Shape sensing of deformable objects for robot manipulation

Jose Manuel Sanchez Loza

UNIVERSITÉ CLERMONT AUVERGNE
École Doctorale Sciences pour l'Ingénieur
Institut Pascal

Thèse

Présentée par

Jose Sanchez

pour obtenir le grade de
Docteur d'Université

Shape Sensing of Deformable Objects for Robot Manipulation

Soutenue le 24 Mai 2019 devant le jury composé de :

Véronique PERDEREAU	Rapporteur	Professeur des universités, ISIR
Daniel SIDOBRE	Rapporteur	Maître de conférences - HDR, Université Paul Sabatier
Giovanni LEGNANI	Examinateur	Professeur des universités, Università di Brescia
Pablo GIL	Examinateur	Associate Professor, Universidad de Alicante
Grigore GOGU	Examinateur	Professeur des universités, SIGMA
Chedli BOUZGARROU	Encadrant	Maître de conférences - HDR, SIGMA
Juan CORRALES RAMÓN	Encadrant	Maître de conférences, SIGMA
Youcef MEZOUAR	Directeur	Professeur des universités, SIGMA

Declaration of Authorship

I, Jose Sanchez, declare that this thesis titled, 'Shape Sensing of Deformable Objects for Robot Manipulation' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Abstract

Deformable objects are ubiquitous in our daily lives. On a given day, we manipulate clothes into uncountable configurations to dress ourselves, tie the shoelaces on our shoes, pick up fruits and vegetables without damaging them for our consumption and fold receipts into our wallets. All these tasks involve manipulating deformable objects and can be performed by an able person without any trouble, however robots have yet to reach the same level of dexterity. Unlike rigid objects, where robots are now capable of handling objects with close to human performance in some tasks; deformable objects must be controlled not only to account for their pose but also their shape. This extra constraint, to control an object's shape, renders techniques used for rigid objects mainly innapplicable to deformable objects. Furthermore, the behavior of deformable objects widely differs among them, e.g. the shape of a cable and clothes are significantly affected by gravity while it might not affect the configuration of other deformable objects such as food products. Thus, different approaches have been designed for specific classes of deformable objects.

In this thesis we seek to address these shortcomings by proposing a modular approach to sense the shape of an object while it is manipulated by a robot. The modularity of the approach is inspired by a programming paradigm that has been increasingly been applied to software development in robotics and aims to achieve more general solutions by separating functionalities into components. These components can then be interchanged based on the specific task or object at hand. Our approach, thus, takes the form of a pipeline to sense the shape of deformable objects.

To validate the proposed pipeline, we implemented three different applications. Two applications focused exclusively on estimating the object's deformation using either tactile or force data, and the third application consisted in controlling the deformation of an object. An evaluation of the pipeline, performed on a set of elastic objects for all three applications, shows promising results for an approach that makes no use of visual information and hence, it could greatly be improved by the addition of this modality.

Acknowledgements

First of all I would like to thank Professor Youcef Mezouar, Professor Juan Antonio Corrales Ramón and Professor Chedli Bouzgarrou for allowing me to pursuit my PhD thesis at Institut Pascal. Their guidance, support and insights certainly improved the quality of my thesis.

I thank my office mates Rawan, Rohit and Mohamed for the daily chats, good company and helpful advice throughout my studies. To Kamal, Carlos and Miguel with whom I got the pleasure to work and, while doing so, learn much from them. Also, I am deeply grateful to Sabrina and Cyrille for helping me navigate multiple bureaucratic hurdles during my studies.

Last but not least, I would like to thank my friends Natalia, Marcel, Verónica and William for their constant encouragement and support; to my family, and especially my parents for the endless help they have given me.

⋆ ⋆ ⋆

This work has been sponsored by the French government research program "Investissements d'Avenir" through the IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25), the IMobS3 Laboratory of Excellence (ANR-10-LABX-16-01). This research was also financed by the European Union through the Regional Competitiveness and Employment program -2014-2020- (ERDF AURA region) and by the AURA region. Therefore, I would like to thank these institutions for the financial support that made this thesis possible.

Contents

List of Figures

List of Tables

Abbreviations

CBSE Component-Based Software Engineering

DLO Deformable Linear Object

EA Evolutionary Algorithm

EB Euler-Bernoulli

FEM Finite Element Method

FT Force-Torque

GM Graph Model

GNG Growing Neural Gas

GP Gaussian Process

GPR Gaussian Process Regression

HOW Histogram of Oriented Wrinkles

IMU Inertial Measurement Unit

LSTM Long Short-Term Memory

MS Mass Spring

MSM Meshless Shape Matching

NURBS Non-Uniform Rational B-Splines

PBD Position-Based Dynamics

PM Probabilistic Model

RGB-D Red Green Blue-Depth

RNN Recurrent Neural Networks

WNN Weightless Neural Network

Chapter 1

Introduction

As deformable objects are ubiquitous in many industries, automating their manipulation would have a great social impact. For instance, robots could perform tasks that are either dangerous or monotonous for workers. Examples of manipulation of deformable objects can be found in the automobile and aerospace industries, where cables and wires must be connected in order to assemble motors; in health care, where clothes are handled to dress disabled people; and in the food industry where meat and produce have to be processed with care. Therefore, plenty of robotic applications have been recently proposed to improve the capability of robots to manipulate deformable objects. For instance, robotic solutions that attempt to automate the manufacture of motors by manipulating cables can be found in [6–8]; and approaches concerned with using robots to perform clothing assistance have been proposed in [9, 10]. As an example of food handling, some roboticists have focused on harvesting bell peppers in an autonomous manner [11, 12].

1.1 What are deformable objects?

Deformable objects are usually considered as those objects that are able to change their shape. However, this definition is too loose. Therefore, in this thesis, we will refer to deformable objects as objects that retain their topology, i.e. their shape can be altered through stretching and twisting but not by being torn or cut. This definition disqualifies objects such as liquids and granular materials (e.g. sand, grains, etc.) as

they can separate and thus alter their topology. Furthermore, we consider as deformable the objects that either have no compression strength, have a large strain or present a large displacement. Examples of objects having no compression strength are cables and clothes; whereas sponges are objects that have a large strain. For an introductory description on strain, the reader is referred to Appendix A.

Although the definition for deformable objects stated above might exclude plenty of objects, there are still significant differences between object such as cables and food, clothes and sponges; and therefore we propose a classification to group objects into categories that capture their similarities. To this end, our proposed classification combines geometric and physical properties. Geometrically, deformable objects can be divided based on the number of significant dimensions that describe their shape. Thus, *beam* objects are described by one dimension, *shell* objects by two dimensions and *volumetric* objects by three dimensions. By also considering their physical properties, we propose the following categories:

1. **Linear:** Beam-like objects that either have no compression strength such as cables, strings and ropes; or they have a large strain such as elastic tubes and beams. These objects are commonly referred in the robotics community as deformable linear objects, or DLO's for short.

2. **Cloth-like:** Shell-like objects not possessing any compression strength. Shirts, pants, towels and fabric sheets are examples of this type of objects.

3. **Planar:** Shell-like objects that present a large strain, or a large displacement, such as paper, cards and foam sheets. Also, thin-shell objects such as empty plastic bottles and hollow rubber balls are considered in this category.

4. **Solid:** Volumetric objects such as a sponges, plush toys and food products fall in this object category.

Figure 1.1 depicts our proposed classification and Figure 1.2 shows examples of deformable objects based on our classification.

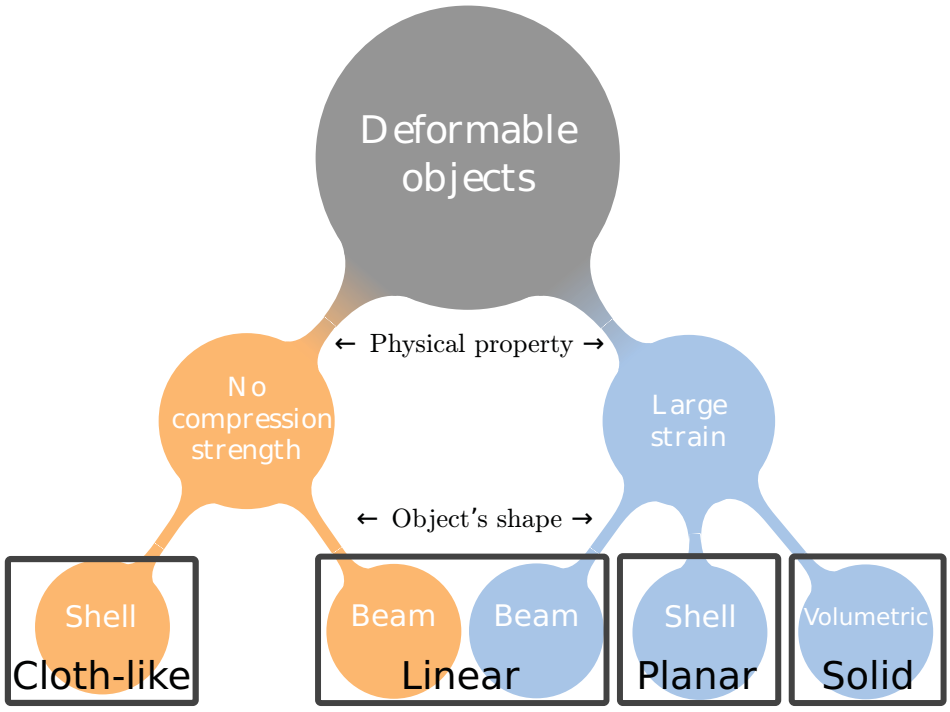


FIGURE 1.1: Proposed classification of deformable objects.

(A) Rope and cable. (B) Towel and pants. (C) Ball and bottle. (D) Sponges and meat.

FIGURE 1.2: Examples of linear, cloth-like, planar and solid deformable objects.

1.2 Difference with rigid objects

Although robotic manipulation of rigid objects has been studied for several decades now [13–16], the algorithms and strategies developed for rigid objects are not always transferable to deformable objects. For instance, force and form closure, two widely used conditions in robot grasping, are not directly applicable to deformable objects. As form closure consists in applying kinematic constraints on an object such that the object cannot perform any relative motion [17], this clearly fails with deformable objects since they have infinite degrees of freedom [18]. To apply force closure, which considers a set of contact points such that contact forces can balance an arbitrary external wrench [19], to deformable objects would entail to continuously recalculate the necessary forces as the object changes its shape due to the contact forces deforming the object [20].

Also, as noted in [21], manipulation of rigid objects focuses mostly on controlling the grasped object's pose. However, manipulating deformable objects requires also controlling the object's shape. This extra requirement, in addition to the inapplicability of the methods developed for manipulation of rigid objects, has led to a diverse set of approaches to manipulate deformable objects.

1.3 Scope of this thesis

The main purpose of this thesis is to control the shape of an object, a task sometimes referred to as *shape servoing*. In order to control an object's shape a feedback signal is required. Ideally, this feedback signal would be the actual shape of the object as it deforms. However, measuring the three-dimensional shape of an object while it deforms remains an open issue. Therefore, we first have focused on estimating the shape of an object as a preliminary goal necessary to perform the subsequent shape control.

Although a "shape sensor" is not available so far, multiple attempts are currently being pursued to estimate a deformable object's shape. Most of these approaches use visual sensing and rely either on the use physical models or on so-called model-free methods (e.g. by attaching fiducial markers on the object). Other methods have avoided the use of a feedback signal by performing the shape control in an open-loop manner and are thus not capable of guaranteeing a successful outcome.

Since approaches based on vision are affected by problems such as occlusions (which occur rather often when manipulating objects), specularity (e.g. shiny objects), objects similar to their background or objects lacking texture; the addition of complementary modalities could greatly improve the performance of these approaches. One such modality is that of touch, e.g. force and tactile sensing. To address these shortcomings, in this thesis we investigate the viability of an approach that estimates the shape of a deformable object using tactile and force sensing.

Furthermore, the approaches currently found in the literature tend to be designed for specific tasks and objects. To avoid this lack of generality, we propose a modular approach that allows for variation of different tasks/objects. The approach consists of modules connected in a pipeline as shown in Figure 1.3 and modularity is achieved via the replacement of modules.

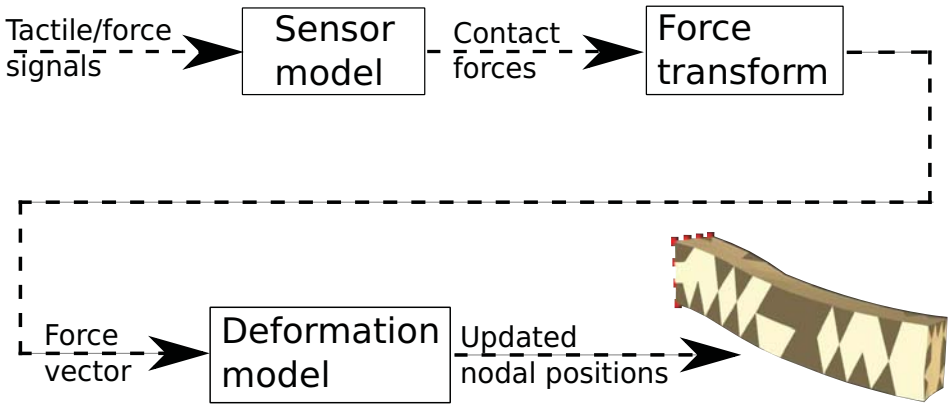


FIGURE 1.3: Proposed pipeline to sense the shape of deformable objects using haptic data.

1.4 Contributions

The contributions of this thesis are summarized below:

1.4.1 Survey and systematization of the state of the art

Works focusing on the manipulation and sensing of deformable objects were reviewed and organized based on the proposed classification shown in Figure 1.1. This organization of the state of the art allowed us to clearly identify potential solutions as well as gaps in current approaches. One such shortcoming is for instance the lack of haptic sensing, e.g. using force and tactile signals, which could greatly benefit robotic perception during the manipulation of deformable objects. Furthermore, current approaches are typically designed for specific objects with well defined tasks (e.g. tying a knot, folding clothes). Thus, developing generic solutions would be highly desirable.

1.4.2 Sensor models

As highlighted above, the addition of haptic sensing to a robot's perception system would improve the shape estimation of an object while is being deformed. To extract haptic information, tactile and force sensors are usually used, however, in order to map the sensors' output to contact information (i.e. magnitude and location of contact forces) accurate sensor models are required. In this thesis, sensor models for an advanced tactile sensor and a commercial force-torque sensor were developed.

1.4.3 Shape sensing pipeline

Due to the variability of tasks and deformable objects, approaches found in the literature suffer a lack of generality. Thus, it is imperative that we develop solutions that are either general or can be modified at run time to handle different objects and tasks. To address this issue we propose a modular pipeline that operates on contact information regardless of the input sensor, provided a sensor model is used to interface with the pipeline. The modularity of the pipeline allows to change the input signals according to the particular task being executed or the manipulated object. For instance, tactile sensing might be more appropriate for performing in-hand manipulation, while force sensing could be used when manipulating large objects. A diagram of the proposed pipeline can be seen in Figure 1.3.

1.4.4 Shape controller architecture

Finally, as the main goal of this thesis is deformation control, we propose a control architecture that uses the developed shape sensing pipeline to deform an object such that it reaches a desired target. Although, preliminary results show great potential, considerable work remains to be done, specially considering how heavily underactuated the problem of shape control is.

1.5 Publications

The results from this thesis have been published in the following articles:

Journal articles

- **Sanchez, Jose**, Juan-Antonio Corrales Ramón, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey", *The International Journal of Robotics Research*, vol. 37, issue 7, pp. 688-716. 2018.

Conference articles

- **Sanchez, Jose**, Carlos M. Mateo, Juan-Antonio Corrales Ramón, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. "Online Shape Tracking based on Tactile Sensing and Deformation Modeling for Robot Manipulation", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Spain, October 1-5, 2018.

- Mohy El Dine*, Kamal, **Jose Sanchez***, Juan-Antonio Corrales Ramón, Youcef Mezouar and Jean-Christophe Fauroux. "Force-Torque Sensor Disturbance Observer using Deep Learning", *International Symposium on Experimental Robotics* Buenos Aires, Argentina, November 5-8, 2018.

1.6 Outline of the thesis

Following this introduction we discuss the state of the art for manipulating and sensing deformable objects in Chapter 2. Chapter 3 describes the sensor models proposed for the tactile and force-torque sensors. The shape sensing pipeline is described in Chapter 4 and its applications, including the deformation controller, are presented in Chapter 4.5. The contributions and limitations of our work, as well as potential research lines and concluding remarks are outlined in Chapter 5.

*Authors contributed equally.

Chapter 2

State of the art

The context of this thesis follows our group research line on robotic manipulation of deformation objects. Previous works have focused on tasks such as meat cutting and grasping of deformable objects. The former was addressed by Nabil et al., where the deformation of the meat was modeled using a mass-spring model and a multi-robot system, consisting of an arm moving a vision system, a cutting arm and an arm holding the meat; was used to separate meat muscles [22]. Their model was then extended by Zaidi et al. to grasp a soft object using a multi-fingered robot hand [23]. Here, the object was also represented as a tetrahedral mesh but with its nodes connected by non-linear springs. They further proposed a contact model to compute the interaction forces between the fingers and the object based on the fingers' positions and velocities, which were used by the deformation model in order to verify the stability of the grasp to lift the object.

Due to the shear amount of works that deal with deformable objects in robotics, we limit the scope of this chapter by focusing on the most relevant approaches to this thesis. To this end, we review works that either control or sense the shape of solid objects (see Figure 1.1), or those approaches that could be extended to perform these tasks. For a further study of the state of the art not limited to these tasks, or type of object, the reader is referred to recent surveys [24, 25].

2.1 Deformation sensing

Sensing of deformable objects can refer to different tasks such as:

- **Parameter estimation:** Physically realistic deformation models require precise knowledge of elasticity parameters such as Young's modulus, Poisson's ratio, Lamé coefficients , etc. Other deformation models, not necessarily reliant on these quantities, might require knowledge of ad-hoc parameters. Thus, a critical preliminary step to estimate deformation is the accurate identification of these material parameters.
- **2D sensing:** Approaches might be interested in following or estimating the contour or surface of an object while it deforms. Note that although the estimation itself is two-dimensional, its representation might be three-dimensional (e.g. bending a sheet of paper).
- **3D sensing:** Refers to the ability to estimate or track the global shape of a deformable object.

As noted above, some approaches require the use of deformation models. These models can be mechanically motivated (continuum or discrete-based), empirically obtained through sensor information (data-driven), or a combination of both (hybrid). For a brief description of the mechanical-based models and how they compare against each other in terms of accuracy and complexity the reader is referred to Appendix A.3. In this section, the sensing approaches are organized based on the type of model they use to represent the shape of a deformable object. A summary of the works covered in this section is presented in Table 2.1.

2.1.1 Mechanical-based models

The sensing approaches reviewed in this section rely on mechanical-based models that can either have a continuous or a discrete representation. The reviewed works here, then, rely on constitutive models[1] such as the Euler-Bernoulli (EB) beam theory model or the finite element method (FEM[2]) for continuous representations; and mass-spring

[1] In these models the elasticity parameters are derived from the material properties of the object.

[2] A description of how the FEM method is used for simulating deformation is covered in more detail in Section 4.3.1.

TABLE 2.1: Classification of the sensing approaches.

	Task			Model			Sensing modality			
	Sensing 2D	**Sensing 3D**	**Parameter estimation**	**Data-driven**	**Hybrid**	**Mech. model**	**Vision 2D**	**Vision 3D**	**Haptic**	**Proprio-ceptive**
[26]			✓			FEM		✓	✓	
[27]		✓	✓			EB		✓		
[28]	✓			GNG			✓		✓	✓
[29]		✓			MS + PM			✓		
[30]		✓			MS			✓		
[31]		✓	✓			EB + NURBS		✓		
[32]	✓			WNN			✓			
[33]		✓				FEM		✓		
[34]	✓				PBD		✓			
[35]	✓				PBD + GP			✓	✓	
[36]		✓				FEM		✓		
[37]		✓		GNG				✓	✓	
[38]	✓		✓		MS + EA		✓		✓	
[39]	✓		✓		PBD		✓			
[40]		✓				FEM		✓		
[41]	✓			GPR				✓		
[42]	✓			GM				✓		

Sensing modalities can refer to (examples): monocular cameras (2D vision), RGB-D cameras (3D vision), force and tactile sensors (haptic) or the configuration of a manipulator (proprioceptive).

(MS) models and position-based dynamics (PBD) for discrete representations. A survey describing these mechanical models can be found in [5].

2.1.1.1 Continuum-based

As previously noted, the mechanical-based models require knowledge of parameters to properly estimate deformations. To this end, Frank et al. proposed an approach to estimate the Young's modulus and the Poisson's ratio of non-rigid objects by combining force and vision sensing with an FEM model [26]. A robot manipulator, with a force-torque sensor, probed different objects to deform them and then compare the observed deformation (obtained by the vision system) with a deformation from the FEM model. The difference between the simulated and observed deformation served as an error function to optimize the parameters' values. Instead of using an FEM model, Fugl et al. used the Euler-Bernoulli beam model to estimate the Young's modulus of an object deforming under gravity [27]. Similar to [26], this approach made use of RGB-D data to measure the object deformation and afterwards minimized the error between the sensed and the simulated deformation to estimate the Young's modulus.

Although FEM models tend to be computationally expensive, recently, it was shown that a real time estimation of an object's shape can be achieved using a FEM model. In a series of papers, Petit et al., assume a known mesh of the object is available and couple the output of an RGB-D sensor with a co-rotational FEM model to track the shape of the object as it deforms [33]. In [36], they added force sensing to estimate the necessary parameters (i.e. Young's modulus and Poisson's ratio) in a preliminary phase. Their work was further extended in [40] to incorporate multiple deformable objects and also deal with collisions. Their approach produced accurate results without excessive computational cost, such that real time estimation was achieved.

2.1.1.2 Discrete-based

The following approaches represent the object as a set of points. If the points, i.e. nodes, are connected by springs they are referred as mass-spring models, while position-based dynamics models indicate that the representation does not require connectivity between nodes.

Leizea et al. in [30], proposed an approach to track deformations using the output of an RGB-D sensor in combination with a mass-spring model. In order to create the mass-spring model, they use a voxel structure[3] created from the object's bounding box, where the vertices in a voxel represented the nodes of the mesh. To estimate the deformation, the displacement of each node (obtained by the vision system) is used to compute forces which are then integrated to produce the new nodal positions.

Güler et al., using a type of PBD simulation called meshless shape matching (MSM), estimated the deformation of an object's surface as it was pushed downwards by a probe [34]. Since MSM only requires position information, they used an optical flow algorithm to compute the position of a set of points that corresponded with the simulated points. In order to tune their simulation they also estimated a "deformability" parameter to control the behavior of the simulated deformation. This approach was later extended in [39] by including an FEM model that served as ground truth to better estimate the deformability parameter.

2.1.2 Data-driven models

These approaches rely on data-driven techniques applied to data gathered by sensors , such as images or point clouds, to estimate an object's deformation.

One of the first works to use machine learning to estimate the deformation of an object was proposed by Cretu et al. [28]. Specifically, they combined a feedforward neural network with a growing neural gas (GNG) network to track the contour of an object while it was deformed by a robotic hand. The feedfoward network was used to map position and force information, from the fingers of the robot hand, to a set of two-dimensional points representing the contour of the object. The GNG network was grown, point by point, until a sufficient number of points could describe the contour appropriately. In [32], Staffa et al., applied neural networks as well to track not only the contour of an object but their surface. They used Weightless Neural Network (WNNs) on a video input to train different classifiers to detect whether a pixel was part of the background or the foreground (i.e. the object to be tracked).

[3] A voxel structure is a three-dimensional grid composed of cells called voxels.

As the previous works relied on monocular cameras that remained at a fixed location, they could only track either the contour or surface of the object. To overcome this limitation, recent approaches have instead used RGB-D sensors to track deformations occurring in a three-dimensional space. Hu et al. estimated the shape of different objects using Gaussian process regression (GPR) to model the object's deformation [41]. Also using RGB-D data, Han et al. estimated the shape of a deforming object by simultaneously tracking and reconstructing the object [42]. Here, the object is first represented as a voxel structure created using a truncated signed distance function computed from the object to the camera. Then, this representation was deformed using a graph model (GM) that considers both the difference between the initial representation and inconsistent transformations in neighboring vertices. Finally, the shape of the object is reconstructed to include the color and texture information.

Recently, an approach proposed by Tawbe and Cretu, using a probe with an attached force sensor and a moving RGB-D sensor, was able to predict the deformation of three-dimensional objects [37]. The approach consists in first generating a mesh using the output of the RGB-D sensor while probing the object to deform it at a specific location. Then, once the mesh is created, the number of points representing the object is reduced based on clusters that divide the object based on the amount of deformation (e.g. the first cluster was the closest to the deformation and the last one was the farthest from the deformation). To predict the deformation they trained a feedforward network for each cluster where the input consists of a three-dimensional force, the angle of the probe and its contact location.

2.1.3 Hybrid models

The following approaches combine discrete-based models with data-driven approaches such as Gaussian processes (GP) and probabilistic models (PM) to sense the shape of an object while it is undergoing deformation.

The approach proposed in [27], was later extended in [31] by using non-uniform rational B-splines (NURBS) as a representation that described the surface of the deforming object, which allowed them to track the object's shape. Arriola-Rios and Wyatt proposed a system that predicts not only the deformations of elastic and plastic objects, but also the parameters of a spring-mass model that represents the object [38]. They first

obtained the model parameters by using an evolutionary algorithm (EA) , and to predict the shape of the object they trained a predictor offline using data from a force sensor (pushing the object) and a monocular camera that tracked the contour of the object. Although their system was able to generalize to different types of deformation (i.e. plastic and elastic), only the deformation on one side of the object was predicted as the vision system remained at a fixed location. In [35], by adding touch sensing, they were able to characterize the deformability of surfaces. Their approach combined RGB-D data with tactile data, obtained by physically interacting with a surface, using Gaussian processes to create a map of how the surface deforms at different locations.

Unlike the previous approaches that tracked deformation on surfaces, Schulman et al. succesfully estimated the deformation of an object in three dimension. Here, the object was described using also a mass-spring model but updating the positions of the vertices via a probabilistic model [29]. In order to update the nodal positions of the mesh, they used point clouds obtained from an RGB-D sensor as observations and, since not all nodes were visible to the RGB-D sensor, they relied on a physics engine simulator to estimate the positions of the non-visible nodes.

2.2 Deformation control

So far, most approaches that deal with manipulating the shape of a deformable object do not rely on a physical-based model of the object they control. Instead, sensing, and in particular vision, has been used to extract feedback signals that are regulated to manipulate the object. Also, not relying on a model, some approaches have used machine learning (e.g. data-driven approaches) to design controllers that manipulate a deformable object. Nevertheless, a few recent approaches have proposed using physical-based models in order to control the shape of an object. This section will first review the approaches based on sensing, followed by data-driven approaches and conclude with the model-based approaches. Table 2.2 shows a classification of the reviewed methods for shape control.

TABLE 2.2: Classification of the shape control approaches.

	Task		Approach			Sensing modality		
	2D	**3D**	**Sensor-based**	**Data-driven**	**Mech. model**	**Vision 2D**	**Vision 3D**	**Proprio-ceptive**
[43]		✓		†				
[44]		✓	VS			✓		
[45]		✓	VS			✓		
[46]		✓	VS				✓	
[47]	✓			HF		✓		
[48]	✓		VS			✓		
[49]		✓			FEM			✓
[50]		✓			FEM			✓
[51]		✓		†				✓
[52]	✓		VS			✓		
[53]					MS			
[54]	✓		VS			✓		
[41]				VS + GPR			✓	

The methods used for the model-free approaches can be classified as follows. **VS:** Visual servoing, **HF:** Histogram features, **GPR:** Gaussian process regression, †: These approaches assume the configuration of the object (e.g. as a set of points) was available for the controller.

2.2.1 Sensor-based

Within the approaches that do not require a mechanical model, different representations have been used as a feedback signal. For instance, the object might be represented as a set of points that are sensed using an RGB-D sensor by covering the object with fiducial markers. These points are then used to extract features that describe the object's deformation.

Navarro-Alarcón et al., in a series of works, proposed a way to control the configuration of a deformable object using visual servoing. Here, the configuration of the object is described using *deformation feature* vectors, based on a set of points tracked using markers. The proposed deformation vectors were the following:

1. **Point-based deformation:** one point on the object is driven to a desired target point.
2. **Distance-based deformation:** one point, or the midpoint between two points, is moved a specified distance.
3. **Angle-based deformation:** rotates a line between two points by a desired angle.
4. **Curvature-based deformation:** an arc of three points on the object can be manipulated to achieve a specific curvature.

These approaches also rely on a deformation Jacobian, which here refers to a matrix mapping the motion of the grippers to the deformation of the object. In [44], the deformation Jacobian is estimated using the Broyden method, which computes the Jacobian once at the beginning and then approximates it at each iteration using the previous Jacobian and the changes of the feature vectors and the end-effector's pose; and, in [45], they proposed a new estimation that used views from multiple cameras. However, both of these approaches were limited as they control the deformation features on a plane, namely in the image space. This was later addressed in [46], by using stereo-vision to track the points in 3D and subsequently define the deformation feature vectors also in 3D. A similar approach proposed by Alambeigi et al. extended its application to heterogeneous objects while being robust to disturbances, e.g. the objects were filled with water beads (heterogenous) and then cut (disturbance) while the controller was

running [54]. To achieve this, instead of relying only on a deformation Jacobian, they combined it with an image Jacobian to consider both the deformation behavior of the object as well as the feedback points obtained by the vision system.

Although the shape of the object can be indirectly controlled by these approaches, e.g. by controlling a few points, they are inadequate to perform tasks that require a true shape control. To overcome this shortcoming, Navarro-Alarcón and Liu, proposed the use of truncated Fourier series to describe the contour of an object and used the Fourier coefficients as feedback signals to control the object's shape [48].

A common assumption the previous approaches made was that, at the beginning of manipulating the object, the robot is grasping the object. To tackle this constraint, Wang et al. proposed a vision-based controller to first make contact with the object and then deform it into a desired configuration without the necessity to have independent controllers for reaching and shaping [52].

2.2.2 Data-driven

Since a deformable object configuration is extremely high-dimensional, researches have instead proposed low-dimensional representations in order to apply machine learning techniques to develop controllers for the manipulation of deformable objects. For instance, Jia et al. proposed a feature called histogram of oriented wrinkles (HOW) that was used by a dual-arm robot to manipulate different deformable objects [47]. The HOW feature was computed using Gabor filters that are convenient for extracting shadow and shape variations (e.g. in the form of wrinkles). As a control law, they computed the velocities of the end-effectors based on the difference between the HOW features of a current and a desired image. To map the velocities to this difference, they approximated the interaction function as a visual feedback dictionary. This dictionary was built offline by pairing the robot configuration with the HOW features at every time step. Then, at runtime, the difference between HOW features was used to retrieve the appropriate command, e.g. the end-effectors' velocities.

Also relying on data to compute a deformation model that maps custom features to the velocities of a robot's end-effectors, the approach described in [41] learns a model online using a modified Gaussian process regression (GPR). The modification of the

GPR consisted in removing uninformative data to achieve a faster computation that allowed the model to be learned online. However, as the model is learned online, an exploration phase at the beginning of the manipulation is necessary to obtain the relevant information. In an experimental evaluation, the approach was shown to outperform, in terms of success and speed, the approaches described in [44, 46] since it used a nonlinear model instead of a linear one. It is worth noting that in this approach the input data were pointclouds (obtained by an RGB-D sensor) rather than two-dimensional images as in the previous approaches.

Other approaches can deal with the high dimensionality of the object, but assuming the configuration of the object is known (e.g. provided by a simulator as set of points). In [43], Berenson proposed an approach to move the object into a desired configuration, using a pair of floating robot's grippers, where the object state was assumed to be known at all times. In this approach, similar to the works by Navarro-Alarcón et al., Berenson computes a deformation Jacobian. Although here the computation is based on the assumption that the position of the points farther away from where the gripper is grasping the object are affected less by the gripper's motion. The deformation Jacobian is then used to compute a set of gripper velocities that minimize the error between desired and current object configuration. Additionally, constraints were added to prevent over-stretching and avoid collisions. This work was recently formulated as an optimization problem in [51] and it was further extended to consider the direction in which the object is pulled, thus improving the performance of the controller.

2.2.3 Mechanical model-based

Usually, physically realistic models, such as the finite element method, are avoided to model deformations of objects since they are computational expensive and thus not appropriate to perform real-time control. However, due to recent improvements in computing power these models have begun to be applied in the field of robotics in control shaping tasks. For instance, Dünser et al. represented an object, grasped by a dual-arm robot, using a FEM model and applied optimization to find the joint angles of the robot that moved the object into a desired shape [50]. To estimate the shape of the object, which was modeled by a neo-Hookean material model, they first computed the energy caused by the deformation, gravity and the force applied by the grippers, and

then searched for the object shape that minimized the energy. Once the shape had been estimated, they optimized an objective function to find the set of joint positions that were closer to reaching the target shape for the object. Since this optimization required the computation of an expensive gradient, they applied sensitivity analysis to reduce the computational burden.

Also using a FEM model to describe an object, but instead of relying on optimization, Ficuciello et al. inverted the model to find the appropriate control commands to perform shape control using a dexterous hand [49]. The fingertips of the robotic hand were considered as end-effectors actuating the motion of the object and the contact forces were regarded as actuators. In order to control these "actuators" they defined the contact forces as Lagrangian multipliers that moved the fingertips such that a desired shape could be reached. Although both of these approaches are able to control the shape of an object in real-time, they perform open-loop control and thus, there is no feedback on the actual shape of the object.

So far, the approaches reviewed have assumed the objects deform in a purely elastic manner, meaning that once the deformation force is removed they return to their rest shape (e.g. the undeformed shape). However, in reality, most objects tend to deform elastically only in a partial manner. That is, that once the deformation force is removed the object does not recover its original shape (e.g. plastic deformation[4]). Those type of objects are refer to as rheological objects. In order to model this mixed behavior, Cocuzza and Tan proposed to model a deformable object as a chain of masses connected by a three-element model[5]. In their approach, the objective is to shape fondant icing on top of a cake [53]. Furthermore, they identify the model parameters (e.g. the values of the spring and the two dampers) by first performing a tensile test of the fondant icing.

2.3 Summary

This section covered the latest approaches on sensing and control of deformable objects with a focus on solid type objects. This consideration was taken to limit the review to the works most relevant to this thesis. For historical references, however, the reader

[4]The different type of deformations are described in Appendix A.1.

[5]A three-element model consists of a spring connected in parallel with a damper and then connected in series with another damper. This model is widely used to represent rheological materials.

is referred to classic surveys focusing on industrial applications [55, 56]. Other surveys have focused on specific tasks, such as planning with deformable objects [57], or on a specific type of object, such as cloth-like objects [58]. Recent works covering a broad set of both tasks and objects can be found in [24, 25].

Sensing approaches were classified based on their reliance on models, where physical-based models are used when accuracy is preferred and geometric models or model-free methods when speed is crucial. Model-free approaches might directly used sensor data or require a preliminary step to collect data in order to apply machine learning techniques. The appropriate model, or lack thereof, for a particular application must be selected by considering the trade-off between accuracy and performance.

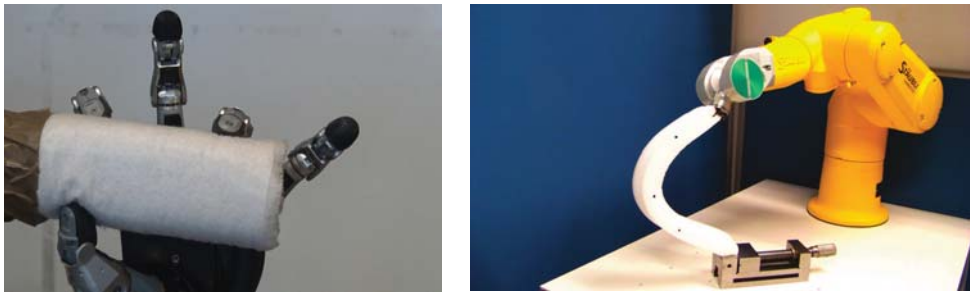
Regarding control approaches, it is evident that a feedback signal is necessary in order to control the shape of a deformable object, however, it is less clear what such signal should look like. For instance, a signal such as a mesh can better describe an object's shape by increasing the number of vertices that comprise the mesh. However, using such a representation as a feedback signal, has the drawback of making the control problem a heavily underactuated one. On the other spectrum, one could choose a low-dimensional representation such as a pair of points with the shortcoming of not accurately representing the actual shape of the object. Thus, similar to the sensing approaches, the definition of a deformation feature (e.g. a feedback signal) is entirely dependent on the task requirements.

For this reason, we propose a modular pipeline (see Chapter 4) with the potential to modify the feedback signal at runtime to fulfill a task appropriately (e.g. sacrificing accuracy for speed or vice versa). Of course, as different feedback signals might require different controllers, the pipeline must also allow the replacing of controllers in an online manner. Furthermore, as demonstrated by the review of the state of the art, and summarized in Table 2.1, most approaches so far have relied on vision systems which are affected by occlusions and are sensitive to lighting conditions. Thus, we develop sensor models, presented in Chapter 3, based on tactile and force sensing to overcome these issues that commonly occur while manipulating objects.

Chapter 3

Sensor modeling

As previously noted in Section 1.3, the proposed pipeline requires sensors models that are able to output contact forces (see Figure 1.3) in order to be integrated with the rest of the pipeline's modules. To this end, we develop sensor models to compute information such as the magnitude and location of contacts. The type of sensors needed depends on the type of manipulation task the robot should execute. For instance, if the robot requires fine and local motions (e.g. in-hand manipulation using a robot hand) tactile sensing is preferred, while if the robot must execute large motions using a robot arm, such as substantially deforming an object, force sensing might be more adequate. Figure 3.1 shows example applications illustrating which modality is preferred depending on the task and the size of the object.



(A) In-hand manipulation tasks, such as the one described in [49], could benefit from information obtained by tactile sensors.

(B) Manipulation of large objects, as proposed in [45], requires arm manipulation where forces might be captured by force sensors.

FIGURE 3.1: Example applications for different sensing modalities.

We begin this chapter by introducing the sensors used in our work and analyzing their characteristics in Section 3.1. It will be shown that their unprocessed output cannot be directly applied to our pipeline due to either having a complex fabrication, as

shown in Section 3.1.1 or by producing undesired non-contact forces, as described in Section 3.1.2. To overcome these issues we rely on machine learning techniques to learn the correlation between a sensor's output and a desired information, e.g. a three-dimensional force. Specifically, we apply recurrent neural networks, which are described in Section 3.2 and the motivation to use them is argued in Section 3.2.1 by acknowledging similar applications in the field of robotics. Then, our proposed tactile and force sensor models are presented in sections 3.3 and 3.4, respectively. And finally, this chapter is concluded with a summary in section 3.5.

3.1 Sensors' characteristics

The tactile sensor used in this thesis along with its sensing capabilities and specifications, will be described in the next section. For a comprehensive review of tactile sensing applications in robotics, the reader is referred to [59]. Then, the robot platform where the force-torque sensor is attached, as well as the sensor's behavior, will be presented in Section 3.1.2.

3.1.1 BioTac - tactile sensor

In this thesis we use a multimodal tactile sensor named BioTac[1], which was developed to imitate the sensing capabilities of a human fingertip. As seen in Figure 3.2, the sensor is equipped with a thermistor, a pressure sensor and 19 impedance sensing electrodes. The BioTac is covered by a flexible skin that holds an incompressible conductive fluid. This construction allows the BioTac to measure three modalities, namely, temperature, force and vibrations. To measure temperature, the thermistor detects changes in temperature caused by objects contacting the core. Force is computed by the fluid changing its distribution when a contact with the skin occurs, this change of distribution is detected as impedance changes by the electrodes. And finally, vibrations are measured when an object slides across the skin of the BioTac by the hydro-acoustic pressure transducer.

The output of the sensor for these modalities is summarized in Table 3.1 along with their ranges, resolution and frequencies. Here, E_n represents the voltage of an n electrode from the 19 impedance electrodes. P_{DC} refers to the absolute fluid pressure and P_{AC}

[1] https://wiki.ros.org/BioTac

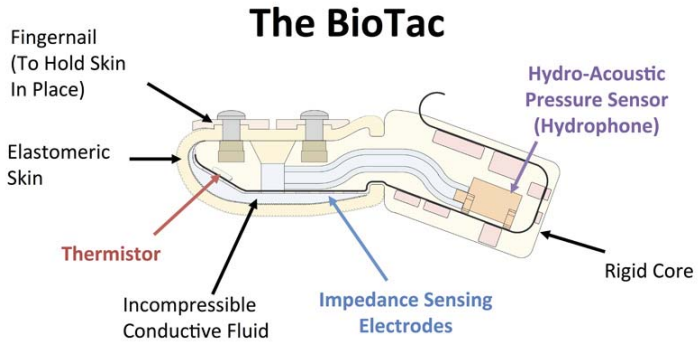


FIGURE 3.2: Schematic diagram of the BioTac sensor. Taken without permission from [1].

is the dynamic fluid pressure (e.g. vibrations). The temperature is represented by T_{DC} and the heat flux by T_{AC}. The value of E_n is computed based on the distribution of the incompressible fluid; since, when an object comes in contact with the sensor it deforms the skin, causing the fluid to change its distribution. Thus, each electrode changes its impedance value based on the amount of fluid around it, e.g. when there is less fluid around the electrode its voltage increases.

TABLE 3.1: Sensing characteristics of the BioTac as reported in [1].

Sensory modality	**Range**	**Resolution**	**Frequency response**
Impedance (E_n)	0 - 3.3V	3.2 mV	0 - 100 Hz
Fluid Pressure (P_{DC})	0 - 100 kPa	36.5 Pa	0 - 1040 Hz
Microvibration (P_{AC})	+/-0.76 kPa	0.37 Pa	10 - 1040 Hz
Temperature (T_{DC})	0 - 75 C	0.1 C	0 22.6 Hz
Thermal Flux (T_{AC})	0 - 1 C/s	0.001 C/s	0.45 22.6 Hz

3.1.2 Robot platform - force-torque sensor

Since the force-torque sensor must be attached to the robot in order to enable force sensing, we first describe the robot platform used in this research and how the force-torque sensor is part of this platform. The robot platform, shown in Figure 3.3, consists of a KUKA LWR arm with an attached Shadow robot hand as an end-effector. The force-torque (FT) sensor, an ATI Gamma[2] sensor, is located between the robot arm and robot hand; and together with an Adafruit (L3GD20H + LSM303)[3] inertial measurement unit (IMU) serve as the robot's external sensors. The IMU is connected to an Arduino board to transmit data to a computer which is also connected to the robot and the FT sensor.

[2] http://www.ati-ia.com/products/ft/ft_models.aspx?id=Gamma
[3] https://www.adafruit.com/product/1714

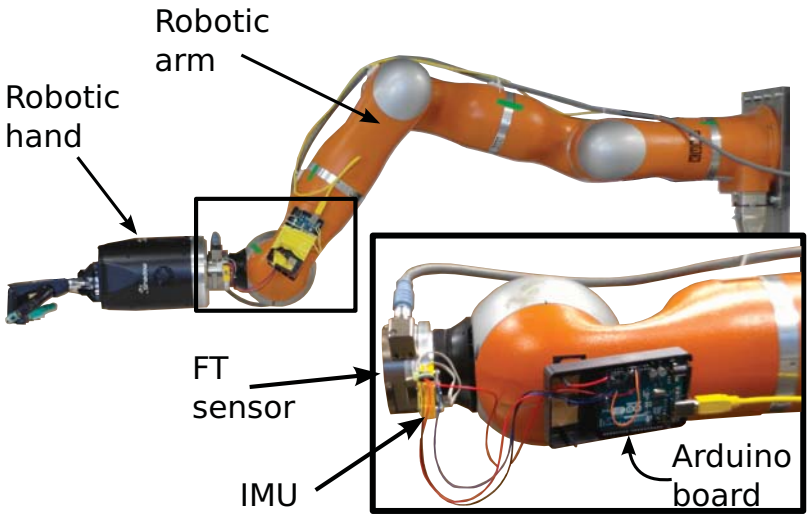


FIGURE 3.3: Robotic platform used to estimate contact forces.

Besides the external sensors, the robot arm also counts with internal sensors (e.g. joint encoders) that are used to obtain information regarding the position and velocity of the end-effector. The output and characteristics of each sensor are as follows:

1. **ATI Gamma:** produces a six-dimensional wrench expressed in the sensor's frame Σ_S at 1,000 Hz.

2. **Adafruit (L3GD20H + LSM303):** generates linear accelerations and angular velocities expressed in the IMU frame Σ_{IMU} at 300 Hz.

3. **Joint encoders:** Provide, through forward and differential kinematics, the end-effector orientation (in quaternion representation) plus linear and angular velocities expressed in the robot frame Σ_O at 500 Hz.

Figure 3.4 shows the output of the FT sensor when the robot arm, with the hand attached as an external load, moves without any contacts. Ideally, all the force in all axes should remain at zero as there are no contacts, however, due to gravitational, inertial, Coriolis and centrifugal forces; the output of the sensor is non-zero and thus it necessary to develop a sensor model that cancels these non-contact forces.

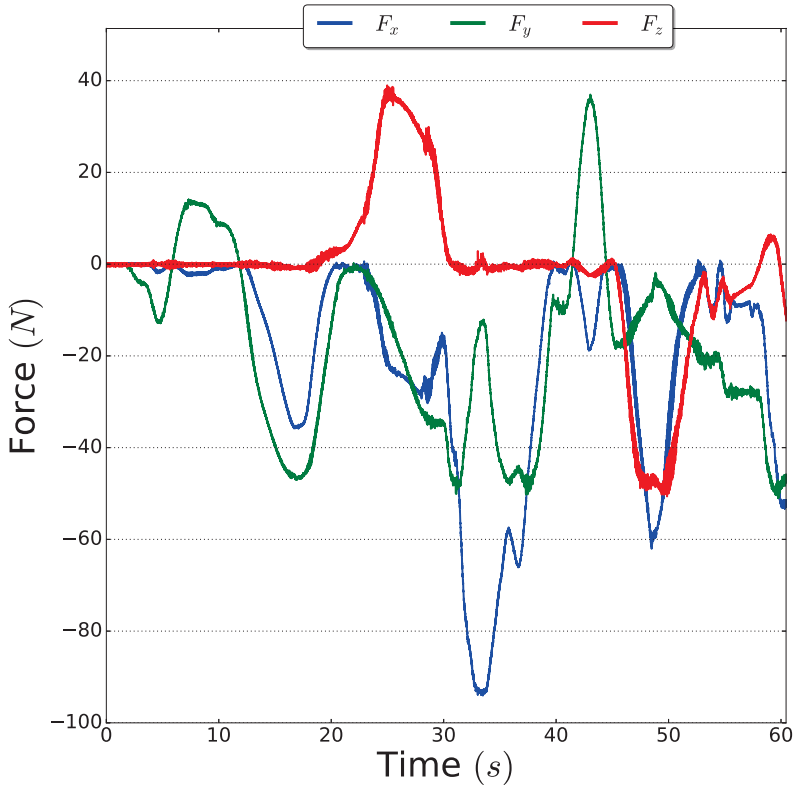


FIGURE 3.4: Force-torque sensor output while the arm was moved without generating contacts.

3.2 Recurrent neural networks

As shown in the previous section, the output of the sensors requires further processing in order to use them in the proposed pipeline. Thus, sensor models are required to map the raw output of the sensors to information compatible with the pipeline's interface. We propose to use recurrent neural networks (RNNs) to learn this mapping and begin this section by covering what RNNs are and how they work. Furthermore, we will provide motivation, in the form of recent applications of RNNs in robotic applications, for their use in this work.

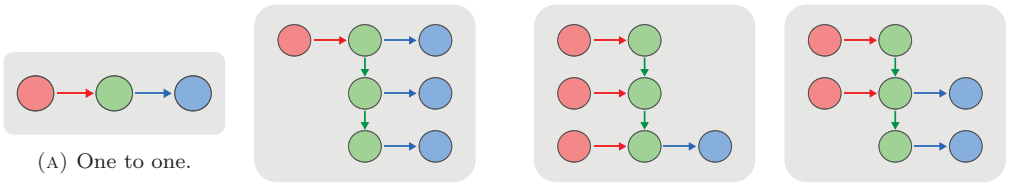
RNNs are a type of neural network that can process sequences, unlike traditional feed-forward networks. In fact, one can consider a feedforward network as a special case of RNN where a single input is mapped to a single output. RNNs are called recurrent since their output depends on previous computations and thus allow them to find patterns in sequences. Depending on the task, different architectures of RNNs can be implemented. Example diagrams for these architectures are shown in Figure 3.5 and their descriptions, along with example applications, are outlined below:

- **One to one:** A simple feedforward network is an example of this architecture, since a single input is mapped to a single output. One application for this kind of networks is image classification, where the input is an image and a label (e.g. a cat) is the output. A diagram of this network is shown in Figure 3.5a.

- **One to many:** Image captioning is an example application for this type of networks, where an image (one input) is given to the network to produce a caption (many outputs). This architecture can be seen in Figure 3.5b.

- **Many to one:** This architecture, shown in Figure 3.5c, is useful for sentiment classification, where a sequence of words are the input to the network and the network outputs whether the sentence is positive or negative.

- **Many to many:** This type of architecture is useful for machine translation were a sentence in one language must be translated into another language where the output sentence might have a different size from the input sentence (e.g. not a word by word translation). One example diagram for this architecture is depicted in Figure 3.5d



(A) One to one. (B) One to many. (C) Many to one. (D) Many to many.

FIGURE 3.5: Type of RNN architectures. The input, hidden and output layers are represented by red, green and blue circles respectively.

3.2.1 RNNs in robotics

Although RNNs have existed for well over two decades, their inefficiency in learning information from long sequences, due mostly to the vanishing gradient problem[4] [60], rendered them inapplicable for non-trivial tasks. However, with the introduction of gating (further discussed in Section 3.2.3), RNNs have been successfully applied on speech recognition problems [61] and on machine translation [62].

[4] As artificial neural networks are usually trained by updating their weights based on the gradient, the smaller the gradient becomes, the lesser effect it will have in changing the values of the weights. Thus, when the gradient is sufficiently small the network stops learning (i.e. updating its weights).

Following these recent successes, researchers have also applied RNNs to robotic tasks where contact information (e.g. forces) was involved. For instance, Erickson et al. used RNNs to learn a force distribution map on a person's limb caused by the forces generated when dressing the person with a hospital gown [10]. Contact transients during snap-fit tasks, e.g. turning on an electric switch or closing an eyeglass case, were detected using RNNs as described in [63].

In this thesis, we will take advantage of the performance of RNNs to learn a map between the sensor's output (e.g. tactile and force signals) and contact forces represented as a three-dimensional vector, i.e. the contact force in the x, y and z axes.

3.2.2 How do RNNs work?

As previously mentioned, and unlike traditional feedforward networks, RNNs are a type of neural networks that are able to process sequences. To do so, they use recurrency since the output of a layer is dependent on previous computations. A diagram showing this dependency is shown in Figure 3.6, where an RNN is unfolded to show how previous outputs are used in the computation of the current and future outputs.

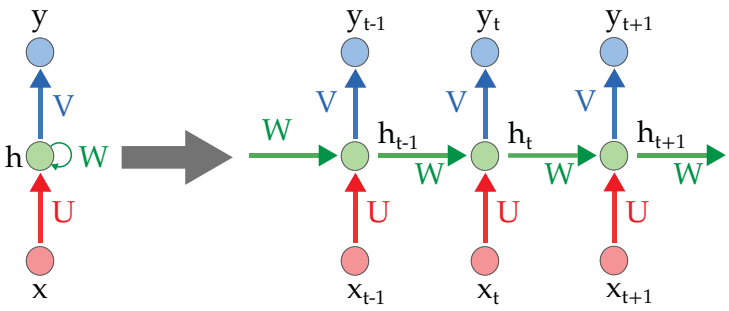


FIGURE 3.6: An unfolded recurrent neural network.

Mathematically, a new hidden state h_t is computed as follows:

$$h_t = f(U \cdot x_t + W \cdot h_{t-1}) \tag{3.1}$$

where x_t is the input vector at time t and f is some nonlinear function such as tanh. The network parameters (e.g weights) for the input, hidden and output layers are contained in the U, W and V matrices, respectively. The output state y_t is computed depending

on the task, e.g. for classification one could used $y_t = softmax(V \cdot h_t)$ to compute the probability of a given class. An example of an RNN architecture is shown in 3.7.

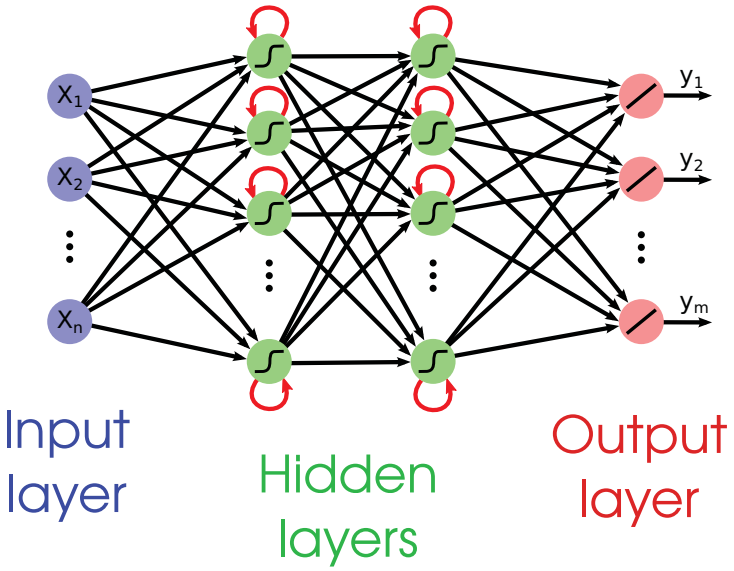


FIGURE 3.7: Example of a RNN architecture.

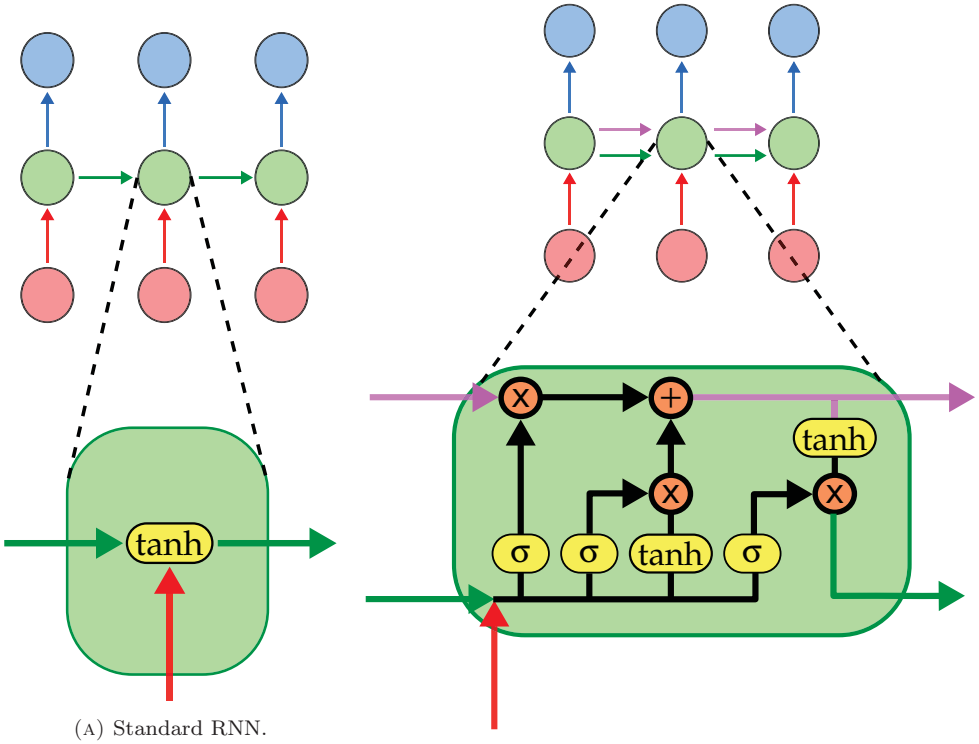
One problem RNNs have, is the fact that the number of layers increases with the number of time steps of the input sequences (see Figure 3.6). This can lead to an inability of the network to learn long term dependencies, e.g. when the relevant information happens too far from the current time step. The reason for this is due to how the network updates its weights. Similar to other neural networks, RNNs are trained using the backpropagation algorithm[5], but with a small difference. Since the network parameters are the same across all time steps (see Figure 3.6) the gradient at each output is dependent of the current time step as well as the previous time steps. The consequence of this is that the gradient will tend to vanish the longer the input sequence is. This effect is known as the vanishing gradient problem.

In order to deal with this problem, researchers have designed algorithms that are able to avoid this problem by using a "gating" mechanism. This gating mechanism basically allows the network to manage its memorizing process by selectively "forgetting" data. One of the most successful architectures that implements this behavior is called long short-term memory (LSTM) and will be explained in the following section.

[5]Backpropagation computes the gradient in order to update the network's weights. It first computes the error at the output and then it propagates it backwards going layer by layer.

3.2.3 LSTMs

Long short-term memory networks, or LSTMs for short, are a type of recurrent neural network that are able to learn long term dependencies by controlling the amount of information to add or remove from the hidden state. In fact, most of the success achieved by RNNs is due to the use of of LSTMs, e.g. the works described in 3.2.1 all use LSTMs. A visual representation of the structure of an LSTM, and a comparison against that of a standard RNN, is shown in 3.8.



(A) Standard RNN.

(B) LSMT.

FIGURE 3.8: Comparison between how a standard RNN and an LSTM compute the hidden state.

As it can be seen in Figure 3.8b, LSTMs introduce an additional variable called *cell state* (shown in purple). To update the value of this cell state, the LSTM uses the following three gates:

1. **Forget**: This gate controls the amount of information that gets discarded from the cell state. Here, a sigmoid (leftmost in Figure 3.8b) takes as input the value of the previous hidden state h_{t-1} (green arrow on the left) and the current input x_t (red arrow) and outputs a value ranging from 0 to 1 to each element in the

previous cell state C_{t-1} (purple line). In this way, this gate updates the cell state by forgetting irrelevant information.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t]) \tag{3.2}$$

2. **Input:** Here, new information is added to the cell state by computing first which values to update and secondly by creating a cell state candidate $\tilde{C}_t$. The values to update are computed using a sigmoid (second from left in Figure 3.8b) and the candidate cell state is created by a tanh function.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t]) \tag{3.3}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t]) \tag{3.4}$$

3. **Output:** Using the outputs of the previous gates, this gate updates the cell state by combining them (rightmost purple arrow in Figure 3.8b):

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{3.5}$$

Finally, using the cell state computed by the gates described above, the hidden state value (rightmost green arrow in Figure 3.8b) can be updated as follows:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t]) \tag{3.6}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{3.7}$$

3.3 Tactile sensor model

This section introduces the approaches developed to estimate both the magnitude and location of contact forces using a tactile sensor, as well as the description of the experimental setup used to validate them.

3.3.1 Force magnitude estimation

In order to estimate the force magnitude of a contact, a sensor model for the BioTac should map the sensor's output to a three-dimensional force. The model therefore can make use of the impedance and pressure signals to compute the magnitude of a contact. Due to the fabrication of the BioTac sensor (see Figure 3.2), developing an analytical formulation that models the fluid dynamics involved in the sensor would be extremely complex. To avoid this, researchers have relied on machine learning algorithms which have been proved to outperform previously analytic formulations for the BioTac sensor [64]. In this thesis, we apply a recurrent neural network, as explained in Section 3.2, to learn a mapping function relating the values of the impedance electrodes and pressure values to a three-dimensional force.

The structure of the network consists of two hidden layers, each composed of 20 long short-term memory units, and a fully connected output layer as shown in Figure 3.7. Here, the inputs to the network are the 19 impedance electrodes and the two pressure signals, and the output is a three-dimensional force. A hyperbolic tangent sigmoid function was used for the hidden layers and a linear activation function was used for the output layer. As noted in Section 3.2, since RNNs are ideal for time series, the input to the network is a sequence of N time steps of the tactile signals, while the output of the network is only the last time step of the force vector. To optimize the learning of the network, we applied the stochastic gradient descent algorithm.

Data collection

To train the network described above it is necessary to first collect data that includes the tactile signals from the BioTac as well as the ground truth values of force. To generate the required dataset, each finger on the Shadow robot hand[6] (which is equipped with BioTac sensors at its tips) was moved to make contact with an ATI Gamma force-torque sensor[7], as shown in Figure 3.9.

To generate contacts on the tactile sensor with different area shapes and sizes, probes with different tips were fixed on the force-torque sensor. Each fingers was separately

[6] https://www.shadowrobot.com/products/dexterous-hand/
[7] http://www.ati-ia.com/products/ft/ft_models.aspx?id=Gamma

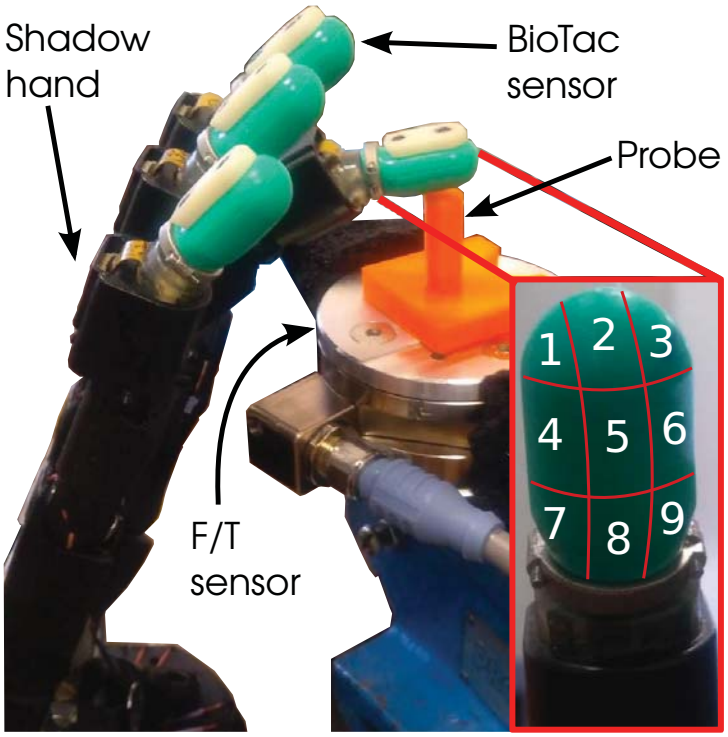


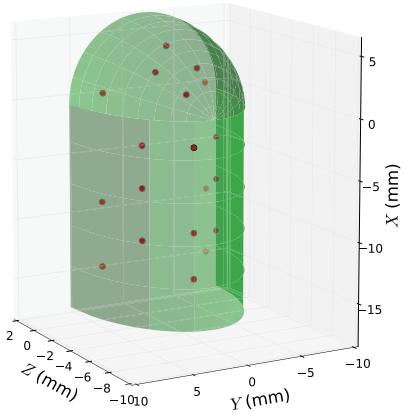
FIGURE 3.9: Setup to collect data for the force magnitude estimation. The data consisted of the tactile signals as inputs and the labels (e.g. ground truth values) were the three-dimensional forces obtained from the force-torque sensor.

moved downwards ten times for ten seconds at nine locations on each probe (see Figure 3.9). This produced a total of 228 recordings, each having close to 12,000 time steps, containing 21 tactile signals (19 impedance and two pressure signals) and a three-dimensional force. As the sensors have different operating rates, the force-torque sensor operates at 1 KHz while the tactile sensor runs at 100 Hz, the data was recorded at the lowest rate. The contact forces generated ranged between 0.1 to 1.0 N[8]. The data was then divided in 80% as the training dataset and 20% as the test dataset. As the validation dataset, 20% of the training dataset was used.

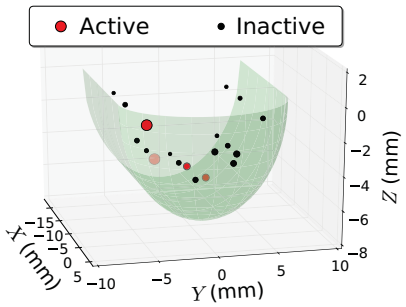
3.3.2 Contact localization

Instead of relying on machine learning to estimate contact location, we leverage the fact that the 19 impedance electrodes are distributed across the BioTac to retrieve the location of a contact. The configuration of the electrodes can be seen in Figure 3.10a.

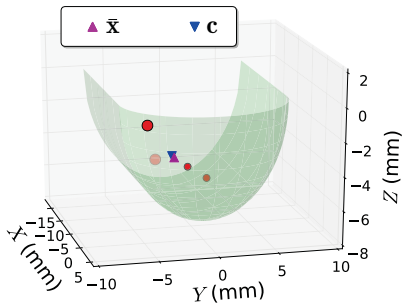
[8]Since these are usual force values that occur when making contact with the deformable objects we are concerned with in this thesis.



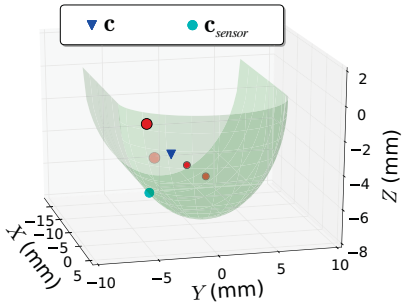
(A) Geometric model of the sensor with the electrodes shown in red.



(B) Active electrodes.



(C) Centroid and contact location.



(D) Projection of the contact to the sensor's surface.

FIGURE 3.10: Sensor model, shown in (A), and the steps of the contact localization algorithm: (B) thresholding of the active electrodes where the electrodes' size is shown proportional to their intensity values, (C) contact localization based on the active electrodes and their geometric centroid, (D) projection of the contact to the sensor's surface.

The first step to estimate the contact location is to filter the active electrodes, e.g. the ones close enough to the contact point such that their values exceed their resting values. These resting values are the initial impedance values, which do not always initialize with the same values and therefore must be subtracted from the current impedance values. This filtering stage is visualized in Figure 3.10b where the size of the electrodes is plotted proportionally to their intensity. Once the active electrodes have been filtered, the geometric centroid is computed as the mean of the positions of the m active electrodes

$$\bar{\mathbf{x}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{p}_{\mathbf{e_i}} \tag{3.8}$$

where $\mathbf{p}_{e_i}$ represents the position of the i-th electrode. Once the centroid is known, it is used to find the contact point by computing the direction vectors $\mathbf{d}_i$ between the centroid and the active electrodes. In order to locate the contact point, the direction

vectors are multiplied by the normalized intensity of the electrodes, thus assuring that the contact point is closer to the electrodes with the highest intensities.

$$\mathbf{d}_i = (\mathbf{p}_{e_i} - \bar{\mathbf{x}})\frac{I_{e_i}}{I_e} \tag{3.9}$$

where I_{e_i} is the intensity value of the i-th electrode and I_e represents the sum of all active electrodes. By summing these displacement vectors we can compute the contact location $\mathbf{c}$

$$\mathbf{c} = \frac{\sum_{i=1}^{m} \mathbf{d}_i}{m} + \bar{\mathbf{x}} \tag{3.10}$$

Both the centroid and contact location are displayed in Figure 3.10c, where the magenta triangle represents the geometric centroid and the blue triangle represents the contact location. Finally, once the contact location has been computed it is necessary to project it onto the surface of the sensor. This is achieved by modeling the sensor surface as a sphere:

$$\mathbf{c}_{sensor} = \mathbf{o} + \frac{r(\mathbf{c} - \mathbf{o})}{\|\mathbf{c} - \mathbf{o}\|} \tag{3.11}$$

where r represents the radius of the sphere (we set $r = 7$ mm) and $\mathbf{o}$ is the origin, except when the contact $\mathbf{c}$ is negative on the X axis, i.e. it is in the cylindrical part of the sensor (see Figure 3.10a). In that case, we set $\mathbf{o} = (x, 0, 0)$ to avoid distortions caused by using a spherical projection on a cylinder, where x is $\mathbf{c}_x$. The contact location, now projected on the sensor's surface, is shown as a turquoise sphere in Figure 3.10d.

3.3.3 Experimental evaluation

This section describes the experimental setups used to evaluate the accuracy of the proposed approaches to estimate the magnitude and location of a contact force.

Force magnitude estimation

To evaluate the ability of the network to estimate the three-dimensional force from the tactile signals of the BioTac, the algorithm was implemented in Python using TFLearn [65]. The input sequence was 50 time steps long and a learning rate of 0.01 was used in the regression (output) layer. The network was trained for 100 epochs. In order to compare this network with similar ones found in the literature, a feedforward deep neural network (DNN) as described by Su et al. in [64] was implemented as well, where the 19 impedance electrode values were used as input. Additionally, two more networks were evaluated by considering also the pressure values. Hence, the following four networks, all using the same parameters described above, were compared:

- dnn_{19} : DNN with impedance values.
- dnn_{21} : DNN with impedance and pressure values.
- rnn_{19} : RNN with impedance values (using a 50 time steps sequence).
- rnn_{21} : RNN with impedance and pressure values (using a 50 time steps sequence).

The results obtained by these four networks are summarized in Table 3.2, and an example of the force magnitude estimation, using the rnn_{21} network, is shown in Figure 3.11.

Table 3.2: Evaluation results of the force estimation.

	RMSE (in mN)			**SMSE**		
	f_x	f_y	f_z	f_x	f_y	f_z
dnn_{19}	41.74	94.38	344.74	1.6127	1.713	2.5225
dnn_{21}	41.95	94.59	344.71	1.6294	1.7207	2.5222
rnn_{19}	18.64	35.91	53.11	0.3213	0.2477	0.0599
rnn_{21}	**18.07**	**31.07**	**51.71**	**0.3018**	**0.1854**	**0.0569**

Contact localization

In order to evaluate the contact localization algorithm, it is necessary to obtain a ground truth measure of the contact location. To this end, a probe was fixed in a known location with respect to the Shadow robot hand and then the kinematic chain of the robot hand was used to compute the relative position between the probe frame (Σ_p) and the sensor frame (Σ_s), as depicted in Figure 3.12.

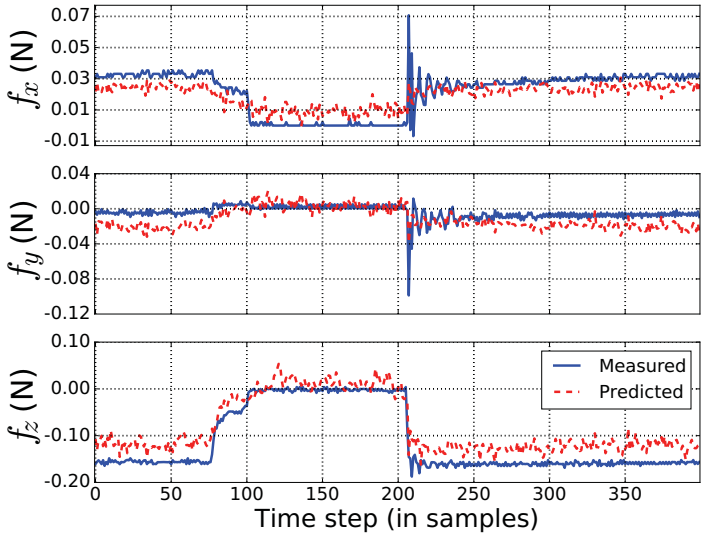


FIGURE 3.11: Force estimation along the three axes.

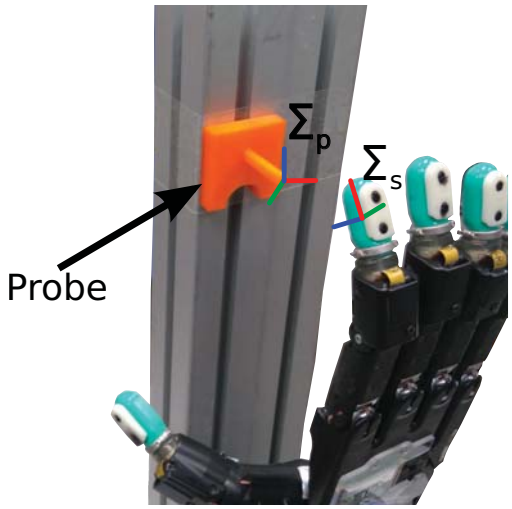


FIGURE 3.12: Setup to evaluate the contact localization algorithm.

The distance between the probe frame and the sensor frame was then compared to the distance between the output of the contact localization algorithm and the sensor frame. The error was then computed as the difference between these distance and expressed with respect of the sensor frame (Σ_s). The algorithm was only evaluated for the first finger by contacting 12 locations across the sensor. For each location, the finger was moved five times and the average error was calculated for the X, Y and Z, as shown in Figure 3.13.

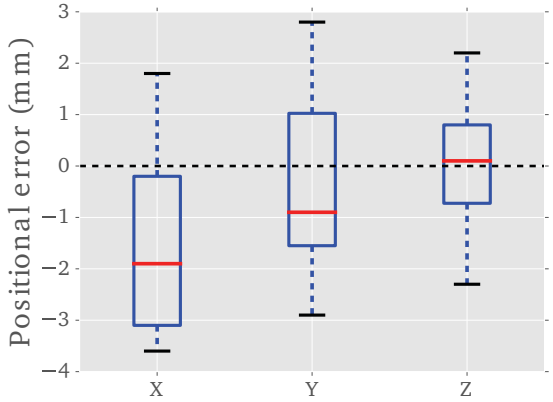


FIGURE 3.13: Errors in the X, Y and Z axes for the contact localization algorithm.

3.3.4 Tactile sensor model

The tactile sensor model estimates the magnitude of contact forces using a RNN, and the location of a contact based on an analytical method. The force estimation results, depicted in Figure 3.11, show the accuracy of RNNs to predict three-dimensional forces from the tactile signals of a BioTac sensor. Furthermore, the use of RNNs clearly outperformed the use of feedforward networks, as Table 3.2 shows, validating our hypothesis that considering temporal signals of the tactile data produces a more accurate estimation. In contrast, the addition of pressure signals did not produce significant improvements, which suggests there is redundancy between the impedance measurements (provided all electrodes are considered) and the pressure sensor of the BioTac.

Although the error when estimating normal forces (f_z) is low, the error for tangential forces (f_x and f_y) is relatively high. This disparity is caused by the manner in which the training data was collected. As most of the data was generated by moving the fingers downwards towards the force-torque sensor (e.g. normal forces), tangential forces were not generated as much. Thus, having less data to learn the tangential forces the network was not able to achieve a higher performance when estimating these forces. This issue can be addressed, for instance, by collecting data while sliding the fingers side to side.

Regarding the accuracy of locating a contact, the proposed model was able to estimate the contact's position, on each axis, withing five millimeters as it is shown in Figure 3.13. The error disparity between the axes can be better understood by referring to Figure 3.10a, which shows the X axis as the long axis of the sensor and thus results in a larger error than the error in the Y axis. The error for the Z axis is significantly

smaller since the estimation of this axis is governed by the geometric projection described in Section 3.3.2, which guarantees that the estimated position will be on the sensor's surface. A limitation of the proposed approach is its dependence on the location of the electrodes, that is, the algorithm cannot properly estimate contacts when they occur on the extreme sides of the sensor (e.g. where there are no electrodes).

3.4 Force sensor model

In this section, we present a model that first estimates the non-contact forces, measured by a force-torque sensor, that later subtracts them from the sensor output in order to obtain the pure contact forces. The section begins by outlining the robotic platform used and motivates the need of a sensor model, as it points out why it is not suitable to rely on the direct output of the force-torque sensor. Then, an analytical method, commonly used to estimate non-contact forces, is described; followed by our proposed approach based on recurrent neural networks. Afterwards, the procedure to collect the necessary data is presented and finally, the experimental validation comparing our approach to the analytical approach is detailed.

Disclaimer: The material presented in this section was developed in collaboration with fellow PhD student Kamal Mohy El Dine.

3.4.1 Analytical model

The wrench output of the force-torque sensor can be expressed as the sum of the contact and non-contact wrenches, namely:

$$\begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{nc} \\ \boldsymbol{\tau}_{nc} \end{bmatrix} + \begin{bmatrix} \mathbf{f}_{c} \\ \boldsymbol{\tau}_{c} \end{bmatrix} \tag{3.12}$$

where $[\mathbf{f}_{nc}, \boldsymbol{\tau}_{nc}]^T$ are the non-contact wrenches generated by gravity, inertia, Coriolis and centrifugal forces; and $[\mathbf{f}_{c}, \boldsymbol{\tau}_{c}]^T$ are the pure contact wrenches due to contact forces and torques. The output of the sensor, is expressed by $\mathbf{f}$ and $\boldsymbol{\tau}$ which are the force and torque values expressed in the sensor frame Σ_S. Both the force and torque values,

expressed in the sensor frame, are comprised of contact and non-contact elements and using the Newton-Euler approach they can be expanded as:

$$\mathbf{f} = \overbrace{m\boldsymbol{\alpha} - m\mathbf{g} + \dot{\boldsymbol{\omega}} \times m\mathbf{c} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times m\mathbf{c})}^{\mathbf{f}_{nc}} + \mathbf{f}_c \tag{3.13}$$

$$\boldsymbol{\tau} = \underbrace{\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) + m\mathbf{c} \times \boldsymbol{\alpha} - m\mathbf{c} \times \mathbf{g}}_{\boldsymbol{\tau}_{nc}} + \boldsymbol{\tau}_c \tag{3.14}$$

where $\boldsymbol{\omega}$ is the angular velocity vector of the sensor with respect to its frame, $\boldsymbol{\alpha}$ and $\dot{\boldsymbol{\omega}}$ are the linear and angular acceleration vectors respectively; $\mathbf{g}$ represents the gravity vector, m is the mass of the load (i.e. the robot hand), $\mathbf{c}$ is its center of mass coordinates vector and $\mathbf{I}$ is a 3×3 symmetric matrix representing the inertia matrix in the sensor frame.

As it can be seen in equations 3.13 and 3.14, the non-contact elements can be estimated provided the ten inertial parameters are known, namely, the values of m, $\mathbf{c}$ and $\mathbf{I}$. To obtain the values for these parameters, identification methods are usually applied, see for instance [66]. In our case, we applied non-linear least squares to identify these parameters. After the identification, the non-contact wrenches can be then removed from the output of the force-torque sensor to obtain only the contact wrenches.

3.4.2 RNNOB

As described in the previous section, the analytical approach depends on an accurate estimation of the inertial parameters. However, this estimation requires precise measurements relating the sensor frame with the robot hand (external load), which might lead to measurement errors. In order to overcome these inaccuracies, we propose an observer that estimates non-contact forces without the need for precise measurements between frames. As noted in Section 3.2, RNNs are ideal for learning time dependencies and thus can be used to correlate the robot's kinematics to a wrench signal (e.g. the output of the force-torque sensor). We exploit this fact to propose an observer called RNNOB, that uses a recurrent neural network to map sensor data into non-contact forces that are then subtracted from the force-torque sensor measurements, as shown in Figure 3.14.

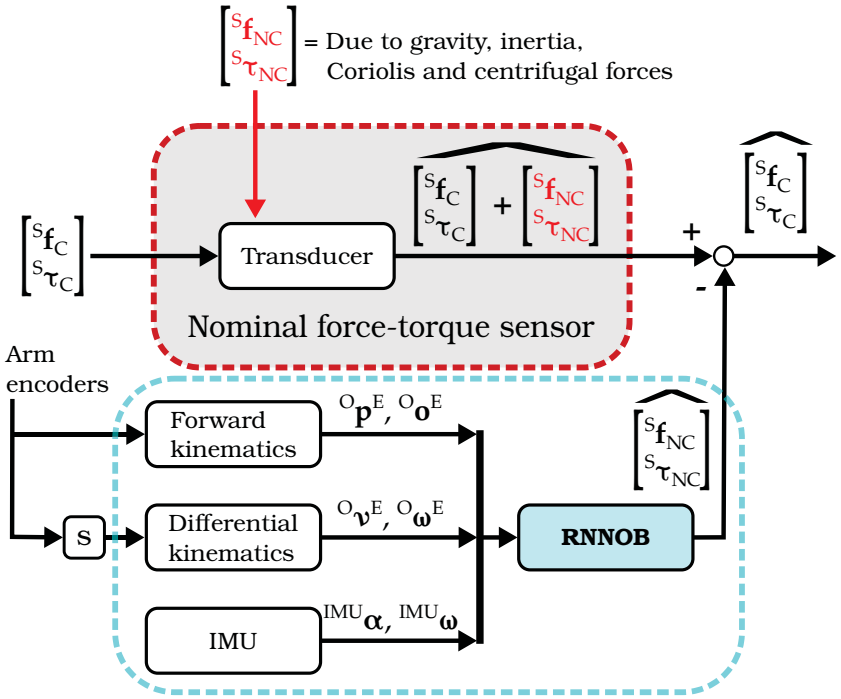
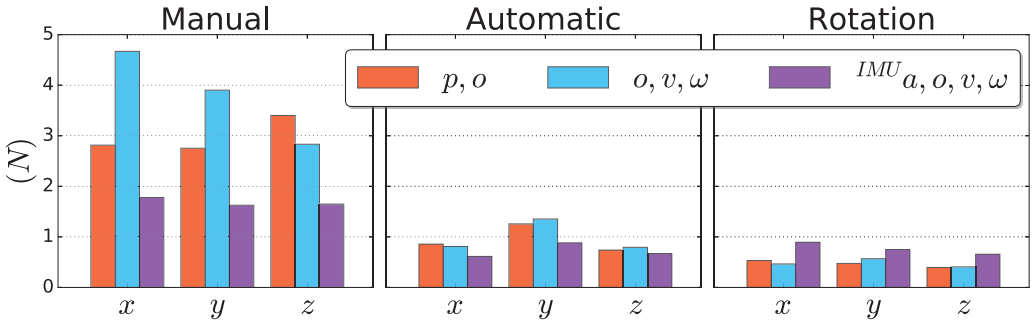


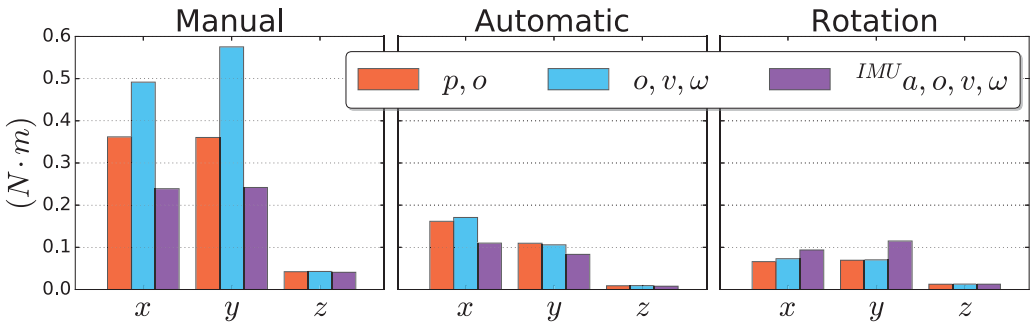
FIGURE 3.14: Diagram showing how the RNNOB cancels the non-contact wrench in order to estimate the pure contact wrench of the force-torque sensor.

The RNNOB takes as input the pose (${}^{o}\mathbf{p}^{E}$, ${}^{o}\mathbf{o}^{E}$) and twist (${}^{o}\nu^{E}$, ${}^{o}\omega^{E}$) of the end-effector expressed with respect to the robot's base frame Σ_o; and the linear acceleration ${}^{IMU}\alpha$ and angular velocity ${}^{IMU}\omega$ of the IMU sensor. The architecture of the RNNOB consists of two hidden layers, with 15 and 10 LSTM units respectively. As an activation function between the hidden layers, the RNNOB uses a hyperbolic tangent sigmoid function and a linear activation function for the output layer. We apply the stochastic gradient descent algorithm in the output layer to learn the wrench output of the force-torque sensor while the robot was moved without generating contacts. Once the non-contact wrench is predicted by the RNNOB, it is subtracted from the output sensor to obtain the pure contact wrench (see Figure 3.14).

The input choice for the network was decided based on the performance of different combinations of features. Figure 3.15 shows three different models using different feature inputs. The first model relies only on the end-effector pose (p, o), where the second model uses the orientation and twist of the end-effector (o, v, ω). Finally, the last and chosen model, includes also the information obtained by the IMU sensor, namely, the linear acceleration $({}^{IMU}a, o, v, \omega)$.



(A) Linear errors for the three types of test motions.



(B) Angular errors for the three types of test motions.

FIGURE 3.15: RMS errors for the RNN models based on: 1) pose (p, o), 2) orientation and twist (o, v, ω) and 3) linear acceleration, orientation and twist $({}^{IMU}a, o, v, \omega)$.

3.4.3 Data collection

As equations 3.13 and 3.14 show, the non-contact forces depend directly on the sensor's angular velocity and acceleration, linear acceleration and its orientation. Hence, it is necessary to excite the sensor, through motions of the robot arm, to generate a wide range of values for these variables. To this end, we generated the following datasets:

- **Manual:** The robot arm was set to gravity compensation mode and then an operator manually moved the arm, without touching the robotic hand, to various poses in the workspace with random velocities and accelerations.
- **Automatic:** The robot arm was commanded to move between random points in its workspace using various trapezoidal velocity profiles without human intervention.
- **Sinusoidal:** The robot arm executed a sinusoidal trajectory on the xy-plane of the robot frame Σ_o, as shown in Figure 3.16, while the end-effector was rotated around each axis sequentially (i.e. roll, pitch and then yaw).

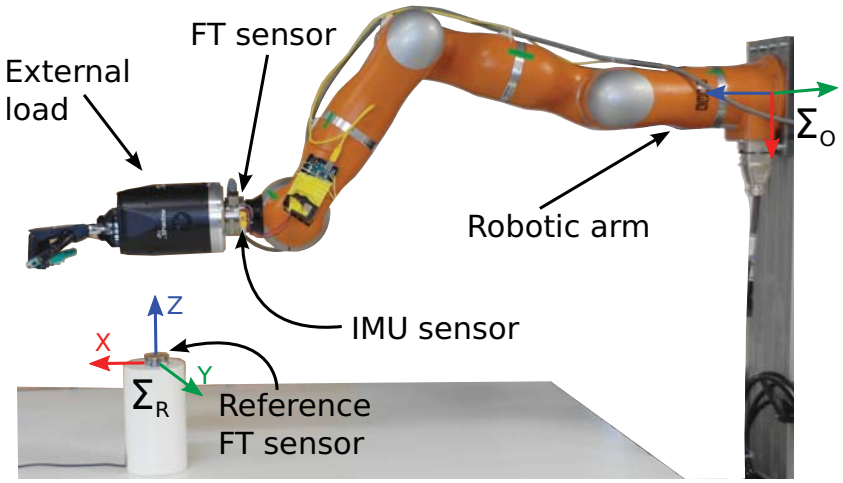


FIGURE 3.16: Test setup used to perform the collision test.

The manual data was collected in five trials, each about five minutes long; the automatic data was collected in ten trials with an average time of two minutes each, and the sinusoidal data was collected in one trial about three minutes long. These datasets were then combined into one training dataset, where one trial of each of the manual and automatic data were separated to create a test dataset, but the sinusoidal data was used entirely for training. A validation set was used by using 20% of the training dataset. The RNNOB was implemented in Python using TFLearn [65], where the sequence length for the input was 20 time steps and a learning rate of 0.01 was used in the output layer. The network was trained for 20 epochs.

Due to the different operating rates of the sensors (as described in Section 3.1.2), the data was recorded at 500 Hz to have a uniform sampling rate. That is, the force-torque sensor was effectively downsampled and the last output of the IMU was kept until a new sample was published.

3.4.4 Experimental evaluation

A comparison between the proposed RNNOB (Section 3.4.2) and the analytical method (Section 3.4.1) for the collected datasets described in the previous section is summarized in Table 3.3.

Besides the testing performed on datasets, two additional tests were performed, namely a rotational motion test and a collision test. The rotational motion test was used to validate the proposed observer against gravitational forces. The measured and estimated

TABLE 3.3: The root mean square error on the datasets for the proposed RNNOB and the analytical method.

		Manual		Automatic		Rotational	
		RNNOB	**Analytic**	**RNNOB**	**Analytic**	**RNNOB**	**Analytic**
Error (N)	f_x	1.6578	2.8934	0.8075	2.4436	0.8405	2.6900
	f_y	1.7235	2.6943	1.1334	2.8558	0.7189	1.2193
	f_z	1.5420	2.2989	0.7829	1.5147	0.7082	1.2204
Error $(N \cdot m)$	τ_x	0.2538	0.3358	0.1384	0.2835	0.0911	0.1408
	τ_y	0.2332	0.3420	0.0995	0.2020	0.1053	0.1601
	τ_z	0.0403	0.0513	0.0089	0.0132	0.0118	0.0301

wrenches (by the analytical and RNNOB approaches) can be seen in Figure 3.19, where the angular rotations are shown at the bottom of the figure.

For the collision test, we used an ATI Mini45[9] force-torque sensor as reference to evaluate the accuracy in estimating pure contact forces. Here, the robot arm was set to gravity compensation mode to move the robotic hand and collide it ten times with the reference force-torque sensor which was fixed on a table as shown in Figure 3.16.

An example plot showing a comparison between the measured and estimated wrenches (from both the analytical and RNNOB approaches) of a manual motion can be seen in Figure 3.17.

Figure 3.18 depicts one of the ten contact tests, where the force in the x axis, as predicted by the RNNOB, is compared to the force as output by the reference force-torque sensor.

3.4.5 Force sensor model

Table 3.3 summarizes the comparison between the analytical method and the proposed RNNOB to estimate non-contact wrenches for the three tests described in 3.4.4. It is clear that the RNNOB outperforms the analytical method not only on each test, but also on each dimension of the estimated wrench (i.e. on estimating the force and torque values in the x, y and z axes). Figure 3.17 visually shows the RNNOB more closely predicting the measured wrenches than the analytical method (see the zoomed area). Furthermore, the RNNOB prove to be more stable when estimating gravitational forces as it is shown in Figure 3.19. Regarding the collision tests, the RNNOB was able to estimate contact forces within an error of $2N$, as it can be seen in Figure 3.18.

[9] https://www.ati-ia.com/products/ft/ft_models.aspx?id=Mini45

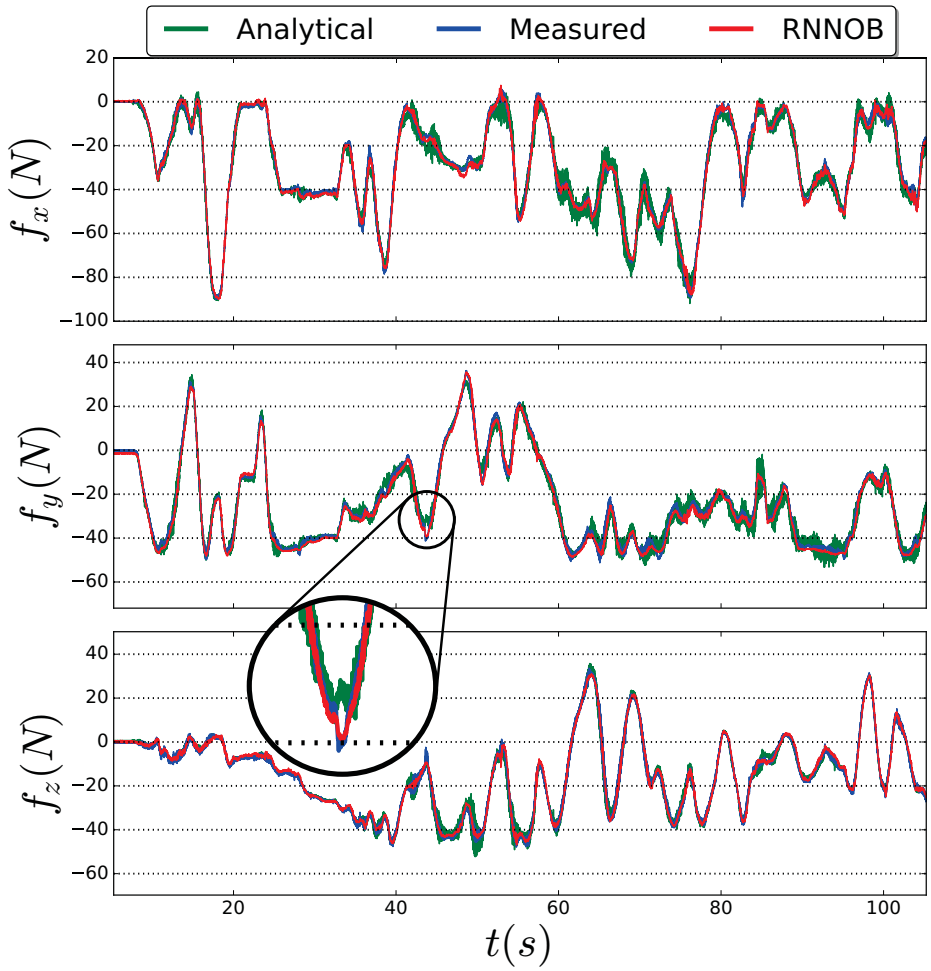


FIGURE 3.17: Non-contact wrench estimation of the proposed RNNOB (red), the analytical-based approach (green) and the measured wrench (blue), as output by the force-torque sensor, for an unseen manual trajectory.

3.5 Summary

This section has presented two sensor models, namely, a tactile sensor model and a force sensor model, which rely on recurrent neural networks to estimate contact information. The characteristics of the sensors were outlined as well, and an experimental validation was conducted to evaluate the accuracy of the proposed models. The performance of the developed sensor models proved to outperform current state of the art solutions, thus highlighting the advantages of using recurrent neural networks. However, this performance comes at the cost of acquiring labeled data which might not be straightforward to obtain for certain applications.

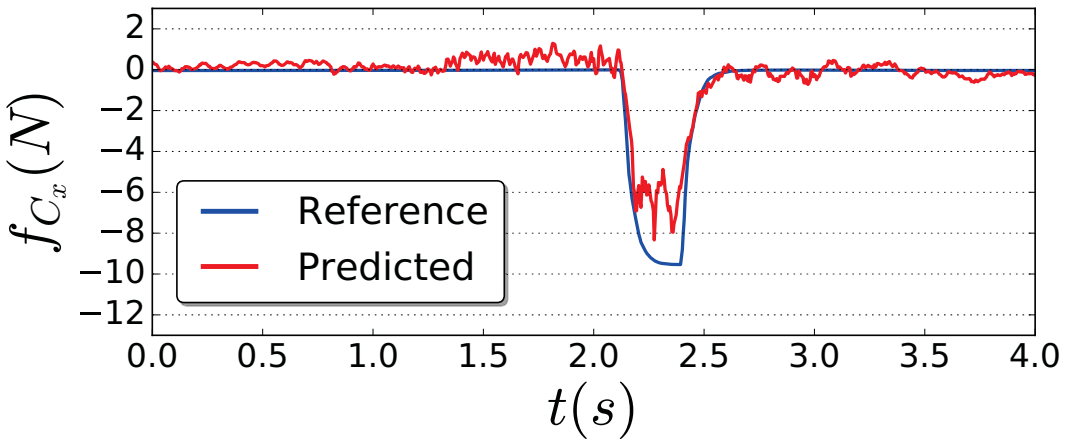


FIGURE 3.18: Contact force estimation of the proposed approach compared to the reference force measurement.

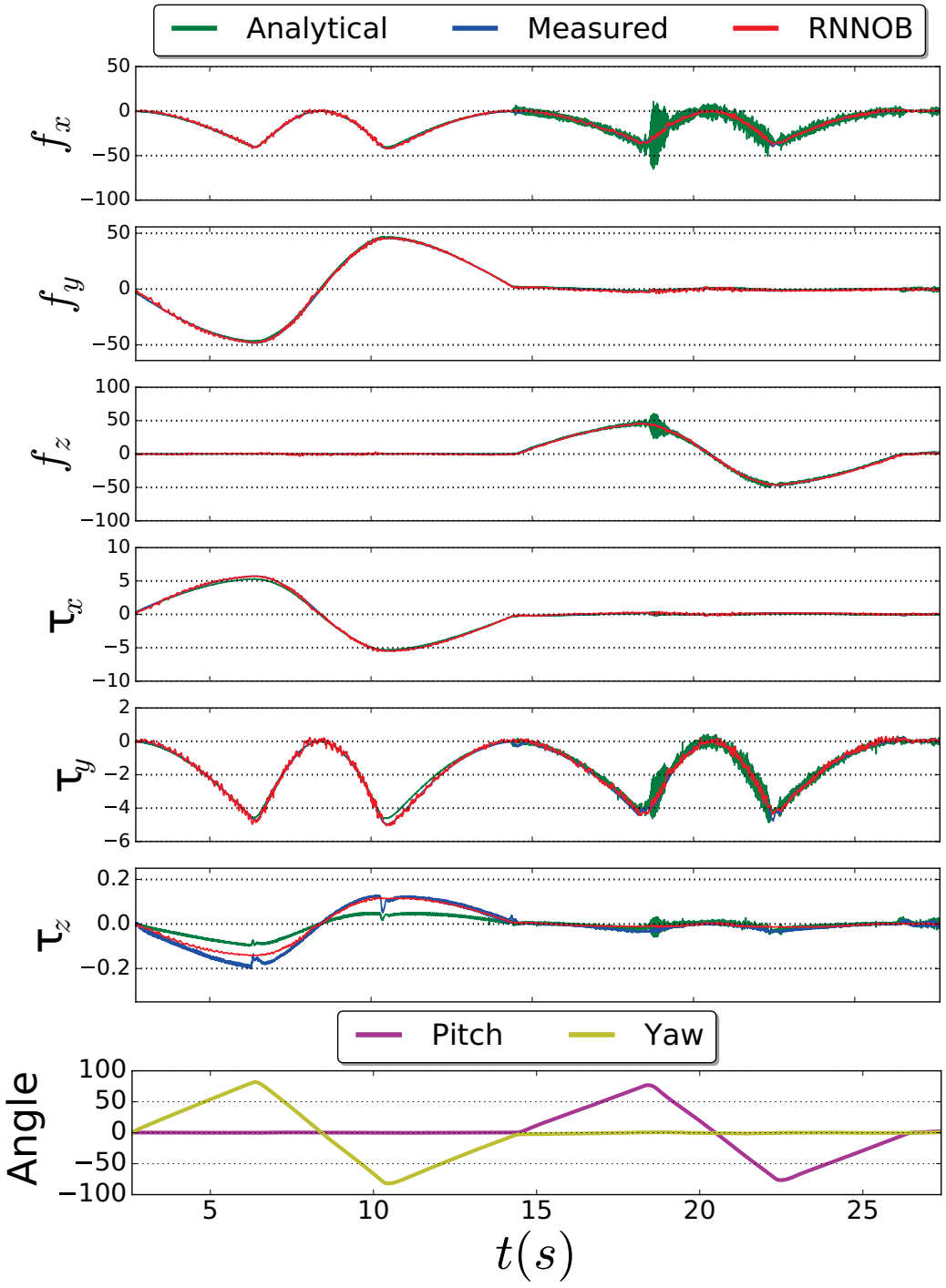


FIGURE 3.19: Non-contact wrench estimation of the RNNOB (red), analytical approach (green) and as measured by the force-torque sensor (blue) for the rotational motion test. The first three rows show the forces (in N) and the next three rows show the torques (in $N \cdot m$). The last row shows the rotations around the sensor's y-axis (pitch) and z-axis (yaw) expressed in degrees. The roll angle is not shown since it has no significant effect as the sensor's x-axis is along the gravity vector $\mathbf{g}$.

Chapter 4

Shape sensing pipeline

In the previous chapter we presented two sensor models that can be used as components of the proposed shape sensing pipeline as depicted in Figure 4.1. In this chapter, we will describe the rest of the components required for the pipeline, namely, a module to transform the contact forces, as obtained from the sensor models, such that they can be applied to a deformation model which updates the shape of the object. We will first justify the design and use of a pipeline based on the advantages of a component-based development paradigm, as well as provide a description of what this paradigm entails. Following this section, the two remaining components of the pipeline, e.g. *force transformation* and *deformation model*, will be outlined as well as the integration of all these components. Example applications are also provided to validate the purpose of the pipeline. Finally, a summary of the chapter is presented at the end.

4.1 Component-based software engineering

Component-based software engineering (CBSE) is a programming paradigm that regards robotic functionalities (e.g. object recognition, manipulation planning, etc.) as components. These components can then be used as building blocks to develop robotic applications in a reusable and maintainable manner, thus reducing software developement time [2]. This is achieved by separating the specification of the component from its implementation, meaning that the code of a component can be upgraded without affecting other components that rely on the upgraded component.

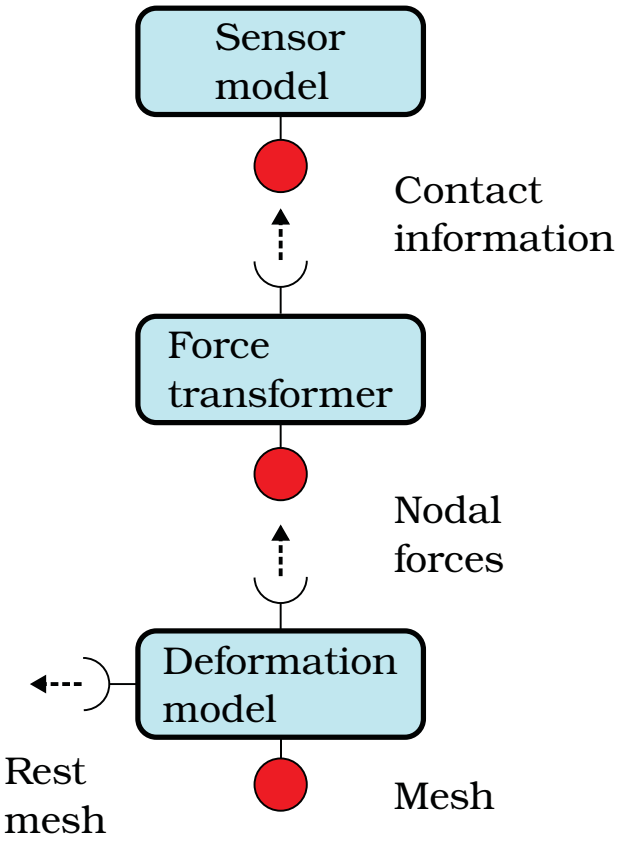


FIGURE 4.1: Proposed pipeline using a component-based representation as detailed in [2]. The red circle denotes the interface a component provides and the half circle represents a required interface.

Furthermore, CBSE serves as a basis for recent software development approaches that have successfully been applied to robotic projects, such as the BRICS component model [67] that builds models using components and it then can automatically transform those component-based models into a specific software framework (e.g. OROCOS[1]).

Other approaches have sought to capitalize on CBSE by proposing software development techniques that, for instance, use domain-specific language to improve deployment[2] for a robot in a robotics competition [68]; or by allowing run-time adaptation of components in order to update the behavior of a robot in reaction to changes in its environment [69]. As this section is meant to provide only a notion of CBSE, the interested reader is referred to [2, 70] which covers this topic in greater detail.

In the context of this thesis, we apply CBSE to our proposed pipeline as shown in Figure 4.2, where a component is described as a computational unit with specified interfaces

[1] http://orocos.org/

[2] In this context, deployment refers to the process of making the software ready for an application, e.g. how components should be executed (as processes or threads) or in which computer (if there are multiple PCs or robots).

that has the ability to exchange information with other components through its interfaces. Interfaces can thus be considered as the external visible parts of the components and allow clients (e.g. components interacting with a given component) to be protected from changes in the component. The interfaces of a component can be provided and/or required. Provided interfaces denote the component's functionalities that are available to other components, while required interfaces describe the dependencies of the component.

In our pipeline (see Figure 4.2), the `sensor model` provides the contact information interface which is required by the `force transformer`. The `deformation model` requires a rest mesh and the interface provided by the `force transformer`, namely nodal forces, to provide the mesh interface.

4.2 Force transformer

This section describes the `force transformer` component, as shown in Figure 4.2, and how it serves as a connection between the `sensor model` component and the `deformation model` component. Thus, this component must provide the nodal forces interface to the `deformation model` that specify the three-dimensional forces acting on each node[3] of a mesh describing the deformable object. Conversely, it requires the contact information interface which contains the forces deforming the object as computed by the `sensor model` and represented on the sensor's frame.

In order to transform the contact information into the required nodal forces, two steps are required, namely, the forces must first be transformed into a common frame and subsequently they must be distributed on the mesh's nodes. The first step can be achieved in a straightforward manner by using the object frame as the common frame to transform the forces that are expressed with respect to a given sensor, e.g.:

$$\mathbf{f}_n^o = \mathbf{T}_n^o \mathbf{f}_n, \tag{4.1}$$

[3]The reasons for this requirement to apply forces directly on the nodes of a mesh are described in Section 4.3.

where $\mathbf{T}_n^o$ is a transformation matrix relating the n-th sensor frame to the object frame. Once the force is expressed with respect to the object frame, it can then be distributed among the surface nodes of the mesh. To do so, it is necessary to decide which nodes will "receive" this force. One way to select the nodes on which to apply the force is by using K-neighbors, as this would determine those k nodes that are closest to the force location. This computation can either be continuously applied (i.e. the force will be applied to different nodes) or just be executed at the beginning, e.g. assuming there is no slippage during grasping and thus the nodes where the force is applied would remain fixed.

Once the nodes have been selected, the force distribution can be simplified by setting the number of nodes to three. This allows the application of a linear shape function $\mathbf{H}$ to distribute the force onto the nodes in an inversely proportional manner based on the area coordinate of the node.

$$\mathbf{H} = \begin{bmatrix} \frac{a_1}{A} & 0 & 0 & \frac{a_2}{A} & 0 & 0 & \frac{a_3}{A} & 0 & 0 \\ 0 & \frac{a_1}{A} & 0 & 0 & \frac{a_2}{A} & 0 & 0 & \frac{a_3}{A} & 0 \\ 0 & 0 & \frac{a_1}{A} & 0 & 0 & \frac{a_2}{A} & 0 & 0 & \frac{a_3}{A} \end{bmatrix} \tag{4.2}$$

$$\begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{f}_3 \end{bmatrix} = \mathbf{H}^T \mathbf{f}^o \tag{4.3}$$

where $\mathbf{f}_i$ is a force vector applied to node i on the X, Y and Z axes and $\mathbf{f}^o$ represents the three-dimensional force with respect to the object. The total area of the triangle where the force is applied is denoted by A, and a_i stands for the sub-triangle area formed between the opposite nodes of the i-node and the contact point. An example of a force being distributed on three surface nodes of a tetrahedral element is shown in Figure 4.2.

4.3 Deformation model

The `deformation model` component, shown in Figure 4.2, requires the nodal forces interface provided by the component described in the previous section and an additional

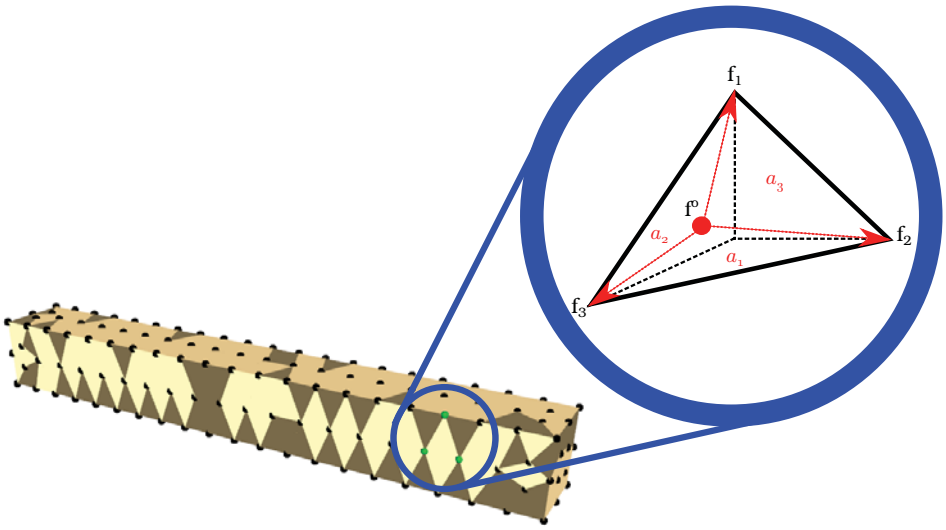


FIGURE 4.2: Visual representation of the force distribution on three nodes of the mesh using a linear shape function.

interface, rest mesh, that describes the initial coordinates of the nodes when the deformable object at its rest state (e.g. undeformed state). It provides in turn the mesh interface, which represents the current shape of the object as it deforms due to the forces applied on it.

Before we detail the computations required for this component we present some technical background[4] on the deformation of elastic materials. The following sections will describe the specific method and model used in this thesis, namely, the finite element method and the co-rotational linear elasticity FEM model are employed to simulate deformations. For other methods and models the reader is referred to [5, 71].

4.3.1 Deformation of elastic objects

A linearly elastic deformable body is defined by its rest configuration (e.g. undeformed shape) and by its material parameters (e.g. Young's modulus and Poisson's ratio for isotropic materials), which determine how the body deforms when external forces are applied to it [5]. The rest configuration of a body can be described as a set of particles or points $\mathbf{X} \in \mathbb{R}^{3n}$. In a general sense, the deformation of a body can be caused by rigid body motions (such as translations and rotations) or due to changes in its shape. However, in this thesis, we will refer to the deformation of a body when there is a relative displacement between the body's particles. This displacement of a particle is

[4] A short introduction to basic deformation terms is presented in Appendix A.2.

specified by a displacement vector, and using a displacement field, which is the collection of displacement vectors for all the particles in a body, we can relate the deformation of a body with its rest configuration by the following Lagrangian description:

$$\mathbf{x} = \boldsymbol{\phi}(\mathbf{X}, t) \tag{4.4}$$

where t represents the time, $\mathbf{x}$ is the deformed configuration, $\mathbf{X}$ is the rest configuration and $\boldsymbol{\phi}$ is a deformation function [71], an illustration of this function is shown in Figure 4.3. By inspecting Equation 4.4, we can derive the Jacobian of the deformation mapping, commonly referred to as the *deformation gradient*:

$$\mathbf{F} = \begin{pmatrix} \frac{\partial \phi_x}{\partial X_x} & \frac{\partial \phi_x}{\partial X_y} & \frac{\partial \phi_x}{\partial X_z} \\ \frac{\partial \phi_y}{\partial X_x} & \frac{\partial \phi_y}{\partial X_y} & \frac{\partial \phi_y}{\partial X_z} \\ \frac{\partial \phi_z}{\partial X_x} & \frac{\partial \phi_z}{\partial X_y} & \frac{\partial \phi_z}{\partial X_z} \end{pmatrix} \tag{4.5}$$

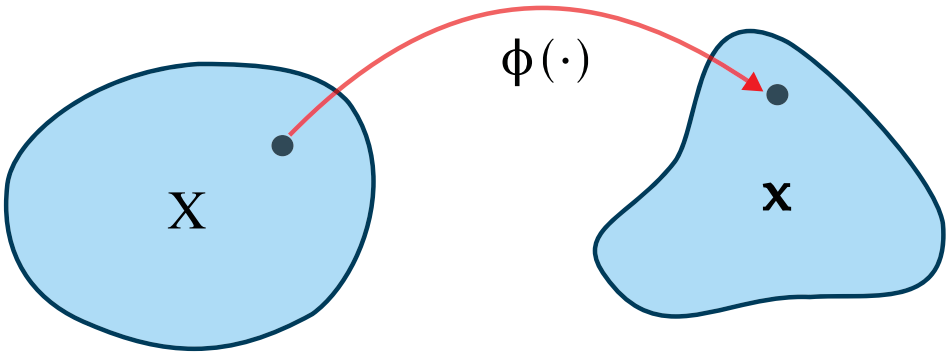


FIGURE 4.3: Example of a deformation map $\boldsymbol{\phi}$ from to the rest configuration $\mathbf{X}$ to the deformed configuration $\mathbf{x}$.

From the deformation map $\boldsymbol{\phi}$, the elastic strain[5], a measure that captures how much a configuration has deformed relative to the rest configuration, can be computed using the Green-Lagrangian strain tensor $\boldsymbol{\varepsilon}$ as follows:

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left[\nabla\boldsymbol{\phi} + (\nabla\boldsymbol{\phi})^T + (\nabla\boldsymbol{\phi})^T \cdot \nabla\boldsymbol{\phi}\right] \approx \frac{1}{2}\left[\nabla\boldsymbol{\phi} + (\nabla\boldsymbol{\phi})^T\right] \tag{4.6}$$

[5]Strain refers to the displacement between the particles of an object with respect to their distribution when the object is at rest, i.e. there is no strain for translation or rotation of the body.

where the approximation neglects the second-order terms to produce a linearized strain tensor and ∇ denotes the gradient. The dynamics of elastic materials are governed by the following partial differential equation (e.g. the equation of motion):

$$\rho\ddot{\mathbf{x}} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{f} \tag{4.7}$$

where ρ is the mass density of the material, $\mathbf{f}$ is the force applied to the object and $\ddot{\mathbf{x}}$ represents the acceleration of the points describing the object. If we assume linear elasticity, the stress $\boldsymbol{\sigma}$ in an element can be computed given the element strain (as defined in Equation 4.6) using Hooke's law:

$$\boldsymbol{\sigma} = \mathsf{C}\boldsymbol{\varepsilon} \tag{4.8}$$

where C is the elasticity tensor that relates the coefficients of the strain and stress tensors, and the Cauchy stress tensor $\boldsymbol{\sigma}$ has the following form:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \tag{4.9}$$

For isotropic materials, C depends only on the Young's modulus and Poisson's ratio [5] and it is defined as:

$$\mathsf{C} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \tag{4.10}$$

where E and ν are the Young's modulus and Poisson's ratio, respectively; and the relation between stress and strain is performed using their associated vectors, namely, $\boldsymbol{\sigma} = [\sigma_{xx},\ \sigma_{yy},\ \sigma_{zz},\ \sigma_{xy},\ \sigma_{yz},\ \sigma_{zx}]^T$ and $\boldsymbol{\varepsilon} = [\varepsilon_{xx},\ \varepsilon_{yy},\ \varepsilon_{zz},\ \gamma_{xy},\ \gamma_{yz},\ \gamma_{zx}]^T$.

Since these deformation quantities are defined in a continuous space, it is necessary, in order to numerically solve for them, to discretize their respective equations. To achieve this discretization, it is common to apply the finite element method which will be described next.

4.3.2 Finite element method

The finite element method (FEM) can be used to simulate the deformation of an object by dividing it into small elements (e.g. tetrahedra or voxels) and then solving for the strains and stress of each element [72]. Once the object is discretized using FEM as a mesh of connected elements, the positions of their vertices and the elastic forces are related via a stiffness matrix $\mathbf{K}_e$ defined for each element e as follows:

$$\mathbf{f}_e = \mathbf{K}_e \mathbf{q}_e \tag{4.11}$$

The stiffness matrix can be computed for each element with the following integral:

$$\mathbf{K}_e = \int_{\Omega_e} \mathbf{B}_e^T \mathsf{C} \mathbf{B}_e \quad d\Omega_e \tag{4.12}$$

with $\mathbf{B}$ being a strain-displacement matrix defined as:

$$\mathbf{B} = [\mathbf{B}_1, \ \mathbf{B}_2, \ \mathbf{B}_n \ldots] = \boldsymbol{\Delta}\mathbf{H} \tag{4.13}$$

where n represents the number of vertices, also called nodes, in the mesh, $\boldsymbol{\Delta}$ is the derivation operator and $\mathbf{H}$ represents a matrix containing shape functions for each node. The strain-displacement for each node i can thus be computed as:

$$\mathbf{B}_i = \begin{bmatrix} \frac{\partial \mathbf{H}_i}{\partial x} & 0 & 0 \\ 0 & \frac{\partial \mathbf{H}_i}{\partial y} & 0 \\ 0 & 0 & \frac{\partial \mathbf{H}_i}{\partial z} \\ \frac{\partial \mathbf{H}_i}{\partial y} & \frac{\partial \mathbf{H}_i}{\partial x} & 0 \\ 0 & \frac{\partial \mathbf{H}_i}{\partial z} & \frac{\partial \mathbf{H}_i}{\partial y} \\ \frac{\partial \mathbf{H}_i}{\partial z} & 0 & \frac{\partial \mathbf{H}_i}{\partial x} \end{bmatrix} \tag{4.14}$$

By summing all the stiffness matrices of all elements we can obtain the stiffness matrix $\mathbf{K}$, which, using Newton's second law, allows for the computation of the elements motion:

$$\mathbf{f}_{ext} = \mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} \tag{4.15}$$

where $\mathbf{f}_{ext}$ is the external force caused by gravity and contacts. The position, velocity and acceleration of each node n is represented by $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$, respectively, with $\mathbf{q} \in \mathbb{R}^{3n}$; $\mathbf{M} \in \mathbb{R}^{3n \times 3n}$ represents the mass matrix and $\mathbf{D}$ is the damping matrix.

4.3.3 Co-rotational linear elasticity

In this thesis, we chose the co-rotational linear elasticity model since it provides a balanced trade-off between computational complexity and accuracy. It achieves this by combining nonlinear characteristics to guarantee rotational invariance with a linear relationship between stress and deformation [71]. The co-rotational linear elasticity model avoids issues encountered in purely linear models (e.g. inflated volumes when the object is subject to large deformations), by assuming the deformation of the object, for each mesh element, to be composed of a rotation plus a small amount of deformation. This rotational component $\mathbf{R}_e \in \mathbb{R}^{3\times3}$ is computed via polar decomposition of the element deformation gradient $\mathbf{F}$. We use the implementation offered by Vega FEM [4], a self-contained C/C++ library for simulating three-dimensional deformable objects, which is formulated with the following linear expression:

$$\mathbf{f}_e = \hat{\mathbf{R}}_e \mathbf{K}_e \left(\hat{\mathbf{R}}_e^T \mathbf{q}_e - \mathbf{q}_e^0 \right) \tag{4.16}$$

where $\hat{\mathbf{R}}_e \in \mathbb{R}^{12\times12}$ is a block diagonal matrix formed by four $\mathbf{R}_e$ matrices, $\mathbf{q}_e$ is the current position and $\mathbf{q}_e^0$ represents the rest configuration of the nodes on an element. Since the elements are defined as tetrahedra in a three-dimensional space, both $\mathbf{f}_e$ and $\mathbf{q}_e$ are 12-dimensional vectors.

4.4 Integration of components

To allow the communication between the components described in the previous sections, we implemented them as nodes in the robot operating system[6] (ROS), a software framework that handles communication via:

- **topics:** are used for continuous data streams (e.g. sensor output, mesh state). The data is published by senders and subscribed by receivers and so the sender (publisher) decides when the data is published, which triggers a callback in the receiver (subscriber).
- **services:** are ideal for performing quick calculations or requesting specific data from a component (referred to as a node in ROS). As services block calls, they should not be used if preemption is required, e.g. when a robot must be stopped to avoid damage to itself or its surroundings.
- **actions:** unlike services, actions can be preempted. During an action, the component executes the desired task while providing feedback. This type of communication is ideal for tasks that require considerable time (e.g. several seconds) to execute while not blocking the component. Examples for the usage of actions, include object recognition and speech recognition.

Since we are concerned mainly with sensor updates (provided by the `sensor model`) and the current state of the mesh, we use topics to establish the communication between the different components in our shape sensing pipeline. As shown in Figure 4.1, the `sensor model` provides the contact information interface which is implemented as a wrench message that is continuously published by this component. This wrench message is composed of a three-dimensional force, a three-dimensional torque and a reference

[6] http://www.ros.org/

frame (in this case the sensor frame). The `force transformer` subscribes to this wrench message, transforms it as described in Section 4.2, and publishes a force array message (nodal forces interface) that contains a three-dimensional force for each node on the mesh. The `deformation model` subscribes to this force array message and to the rest mesh to continuously publish the updated mesh state that describes the deformations of the object being manipulated.

The implementation of the `deformation model` component, requires to step Equation 4.15 forward in time subject to initial and boundary conditions. The initial conditions are the initial positions and velocities of the mesh's nodes and the boundary conditions are constrained nodes specified by the user. The constrained nodes refer to nodes that remain fixed, that is, they do not move even if deformation forces are applied to them. Additionaly, the elasticity parameters of the object's material such as the Young's modulus and Poisson's ratio are required by the model. To integrate this component in ROS, we wrote a wrapper code for the open-source library Vega FEM [4].

An example of the simulation of a deformable object using our proposed shape sensing pipeline is shown in Figure 4.4. Here, a graphical user interface (GUI) replaces the `sensor model` as the provider of the contact information interface, which is then used by the rest of the components.

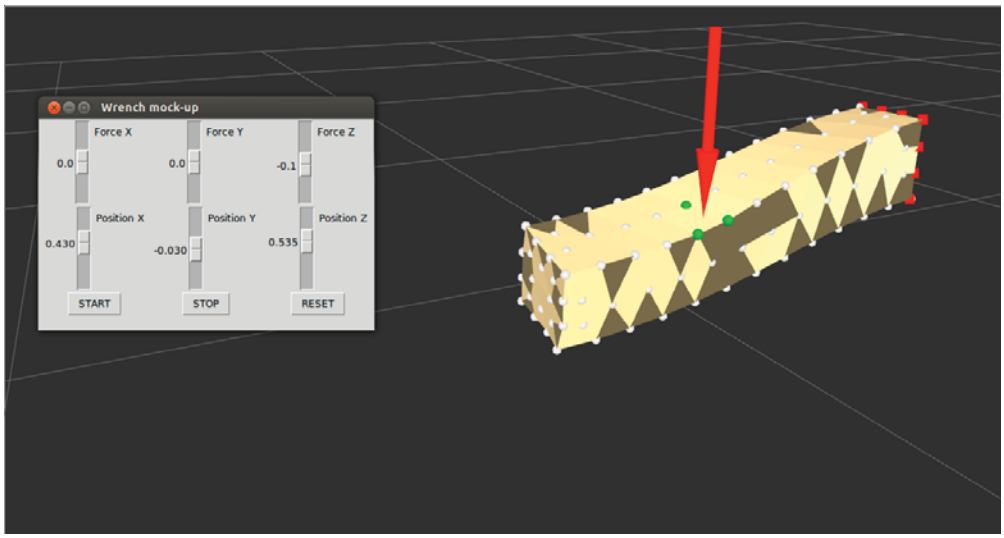


FIGURE 4.4: Screenshot of the simulation of a deformable, where the GUI acts as the `sensor model` component. The nodes of the mesh are shown in white, while the green spheres indicate the nodes where the force is being applied and the red squares represent the constrained nodes.

4.5 Applications

Using the shape sensing pipeline presented in the beginning of the chapter, we can perform robotic tasks on deformable objects such as estimating the deformation of an object while it is manipulated, as well as controlling its shape. In this chapter, we will review three applications where we have used our proposed pipeline, namely, 1) shape estimation of an object using tactile sensing; 2) shape estimation of an object using force sensing and 3) shape control of a deformable object. The implementation of each application will be described in the following sections, along with its evaluation and a discussion of the performance results.

Disclaimer: The meshes used in the experimental evaluation of the applications were generated by Belhassen-Chedli Bouzgarrou and the ground truth measure for Section 4.5.1 using vision was developed by Carlos M. Mateo.

4.5.1 Tactile-based shape sensing

By relying on tactile information, the shape sensing pipeline can be applied to estimate the shape of an object that is being grasped by a robotic hand with tactile sensors on its fingertips, as shown in Figure 4.5. In doing so, in-hand manipulation of deformable objects can profit from this application. For instance, shape sensing using tactile information can be a complimentary skill to the approach proposed by Ficuciello et al. [49], where the shape of an object is controlled using a dexterous robot hand without any feedback on its actual shape.

We implemented the use case of shape sensing of a deformable object for a potential application of in-hand manipulation task using our proposed pipeline, as described in Chapter 4, and using the Shadow Dexterous Hand[7] equipped with tactile sensors on its fingertips (see Figure 4.5). Thus, we make use of the tactile sensor model presented in Section 3.3 and couple it with the rest of the pipeline components to estimate the deformation of an object using tactile data.

[7] https://www.shadowrobot.com/products/dexterous-hand/

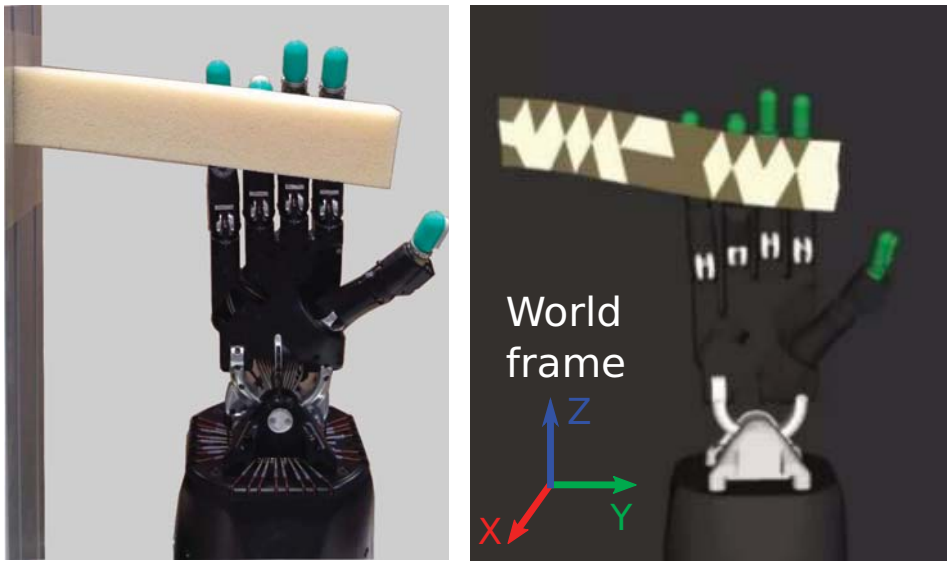


FIGURE 4.5: Shape estimation of a deformable object based on tactile sensing. On the left, the real shape is shown and the shape estimated by our proposed pipeline is shown on the right.

To evaluate the performance of this application we used nine test objects[8] with three shapes (see Figure 4.6 and Table 4.1) and three different material properties (see Table 4.2). The elasticity parameters of the materials were experimentally obtained as described in Appendix B, and the meshes for the objects were generated using the commercial software ANSYS[9].

TABLE 4.1: Geometric information of the test objects used in shape sensing using tactile data.

	Dimensions (cm)			Mesh	
	Length	**Width**	**Height**	**Nodes**	**Elements**
Cube	6	6	6	153	486
Sponge	8	5	2	118	304
Bar	20	4	4	152	385

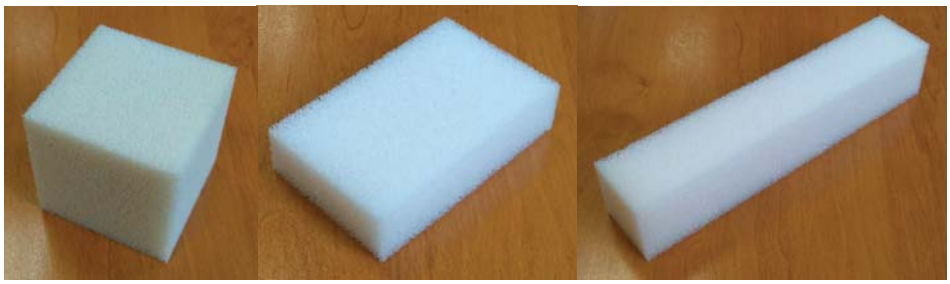


FIGURE 4.6: Test objects: cube (hard), sponge (medium) and bar (soft).

[8]The objects were bought from the following vendor: http://www.moussesurmesure.com/
[9]https://www.ansys.com/

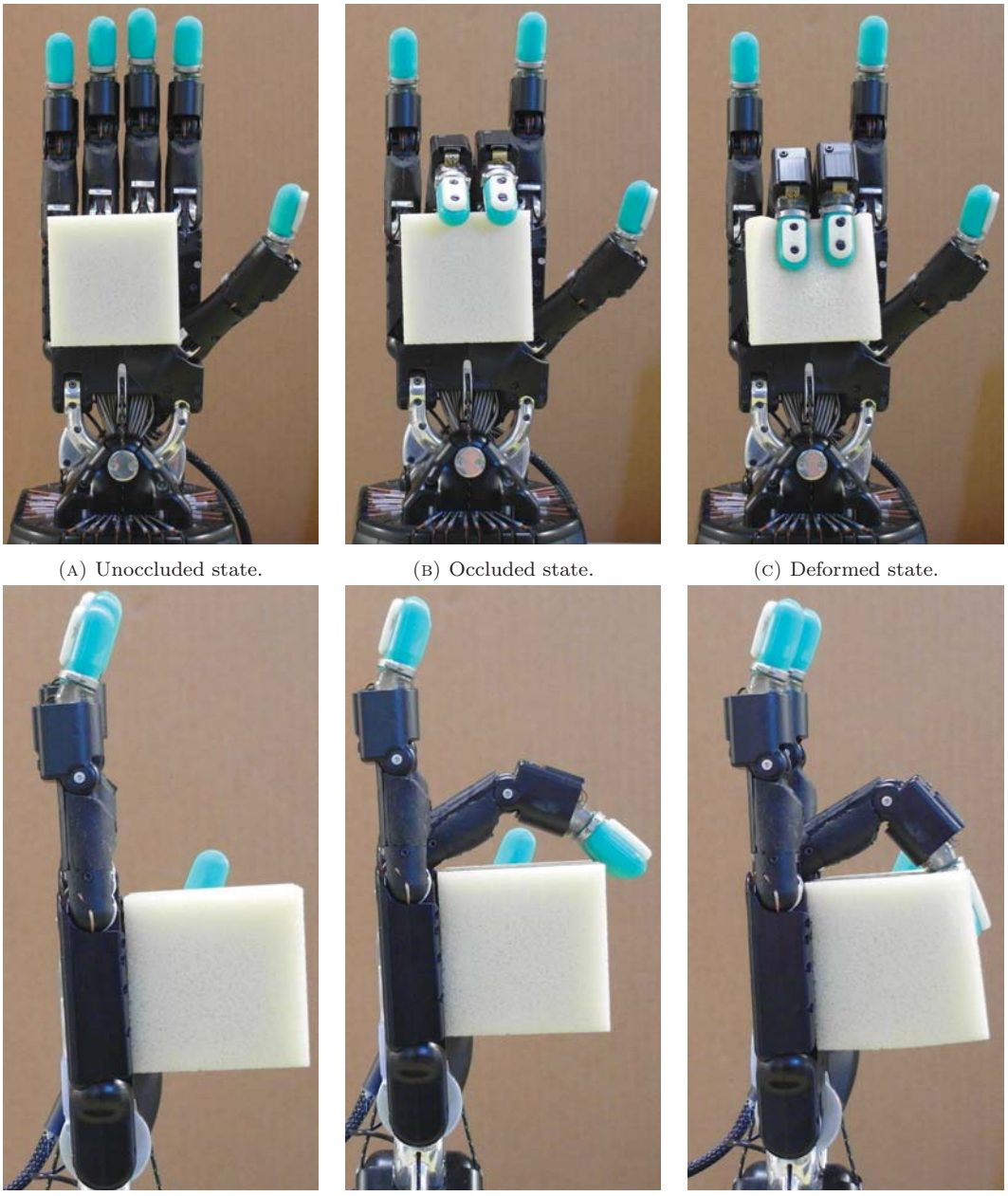
TABLE 4.2: Material properties of the test objects used in the tactile-based shape sensing application.

		Elasticity parameters		
	Material name	**Mass density** (kg/m^3)	**Young modulus** (Pa)	**Poisson ratio**
Hard	HR 45	45	3800	0.15
Medium	Bultex 30	30	3200	0.15
Soft	Bultex 26	26	3000	0.15

The objects were then deformed by moving the fingers of the robot hand while the tactile sensors made contact with the object. The cube objects were grasped using two fingers, while the rest of the objects were pushed by a finger of the robot hand as they were fixed, using double sided tape, on their sides with their longest axis being parallel to the Y axis, as shown in Figure 4.5. The sponge and bar objects start from an *undeformed* state and end in a *deformed* state. Figure 4.7 shows the states used for the cube objects, namely, when the object is fully visible, *unoccluded*; once contact has been made but without deformation, *occluded*; and finally, the *deformed* state.

Since a method to measure the shape of an object while it deforms is not yet available, we propose a vision-based method to asses the accuracy of the shape sensing pipeline using tactile data. The method consists in using the similarity between two point clouds, namely, a *measured* point cloud and a *simulated* point cloud. The measured point cloud is generated using a Microsoft Kinect (v1) RGB-D sensor that is placed in front of the test object while the object was fixed on a test rig and deformed by the fingers of the Shadow hand, as depicted in Figure 4.8. An example of a measured point cloud can be seen in Figure 4.9a. The simulated point cloud was generated by placing a virtual Kinect sensor at the same position as the real sensor and replacing the real object with the mesh output of the pipeline. Then, using ray tracing, the virtual Kinect generated the simulated point cloud from the mesh as shown in Figure 4.9b.

In order to generate the measured point cloud it is necessary to first segment the test object from other objects in the scene and from the background. To do so, we implemented the color-based segmentation proposed in [73], which uses similarity in color and spatial proximity to create clusters, where the cluster representing the object to be tracked is selected manually by the user. This clustering can be seen by the different colored point clouds in Figure 4.10b, where the test object is marked by a yellow circle. The unsegmented point cloud is shown in Figure 4.10a. Once we have both, measured



(A) Unoccluded state. (B) Occluded state. (C) Deformed state.

(D) Unoccluded state. (E) Occluded state. (F) Deformed state.

FIGURE 4.7: A cube-like object tested in the three states. Front view is shown on the top row and a side view is shown on the bottom row.

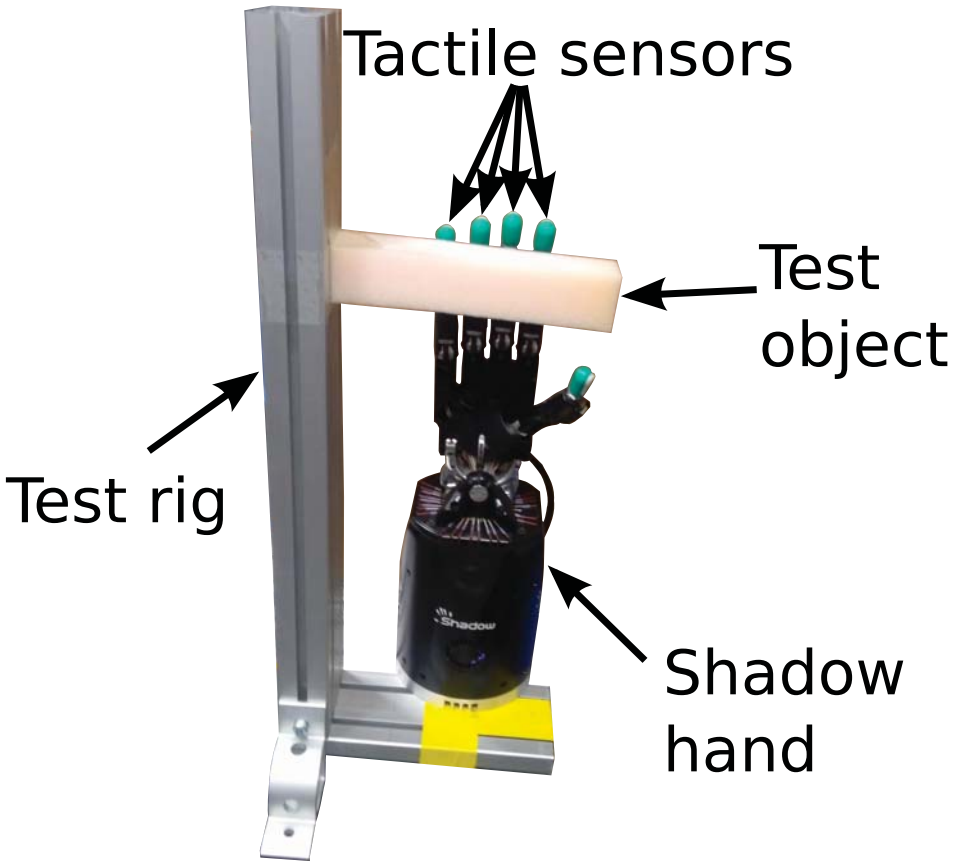
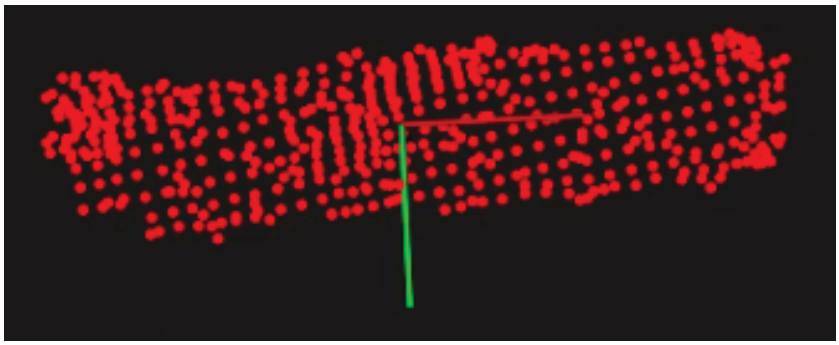


FIGURE 4.8: Experimental setup to evaluate the performance of the shape sensing pipeline with tactile data for a bar-like object.

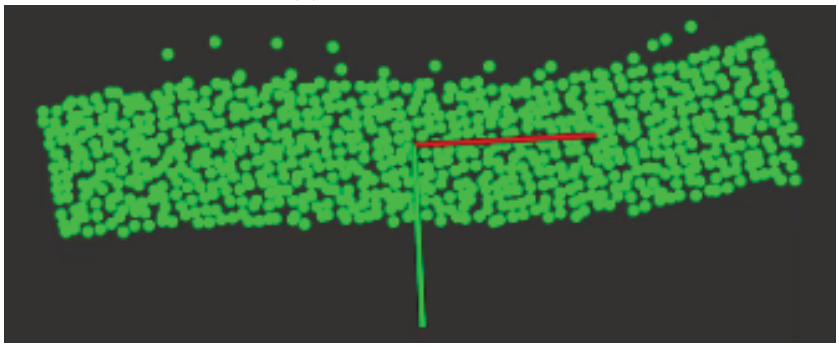
and simulated, point clouds representing the test object we can measure their similarity using octrees[10]. Specifically, we generated an octree from the measured point cloud using a minimum leaf size of 1 *cm* (an example of an octree can be seen in Figure 4.9c). Then, we checked if the points from the simulated point cloud were inside of the octree leaves and define the accuracy as the ratio of points that are inside.

We applied this method to evaluate the accuracy of the shape sensing pipeline on the nine different test objects and the accuracy results are summarized in Figure 4.11. The bar plots show the similarity scores between the measured and simulated point clouds for three states of the cube-like objects (unoccluded, occluded and deformed) and for the deformed and undeformed states for the remaining test objects.

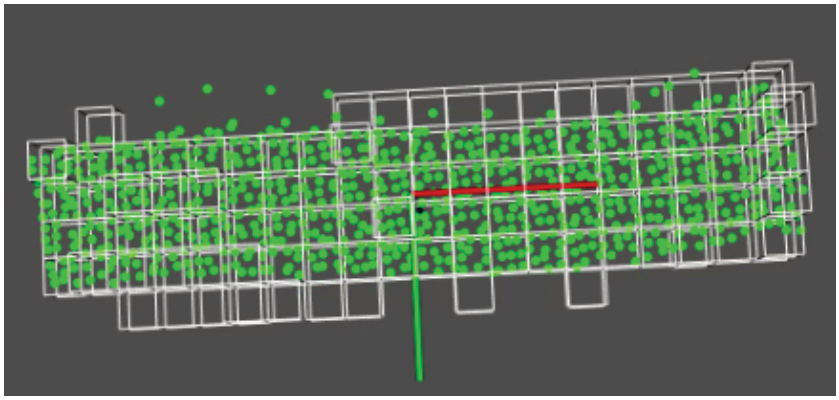
[10] An octree is a tree data structure where its internal nodes have eight leaves (or children).



(A) Measured point cloud.



(B) Simulated point cloud.

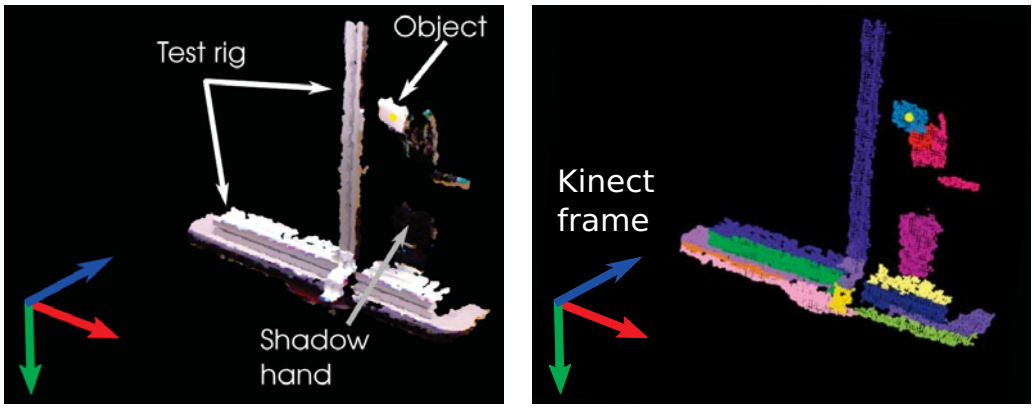


(C) The generated octree, where the minimum leaf size is 1 cm.

FIGURE 4.9: Similarity evaluation of a bar-like object using RGB-D data: (A) point cloud as measured by the Kinect, (B) point cloud generated by a virtual Kinect based on the output mesh of the proposed approach, (C) octree (white) generated from the measured point cloud to measure the similarity with the simulated point cloud (green).

Discussion

At first sight, the results shown in Figure 4.11 appear to demonstrate a poor accuracy of the shape sensing based on tactile data. However, it must be noted that the accuracy of the proposed approach should be measured against the accuracy of the similarity measure described in Section 4.5.1 which does not provide a 100% accuracy for the *occluded* and *undeformed* states. This results in an average accuracy rate of around



(A) Raw point cloud. (B) Segmented point cloud.

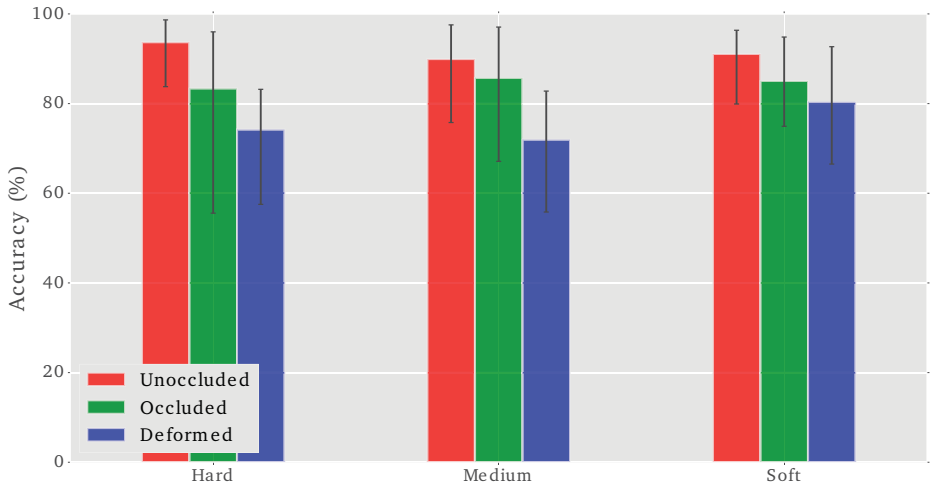
FIGURE 4.10: Point cloud segmentation for a sponge-like object. The reference frame marks the pose of the Kinect sensor.

85%. Also, it must be noted that for the cube objects, occlusions cause the similarity measure to significantly decrease in accuracy as it can be seen in Figure 4.11a.

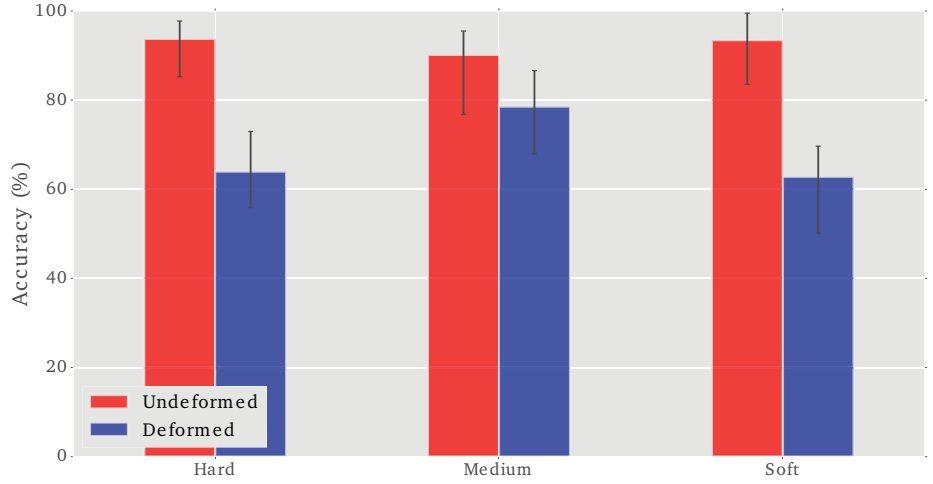
The performance of the pipeline on sponge objects clearly suffered as it can be seen in Figure 4.11b. This is due to the small size of these objects, since the similarity measure builds an octree based on the measured point cloud, which results on fewer leaves of the octree. Thus, the similarity rate is more sensitive to noise errors, i.e. points that are outside of the octree (as depicted in Figure 4.9c).

Regarding the high variance in the accuracy results of the similarity measure shown in 4.11, they are in accordance to those shown when no deformation occurred (e.g. during the *occluded* and *undeformed* states). Therefore, the variance in these results can be attributed to the similarity measure rather than to the shape sensing pipeline.

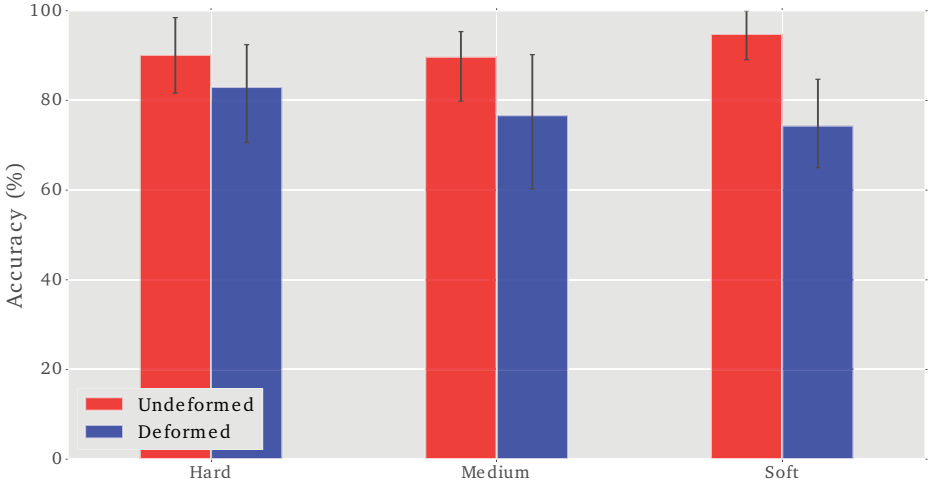
Other factors that contribute to the accuracy errors are, for instance, that the proposed method is highly dependent on the location of the contacts. Thus, errors in the robot model in simulation and between the object's real pose and its pose in simulation have a direct impact on the performance of the pipeline. Another source of error, resulting in the oscillation of the mesh, is due to intermittent contacts caused by the softness of the objects as the tactile sensors fail to detect these contacts. Another error stemming from the tactile sensors is that they cover a small surface of the fingers. This results on the finger contacting the object with parts of the finger's surface that are not covered by the tactile sensor causing the object to deform but without any sensor feedback.



(A) Accuracy results for the cube objects.



(B) Accuracy results for the sponge objects.



(C) Accuracy results for the bar objects.

FIGURE 4.11: Evaluation results of the deformation sensing.

4.5.2 Force-based shape sensing

When objects are significantly large and thus cannot be handled inside a robotic hand, it is necessary to manipulate them using a robot arm or, in some cases, a dual-arm robot. One example of deformable object manipulation using a dual-arm robot is presented in [50], where, similarly as in [49], a model of the object is used to control the shape but without relying on a feedback signal representing the object's shape. To address this deficiency, and as the deformation forces are caused by the motion of an arm rather than robot fingers, we combine the force sensor model described in 3.4 with our shape sensing pipeline.

To evaluate the accuracy of the shape sensing pipeline using force data to estimate deformations, we devised an experiment consisting in a KUKA LWR+4 robot arm [74] with an attached dexterous Shadow Dexterous Hand at its end and an ATI Gamma force-sensor between them. The experimental test, with the setup shown in 4.12, was conducted on four elastic objects with different shapes and material properties. The two shapes of the objects are described in Table 4.3 and the materials properties are shown in Table 4.4. During the test, the test object were fixed on a test rig, where the bar objects were fixed such that their long axis were parallel to the XY plane of the robot frame and the block objects were attached by their bottom side, as seen in Figures 4.13 and 4.14

TABLE 4.3: Geometric information of the test objects used in shape sensing using force data.

	Dimensions (cm)			**Mesh**	
	Length	**Width**	**Height**	**Nodes**	**Elements**
Block	6	40	40	360	1079
Bar	6	6	50	207	536

TABLE 4.4: Material properties of the test objects used in shape sensing using force data.

		Elasticity parameters		
	Material name	**Mass density** (kg/m^3)	**Young modulus** (Pa)	**Poisson ratio**
Hard	HR 45	45	18500	0.15
Soft	Bultex 26	26	9000	0.15

The test for the bar-like objects consisted on moving the robot arm to follow a set of six poses in XZ plane of the robot frame, as shown in Figure 4.13. The block-like

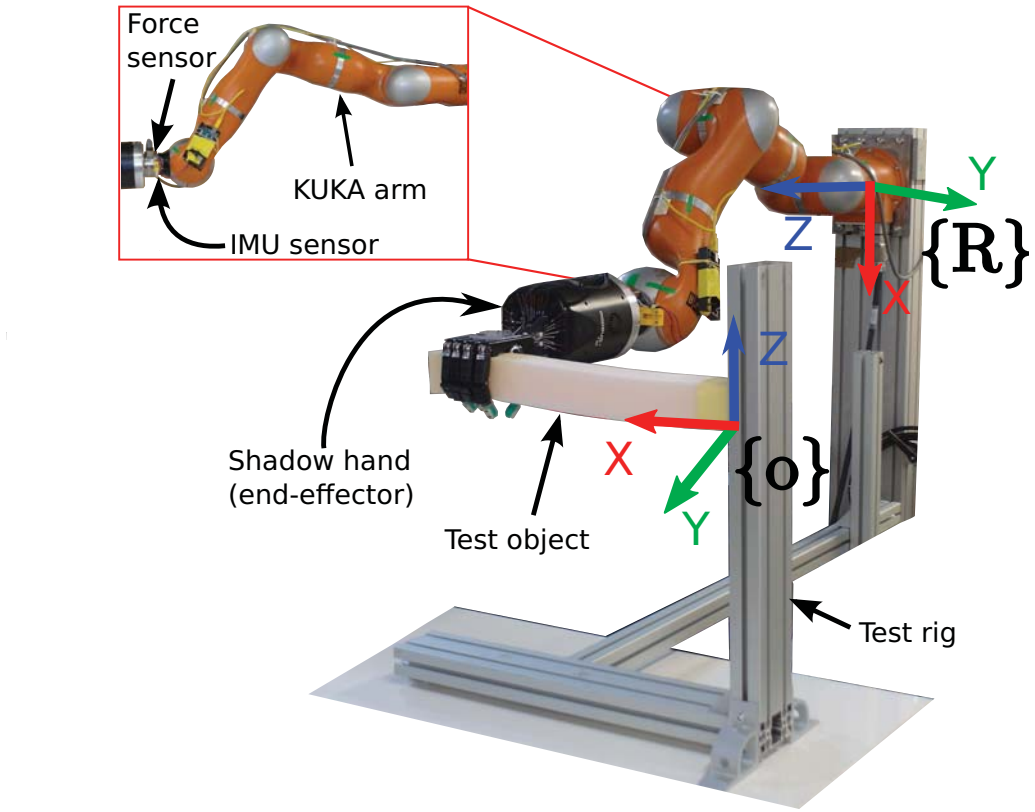


FIGURE 4.12: Experimental setup for a bar-like object.

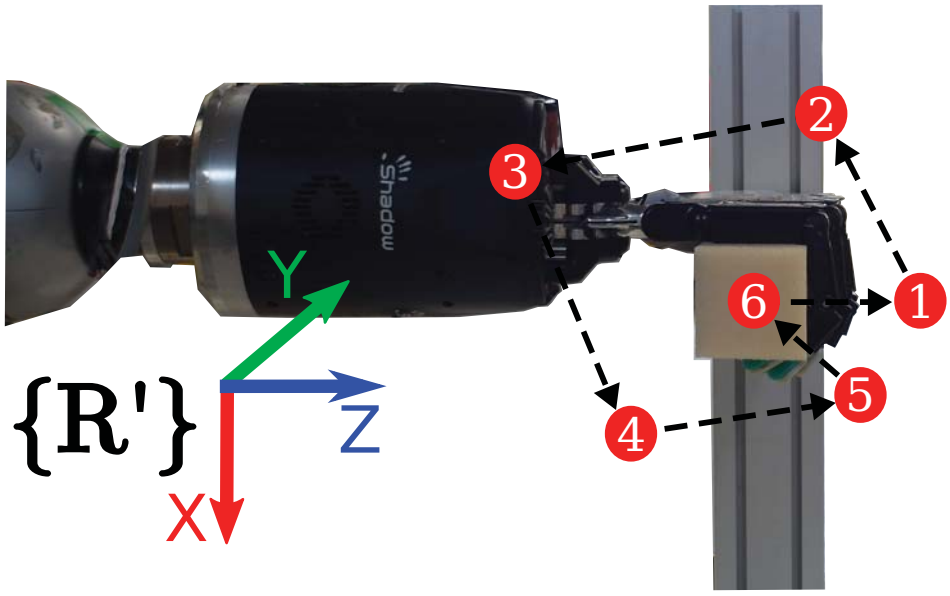


FIGURE 4.13: Example of the path to follow the six test poses by the bar objects during the sensing evaluation. The $\mathbf{R}'$ denotes a reference frame having the same orientation as the robot base frame (see Figure 4.12) but a different translation in order to make it visible.

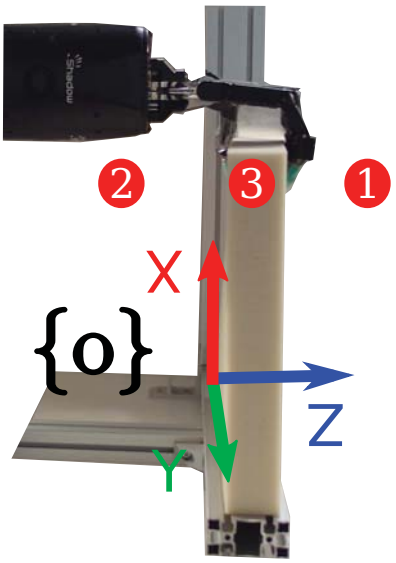


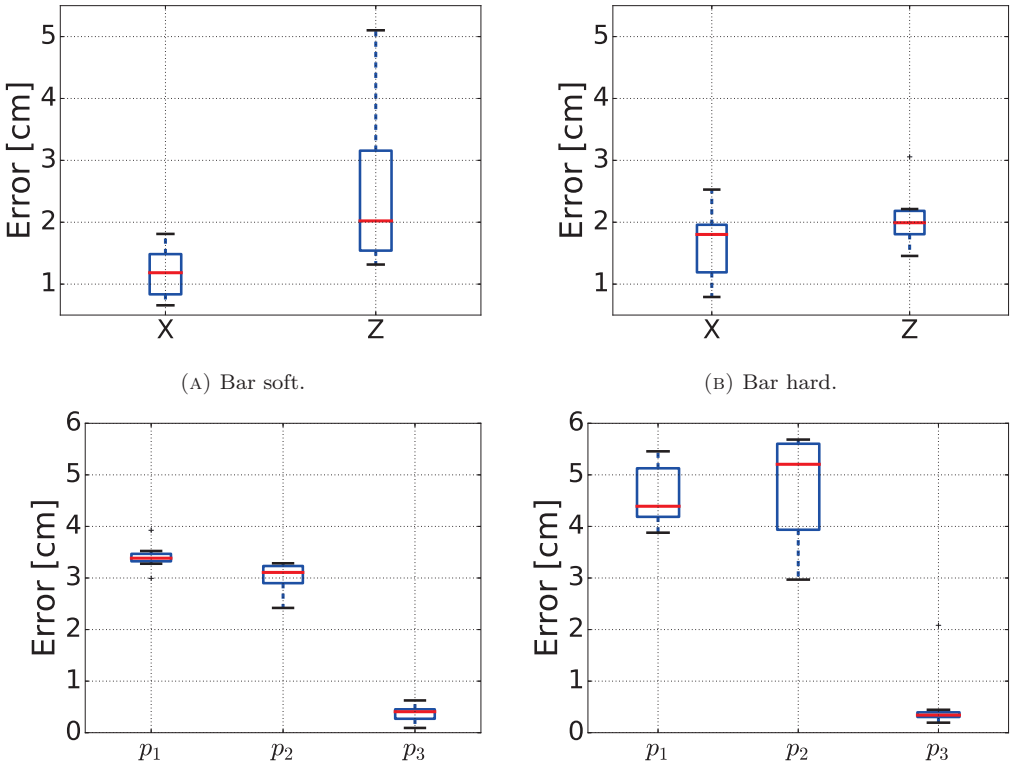
FIGURE 4.14: Test poses used for the block-like objects for the sensing evaluation.

objects were similarly tested by following a linear path along the Z axis, as depicted in Figure 4.13. The reason for this limited test motion is due to the block-like objects being fixed on their bottom side. For both set of tests, the error signal was defined as the distance per axis between the pose where the end-effector grasped the object, used as ground truth, and a pose[11] on the estimated mesh that was coincident with the end-effector pose at the beginning of the test (i.e. when the object was at rest). For each object, seven trials were performed. The results, shown in Figure 4.15, consist of the mean of the absolute error between the reference (end-effector pose) and measured (extracted from the estimated) positions which was computed for each trial. The errors for the bar objects are shown for the X and Z axes, while the errors for the block objects show the average of the seven trials for the three test poses.

Discussion

Unlike the evaluation of the previous application, instead of measuring the similarity of a complete face on the objects we only evaluate the difference in position between two arbitrarily selected points. Figure 4.15 plots the errors in the X and Z axes for the bar objects; and in the Z axis for the block objects. The plots show the average error between the reference and measured positions for all the trials, as described in Section 4.5.2, that is, six positions for the bar objects and three positions for the block objects.

[11] This pose was extracted on the mesh using the method outlined in Section 4.5.3.



(A) Bar soft.

(B) Bar hard.

(C) Block soft.

(D) Block hard.

FIGURE 4.15: Estimation errors for the shape sensing using force data on the four test objects.

The accuracy of the shape sensing using force data for the bar objects is on average less than two centimeters. However, observing Figure 4.15a, the estimation along the Z axis for the *bar soft* object exhibits considerable variation. This large variation can be attributed to the softness of the material, which fail to produce enough reaction forces for the force sensor to capture as the manipulator moved the object through the set of test poses. In contrast, the variation for the *bar hard* object, also along the Z axis, remained small as it can be seen in Figure 4.15b.

The performance of the shape sensing pipeline on block objects reflects consistent results as the variance is small across the different points and the two objects. Since the tests performed on these block objects were limited to only one axis (see Figure 4.14), there was no "crosstalk" between forces on different axes, as it was the case on the bar objects where force measurements on one axis affected the estimations on a different axis. Despite the small variance, the mean errors are considerable higher for the block objects, especially for the *block hard* object as it is shown in Figure 4.15d. These errors were due to the position of the test poses p_1 and p_2, which were at the limit of where

the objects could be stretched. This resulted in higher forces that were input to the deformation model which in turn produced a larger deformation. On the other hand, when the manipulator moved the object to the final pose p_3 (i.e. the initial position), the estimation was quite accurate as depicted in Figure 4.15c and Figure 4.15d.

Although the force-based shape sensing application produces a more accurate estimation of the shape compared to the tactile-based shape sensing application, mainly due to the higher accuracy of the force-torque sensor, it does suffer from similar issues such as the mismatch between the pose of the real object and the pose of the simulated object, the fact that material of the objects is nonlinear and errors due to delays between the components.

4.5.3 Shape control

Another potential application for the shape sensing pipeline is to control the shape of a deformable object. Here, we rely on the previous application (i.e. using the pipeline with force data) to manipulate the objects described in the previous section into a desired configuration. A similar setup using a KUKA robot arm with an attached Shadow robot hand, as the one depicted in Figure 4.12, is used for this shape control application.

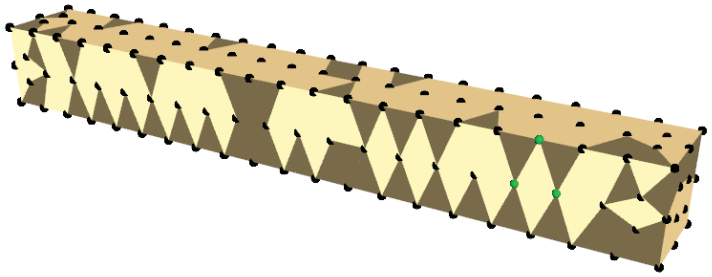


FIGURE 4.16: Simulated mesh of a bar-like object. The mesh nodes are shown in black and the nodes used to extract a pose are shown in green.

As controlling the complete shape of an elastic object, that is, all the nodes of its mesh, is an extremely underactuated problem; we instead control a single pose of the object. This pose is extracted from the object's mesh by selecting three suitable nodes, as the green nodes shown in Figure 4.16, as follows:

$$\mathbf{c} = \frac{1}{3}\sum_{i=1}^{3}\mathbf{p}_i \tag{4.17}$$

$$\mathbf{n} = (\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1) \tag{4.18}$$

$$s = \cos\left(\frac{\pi}{4}\right), \quad \mathbf{v} = \mathbf{n} \cdot \frac{\sin\left(\frac{\pi}{4}\right)}{\|\mathbf{n}\|_2} \tag{4.19}$$

$$\mathbf{x} = [\mathbf{c}, (s, \mathbf{v})]^T \tag{4.20}$$

where $\mathbf{p}_i$ is the position of the node at the $i-$th index of the mesh $\mathbf{q}$ and $\mathbf{n}$ represents the normal of the plane formed by the points. The position and orientation of the pose are given by the centroid $\mathbf{c}$ and the quaternion respectively, where s is the scalar part of the quaternion and $\mathbf{v}$ is the vector part.

Once we have extracted a pose from the mesh, referred to as *current pose* $\mathbf{x}_c$, we can command the robot arm to reach a *desired pose* $\mathbf{x}_d$ by defining the following error signal:

$$\mathbf{e} = \mathbf{x}_d - \mathbf{x}_c \tag{4.21}$$

In order to control the robot motion, we must transform the error $\mathbf{e}$ into a twist command by multiplying it by a diagonal gain matrix $\mathbf{\Lambda}$. This twist, expressed in Cartesian space, can then be mapped into joint space to compute the necessary joint velocities for the robotic arm as:

$$\dot{\boldsymbol{\theta}}_{des} = \mathbf{J}^{+}(\mathbf{\Lambda} \cdot \mathbf{e}) \tag{4.22}$$

where $\mathbf{J}^{+}$ is the MoorePenrose inverse Jacobian used for redundant manipulators. A diagram describing the shape controller can be seen in Figure 4.17.

As noted at the beginning of the section, a similar setup as the one described in Section 4.5.2 was used to evaluate the shape control application. However, we evaluate this application only on the *bar soft* and *block hard* objects (see tables 4.3 and 4.4). For the bar object we set poses on the XZ plane as shown in Figure 4.13. As previously mentioned, due to the block object being fixed at its bottom side, we only commanded

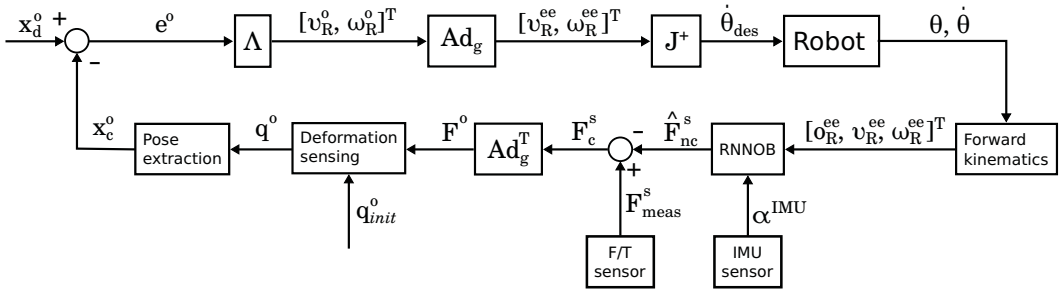


FIGURE 4.17: Block diagram of the proposed deformation controller. The controller uses the output of shape sensing pipeline based on force data to regulate an error signal $\mathbf{e}^o$ that is the difference between the current and desired poses, $\mathbf{x}_c^o$ and $\mathbf{x}_d^o$ respectively, were both are described w.r.t. the object frame. The deformation sensing block uses the initial undeformed configuration $\mathbf{q}_{init}^o$ of the mesh and the estimated contact force $\mathbf{F}^o$, both expressed in the object frame, to update the mesh configuration $\mathbf{q}^o$ as it deforms. From this mesh configuration, $\mathbf{x}_c^o$ is extracted by the method outlined in this section. As the robot expects the end-effector twist expressed in the robot base frame $\{R\}$, the twist expressed on the object frame, namely $[\boldsymbol{\upsilon}_R^o, \boldsymbol{\omega}_R^o]$, must be multiplied by an adjoint matrix $\mathbf{Ad}_g$ relating these two frames in order to obtain the desired twist ($[\boldsymbol{\upsilon}_R^{ee}, \boldsymbol{\omega}_R^{ee}]$).

poses along the Z axis for the block object as it can be seen in Figure 4.14. We used a graphical user interface to set the desired pose $\mathbf{x}_d$ which the current pose $\mathbf{x}_c$ should reach.

The following figures show the behavior of the proposed controller in the command-response plots, as well as the errors between them. The command signal refers to the desired pose $\mathbf{x}_d$ as determined by the user with the GUI; while the response (Mesh) represents the current pose $\mathbf{x}_c$. Additionally, the response (EE), referring to the pose of the end-effector[12], is shown to provide insight to the estimation accuracy of the shape sensing pipeline. The command-response plots show the command signal in green, the mesh response in red and the end-effector response in blue. Figure 4.18 depicts this plot for the bar object along the X and Z axes; and Figure 4.20 plots these signals for the block object along the Z axis.

The error plots for the bar and the block objects are shown in figures 4.19 and 4.21, respectively.

Discussion

As the evaluation of the shape control application directly depends on the performance of the previous application, namely the force-based shape sensing, its behavior is greatly

[12]The initial offset between this end-effector pose and the current pose extracted from the mesh was removed to make the plots clearer.

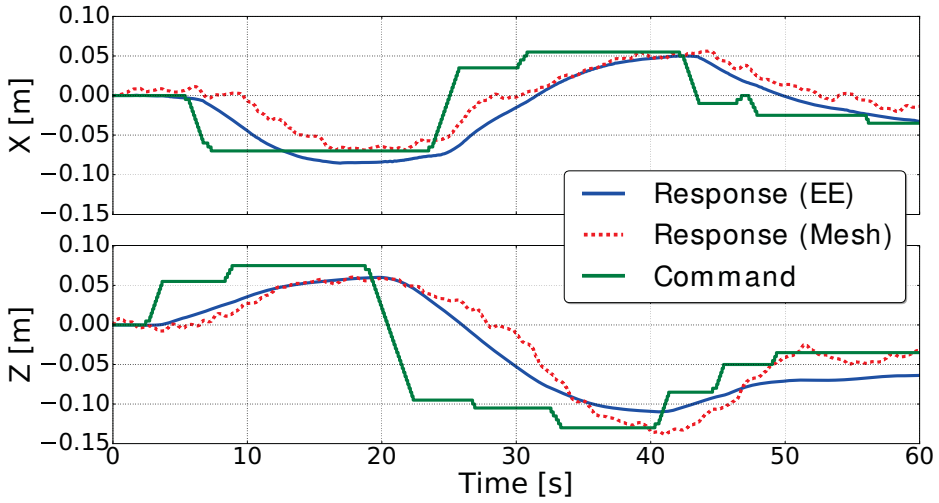


FIGURE 4.18: Command and responses of the mesh and the robot end effector (EE) along the X and Z axes for the bar soft object.

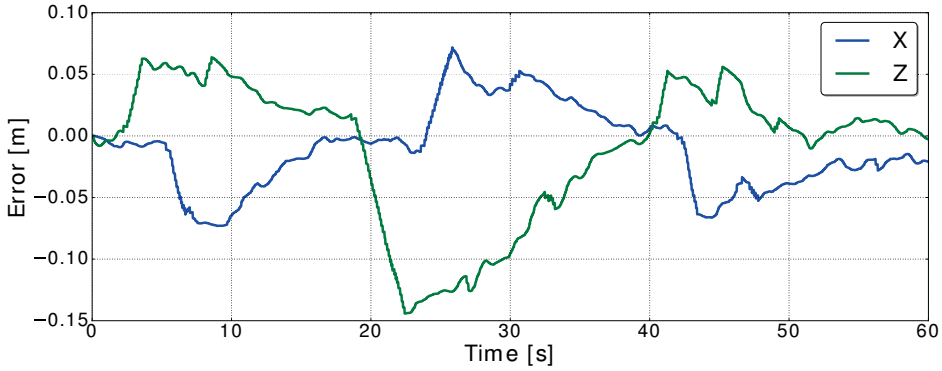


FIGURE 4.19: Control errors along the X and Z axes for the bar soft object.

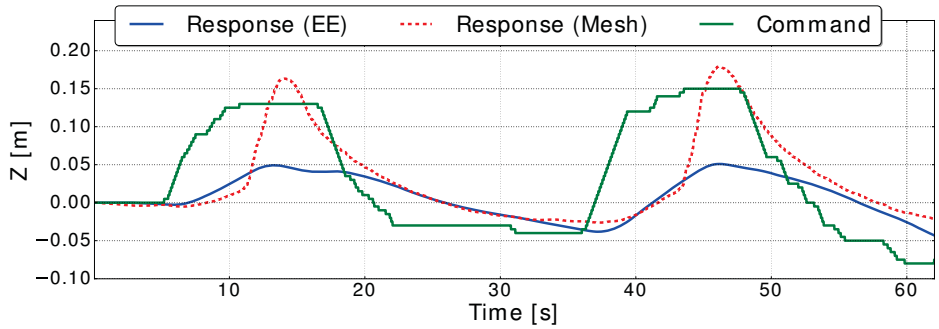


FIGURE 4.20: Command and responses of the mesh and the robot end effector (EE) along the Z axis for the block hard object.

affected by the inaccuracies of the latter. Ideally, that is, if the shape sensing pipeline worked without a flaw, the red dashed line representing the mesh estimation (in Figure 4.18 and Figure 4.20) should exactly match the blue line which represents the pose obtained by the forward kinematics of the robot, i.e. the response (EE). The red dashed

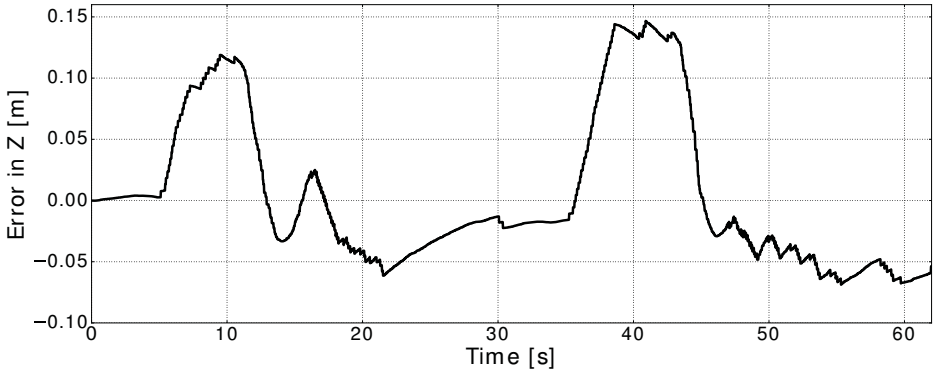


FIGURE 4.21: Control error along the Z for the block hard object.

line then ought to follow the command line (green line), with an expected delay between the two signals. It should be noted that the response (EE) is not the signal being controlled, but rather it serves as a reference to measure the accuracy of the mesh estimation, i.e. the response (Mesh), which is the signal being regulated.

Both figures show a slow response, but the test errors on the *bar soft* object remain relatively small, as it can be seen in Figure 4.19 at the 19, 40 and 60 second marks. The test errors on the *block hard* object present however a significant discrepancy between the response (EE) and the response (Mesh) as shown in Figure 4.20, which results in considerable overshoot at the 15 and 47 seconds mark. This gap between the response (Mesh) and the response (EE) is unsurprising if one refers to performance results shown in Figure 4.15d.

The delayed response of the controller is mainly due to:

1. **Command:** As the command signal is set by a GUI, the target poses for the controller result in a sharp slope, given that the user can vary the target position at a much higher rate than the operating rate of the shape controller (i.e. mesh estimation plus the velocity controller of the arm).

2. **Damping parameters:** The deformation model requires, on top of the Young's modulus and the Poisson's ratio, to define damping parameters to simulate the deformation behavior of the object. However, setting a high damping parameter results in a delayed estimation of the object's deformation. Note that this delay is not caused by a higher computational time of the deformation model, but rather how the behavior of the simulation compares to that of reality.

3. **Sensor noise:** As the raw output of the force-torque sensor produces a noisy signal, its output was filtered with a moving average filter to smoothen the sensor readings. This filter in turn introduced an additional source for delay in the overall behavior of the shape controller.

4.6 Summary

This section presented our proposed shape sensing pipeline. The need for a modular approach was motivated by the divergence of both tasks and objects involved in the manipulation of flexible materials. Each component of the pipeline was detailed and their cohesive integration was outlined. Furthermore, three possible applications using the shape sensing pipeline were implemented and evaluated, namely two shape sensing applications, using tactile and force data, as well as a shape control application. Following the evaluation of the applications, a discussion of the experimental results recognized promising insights, as well as critical limitations.

Chapter 5

Conclusions and future work

This chapter will first summarize the contributions of this thesis. Next, the limitations of our proposed pipeline will be reviewed by analyzing each of its components as well as their interactions, followed by perspectives on how to address such limitations. Finally, the conclusions from this work are drawn in the last section.

5.1 Contributions

In this thesis we proposed a shape sensing pipeline with the objective of approaching a general solution to the problem of manipulating deformable objects. As we showed in Section 2, current approaches are limited to specific tasks and objects and therefore cannot be applied in a general manner. The main purpose of the proposed pipeline is to increase the generality of current solutions by modularly composing behaviors depending on the task at hand. Furthermore, a review of the state of the art exhibited a lack of touch sensing in the field, where vision-based systems currently dominate. To address this issue, we developed two non-vision sensor models that depend on tactile and force information, respectively. Finally, we evaluated the applicability of the shape sensing pipeline on two sensing tasks and one manipulation task.

5.2 Limitations of the approach

Due to the sequential nature of the pipeline, individual errors in each component aggregate as their outputs pass through the rest of the pipeline resulting in a deterioration of the pipeline's performance. In this chapter we will analyze the source of errors for each component involved in the pipeline.

One major issue encountered in the force estimation carried by the sensor models presented in Chapter 3 is their inability to estimate zero forces. This behavior, also noted in [10], is inherent to recurrent neural networks since they learn the force mapping based on the sensor' output which are imperfect due to noise. Additionally, both tactile and force sensors suffer from calibration issues that result in them producing inconsistent readings. For instance, the tactile sensors used in this thesis are filled with a liquid which, due to wear and tear, leaks and thus affects the output of the sensor; whereas the force-torque sensor exhibits a considerable amount of sensitivity due to temperature, as it was shown in [75]. Furthermore, the tactile sensors do not directly measure force but rather pressure, which causes slight and short contacts to not be perceived. These intrinsic failures in the manufacturing of the sensors are difficult to model and hence lead to erroneous force estimations by the proposed models.

The second component of the pipeline, namely the force transformer, converts the output of the sensor models such that it can be used as an input to the deformation model. This requires two calculations, first, to transform the force from the sensors into a common frame of reference, and second to distribute the force on the mesh's nodes, which is a requirement of the deformation model. The former not only requires precise measurements relating frames on the sensors to a common frame (e.g. the object's frame) but also knowledge of the location of such frames which might not always be straightforward to acquire. The latter, nodal distribution of the force, is directly affected by the resolution of the mesh, that is, how many elements the mesh contains and therefore how many nodes. Thus, a higher mesh resolution can achieve a more realistic force distribution but it does so at the expense of a higher computational cost, e.g. a slower estimation by the deformation model.

The third and last component, i.e. the deformation model, assumes the deformable objects to be linearly elastic. However, the behavior of the deformable objects is better

described as viscoelastic, as it is can be seen by the presence of hysteresis in Figure B.1 and Figure B.2. Unlike elastic materials, viscoelastic materials dissipate energy which is unaccounted for in the deformation model used in this thesis. Besides this assumption, the elasticity parameters used by the deformation model, which are empirically obtained, greatly influence its accuracy.

As noted at the beginning of this chapter, the individual errors compound as they flow through the pipeline resulting in an amplification of the pipeline's inaccuracies. The modularity of the pipeline comes at the expense of this drawback as the output of one component is used as the input for the next component. In addition to the error propagation across components, delays on one component directly affect the performance of a component further in the pipeline. An example of the impact a delayed output can cause was discussed in Section 4.5.3, where the delay deteriorated the performance of the shape controller.

To counteract the effects of these shortcomings, possible lines of research are outlined next to extend the works presented in this thesis.

5.3 Future research lines

5.3.1 Sensor model

One way to address the issue of the RNNs' inability to estimate zero values is to collect data with an additional label for contact. In doing so, the algorithm can be trained to detect contacts as well, and thus safely estimate a zero force value. Contact detection for the tactile sensor could also be performed by using proximity queries[1] between the sensor and the object. This can be achieved, for instance, by integrating the Flexible Collision Library [77] within our tactile sensor model. Furthermore, by analyzing the sensor's resolution limits, as recently proposed in [78], the tactile sensor model could be improved to address the uncertainties arising from this limitation.

The performance of the proposed force sensor model (RNNOB) could be improved by combining it with the analytical observer described in Section 3.4.1. Moreover, since the analytical observer is a model-based approach, safety concerns can be ensured.

[1] Proximity queries are methods used for computing the distance between two bodies and checking for collisions [76].

5.3.2 Deformation model

As noted in Appendix B, the elastic behavior of the objects used in this thesis is nonlinear and thus, as the deformation model assumes linear elasticity, its estimation leads to inaccuracies. This could be addressed by replacing the deformation model with a nonlinear model at the expense of increasing computational complexity or by estimating, and then adapting, the elastic parameters online. As Figure B.2 shows, provided an identification process has been previously performed, the Young's modulus could be computed online based on the current amount of force and displacement being exerted on the object as well as considering the geometry of the object. Furthermore, the performance of the deformation model could be boosted by constraining the nodes that are in contact with the manipulator, which could be addressed, for instance, by applying the contact model proposed in [23]. In this manner, the estimation of the deformation model could benefit from knowing the exact position of the nodes where the deformation forces are applied and estimate only the position of the remaining nodes. Building upon this new constraint, since the current approach suffers from the mesh passing through objects (e.g. the fingers on a robotic hand), estimation during in-hand manipulation could be greatly improved.

5.3.3 Pipeline

The performance of the pipeline, as only force and tactile sensing are used, is significantly limited at the moment. This limitation could be addressed by incorporating visual feedback. Current approaches that estimate deformations based on visual feedback, such as [40, 41], could be refactored into components that are compatible with the interfaces defined in our approach. It is however not clear how such an integration should be carried, since it is necessary to merge disparate measurements in a coherent manner, e.g. by respecting the topology of the mesh. A potential solution might be offered by machine learning approaches. The work described in [79], for instance, shows promising results by learning how objects deform using adversarial learning.

In the applications described in Chapter 4.5, the implementation of the pipeline remained static, that is, the components were not replaced while the task was being executed. For more complex tasks it might be necessary to divide them in sub-tasks such as

grasping an object, lifting it and transporting it to a different location to shape its form into a desired configuration. Since these sub-tasks require different sensing and manipulation skills, different components could be switched accordingly in order to accommodate for the specific sub-task at hand. This switching could be performed by a task planner that first subdivides the greater and more complex task into smaller tasks with defined requirements and then schedule the appropriate replacement of components. Furthermore, in the future, instead of only reusing components, complete architectures can be reused as proposed in [80]

5.3.4 Shape control

Besides integrating current shape controllers found in the literature, as the ones proposed in [46, 49], as components in the proposed pipeline, novel controllers can be developed by considering the mesh information. For instance, the displacement of the nodes (or a set of them) can be mapped to the movements of the manipulator holding the object via a so-called *deformation Jacobian*. This deformation Jacobian can in turn be used to control the shape of an object, similar to how the Jacobian of a manipulator is used to control the movement of a robot's end-effector in Cartesian space. An example diagram, showing how a shape controller could be used with our proposed pipeline to drive an object's configuration (i.e. its shape) to a desired one, is shown in Figure 5.1. Here, the plant is considered to be composed of a robot, with its sensors, interacting with a deformable object; and the sensors readings could be in the form of force signals, images, point clouds, etc. Different sensor models could be then developed for the specific sensors of a particular robot.

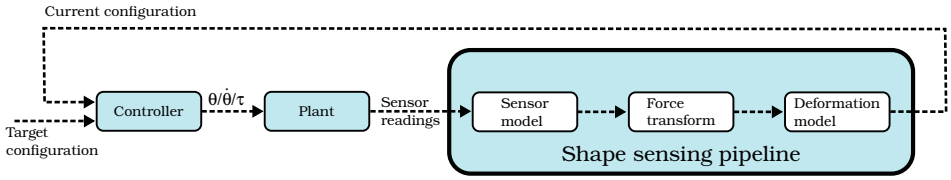


FIGURE 5.1: Integration of a controller with the shape sensing pipeline.

Other characteristics of the deformation could also be controlled, for example, the energy produced by the deformation could be monitored such that the manipulator does not bend the object in an excessive manner. By adding this energy constraint, a controller as

proposed in [81] could be used to guarantee a shape controller is safe to use for delicate objects.

5.4 Final conclusions

Although a lot of progress has been made in recent years, the manipulation of deformable objects still remains an open challenge for robotic manipulators. Two main research lines have been pursued in order to solve this problem, namely, sensing and controlling the shape of a deformable object. In this thesis, we addressed the former research line with the purpose of enabling progress on the latter. We proposed a pipeline to estimate the shape of an object while it is manipulated by leveraging the contact forces generated during the manipulation. These contact forces were estimated using novel sensor models based on recurrent neural networks that take tactile and force data as input. The contact forces where then applied to a deformation model to estimate the object's shape in an online manner.

The design of the pipeline was motivated by a software paradigm named component-based software engineering, which has been increasingly adopted by the robotics community as it reduces developing time by promoting re-use of code. It also allows modularity, since the components can be interchanged in order to achieve a more general solution, e.g. by replacing the deformation model for a particular class of object.

Our proposed pipeline was applied to two shape sensing scenarios, using tactile and force sensing respectively; and to a shape control task. Despite the limited performance of the pipeline on real-life scenarios, the results are promising since no visual feedback was used to correct for inaccuracies, which, given the design of the pipeline, could be integrated to improve the overall performance of the proposed approach. In addition to including visual information, other potential improvements were outlined for future research lines.

Appendix A

Basics of deformation

This appendix introduces basic concepts and terminology on deformation. A deformation occurs when an external force[1] is applied to an object which causes the object to change its shape. Moreover, a deformation can be classified as plastic, elastic or elasto-plastic; depending on the object's response when the external force is removed.

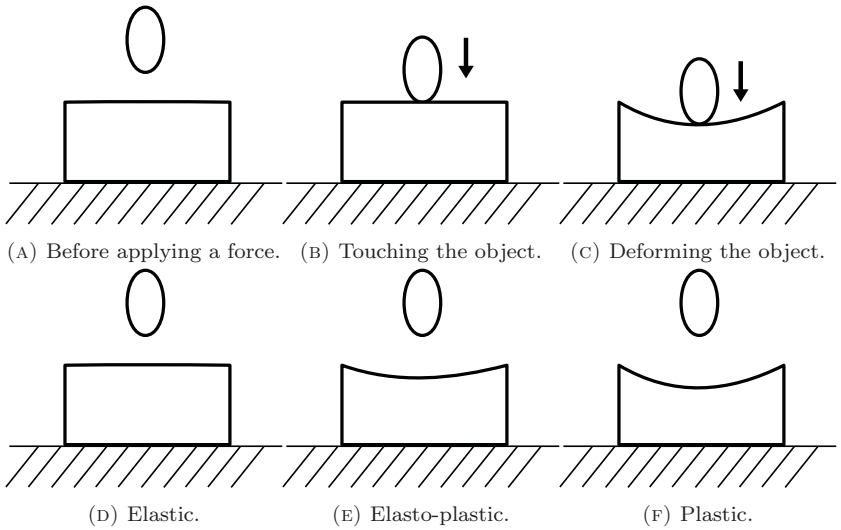
A.1 Deformation types

A *plastic* deformation refers to a permanent deformation, that is, an object maintains the shape caused by a deforming force even when the force is removed. On the contrary, an *elastic* deformation results on the object returning to its undeformed shape once the deforming force is removed [3]. And an *elasto-plastic* deformation is a combination of both, elastic and plastic, deformations; where the object does not return to its original shape, but it does not hold the deformation entirely. A visual example of these deformation types is shown in Figure A.1.

A.2 Elasticity

Elasticity measures a body's ability to recover its shape once deformation forces are removed. Formally, elasticity describes the relation between stress and strain. Strain

[1] In this thesis we are not concerned with deformations produced by other physical phenomena such as temperature.



(A) Before applying a force. (B) Touching the object. (C) Deforming the object.

(D) Elastic. (E) Elasto-plastic. (F) Plastic.

FIGURE A.1: **Top row:** an object being deformed by an external force. **Bottom row:** the resulting types of deformation once the external force is removed.

(ϵ) is the amount of deformation induced by a force on a body; and *stress* (σ) is the ratio between the applied force F and the cross-section area A_0 [3]. Figure A.2 shows an object being deformed by a tensile load, which causes lateral and axial strains. For linear elastic deformations, which occur when the stress and strain are proportional [3], stress and strain are related by Hooke's law [3]:

$$\sigma = E\epsilon$$

where E is the modulus of elasticity, also called Young's modulus, and is measured in pressure units such as Pascal (N/m^2) [82].

Another important elasticity parameter is the Poisson's ratio (ν), an adimensional number that relates the ratio between axial and lateral strains [3]. In Figure A.2, the axial strain is represented by the change of length ($\frac{L-L_0}{L_0}$), where lateral strains occur perpendicularly to the applied force F.

The Young's modulus E and the Poisson's ratio ν are common parameters in modeling the deformation of an isotropic object, where the deformation's elastodynamics are represented by a set of partial differential equations solved through discretization techniques in order to approximate the displacement field. However, these parameters are only valid for linear elasticity. Linear deformation can refer either to a geometric or a material linearity. Geometrical linearity is not appropriate for large deformations, since

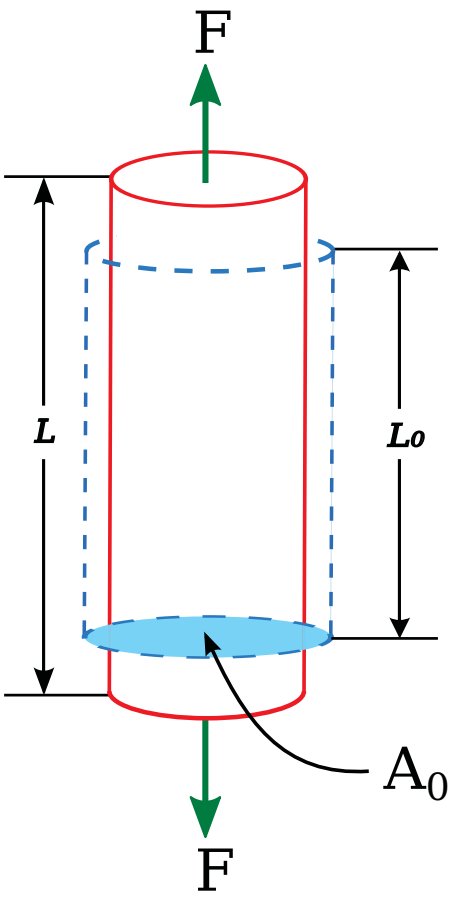


FIGURE A.2: A tensile load (F) producing axial and lateral strains. The blue dashed lines represent the original, undeformed, shape and the red solid lines represent the deformed shape [3].

only small deformations can be modeled accurately [5]. On the other hand, material linearity refers to a deformation that retains a linear stress-strain relation [3]. In this thesis we assume the deformable objects have material linearity.

A.3 Deformation models

Modeling a deformation can be done with a variety of techniques. These techniques require a deformation model and a representation of the object's shape, usually by a set of particles or a mesh. A mesh represents an object as set of points (vertices), edges and faces or elements for a two dimensional or a three dimensional object, respectively. The faces are usually triangles or quadrilaterals, and the elements are commonly represented as tetrahedra or hexahedra. The deformation models provide a function to compute the position of every vertex based on their current position and an input force [5].

Deformation models that do not require a mesh are termed mesh-free (or meshless), and particle based models [83] are an example of a meshless model. The mesh-based models are categorized either as continuum or lumped (discrete) variable models, according to the consistency of the mass and stiffness parameters with the approximated displacement fields in the elements of the mesh. The discrete based models are mainly represented as Mass-Spring-Damper (MSD) systems, where the vertices are treated as mass particles and the edges are considered as springs. Continuum based methods are commonly modeled with finite element methods (FEM), where the object is split into a set of discrete geometric parts called finite elements in order to approximate the object's shape [84]. A comparison between different physically-based models is shown in Figure A.3.

MSD models are more intuitive and simpler to implement than FEM-based models, however FEM-based models are able to produce more physically realistic simulation [5, 84]. Furthermore, MSD models are unable to preserve volume and tend to easily invert [4, 84].

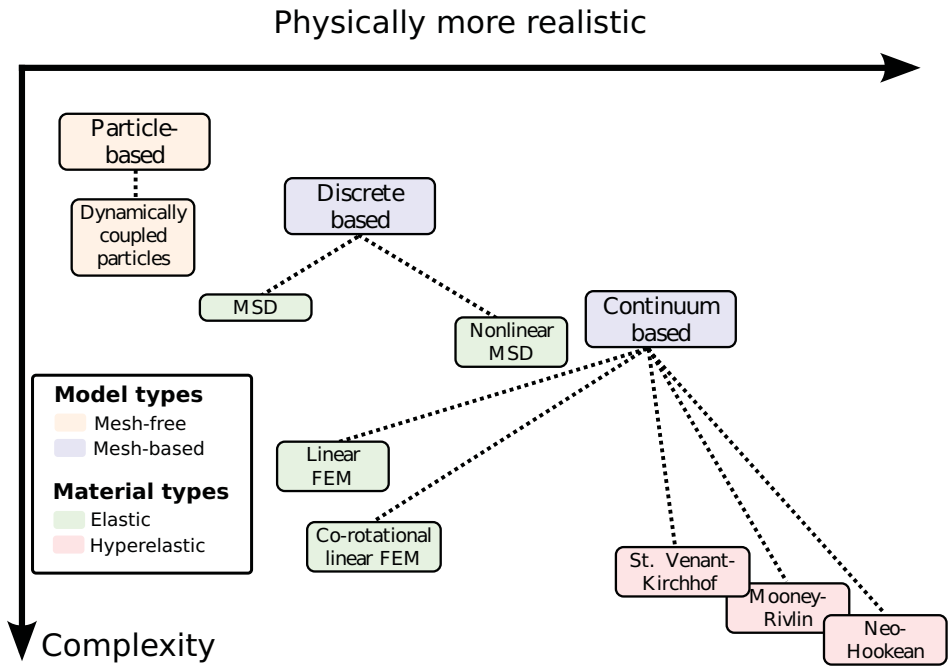


FIGURE A.3: Comparison of physically-based deformation models based on the evaluation results from [4] and the classification presented in [5].

Since this appendix covered only a brief summary on a variety of topics regarding deformation and its simulation, the interested reader is referred to [5, 84] for more comprehensive surveys of deformable models and modeling techniques in computer graphics; and a technical review of mechanical properties and elastic behavior can be found in [3, 82].

Appendix B

Elastic behavior of the test objects

The elastic behavior of isotropic materials can be represented by two parameters, such as the Young's modulus and the Poisson's ratio [85]. One common way to obtain these parameters is to perform a compression test which consists on pushing down on an object while simultaneously recording the displacement[1] of the object and the applied force.

We attempted to estimate the material properties of the cube-like objects described in Section 4.5.1 by performing a compression test. For the test, the object was placed inside of a press that can be programmed to move on a vertical axis until a desired height and is equipped with a force sensor. The press was set to first move down to approximately compress the object 36 mm, then to move upwards until the compression was around 4 mm and finally the press was moved down until 44 mm of compression were reached. The results of this test can be seen in Figure B.1, where the compression is plotted against the measured force. Similarly, the stress-strain curves for the three materials are shown in Figure B.2. The plots show the elastic behavior for three different materials, namely, *hard* (HR 45), *medium* (Bultex 30) and *soft* (Bultex 26).

It can be seen from these figures, that the behavior of all three materials is not only nonlinear but also presents hysteresis. Hence, modeling these materials as linear will result in inaccuracies. Nevertheless, we can use the test findings to obtain a range

[1] Here, displacement refers to the compression of the object, i.e. the object is fixed at one side while the opposite side is moved towards the fixed side.

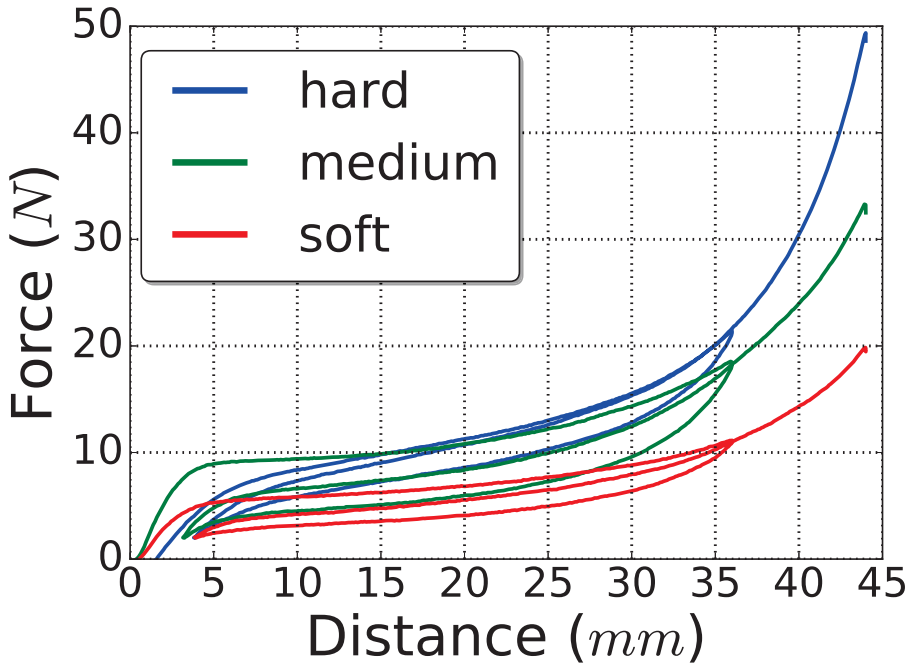


FIGURE B.1: The X axis shows the compression distance while the Y displays the force applied to the objects.

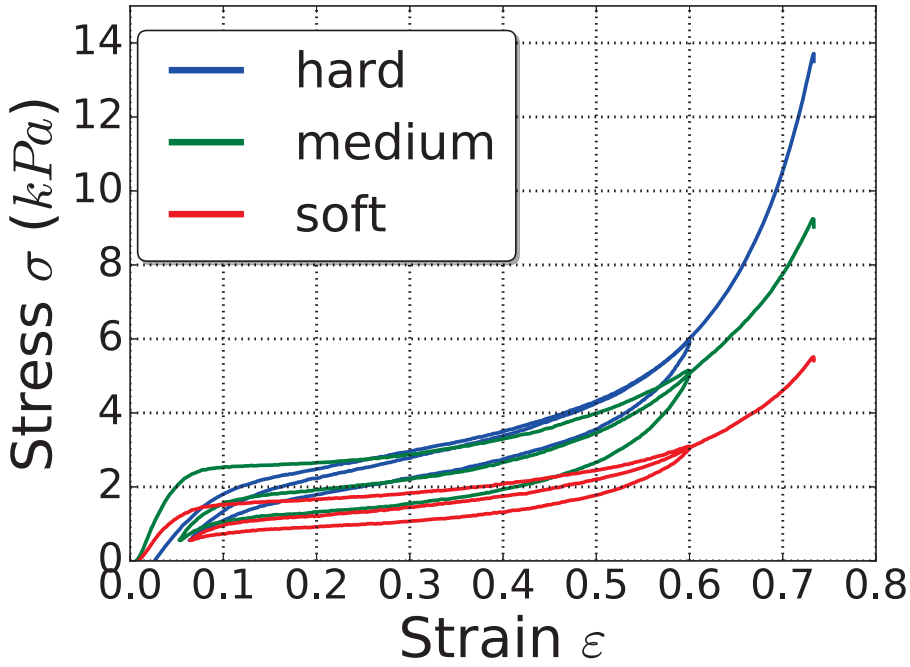


FIGURE B.2: Stress-strain curve.

estimation of the Young's modulus for each material and then tune them individually until the simulation behavior better matches reality.

Bibliography

[1] Jeremy A. Fishel. *Design and use of a biomimetic tactile microvibration sensor with human-like sensitivity and its application in texture discrimination using Bayesian exploration.* Ph.D. thesis, University of Southern California, 2012.

[2] Davide Brugali and Patrizia Scandurra. Component-Based Robotic Engineering (Part I). *IEEE Robotics and Automation Magazine*, 16(4):84–96, 2009.

[3] William D. Jr. Callister. Mechanical Properties of Metals. In *Materials Science and Engineering: An Introduction*, chapter 6, page 832. Wiley Publishers, 7th edition, 2006. ISBN 978-0006970118.

[4] F. S. Sin, D. Schroeder, and J. Barbič. Vega: Non-linear FEM deformable object simulator. *Computer Graphics Forum*, 32(1):36–48, 2013.

[5] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[6] Matthias Rambow, Thomas Schauss, Martin Buss, and Sandra Hirche. Autonomous manipulation of deformable objects based on teleoperated demonstrations. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2809–2814, 2012.

[7] Olivier Roussel and Michel Ta. Deformable Linear Object manipulation planning with contacts. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–5, 2014.

[8] Ankit J. Shah and Julie A. Shah. Towards manipulation planning for multiple interlinked deformable linear objects. In *IEEE International Conference on Robotics and Automation*, pages 3908–3915, may 2016. ISBN 978-1-4673-8026-3.

[9] Wenhao Yu, Ariel Kapusta, Jie Tan, Charles C Kemp, Greg Turk, and C Karen Liu. Haptic Simulation for Robot-Assisted Dressing. *IEEE International Conference on Robotics and Automation*, 2017.

[10] Zackory Erickson, Alexander Clegg, Wenhao Yu, Greg Turk, C. Karen Liu, and Charles C. Kemp. What does the person feel? Learning to infer applied forces during robot-assisted dressing. In *IEEE International Conference on Robotics and Automation*, pages 6058–6065. Singapore, 2017. ISBN 9781509046331.

[11] C. Wouter Bac, Jochen Hemming, B.A.J. van Tuijl, Ruud Barth, Ehud Wais, and Eldert J. van Henten. Performance Evaluation of a Harvesting Robot for Sweet Pepper. *Journal of Field Robotics*, 34(6):1123–1139, sep 2017.

[12] Christopher Lehnert, Andrew English, Christopher McCool, Adam W. Tow, and Tristan Perez. Autonomous Sweet Pepper Harvesting for Protected Cropping Systems. *IEEE Robotics and Automation Letters*, 2(2):872–879, 2017.

[13] Antonio Bicchi and V. Kumar. Robotic grasping and contact: a review. In *2000 IEEE International Conference on Robotics and Automation*, volume 1, pages 348–353. IEEE, 2000. ISBN 0-7803-5886-4.

[14] Efrain Lopez-Damian, Daniel Sidobre, and Rachid Alami. A grasp planner based on inertial properties. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 754–759, 2005.

[15] Xavier Broquère, Daniel Sidobre, and Ignacio Herrera-Aguilar. Soft motion trajectory planner for service manipulator robot. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 2808–2813, 2008.

[16] Juan Antonio Corrales Ramon, Veronique Perdereau, and Fernando Torres Medina. Multi-fingered robotic hand planner for object reconfiguration through a rolling contact evolution model. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 625–630, 2013.

[17] V.-D. Nguyen. Constructing Force- Closure Grasps. *The International Journal of Robotics Research*, 7(3):3–16, jun 1988.

[18] Feng Guo, Huan Lin, and Yan Bin Jia. Squeeze grasping of deformable planar objects with segment contacts and stick/slip transitions. *2013 IEEE International Conference on Robotics and Automation*, pages 3736–3741, 2013.

[19] Antonio Bicchi. On the Closure Properties of Robotic Grasping. *The International Journal of Robotics Research*, 14(4):319–334, aug 1995.

[20] Yan Bin Jia, Feng Guo, and Jiang Tian. On two-finger grasping of deformable planar objects. *2011 IEEE International Conference on Robotics and Automation*, pages 5261–5266, may 2011.

[21] Fouad F. Khalil, Phillip Curtis, and Pierre Payeur. Visual monitoring of surface deformations on objects manipulated with a robotic hand. *2010 IEEE International Workshop on Robotic and Sensors Environments*, pages 1–6, oct 2010.

[22] Essahbi Nabil, Bouzgarrou Belhassen-chedli, and Gogu Grigore. Robotics and Computer-Integrated Manufacturing Soft material modeling for robotic task formulation and control in the muscle separation process. *Robotics and Computer Integrated Manufacturing*, 32:37–53, 2015.

[23] Lazher Zaidi, Juan Antonio Corrales, Belhassen Chedli Bouzgarrou, Youcef Mezouar, and Laurent Sabourin. Model-based strategy for grasping 3D deformable objects using a multi-fingered robotic hand. *Robotics and Autonomous Systems*, 95:196–206, 2017.

[24] Jose Sanchez, Juan-Antonio Corrales, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey. *The International Journal of Robotics Research*, 37(7):688–716, jun 2018.

[25] Félix Nadon, Angel Valencia, and Pierre Payeur. Multi-Modal Sensing and Robotic Manipulation of Non-Rigid Objects: A Survey. *Robotics*, 7(4):74, nov 2018.

[26] Barbara Frank, Rudiger Schmedding, Cyrill Stachniss, Matthias Teschner, and Wolfram Burgard. Learning the elasticity parameters of deformable objects with a manipulation robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1877–1883, oct 2010. ISBN 978-1-4244-6674-0.

[27] Andreas Rune Fugl, Andreas Jordt, Henrik Gordon Petersen, Morten Willatzen, and Reinhard Koch. Simultaneous Estimation of Material Properties and Pose for Deformable Objects from Depth and Color Images. In Axel Pinz, Thomas Pock, Horst Bischof, and Franz Leberl (editors), *Pattern Recognition*, volume 7476 LNCS, pages 165–174. Springer Berlin Heidelberg, 2012. ISBN 9783642327162.

[28] Ana Maria Cretu, Pierre Payeur, and Emil M. Petriu. Soft object deformation monitoring and learning for model-based robotic hand manipulation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42(3):740–753, 2012.

[29] John Schulman, Alex Lee, Jonathan Ho, and Pieter Abbeel. Tracking deformable objects with point clouds. In *IEEE International Conference on Robotics and Automation*, pages 1130–1137, may 2013. ISBN 978-1-4673-5643-5.

[30] Ibai Leizea, Hugo Alvarez, Iker Aguinaga, and Diego Borro. Real-time deformation, registration and tracking of solids based on physical simulation. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 165–170, 2014. ISBN 9781479961849.

[31] Leon Bodenhagen, Andreas R Fugl, Andreas Jordt, Morten Willatzen, Knud A Andersen, Martin M Olsen, Reinhard Koch, Henrik G Petersen, and Norbert Krüger. An adaptable robot vision system performing manipulation actions with flexible objects. *IEEE Transactions on Automation Science and Engineering*, 11(3):749–765, jul 2014.

[32] M Staffa, S Rossi, M Giordano, M De Gregorio, and B Siciliano. Segmentation performance in tracking deformable objects via WNNs. *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2462–2467, 2015.

[33] Antoine Petit, Vincenzo Lippiello, and Bruno Siciliano. Real-time tracking of 3D elastic objects with an RGB-D sensor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 320992, pages 3914–3921, sep 2015. ISBN 978-1-4799-9994-1.

[34] Püren Güler, Karl Pauwels, Alessandro Pieropan, Hedvig Kjellstrom, and Danica Kragic. Estimating the deformability of elastic materials using optical flow and

position-based dynamics. In *IEEE-RAS International Conference on Humanoid Robots*, pages 965–971, nov 2015. ISBN 978-1-4799-6885-5.

[35] Sergio Caccamo, Püren Güler, Hedvig Kjellström, and Danica Kragic. Active perception and modeling of deformable surfaces using Gaussian processes and position-based dynamics. *IEEE-RAS International Conference on Humanoid Robots*, pages 530–537, 2016.

[36] Antoine Petit, Fanny Ficuciello, Giuseppe Andrea Fontanelli, Luigi Villani, and Bruno Siciliano. Using Physical Modeling and RGB-D Registration for Contact Force Sensing on Deformable Objects. *Int. Conference on Informatics in Control, Automation and Robotics*, 2017.

[37] Bilal Tawbe and Ana Maria Cretu. Acquisition and neural network prediction of 3D deformable object shape using a kinect and a force-torque sensor. *Sensors (Switzerland)*, 17(5), 2017.

[38] Veronica E. Arriola-Rios and Jeremy L. Wyatt. A Multimodal Model of Object Deformation under Robotic Pushing. *IEEE Transactions on Cognitive and Developmental Systems*, 9(2):153–169, 2017.

[39] Püren Güler, Alessandro Pieropan, Masatoshi Ishikawa, and Danica Kragic. Estimating deformability of objects using meshless shape matching. In *IEEE International Conference on Intelligent Robots and Systems*, pages 5941–5948, 2017. ISBN 9781538626825.

[40] Antoine Petit, Stephane Cotin, Vincenzo Lippiello, and Bruno Siciliano. Capturing Deformations of Interacting Non-rigid Objects Using RGB-D Data. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 491–497. IEEE, oct 2018. ISBN 978-1-5386-8094-0.

[41] Zhe Hu, Peigen Sun, and Jia Pan. Three-Dimensional Deformable Object Manipulation Using Fast Online Gaussian Process Regression. *IEEE Robotics and Automation Letters*, 3(2):979–986, apr 2018.

[42] Tao Han, Xuan Zhao, Peigen Sun, and Jia Pan. Robust Shape Estimation for 3D Deformable Object Manipulation. pages 1–9, 2018.

[43] Dmitry Berenson. Manipulation of deformable objects without modeling and simulating deformation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4525–4532, nov 2013. ISBN 978-1-4673-6358-7.

[44] David Navarro-Alarcon, Yun-Hui Liu, Jose Guadalupe Romero, and Peng Li. Model-Free Visually Servoed Deformation Control of Elastic Objects by Robot Manipulators. *2013 IEEE Transactions on Robotics*, 29(6):1457–1468, dec 2013.

[45] David Navarro-Alarcon, Yun Hui Liu, Jose Guadalupe Romero, and Peng Li. On the visual deformation servoing of compliant objects: Uncalibrated control methods and experiments. *International Journal of Robotics Research*, 33(11):1462–1480, sep 2014.

[46] David Navarro-Alarcon, Hiu Man Yip, Zerui Wang, Yun-Hui Liu, Fangxun Zhong, Tianxue Zhang, and Peng Li. Automatic 3-D Manipulation of Soft Objects by Robotic Arms With an Adaptive Deformation Model. *IEEE Transactions on Robotics*, 32(2):429–441, apr 2016.

[47] Biao Jia, Zhe Hu, Jia Pan, and Dinesh Manocha. Manipulating Highly Deformable Materials Using a Visual Feedback Dictionary. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Section IV, pages 239–246. IEEE, may 2018. ISBN 978-1-5386-3081-5.

[48] David Navarro-Alarcon and Yun Hui Liu. Fourier-Based Shape Servoing: A New Feedback Method to Actively Deform Soft Objects into Desired 2-D Image Contours. *IEEE Transactions on Robotics*, pages 1–8, 2017.

[49] Fanny Ficuciello, Alessandro Migliozzi, Eulalie Coevoet, Antoine Petit, and Christian Duriez. FEM-Based Deformation Control for Dexterous Manipulation of 3D Soft Objects. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4007–4013. IEEE, oct 2018. ISBN 978-1-5386-8094-0.

[50] Simon Duenser, James M Bern, Roi Poranne, and Stelian Coros. Interactive Robotic Manipulation of Elastic Objects. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3476–3481. IEEE, oct 2018. ISBN 978-1-5386-8094-0.

[51] Mengyao Ruan, Dale Mc Conachie, and Dmitry Berenson. Accounting for Directional Rigidity and Constraints in Control for Manipulation of Deformable Objects without Physical Simulation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 512–519. IEEE, oct 2018. ISBN 978-1-5386-8094-0.

[52] Zerui Wang, Xiang Li, David Navarro-Alarcon, and Yun-hui Liu. A Unified Controller for Region-reaching and Deforming of Soft Objects. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 472–478. IEEE, oct 2018. ISBN 978-1-5386-8094-0.

[53] Silvio Cocuzza and X. T. Yan. First engineering framework for the out-of-plane robotic shaping of thin rheological objects. *Robotics and Computer-Integrated Manufacturing*, 53(May 2017):108–121, 2018.

[54] Farshid Alambeigi, Zerui Wang, Rachel Hegeman, Yun-Hui Liu, and Mehran Armand. A Robust Data-Driven Approach for Online Learning and Manipulation of Unmodeled 3-D Heterogeneous Compliant Objects. *IEEE Robotics and Automation Letters*, 3(4):4140–4147, 2018.

[55] Dominik Henrich and Heinz Wörn. *Robot Manipulation of Deformable Objects*. Advanced Manufacturing. Springer London, London, 1 edition, 2000. ISBN 978-1-4471-1193-1.

[56] Mozafar Saadat and Ping Nan. Industrial applications of automatic manipulation of flexible materials. *Industrial Robot: An International Journal*, 29(5):434–442, oct 2002.

[57] P. Jimenez. Survey on model-based manipulation planning of deformable objects. *Robotics and Computer-Integrated Manufacturing*, 28(2):154–163, 2012.

[58] P. Jiménez. Visual grasp point localization, classification and state recognition in robotic manipulation of cloth: An overview. *Robotics and Autonomous Systems*, 92:107–125, 2017.

[59] Zhanat Kappassov, Juan-Antonio Corrales, and Véronique Perdereau. Tactile sensing in dexterous robot hands Review. *Robotics and Autonomous Systems*, 74:195–220, 2015.

[60] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[61] Alex Graves, Abdel-Rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013. ISBN 978-1-4799-0356-6.

[62] Nal Kalchbrenner and Phil Blunsom. Recurrent Continuous Translation Models. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMMLP)*, (October):1700–1709, 2013.

[63] Martin Karlsson, Anders Robertsson, and Rolf Johansson. Detection and Control of Contact Force Transients in Robotic Manipulation without a Force Sensor. In *IEEE International Conference on Robotics and Automation*, pages 21–25. Brisbane, Australia, 2018.

[64] Zhe Su, Karol Hausman, Yevgen Chebotar, Artem Molchanov, Gerald E Loeb, Gaurav S Sukhatme, and Stefan Schaal. Force estimation and slip detection/classification for grip control using a biomimetic tactile sensor. In *IEEE-RAS International Conference on Humanoid Robots*, October, pages 297–303, nov 2015. ISBN 978-1-4799-6885-5.

[65] Damien Aymeric and Others. TFLearn, 2016.

[66] Daniel Kubus, Torsten Kröger, and Friedrich M. Wahl. On-line estimation of inertial parameters using a recursive total least-squares approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3845–3852. Nice, France, 2008. ISBN 9781424420582.

[67] Herman Bruyninckx, Markus Klotzbücher, Nico Hochgeschwender, Gerhard Kraetzschmar, Luca Gherardi, and Davide Brugali. The BRICS component model. *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13*, pages 1758–1764, 2013.

[68] Nico Hochgeschwender, Luca Gherardi, Azamat Shakhirmardanov, Gerhard K. Kraetzschmar, Davide Brugali, and Herman Bruyninckx. A model-based approach to software deployment in robotics. *IEEE International Conference on Intelligent Robots and Systems*, pages 3907–3914, 2013.

[69] Luca Gherardi and Nico Hochgeschwender. RRA: Models and tools for robotics run-time adaptation. *IEEE International Conference on Intelligent Robots and Systems*, 2015-Decem:1777–1784, 2015.

[70] Davide Brugali and Azamat Shakhimardanov. Component-Based Robotic Engineering (Part II). *IEEE Robotics and Automation Magazine*, 17(1):100–112, 2010.

[71] Eftychios Sifakis and Jernej Barbic. FEM simulation of 3D deformable solids. In *ACM SIGGRAPH 2012 Posters on - SIGGRAPH '12*, pages 1–50. ACM Press, New York, New York, USA, 2012. ISBN 9781450316781.

[72] Peter Kaufmann, Sebastian Martin, Mario Botsch, and Markus Gross. Flexible simulation of deformable models using discontinuous Galerkin FEM. *Graphical Models*, 71(4):153–167, 2009.

[73] Qingming Zhan, Liang Yubin, and Yinghui Xiao. Color-Based Segmentation of Point Clouds. *Laser scanning 2009, IAPRS*, XXXVIII, P:248–252, 2009.

[74] Rainer Bischoff, Johannes Kurth, Günter Schreiber, Ralf Koeppe, Alin Albu-Schäffer, Alexander Beyer, and Oliver Eiberger. The KUKA-DLR Lightweight Robot Arm - A New Reference Platform for Robotics Research and Manufacturing. *Robotics (ISR)*, pages 1–8, 2010.

[75] Torsten Kröger, Daniel Kubus, and Friedrich M. Wahl. 12D force and acceleration sensing: A helpful experience report on sensor characteristics. In *IEEE International Conference on Robotics and Automation*, pages 3455–3462. Pasadena, USA, 2008. ISBN 9781424416479.

[76] Gino van den Bergen. Proximity queries and penetration depth computation on 3d game objects. *Game developers conference 2001*, 2001.

[77] Jia Pan, Sachin Chitta, and Dinesh Manocha. FCL: A general purpose library for collision and proximity queries. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3859–3866, 2012.

[78] Qian Wan and Robert D Howe. Modeling the Effects of Contact Sensor Resolution on Grasp Success. *IEEE Robotics and Automation Letters*, 3(3):1933–1940, jul 2018.

[79] Zhihua Wang, Stefano Rosa, Bo Yang, Sen Wang, Niki Trigoni, and Andrew Markham. 3D-PhysNet: Learning the Intuitive Physics of Non-Rigid Object Deformations. *arXiv Computer Vision and Pattern Recognition*, (Nips), 2018.

[80] Luca Gherardi and Davide Brugali. Modeling and reusing robotic software architectures: The HyperFlex toolchain. *IEEE International Conference on Robotics and Automation*, pages 6414–6420, 2014.

[81] Lucas Joseph, Vincent Padois, and Guillaume Morel. Towards X-Ray Medical Imaging with Robots in the Open: Safety Without Compromising Performances. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6604–6610, 2018.

[82] Donald Askeland and Pradeep Fulay. Mechanical Properties and Behavior. In *The Science and Engineering of Materials*, chapter 6, page 888. Cengage Learning, 6th edition, 2005. ISBN 0534553966.

[83] David Love Tonnesen. *Dynamically Coupled Particle Systems for Geometric Modeling, Reconstruction, and Animation.* Ph.D. thesis, University of Toronto, 1998.

[84] Patricia Moore and Derek Molloy. A Survey of Computer-Based Deformable Models. *International Machine Vision and Image Processing Conference (IMVIP 2007)*, pages 55–66, sep 2007.

[85] Veikko Lindroos, Markku Tilli, Ari Lehto, and Teruaki Motooka. *Handbook of Silicon Based MEMS Materials and Technologies*. Applied Science Publishers, 2010. ISBN 978-0-8155-1594-4.