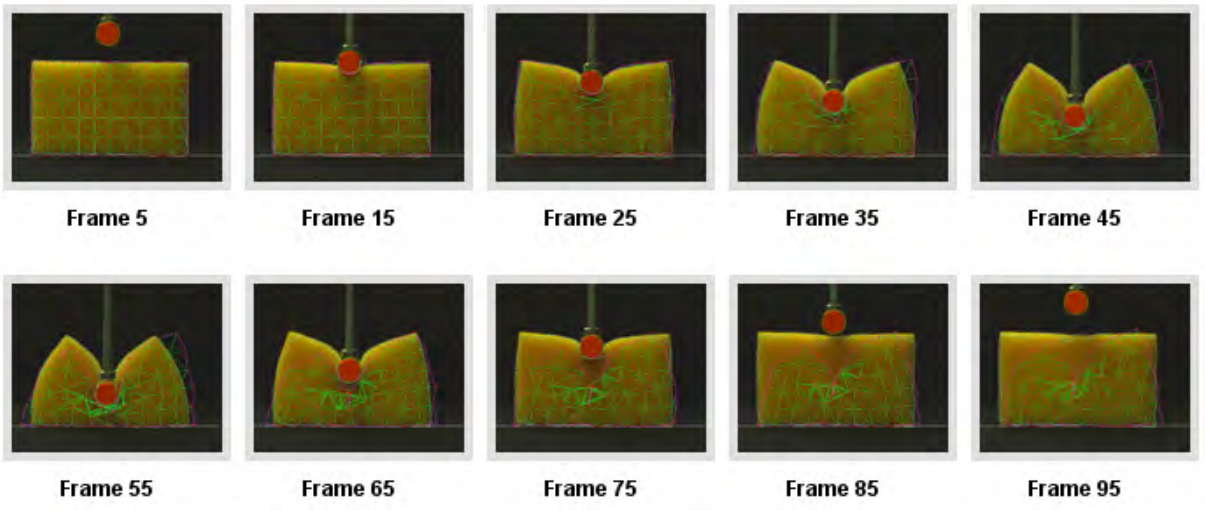

FIGURE B.2: *Without geometric constraints.* $KL = 15763.9$, $kd = 0.118432$, $KA = 27386.2$, $KFI = 37.6151$, $yield = 0.240586$, $creep = 0.860416$, $maxAlpha = 0.640253$, $maxForce = 8536.97$, $mark = 0.929229$.

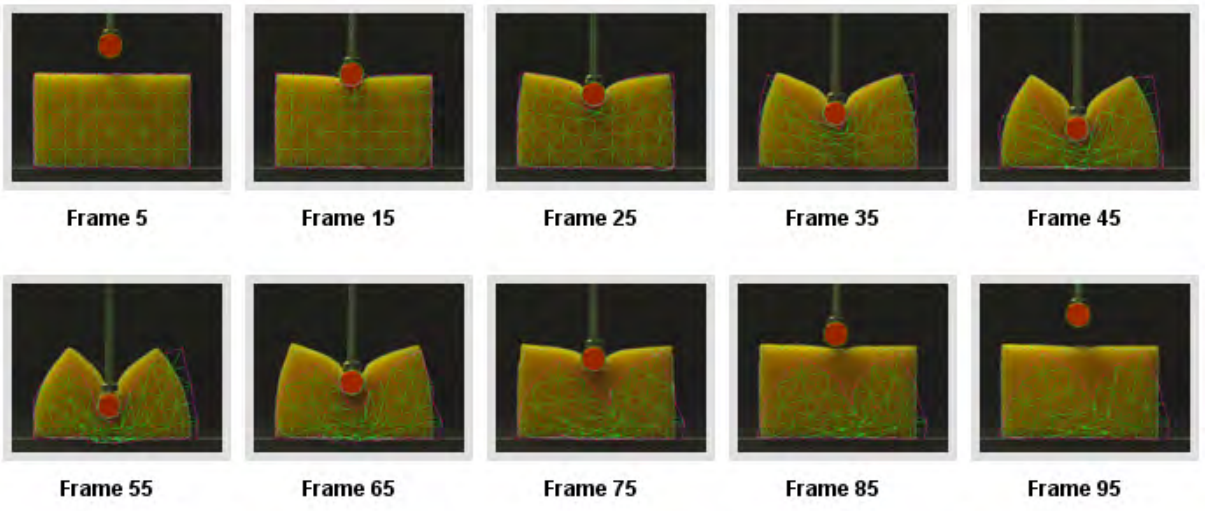

FIGURE B.3: *With geometric constraints.* $KL = 2034.45$, $kd = 0.668915$, $KA = 9203.46$, $KFI = 262.904$, $yield = 0.622656$, $creep = 0.213806$, $maxAlpha = 0.118774$, $maxForce = 1011.32$, $mark =' 0.91356$

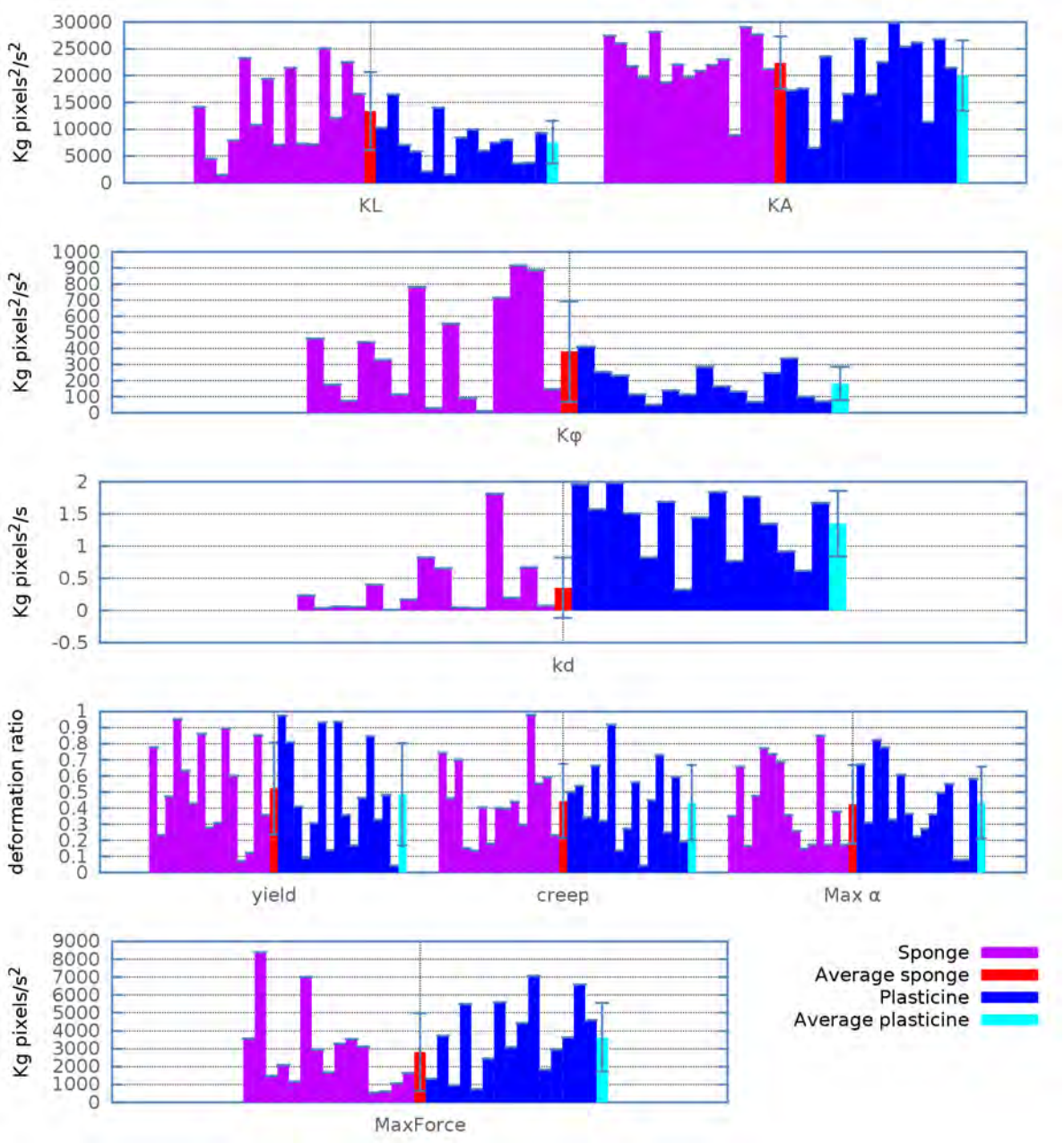

FIGURE B.4: *Values of the spring constants and plasticity parameters for the best models of 15 runs of the learning algorithm, with meshes of* $20 \times 10 \times 2$ *triangles. The last columns of each set correspond to the average value of all the previous ones and the bars, to the standard deviation.*

From this graph, the most visible difference between the sponge and the plasticine is in the amount of damping, rather than in the values of the plastic parameters. It turns out that the size of the triangles of the mesh is too small for the plasticity coefficients to play a relevant role.

Bibliography

[1] *States of Aggregation*, 3rd ed. The Great Soviet Encyclopedia. The Gale Group, Inc., 1970-1979.

[2] ARRIOLA-RIOS, V. E., DEMERY, Z. P., WYATT, J., SLOMAN, A., AND CHAPPELL, J. Salient features and snapshots in time:an interdisciplinary perspective on object representation. In *Computing Nature: Turing Centenary Perspective* (January 2013), G. Dodig-Crnkovic and R. Giovagnoli, Eds., Studies in Applied Philosophy, Epistemology and Rational Ethics (SAPERE).

[3] BÄCK, T., AND SCHWEFEL, H.-P. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation 1*, 1 (December 1993), 1–23.

[4] BALANIUK, R., AND SALISBURY, K. Dynamic simulation of deformable objects using the long elements method. In *10th Symposium On Haptic Interfaces For Virtual Environment And Teleoperator Systems, Proceedings* (2002), pp. 58–65.

[5] BARR, A. H. Superquadrics and anglepreserving transformations. *IEEE Computer Graphics and Applications 1* (1981), 11–22.

[6] BLAKE, A., BASCLE, B., ISARD, M., AND MACCORMICK, J. Statistical models of visual shape and motion. *Proc. Roy. Soc. Lond. 356* (1998), 1283–1302.

[7] BLAKE, A., AND ISARD, M. *Active Contours*. Springer, 1998.

[8] BOURGUIGNON, D., AND CANI, M.-P. Controlling anisotropy in mass-spring systems. In *11th Eurographics Workshop on Computer Animation and Simulation, EGCAS*

2000, August, 2000 (Interlaken, Suisse, 2000), N. Magnenat-Thalmann, D. Thalmann, and B. Arnaldi, Eds., Springer Computer Science, Springer-Verlag, pp. 113–123.

[9] BURION, S., CONTI, F., PETROVSKAYA, A., BAUR, C., AND KHATIB, O. Identifying physical properties of deformable objects by using particle filters. *2008 Ieee International Conference On Robotics And Automation 1-9* (2008), 1112–1117.

[10] CALVO, N. Curvas y superficies para modelado geométrico. Computación Gráfica, FICH-UNL.

[11] CATMULL, E., AND CLACK, J. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design 10*, 6 (November 1978), 350–355.

[12] CHAPPELL, J., DEMERY, Z. P., ARRIOLA-RIOS, V., AND SLOMAN, A. How to build an information gathering and processing system: Lessons from naturally and artificially intelligent systems. *Behavioural Processes 89*, 2 (February 2012), 179–186.

[13] CONTI, F., KHATIB, O., AND BAUR, C. Interactive rendering of deformable objects based on a filling sphere modeling approach. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on* (sept. 2003), vol. 3, pp. 3716 – 3721.

[14] COOTES, T., AND TAYLOR, C. Statistical models of appearance for computer vision.
http://www.isbe.man.ac.uk/ bim/Models/app_models.pdf, March 2004.

[15] COOTES, T., TAYLOR, C., COOPER, D., AND GRAHAM, J. Active shape models-their training and application. *Computer Vision and Image Understanding 61*, 1 (1995), 38 – 59.

[16] CORDERO VALLE, J. M., AND CORTS PAREJO, J. *Curvas y Superficies para Modelado Geomtrico*. Alfaomega – Ra-Ma, 2003.

[17] COSTA, I. F., AND BALANIUK, R. Lem-an approach for real time physically based soft tissue simulation. In *IEEE International Conference on Robotics and Automation, ICRA 2001. Proceedings of the* (Seoul, Korea, May 2001), vol. 3, pp. 2337–2343.

[18] CREMERS, D. Dynamical statistical shape priors for level set-based tracking. *Ieee Transactions On Pattern Analysis And Machine Intelligence 28*, 8 (Aug. 2006), 1262–1273.

[19] CRETU, A.-M., PAYEUR, P., AND PETRIU, E. M. Neural gas and growing neural gas networks for selective 3d sensing: a comparative study. *Sensors & Transducers Journal 5*, Special issue (March 2009), 119–134.

[20] CRETU, A.-M., PAYEUR, P., AND PETRIU, E. M. Soft object deformation monitoring and learning for model-based robotic hand manipulation. *IEEE Transactions on Systems, Man, and Cybernetics 42*, 3 (June 2012), 740–753.

[21] CRETU, A.-M., PETRIU, E., PAYEUR, P., AND KHALIL, F. Estimation of deformable object properties from shape and force measurements for virtualized reality applications. In *Haptic Audio-Visual Environments and Games (HAVE), 2010 IEEE International Symposium on* (oct. 2010), pp. 1 –6.

[22] DE BOOR, C. B(asic)-spline basis.
http://www.cs.unc.edu/ dm/UNC/COMP258/Papers/bsplbasic.pdf. Essay.

[23] DE BOOR, C. Splines as linear combinations of b-splines. a survey. *Approximation Theory II* (1976), 1–47.

[24] DELINGETTE, H. General object reconstruction based on simplex meshes. *International Journal of Computer Vision 32* (1999), 111–142.

[25] FAN, J., AND YAO, Q. *Nonlinear Time Series: Nonparametric and Parametric Methods*. Springer Series in Statistics. Springer, 2005.

[26] FRANK, B., SCHMEDDING, R., STACHNISS, C., TESCHNER, M., AND BURGARD, W. 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).

[27] FRANK, B., STACHNISS, C., ABDO, N., AND BURGARD, W. 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).

[28] FRITZKE, B. *Handbook of Neural Computation*. E. Fiesler, 1997, ch. Unsupervised ontogenic networks, p. C2.4.

[29] FRUGOLI, G., GALIMBERTI, A., RIZZI, C., AND BORDEGONI, M. *Robot Manipulation of Deformable Objects*. Springer London, 2000, ch. Discrete Element Approach for Non-Rigid Material Modeling, pp. 29–41.

[30] FUHRMANN, A., GRO, C., AND LUCKAS, V. Interactive animation of cloth including self collision detection. *Journal of WSCG 11*, 1 (February 2003), 141–148.

[31] GASCUEL, M.-P. An implicit formulation for precise contact modeling between flexible solids. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), SIGGRAPH '93, ACM, pp. 313–320.

[32] GIBSON, S. F. F., AND MIRTICH, B. A survey of deformable modeling in computer graphics. Tech. rep., MERL (Mitsubishi Electric Research Laboratory), 1997.

[33] GLADILIN, E., PEKAR, V., ROHR, K., AND STIEHL, H. S. A comparison between bem and fem for elastic registration of medical images. *Image Vision Comput. 24* (April 2006), 375–379.

[34] GOLDSTEIN, H. *Classical Mechanics*, 2nd ed. Addison Wesley Series in Physics. Addison Wesley, 1980.

[35] GONZLEZ-VIAS, W., AND MANCINI, H. L. *An Introduction To Materials Science*. Princeton University Press, U.S.A., 2004. Translation of: Ciencia de los Materiales.

[36] GREMINGER, M. A., AND NELSON, B. J. A deformable object tracking algorithm based on the boundary element method that is robust to occlusions and spurious edges. *Int. J. Comput. Vision 78* (June 2008), 29–45.

[37] HENRICH, D., AND WÖRN, H., Eds. *Robot Manipulation of Deformable Objects*. Springer-Verlag London, Berlin, Heidelberg, 2000.

[38] HIGGINS, R. A. *Properties of Engineering Materials*, 2nd ed. Edward Arnold, Great Britain, 1994.

[39] HIRAI, S. *Robot manipulation of deformable objects*. Springer London, 2000, ch. Energy-Based Modeling of Deformable Linear Objects, pp. 11–27.

[40] HOWARD, A., AND BEKEY, G. Intelligent learning for deformable object manipulation. *Autonomous Robots 9*, 1 (AUG 2000), 51–58. 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA 99), MONTEREY, CALIFORNIA, NOV 08-09, 1999.

[41] JAIN, A., ZHONG, Y., AND DUBUISSON-JOLLY, M. Deformable template models: A review. *SIGNAL PROCESSING 71*, 2 (DEC 1998), 109–129.

[42] JOY, K. I. Catmull-clack surfaces.
http://www.cs.ucr.edu/ vbz/resources/Catmull-Clark.pdf,
http://www.cs.unc.edu/ dm/UNC/COMP258/LECTURES/Catmull-Clark.pdf.

[43] KASS, M., WITKIN, A., AND TERZOPUOLOS, D. Snakes: Active contour models. *International Journal Of Computer Vision 1*, 4 (1988), 321–331.

[44] KATIC, D., AND VUKOBRATOVIC, M. *Intelligent Control of Robotic Systems*, vol. 25 of *Intelligent Systems, Control and Automation: Science and Engineering*. Springer, 2003.

[45] KHALIL, F. F., AND PAYEUR, P. Dexterous robotic manipulation of deformable objects with multi-sensory feedback–a review. *Robot Manipulators, Trends and Development Vukovar, Croatia: In-Tech* (2010), 587–619. A. Jimenez and B. M. Al Hadithi, Eds.

[46] KITTEL, C. *Introduction to Solid State Physics*, 8th ed. John Wiley & Sons, Inc, 2005.

[47] KUMAR, S., HAN, S., GOLDGOF, D., AND BOWYER, K. On recovering hyperquadrics from range data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 17*, 11 (November 1995), 1079–1083.

[48] LANGER, S. A., AND JR., E. R. F. Oof: An image-based finite-element analysis of material microstructures. *Computing in Science and Engineering 3* (2001), 15–23.

[49] LEVENTON, M. E., GRIMSON, W. E. L., AND FAUGERAS, O. D. Statistical shape influence in geodesic active contours. In *2000 Conference on Computer Vision and Pattern Recognition (CVPR 2000)* (2000), pp. 1316–1323.

[50] LUO, Y. H., AND NELSON, B. J. Fusing force and vision feedback for manipulating deformable objects. *Journal Of Robotic Systems 18*, 3 (2001), 103–117.

[51] MALASSIOTIS, S., AND STRINTZIS, M. G. Tracking textured deformable objects using a finite-element mesh. *Ieee Transactions On Circuits And Systems For Video Technology 8*, 6 (1998), 756–774.

[52] MARAFFI, C. *Maya Character Creation, Modeling and Animation Controls*. Stephanie Wall, New Riders Publisher, 2004.

[53] MCINERNEY, T., AND TERZOPOULOS, D. Deformable models in medical image analysis: a survey. *Med Image Anal 1*, 2 (1996), 91–108.

[54] MONTAGNAT, J., DELINGETTE, H., AND AYACHE, N. A review of deformable surfaces: topology, geometry and deformation. *Image and Vision Computing 19*, 14 (2001), 1023–1040.

[55] MORRIS, D., AND SALISBURY, K. Automatic preparation, calibration, and simulation of deformable objects. *Computer Methods In Biomechanics And Biomedical Engineering 11*, 3 (2008), 263–279.

[56] MUELLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. Meshless deformations based on shape matching. In *Proc. SIGGRAPH* (Los Angeles, USA, July 31 - Aug. 4 2005), pp. 471–478.

[57] NEALEN, A., MUELLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. Physically based deformable models in computer graphics. *COMPUTER GRAPHICS FORUM 25*, 4 (December 2006), 809–836. 18th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2005), Natal, BRAZIL, OCT 09-12, 2005.

[58] NEUMAIER, A., AND SCHNEIDER, T. Estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Transactions on Mathematical Software (TOMS) 27*, 1 (March 2001), 27–57.

[59] NEWCOMBE, R. A., AND DAVISON, A. J. Live dense reconstruction with a single moving camera. In *CVPR* (2010).

[60] NORTH, B., AND BLAKE, A. Learning dynamical models using expectation-maximisation. In *ICCV '98 Proceedings of the Sixth International Conference on Computer Vision* (1998), pp. 384–389.

[61] O'BRIEN, J. F., BARGTEIL, A. W., AND HODGINS, J. K. Graphical modeling and animation of ductile fracture. *ACM Trans. Graph. 21*, 3 (2002), 291–294.

[62] O'BRIEN, J. F., AND HODGINS, J. K. Graphical modeling and animation of brittle fracture. In *Proceedings of ACM SIGGRAPH 1999* (Aug. 1999), ACM Press/Addison-Wesley Publishing Co., pp. 137–146.

[63] O'NEIL, P. V. *Advanced Engineering Mathematics*, 4th ed. PWS Publishing Company, 1995.

[64] ONO, E. *Robot Manipulation of Deformable Objects*. Springer London, 2000, ch. 3 Automated Sewing System and Unfolding Fabric, pp. 135–158.

[65] PAI, D. K., DOEL, K. V. D., JAMES, D. L., LANG, J., LLOYD, J. E., RICHMOND, J. L., AND YAU, S. H. Scanning physical interaction behavior of 3d objects. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 87–96.

[66] PENTLAND, A., AND WILLIAMS, J. Good vibrations: Modal dynamics for graphics and animation. In *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, 215–222.* (New York, NY, USA, 1989), SIGGRAPH '89, ACM.

[67] PRESS, W., TEUKOLSKY, S., VETTERLING, W., AND FLANNERY, B. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007, ch. Section 17.6. Multistep, Multivalue, and Predictor-Corrector Methods in Integration of Ordinary Differential Equations, pp. 942–945.

[68] RADEVA, P., AMINI, A. A., AND HUANG, J. Deformable b-solids and implicit snakes for 3d localization and tracking of spamm mri data. *Computer Vision and Image Understanding 66*, 2 (May 1997), 163–178.

[69] RAVISHANKAR, S., JAIN, A., AND MITTAL, A. Multi-stage contour based detection of deformable objects. *Computer Vision - Eccv 2008, Pt I, Proceedings 5302* (2008), 483–496.

[70] RINPACHE, T. W. *Healing with Form, Energy and Light.* Snow Lion Publicatinos, 2002.

[71] ROYLANCE, D. Constitutive equations, October 2000.

[72] SAADAT, M., AND NAN, P. Industrial applications of automatic manipulation of flexible materials. *Industrial Robot 29*, 5 (2002), 434–442.

[73] SAHA, M., AND ISTO, P. Manipulation planning for deformable linear objects. *Ieee Transactions On Robotics 23*, 6 (2007), 1141–1150.

[74] SCHUMAKER, L. L. Splines and their impact. http://www.maa.org/dist-lecture/Schumaker-Jan07-short.pdf, January 2007.

[75] SEDERBERG, T. W., AND PARRY, S. R. Free-form deformation of solid geometric models. In *SIGGRAPH '86 Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (August 1986), vol. 20, pp. 151–160.

[76] SEDERBERG, T. W., ZHENG, J., BAKENOV, A., AND NASRI, A. T-splines and t-nurccs. *ACM Trans. Graph. 22*, 3 (July 2003), 477–484.

[77] SETHIAN, J. A. Tracking interfaces with level sets: An "act of violence" helps solve evolving interface problems in geometry, fluid mechanics, robotic navigation and materials sciences. *American Scientist 85*, 3 (May–June 1997), 254–263.

[78] SHENE, C. K. Cs3621 introduction to computing with geometry notes. http://www.cs.mtu.edu/ shene/COURSES/cs3621/NOTES/.

[79] SINHA, P. R., AND BAJCSY, R. K. Robotics exploration of surfaces to measure mechanical properties. In *Proceedings of the IARP Workshop on Robotics in Agriculture and the Food Industry* (Avignon, France, June 1990).

[80] SLOMAN, A. From "baby stuff" to the world of adult science: Developmental ai from a kantian viewpoint, 2009.

[81] SONG, Y., AND BAI, L. 3d modeling for deformable objects. *Articulated Motion And Deformable Objects, Proceedings 5098* (2008), 175–187.

[82] STRANG, G., AND FIX., G. J. *An analysis of the finite element method*. Englewood Cliffs ; London : Prentice-Hall, 1973.

[83] SUN, W., CETIN, M., CHAN, R., AND WILLSKY, A. S. Learning the dynamics and time-recursive boundary detection of deformable objects. *Ieee Transactions On Image Processing 17*, 11 (2008), 2186–2200.

[84] SZKELY, G., KELEMEN, A., BRECHBHLER, C., AND GERIG, G. Segmentation of 3d objects from mri volume data using constrained elastic deformations of flexible fourier surface models. In *Proceedings of the First International Conference on Computer Vision, Virtual Reality and Robotics in Medicine, CVRMed'95* (April 1995).

[85] TERZOPOULOS, D., AND FLEISCHER, K. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. *SIGGRAPH Comput. Graph. 22*, 4 (June 1988), 269–278.

[86] TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. Elastically deformable models. *SIGGRAPH Comput. Graph. 21*, 4 (Aug. 1987), 205–214.

[87] TESCHNER, M. Deformable objects - an introduction to mass-point systems.

[88] TESCHNER, M., HEIDELBERGER, B., MULLER, M., AND GROSS, M. 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.

[89] TSAI, A., JR, A. Y., III, W. W., TEMPANY, C., TUCKER, D., FAN, A., GRIMSON, W., AND WILLSKY, A. Model-based curve evolution technique for image segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (2001), pp. 463–468.

[90] UNSER, M. Splines: a perfect fit for signal and image processing. *Signal Processing Magazine, IEEE 16*, 6 (November 1999), 22–38.

[91] WADLEY, L., HODGSKISSB, T., AND GRANTB, M. Implications for complex cognition from the hafting of tools with compound adhesives in the middle stone age,

south africa. *Proceeding of the National Academy of Sciences of the United States of America 106*, 24 (June 2009), 9590–9594.

[92] WANG, H., WANG, Y., AND ESEN, H. Modeling ofdefonnable objects in haptic rendering system for virtual reality. In *Proceedings of the 2009IEEE International Conference onMechatronics andAutomation* (August 2009).

[93] WIKIPEDIA. Stress modelling (cauchy). http://en.wikipedia.org/wiki/Stress_(mechanics)#Cauchy_stress_tensor.

[94] YUILLE, A. L., HALLINAN, P. W., AND COHEN, D. S. Feature extraction from faces using deformable templates. *International Journal of Computer Vision 8*, 2 (1992), 99–111.

[95] ZEMANSKY, M. W., AND DITTMAN, R. H. *Heat and Thermodynamics*, 7th ed. McGrawHill, 1997.

[96] ZERBATO, D., GALVAN, S., AND FIORINI, P. Calibration of mass spring models for organ simulations. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on* (Oct - Nov 2007), pp. 370 – 375.

[97] ZHONG, Y., ANF GURSEL ALICI, B. S., AND SMITH, J. A cellular neural network methodology for deformable object simulation. *IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE 10*, 4 (October 2006), 749–762.

[98] ZIENKIEWICZ, O., TAYLOR, R., AND ZHU, J. *Finite Element Method - Its Basis and Fundamentals*, 6th edition ed. Elsevier, 2005.