Università
degli Studi
della Campania
Luigi Vanvitelli

Scuola Politecnica e
delle Scienze di Base

Dipartimento di
Ingegneria

Corso di Laurea Magistrale in

Ingegneria Informatica

Tesi di Laurea in Robotica e Automazione

Online estimation of physical properties of unknown objects using tactile sensors

Candidato
Maria Domenica Pone
Matr. A18/234

Relatore
Prof. Ciro Natale

A. A. 2017/2018

Abstract

In many robotic manipulation tasks, knowledge of physical properties of the manipulated object can increase robot autonomy and enable robust task execution.

Manipulation of a completely unknown object can be very difficult since prediction of object motion is very hard without knowledge of dynamic parameters, such as mass, inertia and friction parameters. Availability of tactile sensors can allow the robot to estimate such physical properties through an exploration procedure, just like humans learn manipulating objects by exploring them using the sense of touch. The thesis describes algorithms for the estimation of physical properties of unknown objects using tactile sensors attached to a gripper of a robotic arm. In particular, mass, center of mass, friction coefficient and stiffness have been estimated using only forces and torques measurements given by tactile sensors. Finally, a Machine Learning classifier has been created in order to recognize an unknown object through the physical properties previously estimated.

Sunto

In molte applicazioni di manipolazione robotica, la conoscenza delle proprietà fisiche dell'oggetto manipolato può aumentare l'autonomia del robot e consentire l'esecuzione di operazioni robuste.

La manipolazione di un oggetto completamente sconosciuto può essere molto complicata, poiché la predizione del moto dell'oggetto risulta difficile senza la conoscenza di parametri dinamici come massa, parametri di inerzia e attrito. Avere a disposizione sensori tattili può permettere al robot di stimare tali proprietà fisiche attraverso una procedura di esplorazione, così come l'uomo impara a manipolare gli oggetti esplorandoli attraverso il tatto. La tesi descrive algoritmi per la stima di proprietà fisiche di oggetti non noti, usando sensori tattili montati sul gripper di un braccio robotico. In particolare, sono state stimate massa, centro di massa, rigidezza e attrito dell'oggetto. Usando un algoritmo di Machine Learning, è stato, infine, costruito un classificatore capace di identificare un oggetto attraverso le proprietà fisiche precedentemente stimate.

Contents

List of figures

Chapter 1 Introduction

In this chapter the problem of the robotic grasping is introduced. Furthermore, the approach proposed to deal with this problem is explained, focusing on the importance of tactile sensors in this field. Then, the thesis structure is described.

1.1 Problem description

Research in the area of robotic grasping has gradually shifted from structured manufacturing environments toward unstructured everyday environments. Grasping arbitrary objects with robotic hands remains a difficult task with many open research problems. While there have been several detailed analyses of the kinematic and dynamic constraints necessary to effect stable grasps, most require a high level of sensory input and feedback from the grasping device to perform dexterous manipulation. This sensory information required, typically include contact point estimation, stiffness and friction estimation, etc.

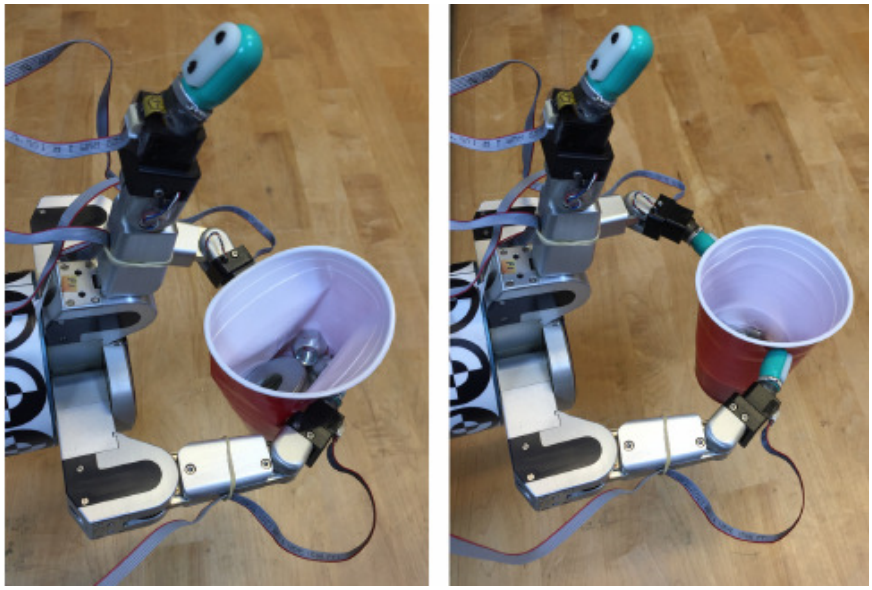

Figure 1: A robotic arm grasping a fragile object

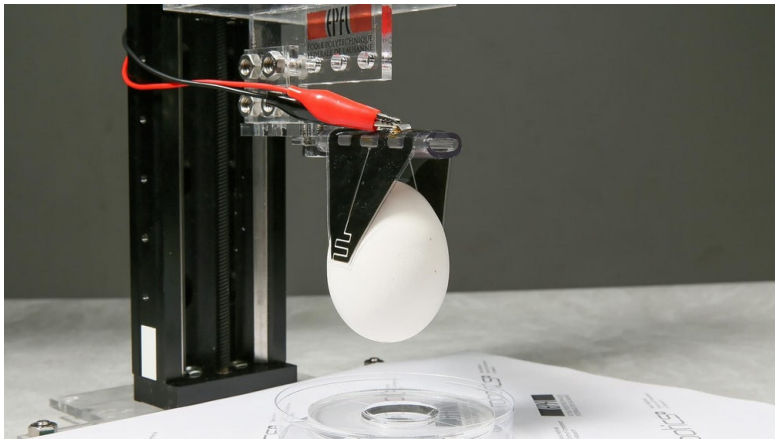

Figure 2: Grasping a fragile object

During interaction with the environment a significant portion of the information about contact comes through the detection of normal and shear forces. Hence, the importance of tactile sensors. These are a category of sensors that acquire tactile information through physical touch. The measured characteristics can be properties such as temperature, vibration, softness, texture, shape, composition as well as shear and normal forces. A tactile sensor may measure one or more of these properties. The importance of touch/tactile sensing can be explained by performing a simple experiment presented by Westling & Johannson in [1]. The outcome of this experiment was to explore the objects after making the hand numb through local anaesthesia. The skin on some volunteers' hand was anesthetized so that mechanoreceptors – specialized nerve endings that respond to mechanical stimulation (force) - activity was no longer available to the brain. It was observed that even though volunteers could see what they were doing, they could no longer maintain a stable grasp of objects. The movements become inaccurate and unstable when *'sense of touch'* was lost. Real-world objects exhibit rich physical interaction behaviours on touch. These behaviours depend on how heavy and hard the object is when hold, how its surface feels when touched, how it deforms on contact and how it moves when pushed etc. *'Sense of touch'* allows not only to assess the size, shape, and the texture of objects, but also helps in developing awareness of the body.

Visual feedback has proven to be also an important source of sensory information necessary for grasp generation and control, although vision provides important information, it is not always enough or trivial to obtain. In addition, the accuracy may be limited, due to imperfect calibration and occlusions. Errors in estimation of object physical properties are common even for known objects and these errors may cause failures in grasping. Clearly, vision does not provide important information on object properties such as deformability or material texture. However, vision sensors can be used to improve grasping performance during grasp execution rather than only tactile sensors.

1.2 Proposed Method

The outcome of this thesis is to build an online procedure able to estimate some physical properties of an unknown object and then to create a Machine Learning classifier that can recognize the object through its properties previously estimated. In this context, an *unknown object* is an object of which nothing is known except the position in the environment. The physical properties estimated are the mass, center of mass, stiffness and friction. The entire algorithm is based on the force and torque measurements given by tactile sensors. These are attached to the fingers of a gripper mounted on a robotic arm. To estimate the mass and the center of mass, the object has been positioned between the gripper fingers and it has been grasped and rotated several times. Instead, to estimate the stiffness, the object has been positioned in the same configuration, but it has been just 'squeezed' to obtain the object deformation, given a certain displacement. Finally, the friction has been estimated by sliding the gripper fingers on the object surface while acquiring the normal and the tangential forces. Once the algorithm has been tested in MATLAB, the entire procedure has been built in ROS (Robot Operating Systems). Furthermore, a Machine Learning algorithm, Random Forest, has been

used to create a classifier able to recognize the object, simply knowing its mass and stiffness. One of the possible applications of this procedure can be in various professional services, where manipulation of many types of objects is necessary, e.g., in the logistics domain.

1.3 Thesis structure

The thesis is structured as follows:

- **Chapter 2 - Hardware and software tools**

  It introduces the hardware and software tools adopted of this thesis.

- **Chapter 3 - General approaches for the estimation of physical properties**

  It contains an overview on materials and their properties. Then, several approaches for the estimation of physical properties of an object are reviewed.

- **Chapter 4 - Proposed algorithm for the estimation of physical properties**

  This section illustrates the algorithms used in this work for the estimation of physical properties of an object.

- **Chapter 5 - Classifier for object recognition**

  It explains how has been devised a Machine Learning classifier able to recognize an object through its mass and stiffness previously estimated.

- **Chapter 6 - Experimental results**

  This chapter shows the results of the procedures explained in the previous chapters.

- **Chapter 7 - Conclusions**

  The last chapter contains the conclusions and further improvements of this work.

Chapter 2 Software and Hardware tools

This chapter illustrates the software and hardware tools used during this work. The aim is to highlight their most important features, providing a quick overview of the tools adopted.

2.1 Software description

2.1.1 ROS

ROS is an open-source, meta-operating system for robots as: manipulators, mobile robots, autonomous cars, social robots, humanoids, and others. It provides the services of an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionalities, message-passing between processes, and package management. ROS implements a peer-to-peer network, namely graph, in which the processes, called **nodes**, can communicate at runtime via publish/subscribe pattern. ROS leverages the communication between processes on different styles, including synchronous communication over services, asynchronous streaming of data over topics, and storage of data on the parameter server. However, ROS is not a real time framework: although it is possible to integrate ROS with real time code, there aren't mechanisms to ensure natively real time communications. Moreover, the sensors and actuators used in robotics have also been adapted to be used with ROS. It is based on graph architecture with a centralized topology where processing takes place in nodes that may receive or post, such as multiplex sensor, control, state, planning, actuator, and so on. The library is geared towards a **Unix-like system** (Ubuntu Linux is listed as supported while other variants such as Fedora and Mac OS X).

2.1.1.1 Ros levels

One of the main advantages of using ROS instead of other competitor solutions, is the ability of grouping processes into **Packages** and **Stacks**, which can be easily shared and distributed. This is possible because, it is a combination of three levels: **filesystem level**, **computation graph level** and **community level**. Thanks to this division it is possible to make independent decisions about development and implementation, leaving to developers the ability of integrate different kind of code. Below, are showed these levels and how it is possible to manage ROS nodes.

The first level is Filesystem layer. In this layer a group of concepts is used to explain how ROS is internally formed, the folder structure, and the minimal files that it needs to work.

The second level is the Computation Graph level where communication between processes and systems happens.

The third level is the Community level where there are the tools and concepts to share knowledge, algorithms, and code from any developer. This level is important because ROS can grow quickly with great support from the community.

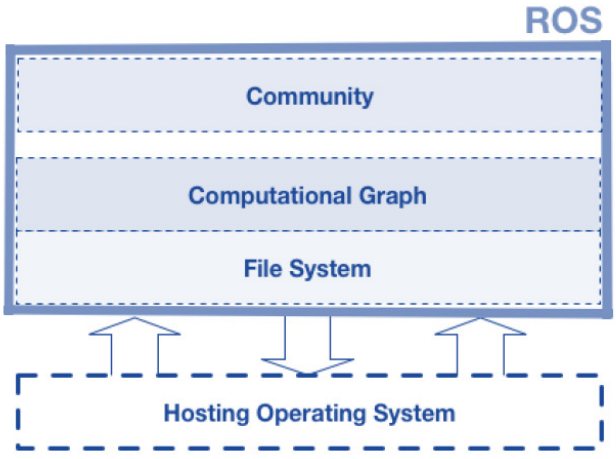

Figure 3: Ros levels

2.1.1.1.1 Filesystem level

ROS is built upon a Unix operation system. This means, that it needs a mechanism to communicate at application level with the OS. As the name suggests, filesystem level covers ROS physical resources that you encounter on 'disk', such as:

• **Packages**: Packages are the main unit for organizing software in ROS. A package may contain ROS runtime processes (**nodes**), a ROS-dependent library, datasets, configuration files, or anything else that is usefully organised together.

• **Metapackages**: Specialised Packages which only serve to represent a group of related other packages. Most commonly metapackages are used as a backwards compatible place holder for converted rosbuild Stacks.

• **Package Manifests**: Manifests (package.xml) provide metadata about a package, including its name, version, description, license information, dependencies, and other meta information like exported packages.

• **Message types**: Message descriptions, define the data structures for messages sent in ROS.

• **Service types**: service descriptions, define the request and response data structures for services in ROS.

2.1.1.1.2 Computation Graph Level

The computation graph level is the most important. As said, the computational graph is a peer-to-peer network of ROS processes that are processing data together. The basic Computation Graph components of ROS are nodes, master, parameter server, messages, services, and topics, all of which provide data to the 'graph' in different ways. More in detail:

• **Nodes**: Nodes are the basic processes that perform computation. ROS is designed to be modular at a fine-grained scale; usually a robot control system

comprises many nodes. For example, one node controls a camera, one node controls the temperature sensors and so on.

• **Master**: The ROS Master provides name registration and lookup to the rest of the Computation Graph. In practice, the master is a DNS server for nodes. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.

• **Parameter Server**: The Parameter Server allows data to be stored by keying a central location. Even though it has to be considered as an independent component, currently it is part of the Master.

• **Messages**: Nodes communicate with each other by means of messages. A message is simply a data structure, comprising typed fields.

• **Topics**: Messages are routed via a transport system with publish/subscribe mechanism. A node sends out a message by publishing it to a given topic. The topic is a name that is used to identify the content of the message. A node that is interested in a certain kind of data will subscribe to the appropriate topic. It's important to notice that there may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics. It's important to notice that publishers and subscribers are not aware of each other's existence.

• **Services**: The topics publish/subscribe model is a very flexible communication paradigm, but its many-to-many, for one-way transport its more appropriate a request/reply interaction. This kind of communications are done via services, which are defined by a pair of message structures: one for the request and one for the reply. A providing node offers a service under a name and a client uses the service by sending the request message and waiting the reply. So, we know that the master acts as name service; as such, it stores topics and services registration information as well as nodes addresses. Each node needs to communicate with the master, in order to receive information about other registered nodes and make

connections as needed. The Master will also make *callbacks* to these nodes when this registration information changes, which allows nodes to dynamically create connections as new nodes are launched. Once a node receives the address of another one from the master, it can establish a direct connection. As example, if a node wants to subscribe to a topic, it needs to send a request connection to the node that publishes that topic and will establish that connection over an agreed upon connection protocol. The most common protocol used in a ROS is called TCPROS, which uses standard TCP/IP sockets. This kind of architecture allows for decoupled operations; however, we need to introduce the primary means by which each node can connect to another, i.e. names. Names are used for all the ROS's components: nodes, topics, services, and parameters all have names.

In detail, ROS provides a hierarchical naming structure that is used for all resources in a ROS computation graph. These names are very powerful in ROS and central to how larger and more complicated systems are composed in ROS, so it is critical to understand how these names work and how we can manipulate them. Leveraging on names, we can provide encapsulation. Each resource is defined within a namespace, which it may share with many other resources. In general, resources can create other resources within their namespace, and they can access resources within or above their own namespace. This encapsulation isolates different portions of the system from accidentally picking the wrong named resource. This is the reason why it is possible to reuse code from other implementation; from a developer point of view, it is enough to write the nodes that work together as if they are all in the top-level namespace. When these nodes are integrated into a larger system, they can be pushed down into a namespace that defines their collection of code.

2.1.1.1.3 Community level

Finally, there is the community level. This is a 'pretended' level that is not physically present in ROS, but it represents ROS's developer community that shares code structure and knowledge for the definition of more complex community-based systems. In detail it's made of: ROS wiki, mailing lists, software repositories, etc

2.1.1.2 Client libraries

Actually, ROS is an application framework build upon a Unix operating system. As such, it is necessary a library to communicate with the ROS entities. There are several libraries for ROS that range a remarkable number of programming languages, although current focus is on providing robust C++ (**roscpp**) and Python (**rospy**) support. By means of these libraries, developers can write ROS nodes, publish and subscribe to topics, write and call services, and use the Parameter Server. By observing the ROS's repositories, we can notice that roscpp is the most widely used ROS client library and it's mostly deployed for high performance solutions. On the other hand, rospy favours implementation speed over runtime performance, so it is possible to quickly prototype and test algorithms within ROS.

After this general overview of ROS, we dive into a technical analysis of ROS's graph components. In detail, the following section presents a focus on: the master, parameter server, nodes and topic/service connections.

The reason why XMLRPC has been chosen instead of other HTTP/HTTPS protocols is mainly because it lightweight; In fact, it doesn't require a stateful connection and it is supported by a wide range of programming languages

2.1.1.3 Master

The master is probably the most important component of the current implementation of ROS; it exposes several functions to the other agents as registration APIs for register as publishers, as subscribers, and service providers. Furthermore, it includes the mechanisms that allow the discoverability of agents in the network. the master is a trust identity in the network that is queried by the nodes via its address (URI). In fact, each entity in the graph is qualified by a URI that corresponds to the host.

2.1.1.4 Parameter Server

The Parameter Server can store basic scalars, lists, and base64-encoded binary data. It can also store dictionaries (i.e. structs), but these have a special meaning; in fact, dictionary-of-dictionary is the structure used for the representation of the namespace; in detail, each dictionary represents a level in the naming hierarchy (aka namespace), and the other dictionary represents the internal structure of that namespace 'branch'.

2.1.1.5 Node

A general node has three types of APIs that expose it to the other agents:
1. **Slave API**: this kind of APIs are used by the node to receive callbacks from the Master, and for negotiating connections with other nodes.
2. **Transport protocol API**: namely TCPROS and UDPROS. These methods are used by nodes as communications primitives. As example, if a node wants to establish a topic connection with another, it uses these protocols to negotiate a connection. The difference between the two protocols is, trivially, the underling mechanisms: TCP - persistent, stateful TCP/IP socket connections - in TCPROS and UDP in UDPROS.

3. **Command-line API**: lastly, each node supports command-line remapping arguments, which enables names within a node to be configured at runtime. As the other entities of the graph, also every node has a URI, which corresponds to the host: port; but this is not used to transport topic or service data, but it only provides Slave APIs to negotiate connections with other nodes and also communicate with the MasterAPIs.

2.1.1.6 Communications

In this section, we want to focus on the way in which communications are handled in ROS. When we talk about networks, in general, TCP is widely used because it provides a simple reliable communication stream: TCP packets always arrive in order, and lost packets are resent until they arrive. However, while these features are great for reliable wired networks, they become less efficient when the underlying network is a lossy Wi-Fi or GSM connection. In this situation, UDP is more appropriate. So, with no surprise, also ROS doesn't commit to a single transport protocol but embrace both TCP and UDP. As previously said, when a node needs to connect to another one, it negotiates a connection using the appropriate transport. The result of the negotiation is that the two nodes are directly connected, with messages streaming from publisher to subscriber via ROS's proprietary implementations of TCPROS and UDPROS. This means that each transport has its own protocol for how the message data is exchanged. For example, using TCPROS, the negotiation would involve the publisher giving the subscriber the IP address and port on which to call connect. The subscriber then creates a TCP/IP socket to the specified address and port. The nodes exchange a Connection Header that includes information like the MD5sum of the message type and the name of the topic, and then the publisher begins sending serialised message data directly over the socket. It is possible to summarize the whole negotiating process as a two-step communication between the nodes and the

master. In the first part, the subscriber node queries the master (DNS server) for the publisher node address, and then a negotiation between the nodes starts. This process leaves some vulnerabilities that we need to handle. First of all, it may happen that a subscriber avoids querying the master because it already knows the address of the publisher; as such, we are not able to build a stateful solution based on the master only, but we need to implement additional mechanisms to secure node-to-node communications.

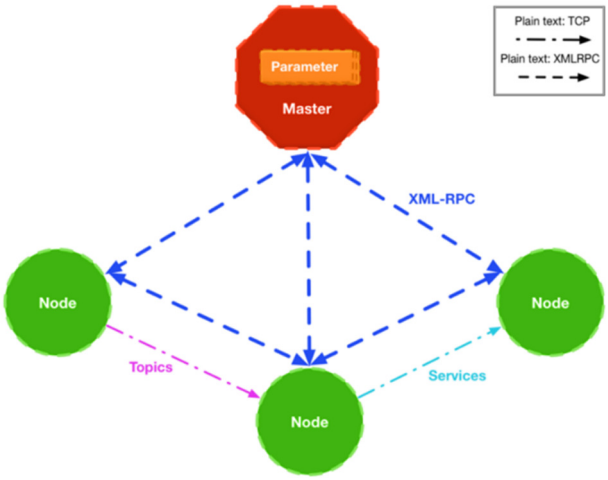

Figure 4: ROS Communications

2.1.2 MATLAB

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by **MathWorks**. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

In this work MATLAB has been used to test the validity of the algorithm before passing to the ROS version. In particular, ***polyfit*** and ***pinv*** have been used.
Polyfit (polynomial curve fitting) returns the coefficients for a polynomial $p(x)$ of degree n that is best fit in least-squares sense. It has been used to calculate the model of the sensor pad and object in terms of their stiffness.
Pinv, instead, returns the Moore-Penrose Pseudoinverse of a matrix and it has been used to calculate the center of mass of an object.

2.1.3 Python

Python is a popular programming language. It was created in 1991 by Guido van Rossum. It is used for: web development, software development, mathematics etc Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc). It has a simple syntax similar to the English language and it has syntax that allows developers to write programs with fewer lines than some other programming languages. It runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick. Python can be treated in a **procedural** way, an **object-orientated** way or a **functional** way.
The most recent major version of Python is Python 3. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
It is possible to write Python in an Integrated Development Environment, such as *Thonny, Pycharm, Netbeans* or *Eclipse* which are particularly useful when managing larger collections of Python files.
Python was designed to for **readability** and has some similarities to the English language with influence from mathematics.
Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.

It relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose. In this work Python has been used to create the Machine Learning classifier through the **Scikit learn** library. Furthermore, a **rospy** node has been created to obtain a classifier able to identify online an object when the procedure for the estimation of physical properties terminates.

2.1.3.1 Scikit learn

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

2.1.3.1.1 Random forest classifier

Random Forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.
To build a Random Forest in Python, the *Scikit learn* library has been used. In particular, *RandomForestClassifier* has been adopted and, as reported in [3], it has the following parameters:

- **n_estimators**: The number of trees in the forest.
- **criterion**: The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

- **max_depth:** The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- **min_samples_split**: The minimum number of samples required to split an internal node:
- **min_samples_leaf**: The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches.
- **min_weight_fraction_leaf**: The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node.
- **max_features**: The number of features to consider when looking for the best split.
- **max_leaf_nodes**: Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.
- **min_impurity_decrease**: A node will be split if this split induces a decrease of the impurity greater than or equal to this value.
- **bootstrap**: Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.
- **oob_score**: Whether to use out-of-bag samples to estimate the generalization accuracy.

2.2 Hardware description

2.2.1 KUKA IIWA LBR R800

A robot system comprises all the assemblies of an industrial robot, including the manipulator (mechanical system and electrical installations), controller, connecting cables, end effector (tool) and other equipment.

The industrial robot consists of the following components:

- Manipulator
- KUKA Sunrise Cabinet robot controller
- KUKA smartPAD control panel
- Software
- Options, accessories

Figure 5: Robot system

The LBR iiwa is classified as a **lightweight** robot and is a jointed-arm robot with 7 axes. All drive units and current-carrying cables are installed inside the robot. Every axis contains multiple sensors that provide signals for robot control (e.g. Position control and impedance control) and that are also used as a protective function for the robot. Every axis is monitored by sensors: axis range

sensors ensure that the permissible axis range is adhered to, torque sensors ensure that the permissible axis loads are not exceeded, and temperature sensors monitor the thermal limit values of the electronics. In the case of an unfavourable combination of permanently high demand on robot power and external temperature influences, the LBR is protected by this temperature monitoring which switches it off if the thermal limit values are exceeded. Following a cooling time, the LBR can be restarted with no need for additional measures.

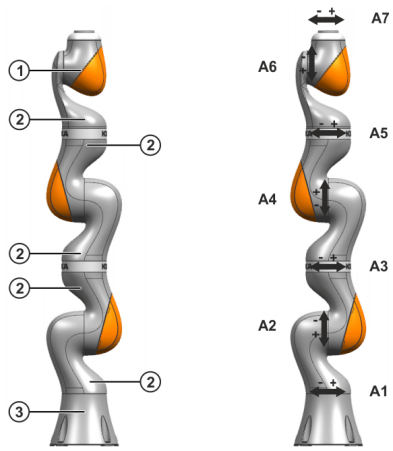

Figure 6: IIWA LBR R800

The kinematic system of both robot variants is of **redundant** design due to its 7 axes and consists of the following principal components:

1.**In-line wrist**: The robot is fitted with a 2-axis in-line wrist. The motors are located in axes A6 and A7.

2.**Joint module**: The joint modules consist of an aluminium structure. The drive units are situated inside these modules. In this way, the drive units are linked to one another via the aluminium structures.

3. **Base frame**: The base frame is the base of the robot. Interface A1 is located at the rear of the base frame. It constitutes the interface for the connecting cables between the robot, the controller and the energy supply system.

In the following table are resumed the principal features of KUKA LBR R800.

Robot Specifications		Robot Motion Speed		Robot Motion Range	
Axes	7	J1	$98^{o}/s$	J1	± 170
Payload	7 kg	J2	$98^{o}/s$	J2	$\pm 120^{o}$
H-reach	800 mm	J3	$100^{o}/s$	J3	$\pm 170^{o}$
Repeatability	±0.10mm	J4	$130^{o}/s$	J4	$\pm 120^{o}$
Robot Mass	24 kg	J5	$140^{o}/s$	J5	$\pm 170^{o}$
Structure	Articulated	J6	$180^{o}/s$	J6	$\pm 120^{o}$
Mounting	Floor, Inverted	J7	$180^{o}/s$	J7	$\pm 170^{o}$

Table 1: Robot specification

2.2.1.1 Mode selection

The industrial robot can be operated in the following modes:

- Manual Reduced Velocity (T1)
- Manual High Velocity (T2)
- Automatic (AUT)
- Controlled robot retraction (CRR)

2.2.1.1.1 Manual mode

Manual mode is the mode for setup work. Setup work is composed by all the tasks that have to be carried out on the industrial robot to enable automatic operation. New or modified programs must always be tested first in Manual Reduced Velocity mode (T1). The manipulator and its tooling must never touch or project beyond the safety fence. Workpieces, tooling and other objects must not become jammed as a result of the industrial robot motion, nor must they lead to short-circuits or be liable to fall off. All setup work must be carried out, where possible, from outside the safe-guarded area. In Manual High Velocity mode (T2): This mode may only be used if the application requires a test at a velocity higher than Manual Reduced Velocity. Teaching is not permissible in this operating mode.

2.2.1.1.2 Automatic mode

Automatic mode is only permissible in compliance with the following safety measures: All safety equipment and safeguards are present and operational. There are no persons in the system.

2.2.1.1.3 CRR

CRR is an operating mode which can be selected when the robot is stopped by the safety controller for one of the following reasons:

- Industrial robot violates an axis-specific or Cartesian monitoring space.
- Orientation of a safety-oriented tool is outside the monitored range,
- Robot violates a force or torque monitoring function.

2.2.2 WSG 50

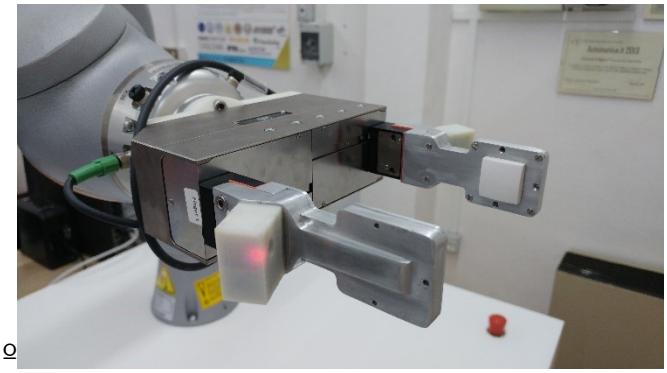

Figure 7: WSG50

The WSG 50 is a gripper from **WEISS Robotics** that has a long stroke, precise and highly dynamic movement, high stiffness and low weight in a compact design. Its electric motor allows highly dynamic gripping operations with maximum precision. Speed and gripping force are controlled continuously, which is why the WSG 50 is suitable for handling sensitive parts as well as for applications with a huge spectrum of different parts.

By constantly monitoring the finger motion and actual gripping forces, the WSG 50 automatically recognizes contact with the gripping object and monitors it throughout the whole handling process. This frees the system controls of any gripping functions. External positioning switches are no longer necessary. Hereby process simplicity is achieved with minimal integrational effort.

The WSG 50 integrates both a powerful control unit as well as the power stage into a compact electric gripper module. The WSG 50 gripper has a rich connectivity. It can either be controlled by text-based Ethernet TCP or UDP connections or over its field bus interfaces for PROFIBUS and optionally PROFINET. For legacy applications, CAN-Bus and RS-232 are supported as well. Operating software, configuration and application data are saved on an interchangeable MicroSD memory card and are easily portable. With its separate power supplies for logic and drive, safe torque off (STO) feature can easily implemented. Thus, in case of an emergency shutdown of the machine, the gripper's control unit will still be online while the motor is guaranteed torque-free. Because of large scale integration, neither an additional control unit nor any external servo amplifier are required for the WSG 50 to do its job. Best conditions to be easily integrated into your machine.

The WSG 50 is the first electric gripper with an integrated Ethernet interface. Because of its integrated web server, it can be configured using a common web browser.

The WSG 50 is also the first gripper to offer an electronic data interface for sensors integrated into the basic jaws. Using this interface, sensors can be directly integrated into the gripping control. The sensor port is connected to the WSG 50's internal control. This way, sensor signals can be tightly integrated into the gripping process.

The WSG 50 can therefore regulate the effective gripping force with precision and allows a safe handling of highly sensitive parts.

Technical features of WSG 50	
Stroke	110 mm (55 mm per finger)
Gripping Speed	5 - 420 mm/s
Gripping Force	5 - 80 N
Protection Class	IP 20
Temperature Range	5 to 50 °C
Dimensions	146 x 50 x 96.5 mm (L x B x H)
Weight	1.2 kg
Power Supply	24 V ±10%
Interfaces	10/100 MBit/s Ethernet, serial (RS-232), CAN-Bus, Profibus, PROFINET (optional), Modbus/TCP (optional)

Table 2: Technical features of WSG 50

It contains a powerful scripting extension that provides the access of a number of internal functions. Using the script editor that is integrated into the web-based configuration interface, it is possible to write your own programs and save them onto the WSG 50's microSD memory card. Special movement profiles as well as expansions of the control set or the addition of extra sensors into the gripping process are all possible.

2.2.3 Tactile sensor

2.2.3.1 Why a tactile sensor?

The dexterous manipulation is a very challenging robotic task, especially when the objects to manipulate have mechanical properties (e.g., weight, shape, stiffness, friction coefficient) very different from each another and/or not a priori known. In these cases, the use of exteroceptive sensors, able to provide data concerning object properties, is necessary to implement suitable control strategies. Tactile sense is used by humans to grasp and manipulate objects avoiding slippage, or to blindly operate in a dynamic environment. A definition of tactile

sensor is given by Lee and Nichols in [6]: "*A device or system that can measure a given property of an object or contact event through physical contact between the sensor and the object*". An artificial tactile sensor, by mimicking the human touch, should possess the capability to measure both dynamic and geometric quantities, i.e. contact forces and torques as well as spatial and geometrical information about the contacting surfaces. Each of these may be measured either as an average quantity for some part of the robot or as a spatially resolved, distributed quantity across a contact area.

2.2.3.2 Design and working principle

The sensor used in this work is based on a two-layer structure, i.e. a printed circuit board with optoelectronic components below a deformable silicon layer with a suitably designed geometry. The mechanical structure of the sensor has been optimized in terms of geometry and material physical properties to provide the sensor with different capabilities. The first capability is to work as a six-axis

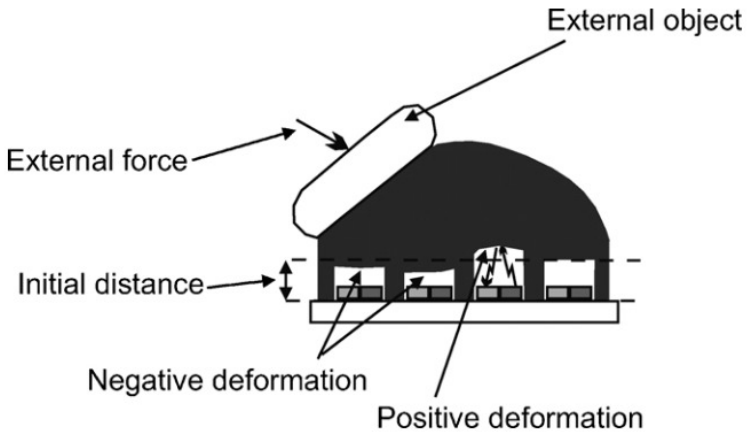

Figure 8: Tactile sensor working principle

force/torque sensor; additionally, the sensor can be used as a tactile sensor providing a spatially distributed information exploited to estimate the geometry of the contact with a stiff external object. The sensor is based on the use of optoelectronic technologies and it aims to overcome most of the problems

encountered in other prototypes, mainly: difficulty of the integration into small spaces, high costs, repeatability and complex conditioning electronics.

It has different capabilities, i.e. it can measure the six components of the force and torque vectors applied to it and it can be used as a tactile sensor providing a spatial and geometrical information about the contact with a stiff external object. A deformable elastic layer is positioned above a matrix of sensible points (**the taxels**) so as to transduce the force and torque vectors into deformations which are then measured by the sensible elements. The taxels, spatially distributed below the deformable layer, provide a set of signals corresponding to a distributed information called "**tactile map**" about the sensor deformations. Practically, the deformable layer transduces an external force and/or torque into a deformation of its bottom facet through its stiffness. An external force applied to the deformable layer produces local variations of the bottom surface of the elastic material and the couples of optical devices measure the deformations in a discrete number of points. The tactile sensor is based on the use of light-emitting diode (LED)–phototransistor couples and a deformable elastic layer positioned above the optoelectronics devices. A support layer is used to fix the initial distance between the electronic layer and the deformable layer to optimize the sensitivity and also to avoid cross-talk disturbances.

The optoelectronic components are organized in a matrix structure. For each couple, the LED illuminated the reflective surface, which coincided with the bottom facet of the deformable layer.

In particular, these vertical displacements produce a variation of the reflected light intensity and, accordingly, of the photocurrent measured by the photodetector. The vertical displacements, measured for each taxel, could be positive or negative; that is, the deformable layer could locally go up or down depending on the amplitudes of the tangential and normal force components. Finite element (FE) analysis could be used to actually reconstruct the external

force components through measurement of the elastic layer vertical deformation in a discrete number of points.

The maximum force level can be adapted by changing the hardness of the deformable layer. The maximum measurable force is limited by the maximum vertical deformation of the reflecting surface of each taxel, so the former can be changed by acting on the deformable layer geometry.

When an external force and torque is applied to the sensor, the distance of the reflecting surface of each cell from the corresponding LED/phototransistor couple the electronic layer can be subjected to a positive or a negative variation. These distance variations imply changes of the reflected and, accordingly, of the voltages measured on the phototransistor collector. Denoting with v_k the voltage variation of the k^{th} taxel, $v_k > 0$ denotes an increasing distance (and then a decreasing photo-currect), while $v_k < 0$ denotes a decreasing distance (and an increasing photo-currect) between the reflecting surface the electronic layer (obviously $v_k = 0$ denotes no variation).

The use of a rigid interface does not allow manipulation of fragile objects and adaptation to different object shapes. Moreover, a rigid interface allows measurement of pure torsional moments only by using a flat extended surface and by ensuring that the contact occurs with this flat surface perfectly parallel with the object. Therefore, a solution with a soft pad as contact interface becomes a mandatory requirement. The design of this soft pad has to provide a domed shape to improve the sensor adaptation to different object shapes. e second requirement concerns the transduction method. Whereas the objective is not only the estimation of contact force and moment, the classic solution of a force/torque sensor with mechanical frames with bonded sensing elements (usually strain gages or capacitive sensors) has to be overcome. A distributed measurement of the soft pad deformation due to a contact appears to be the best solution to reconstruct the maximum number of information about the manipulated object properties.

Since this solution is not implementable, the alternative is to measure the deformation of the soft pad in a discrete number of points by using sensing elements spatially distributed on a plane positioned on the bottom side of the pad. These distributed measurements are correlated to the contact state and allow, after a suitable calibration, reconstruction of several mechanical properties of the object.

2.2.3.3 Technical features

The sensor is mainly constituted by three components: a Printed Circuit Board (PCB), a rigid grid and a deformable cap. For each, the emitter/receiver couple is constituted by a unique optoelectronic component: a Surface Mount Technology (SMT) photo-reflector. In particular, the optoelectronic section of the PCB integrates 25 taxels, organized in a 5 × 5 matrix. As in previous prototypes, for each taxel, the conditioning electronics is constituted by two resistors: one to drive the LED and a second to transduce the photocurrent measured by the PT into a voltage directly compatible with an Analog-to-Digital (A/D) converter. The same 12− bit A/D converters (manufactured by Analog Devices) with 16 channels and a Serial Peripheral Interface (SPI), used in previous works, has been integrated in the PCB design. In order to integrate the tactile sensor in a standard parallel gripper, for this work, a microcontroller-based section has been integrated into the PCB design. In particular, an interfacing section constituted by the microcontroller PIC16F1824, manufactured by Microchip Technology, has been added on a separate rigid board, connected to the previous described part via a flexible section. The integration of the microcontroller allows to obtain a fully integrated sensor with a programmable device used to interrogate the sensor via a standard serial interface already available in most commercial grippers. The board is completed by a standard low-noise voltage regulator with an input

voltage range up to 12 V (typical range of supply voltage available on commercial grippers) and an output voltage equal to 3.3 V to supply the whole PCB.

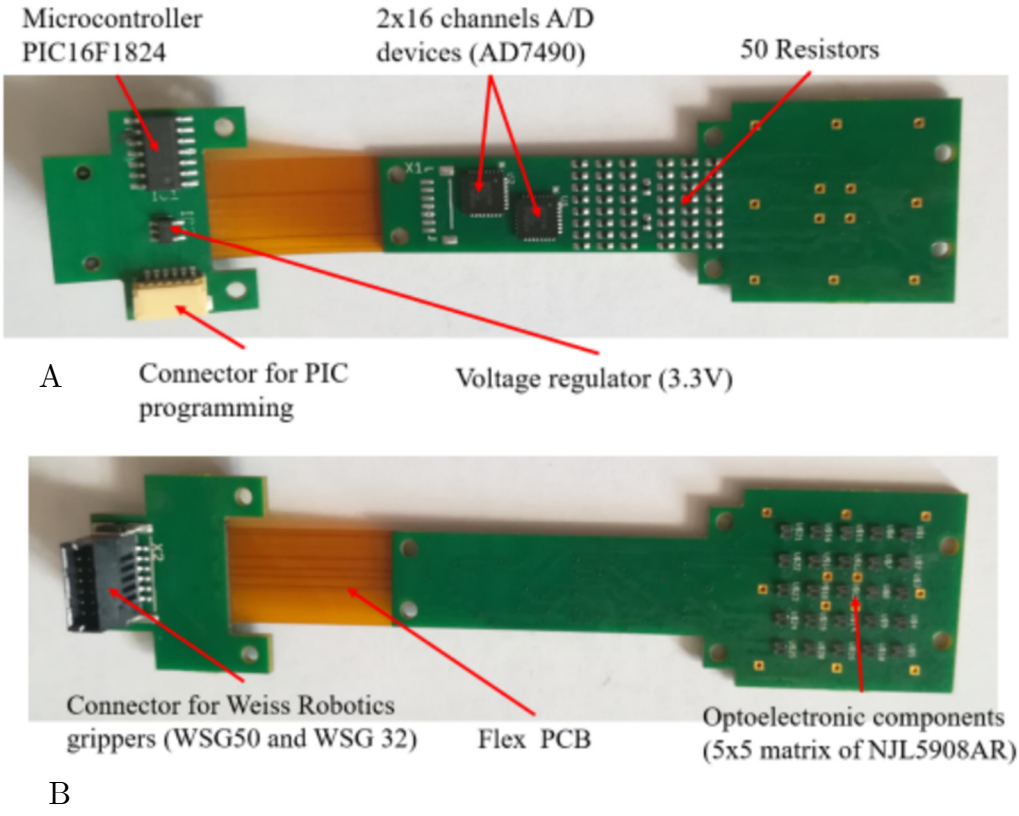

Figure 9: (A) PCB top view (B) bottom view

2.2.3.4 Characteristics of the deformable pad

A mechanical structure constituted by the deformable layer and the rigid grid is connected above the PCB. The deformable layer is mainly made of white silicone with a domed top side and a square base.

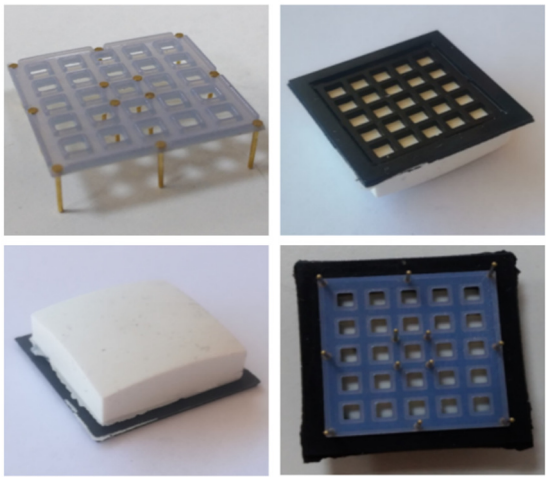

Figure 10: Sensor design

The mechanical properties of the silicone determine the full-scale and the sensitivity of the sensor. This prototype uses a shore hardness of 26 A. When external forces and/or moments are applied to the deformable layer, they produce vertical displacements of the white ceilings for all cells. The distances between the top of photo-reflectors and the white surfaces change, by producing variations of the reflected light and, accordingly, of the voltage signals measured by the PTs. The addition of the third component (i.e., the rigid grid) became necessary due to the electromechanical characteristic of the optical components. In particular, photo-reflector has a non-monotonic characteristic (see Fig. 11), which relates the measured voltage to the distance of a reflecting surface positioned in front of the component.

As a consequence, the rigid grid has to ensure that the reflecting surface never reaches distances from the component that fall into the non-monotonic area, highlighted by the red bars. Considering that the height of a component is 0.5 mm, the rigid grid has been designed with a thickness of 0.8 mm. With this choice the minimum reachable distance between a reflecting surface and a photo-reflector is $d_m = 0.3$ mm.On the other side, the silicone layer has been designed so that, in rest condition, the sum of the grid thickness and of the cell walls fixes the white ceilings at an initial distance $d_0 = 1$ mm from the emitting surface of the optical components. The integrated design of these two components allows to force the photo-reflectors to work in the monotonic working area, highlighted by the green bars.

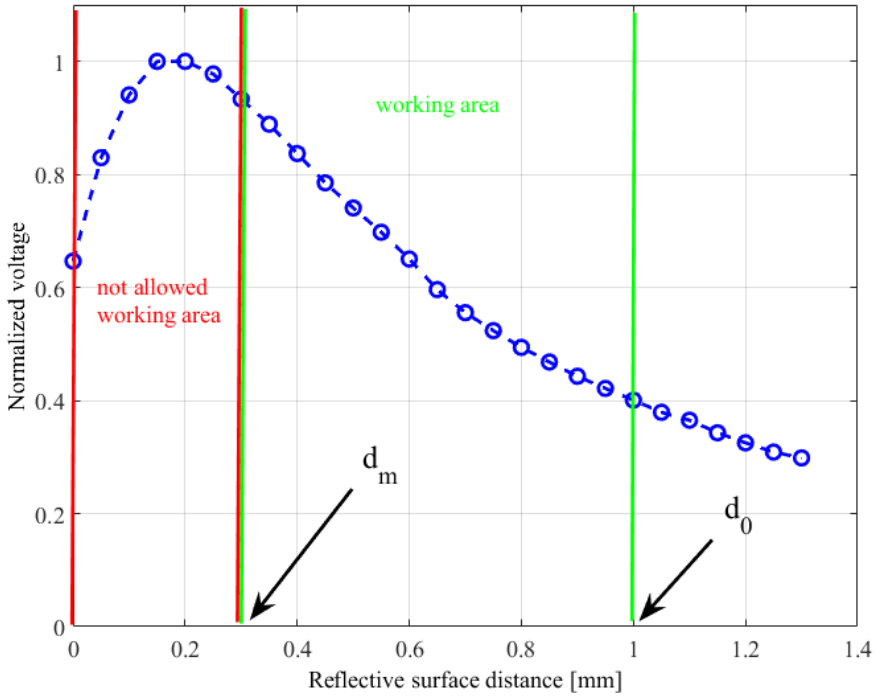

Figure 11 Characteristic for a single taxel: normalized voltage vs reflective surface distance

2.2.3.5 Integration of the sensor in a commercial gripper

The connector compatible with the sensor port available on the commercial grippers WSG-series, manufactured by Weiss Robotics, has been integrated, in order to provide the 5 V voltage supply to the sensor and for the physical implementation of the serial interface. The assembled force/tactile sensor is finally fixed inside an aluminum case suitably designed to house the sensor and for the mechanical connection to the WSG-series flange. Figure 12 reports a picture of the sensorized finger fully integrated with the gripper.

The microcontroller section available on the PCB allows two possible connections to exchange data with the main PC. In the fully integrated version, the PCB takes the voltage supply directly from the sensor port available on the WSG-series flange. The same port is used to implement a standard serial communication between the gripper and the sensor. The microcontroller interrogates the A/D converters via an SPI interface and transmits the raw via its serial port. The gripper is programmable by using the LUA programming language, that is an

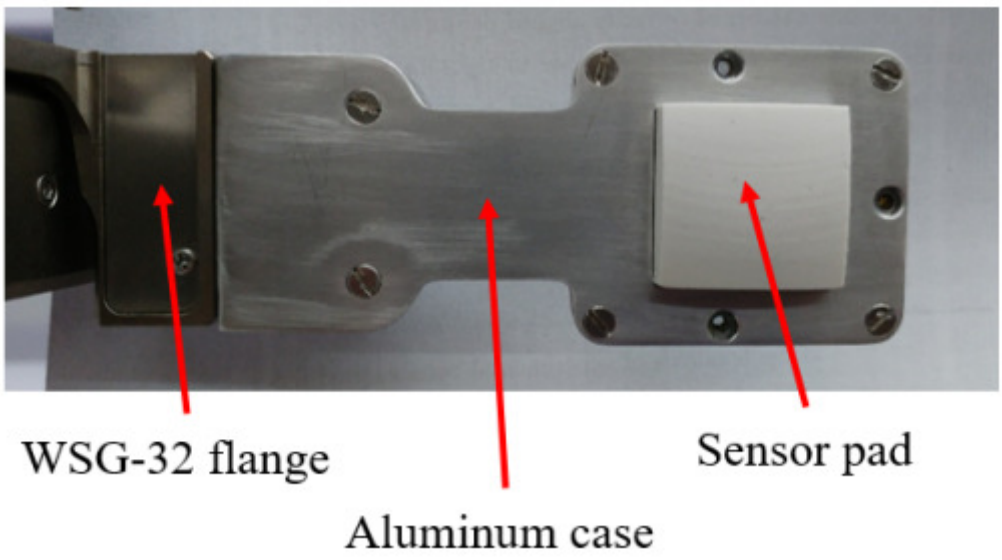

Figure 12: Integration of the finger in a commercial gripper

interpreted language suitably designed for embedded applications. The second possible connection from the microcontroller to the robot control PC foresees the

use of a standard USB-to-serial converter with an external cable, that directly connects the microcontroller to the main PC. In this case, the power supply and the serial transmission are implemented directly from the PC.

The solution to limitations related to the serial port latency time is to interface the microcontroller with a serial-to-WIFI adapter, in order to use a wireless connection directly with the PC. On the control PC, two different ROS nodes have been developed: one to interact with the gripper, if the first solution is selected, and another one to directly interact with the microcontroller in the second case. In both cases the ROS nodes receive raw data (i.e., the 50 bytes acquired by the A/D converters) and the first elaboration consists in the reconstruction of actual voltage values, which are published to be available for the whole ROS network.

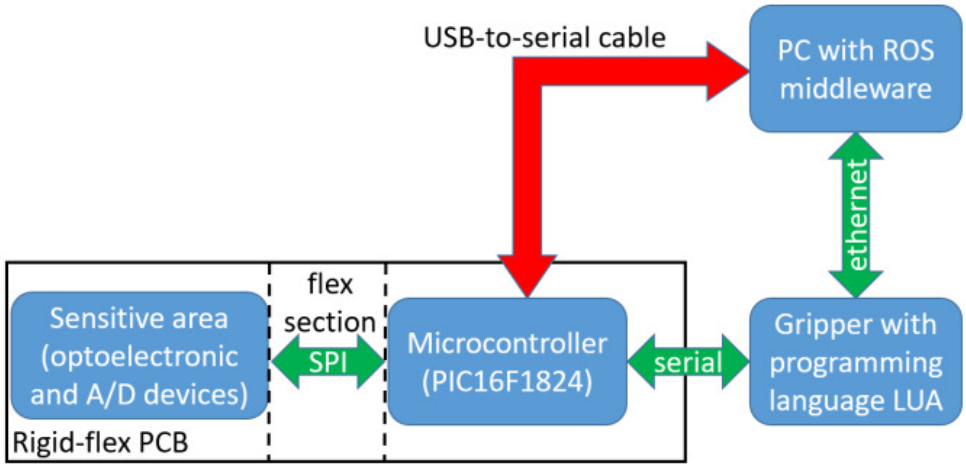

Figure 13: Data flow scheme of possible connections from the sensor to the control pc

Chapter 3 General approaches for the estimation of physical properties

This chapter contains an overview on the materials and their classification, to highlight their different behaviours when a certain force is applied to them. Then, different approaches for the estimation of physical properties are showed.

3.1 Overview of materials and their properties

In the robotic field it is necessary to deal with different objects made with materials with unknown properties. Hence, it is important to estimate these properties starting from an unknown object.

Materials have been conveniently grouped into three basic classifications:

Metals: they are composed of one or more metallic elements, and often also non-metallic elements for example, carbon, nitrogen, and oxygen in relatively small amounts. Atoms in their alloys are arranged in a very orderly structure in comparison to the ceramics and polymers

Figure 14: Metals

Ceramics: are compounds between metallic and non-metallic elements; they are most frequently oxides, nitrides, and carbides. They are characterized by an high temperature stability, high mechanichal strenght and high hardness-brittleness.

Figure 15: Ceramics

Polymers: include the familiar plastic and rubber materials. Many of them are organic compounds that are chemically based on carbon, hydrogen, and other non-metallic elements.

Figure 16: Polymers

This scheme is based primarily on chemical and atomic structure, and most materials fall into one distinct grouping or another, although there are some intermediates. In addition, there are the **composites**, combinations of two or more of the above three basic material classes.

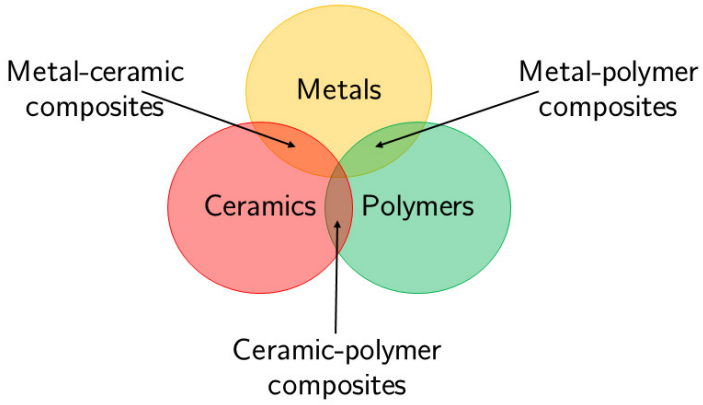

Figure 17: Classification of materials

All materials have their own characteristics. The knowledge of the latter is useful to use the most suitable material to each specific application. In particular, the most important properties of materials can be distinguished in the following way.

Chemical-structural properties: They concern the chemical composition and their internal structure. They fall within the chemical properties, and also the phenomena that are produced between the material and the environment external (oxidation, corrosion, etc.).

Technological properties: They concern the aptitude of the materials to undergo the various technological processes through which the mechanical pieces

are produced. The main technological properties are the fusibility, weldability, plasticity, machinability, malleability, ductility, extrusibility, feasibility, folding, etc.

Physical properties: They refer to the generic characteristics of materials, in relation to external factors such as gravity, electricity, etc. The main physical properties are the temperature of fusion, density, thermal expansion, etc.

Mechanical properties: They concern the ability of the material to resist the action of forces or external solicitations. Let us now list the different types of forces in order to define the corresponding mechanical characteristics:

- **Static forces**: They are applied constantly or vary slowly over time.
- **Dynamic forces**: They are applied in short times (<0,1s, impact forces) for example hammering. The ability of materials to counteract the effects of dynamic forces is called *resilience*.
- **Periodic forces**: They are variable periodically over time and with a high frequency. The ability of materials to counteract the effects of periodic forces is called fatigue *resistance*.
- **Concentrated forces**: They are applied in restricted or punctual areas. The ability of materials to counteract the effects of concentrated forces is called *stiffness*.
- **Frictional forces**: They occur between the contact surfaces and two sliding or sliding moving bodies (rolling friction). The ability of materials to counteract the effects of frictional forces is called *resistance to wear*.

3.1.1 Stress-strain diagram

The relationship between the stress (force per unit area) and strain (resulting compression/stretching, known as deformation) that a particular material displays is known as that particular material **stress–strain curve**. It is unique for each material and is found by recording the amount of deformation (strain)

at distinct intervals of tensile or compressive loading (stress). These curves reveal many of the properties of a material.

Stress–strain curves of various materials vary widely, and different tensile tests conducted on the same material yield different results, depending upon and also the speed of the loading. It is possible, however, to distinguish some common characteristics among the stress–strain curves of various groups of materials and, on this basis, to divide materials into two broad categories; namely, the ductile materials and the brittle materials.

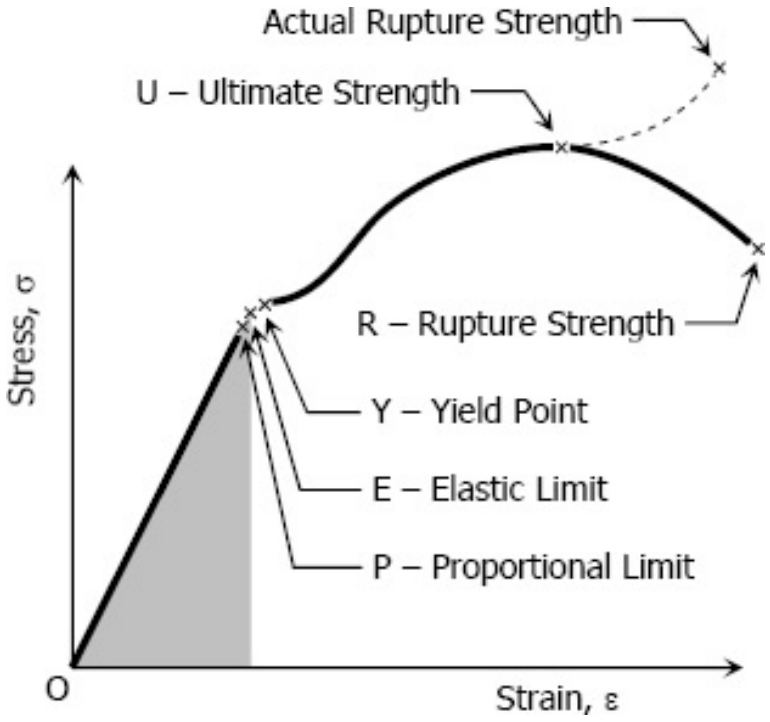

Figure 18:Stress-strain diagram

There are mainly different points in the graph.

Proportional limit: From the origin O to the point called proportional limit, the stress-strain curve is a straight line. This linear relation between elongation and the axial force and is called Hooke's Law so that within the proportional limit, the stress is directly proportional to strain or

$$\sigma = k\epsilon$$

The constant of proportionality k is called the Modulus of Elasticity E or Young Modulus and is equal to the slope of the stress-strain diagram from O to P. Then

$$\sigma = E\epsilon$$

Elastic limit: The elastic limit is the limit beyond which the material will no longer go back to its original shape when the load is removed, or it is the maximum stress that may be developed such that there is no permanent or residual deformation when the load is entirely removed.

Elastic and plastic ranges: The region in stress-strain diagram from O to P is called the elastic range. The region from P to R is called the plastic range.

Yield Point: Yield point is the point at which the material will have an appreciable elongation or yielding without any increase in load.

Ultimate Strength: The maximum ordinate in the stress-strain diagram is the ultimate strength or tensile strength.

Rapture Strength: Rapture strength is the strength of the material at rupture. This is also known as the breaking strength.

Modulus of Resilience: Modulus of resilience is the work done on a unit volume of material as the force is gradually increased from O to P, in $\mathrm{N \cdot m/m^3}$. This may be calculated as the area under the stress-strain curve from the origin O to up to the elastic limit E (the shaded area in figure 18). The resilience of the material is its ability to absorb energy without creating a permanent distortion.

Modulus of Toughness: Modulus of toughness is the work done on a unit volume of material as the force is gradually increased from O to R, in $\mathrm{N \cdot m/m^3}$. This may be calculated as the area under the entire stress-strain curve (from O to R). The toughness of a material is its ability to absorb energy without causing it to break.

Working Stress, Allowable Stress, and Factor of Safety: Working stress is defined as the actual stress of a material under a given loading. The maximum safe stress that a material can carry is termed as the allowable stress. The allowable stress should be limited to values not exceeding the proportional limit. However, since proportional limit is difficult to determine accurately, the

allowable tress is taken as either the yield point or ultimate strength divided by a factor of safety. The ratio of this strength (ultimate or yield strength) to allowable strength is called the factor of safety.

3.1.2 Elasticity and Hooke's law

Elasticity is the property of a material to deform under the action of an imposed stress state (for example, due to external forces applied) and then to reacquire its original form to the failure of the urging cause. Elasticity applies to both solid materials and fluids.

Otherwise, plasticity describes the deformation of a (solid) material undergoing non-reversible changes of shape in response to applied forces.

Hooke's law, the law of elasticity discovered by the English scientist Robert Hooke in 1660, which states that, for relatively *small deformations* of an object, the displacement or size of the deformation is directly proportional to the deforming force or load. Under these conditions the object returns to its original shape and size upon removal of the load. Elastic behaviour of solids according to Hooke's law can be explained by the fact that small displacements of their constituent molecules, atoms, or ions from normal positions is also proportional to the force that causes the displacement.

The deforming force may be applied to a solid by stretching, compressing, squeezing, bending, or twisting. Thus, a metal wire exhibits elastic behaviour according to Hooke's law because the small increase in its length when stretched by an applied force doubles each time the force is doubled. Mathematically, Hooke's law states that the applied force F equals a constant k times the displacement or change in length x, or

$$F = kx$$

The value of k depends not only on the kind of elastic material under consideration but also on its dimensions and shape.

At relatively large values of applied force, the deformation of the elastic material is often larger than expected on the basis of Hooke's law, even though the material remains elastic and returns to its original shape and size after removal of the force. Hooke's law describes the elastic properties of materials only in the range in which the force and displacement are proportional. Sometimes Hooke's law is formulated as

$$F = -kx$$

In this expression F no longer means the applied force but rather means the equal and oppositely directed restoring force that causes elastic materials to return to their original dimensions. Note that for engineering purposes we often assume the cross-section area of the material does not change during the whole deformation process. This is not true since the actual area will decrease while deforming due to elastic and plastic deformation.

3.2 Analysis of object physical properties

3.2.1 Mass

Mass is both a property of a physical body and a measure of its resistance to acceleration (a change in its state of motion) when a net force is applied. **Mass** is a measurement of the amount of matter something contains. The basic SI unit of mass is the kilogram (kg). In physics, mass is not the same as weight, even though mass is often determined by measuring the object's weight using a spring scale, rather than balance scale comparing it directly with known masses. Mass always remains the same, it is closely related to inertia and it cannot be measured directly. Instead, weight is the measurement of the pull of gravity on an object and it can change with the location, it is can be directly measured using a scale.

3.2.2 Center of mass

The center of mass of a distribution of mass in space is the unique point where the weighted relative position of the distributed mass sums to zero, or the point where if a force is applied it moves in the direction of the force without rotating. The distribution of mass is balanced around the center of mass and the average of the weighted position coordinates of the distributed mass defines its coordinates. Calculations in mechanics are often simplified when formulated with respect to the center of mass. It is a hypothetical point where entire mass of an object may be assumed to be concentrated to visualise its motion. In other words, the center of mass is the particle equivalent of a given object for application of Newton's laws of motion.

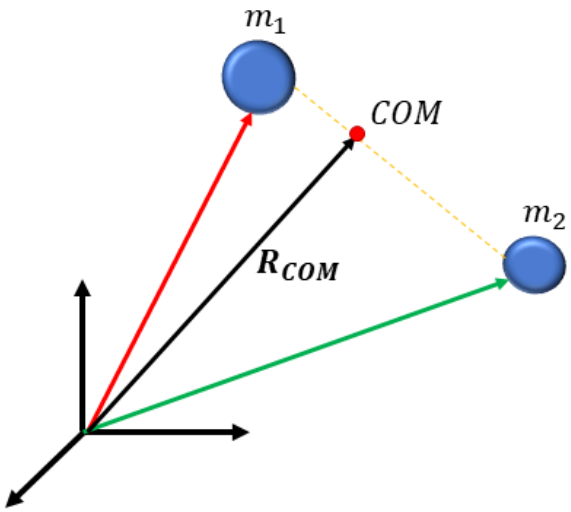

Figure 19: Center of mass of a system with two particles

In the case of a system of particles P_i $i = 1, \ldots, n$, each with mass m_i that are located in space with coordinates r_i $i = 1, \ldots, n$, the coordinates R of the center of mass satisfy the condition

$$\Sigma_i^n m_i (r_i - R) = 0$$

Solving the equation for R yelds the formula

$$R = \frac{1}{M}\Sigma_i^n m_i r_i$$

where M is the sum of the masses of all the particles.

If the mass distribution is continuous with the density $\rho(r)$ within a solid Q, then the integral of the weighted position coordinates of the points in this volume relative to the center of mass R over the volume V is zero, that is

$$\iiint \rho(\boldsymbol{r})(\boldsymbol{r} - \boldsymbol{R})dV = 0$$

Solve this equation for the coordinates R to obtain

$$\mathrm{R} = \frac{1}{\mathrm{M}} \iiint \rho(\mathbf{r})\mathbf{r}\, \mathrm{dV}$$

where M is the total mass in the volume.

If a continuous mass distribution has uniform density, which means ρ is constant, then the center of mass is the same as the centroid of the volume.

In the case of a single rigid body, the center of mass is fixed in relation to the body, and if the body has uniform density, it will be located at the centroid. The center of mass may be located outside the physical body, as is sometimes the case for hollow or open-shaped objects, such as a horseshoe. In the case of a distribution of separate bodies, such as the planets of the Solar System, the center of mass may not correspond to the position of any individual member of the system.

The center of mass is a useful reference point for calculations in mechanics that involve masses distributed in space, such as the linear and angular momentum of planetary bodies and rigid body dynamics. In orbital mechanics, the equations of motion of planets are formulated as point masses located at the centers of mass. The center of mass frame is an inertial frame in which the center of mass of a system is at rest with respect to the origin of the coordinate system.

3.2.3 Stiffness

Stiffness measures the resistance of materials to deformation in response to applied forces. Accurate knowledge of stiffness can be important for the

performance of robotic tasks involving explicit force control, although stiffness is the most important contact parameter for tasks with low dynamics. Mismatches between the real and nominal (that is, the stiffness value used in control design) stiffnesses can degrade force tracking and provoke bouncing and instability once contact is established between the manipulator and the environment. Accurate parameter knowledge is also relevant for improving telepresence in haptic feedback systems. Examples of tasks where these factors are relevant can be found in the area of telemanipulated robotic-assisted minimally invasive surgery (RMIS).

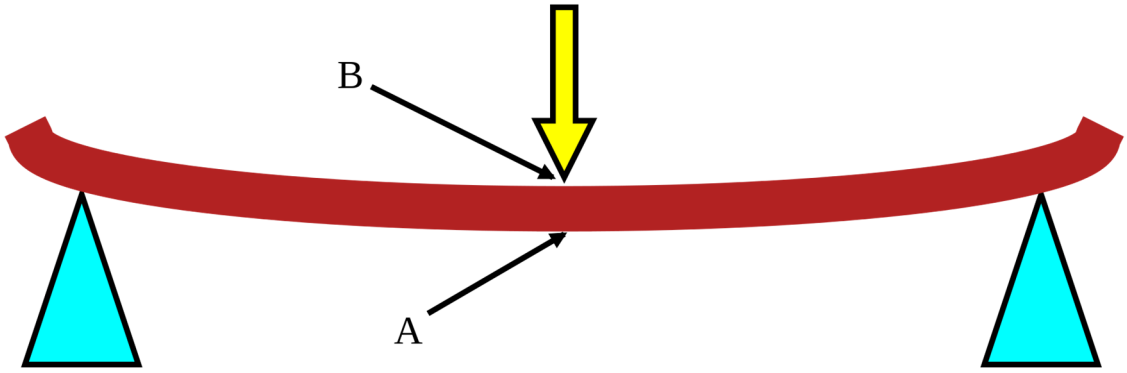

Figure 20: Scheme of beam subjected to bending

The stiffness, k, of a body is a measure of the resistance offered by an elastic body to deformation. For an elastic body with a single degree of freedom (DOF) (for example, stretching or compression of a rod), the stiffness is defined as

where, F is the force on the body

δ is the displacement produced by the force along the same degree of freedom (for instance, the change in length of a stretched spring)

In the International System of Units, stiffness is typically measured in $\frac{N}{m}$

Generally speaking, deflections (or motions) of an infinitesimal element (which is viewed as a point) in an elastic body can occur along multiple DOF (maximum of six DOF at a point). For example, a point on a horizontal beam can undergo both a vertical displacement and a rotation relative to its undeformed axis. When there are M degrees of freedom a $M \, x \, M$ matrix must be used to describe the

stiffness at the point. The diagonal terms in the matrix are the direct-related stiffnesses along the same degree of freedom and the off-diagonal terms are the coupling stiffnesses between two different degrees of freedom (either at the same or different points) or the same degree of freedom at two different points. In industry, the term **influence coefficient** is sometimes used to refer to the coupling stiffness.

It is noted that for a body with multiple DOF, the equation above generally does not apply since the applied force generates not only the deflection along its own direction (or degree of freedom) but also those along with other directions.

For a body with multiple DOF, in order to calculate a particular direct-related stiffness (the diagonal terms), the corresponding DOF is left free while the remaining should be constrained. Under such a condition, the above equation can be used to obtain the direct-related stiffness for the degree of freedom which is unconstrained. The ratios between the reaction forces (or moments) and the produced deflection are the coupling stiffnesses.

A description including all possible stretch and shear parameters is given by the elasticity tensor. The inverse of stiffness is flexibility or compliance, typically measured in units of metres per Newton. In rheology, it may be defined as the ratio of strain to stress, and so take the units of reciprocal stress, e.g. $\frac{1}{\mathrm{Pa}}$.

While it is a simplified model of reality, Hooke's Law is nevertheless useful in practice in robotic system control design, as it is amenable to analytical treatment and, more importantly, includes the stiffness parameter, that dominates force response for prolonged contacts with low indentation velocity. Low contact velocity is typical in most robotic interaction scenarios, to avoid damage to the manipulator or environment.

Hooke's Law can be generalized to account for energy dissipation by including a linear damping component, obtaining the Kelvin-Voigt (or spring-dashpot) model [7].

$$F_n = B\dot{\delta} + K\delta$$

where B is the damping coefficient, and $\dot{\delta}$ is the time derivative of the indentation. However, the linear damping term introduces physically unrealistic force behaviour: The damping term generates a discontinuous force at the beginning of impact, due to non-null contact velocity.

As the force due to the elastic term gradually decreases just before contact is broken at the end of a contact episode, the result will be dominated by a negative force holding the objects together, generated by the damping term.

Additionally, it has been demonstrated experimentally that the coefficient of restitution, used in discrete techniques to model energy loss during impact, should be dependent on velocity [8]. However, this is not the case if the equivalent coefficient of restitution is computed assuming a contact described by a spring-dashpot model. Nevertheless, despite these short-comings, it is a popular modelling option, since it is a simple linear model that allows energy dissipation in a perfectly elastic contact to be accounted for. Other linear contact models can be constructed by alternative combinations of linear spring and damper elements. These include the so-called Standard Linear Solid model, a parallel composition of a Hookean spring and a single spring-dashpot element, and its generalization, the Maxwell-Wiechert model (or generalized Maxwell model) which includes an arbitrary number of the latter elements. A nonlinear generalization of Hooke's Law, also ignoring energy dissipation, is Hertz's model (also known as the power law model)

$$F_n = K\delta^m.$$

Unlike Hooke's Law, it describes a nonlinear relation between local indentation and normal force, with K and m depending on geometrical and material properties of the colliding bodies. These parameters can be computed analytically for specific conditions of axisymmetric contact of certain solids, such as sphere-sphere central

collision, however no solution exists in the general case. By adding a damping term to Hertz's Law, energy dissipation can be modelled. A simple extension is the impact-pair model, described by Dubowski and Freudenstein [9], where a linear damping term is added to a nonlinear Hertzian spring. However, just like in the case of the Kelvin-Voigt model, the linear dampers introduce unrealistic force discontinuities. To address these problems, Hunt and Crossley [17] in proposed a nonlinear damping term, obtaining the model in the following,

$$F_n = B\delta^p\dot{\delta}^n + K\delta^m$$

This model addresses limitations of the Kelvin-Voigt and impact-pair models by making damping depend on indentation, ensuring continuous force, beginning and ending at zero in every contact episode, although negative sticking forces may still be generated]. An alternative model based on similar principles but different formulation of the nonlinear damping term was described by Lee and Wang in [10]. All the above models consider elastic contact. Adaptation to plastic contact can be achieved by differentiating the models of the compression and restitution phases of contact, accounting for persistent deformation of the material]. In practice, these models are less commonly used, not only because of parameterization difficulties but also because persistent deformation is not expected nor relevant in most applications. Analytical parameterization of Hertz's model in generic contact scenarios is not possible. An alternative approach for representing nonlinear elastic behaviour is the empiric model. This model attempts to relate the stiffness coefficient to the area of contact A,

$$Fn = K(A)\delta$$

where K(A) is proportional to the square root of A and dependent of contact geometry and material properties of the colliding objects.

3.2.4 Friction

Friction is the force resisting the relative motion of solid surfaces, fluid layers, and material elements sliding against each other.

There are several types of friction:

- **Dry friction** is a force that opposes the relative lateral motion of two solid surfaces in contact. Dry friction is subdivided into static friction ("stiction") between non-moving surfaces, and kinetic friction between moving surfaces. With the exception of atomic or molecular friction, dry friction generally arises from the interaction of surface features, known as asperities.
- **Fluid friction** describes the friction between layers of a viscous fluid that are moving relative to each other.
- **Lubricated friction** is a case of fluid friction where a lubricant fluid separates two solid surfaces.
- **Skin friction** is a component of drag, the force resisting the motion of a fluid across the surface of a body.
- **Internal friction** is the force resisting motion between the elements making up a solid material while it undergoes deformation.

3.2.4.1 Dry friction

Dry friction is a force that opposes the relative lateral motion of two solid surfaces in contact. Dry friction is subdivided into static friction ("stiction") between non-moving surfaces, and kinetic friction between moving surfaces. With the exception of atomic or molecular friction, dry friction generally arises from the interaction of surface features, known as asperities.

Coulomb friction is an approximate model used to calculate the force of dry friction. It is governed by the model:

$$F_f < \mu F_n$$

where:

- F_f is the force of friction exerted by each surface on the other. It is parallel to the surface, in a direction opposite to the net applied force.
- μ is the coefficient of friction, which is an empirical property of the contacting materials.
- F_n is the normal force exerted by each surface on the other, directed perpendicular (normal) to the surface.

The Coulomb friction, F_f may take any value from zero up to μF_n, and the direction of the frictional force against a surface is opposite to the motion that surface would experience in the absence of friction. Thus, in the static case, the frictional force is exactly what it must be in order to prevent motion between the surfaces; it balances the net force tending to cause such motion. In this case, rather than providing an estimate of the actual frictional force, the Coulomb approximation provides a threshold value for this force, above which motion would commence. This maximum force is known as traction.

The force of friction is always exerted in a direction that opposes movement (for kinetic friction) or potential movement (for static friction) between the two surfaces.

3.2.4.2 Normal force

The normal force is defined as the net force compressing two parallel surfaces together; and its direction is perpendicular to the surfaces. In the simple case of a mass resting on a horizontal surface, the only component of the normal force is the force due to gravity, where

$$N = mg$$

In this case, the magnitude of the friction force is the product of the mass of the object, the acceleration due to gravity, and the coefficient of friction. However,

the coefficient of friction is not a function of mass or volume; it depends only on the material.

If an object is on a level surface and the force tending to cause it to slide is horizontal, the normal force N between the object and the surface is just its weight, which is equal to its mass multiplied by the acceleration due to earth's gravity, g. If the object is on a tilted surface such as an inclined plane, the normal force is less, because less of the force of gravity is perpendicular to the face of the plane. Therefore, the normal force, and ultimately the frictional force, is determined using vector analysis, usually via a free body diagram. Depending on the situation, the calculation of the normal force may include forces other than gravity.

3.2.4.2.1 Coefficient of friction

The coefficient of friction (COF), often symbolized by the Greek letter μ, is a dimensionless scalar value which describes the ratio of the force of friction between two bodies and the force pressing them together. The coefficient of friction depends on the materials used. Coefficients of friction range from near zero to greater than one. It is an axiom of the nature of friction between metal surfaces that it is greater between two surfaces of similar metals than between two surfaces of different metal, hence, brass will have a higher coefficient of friction when moved against brass, but less if moved against steel or aluminium.

For surfaces at rest relative to each other $\mu = \mu_s$ is the **coefficient of static friction**. This is usually larger than its kinetic counterpart. The coefficient of static friction exhibited by a pair of contacting surfaces depends upon the combined effects of material deformation characteristics and surface roughness, both of which have their origins in the chemical bonding between atoms in each of the bulk materials and between the material surfaces and any adsorbed material. The fractality of surfaces, a parameter describing the scaling behaviour

of surface asperities, is known to play an important role in determining the magnitude of the static friction.

For surfaces in relative motion $\mu = \mu_k$ is **the coefficient of kinetic friction**. The Coulomb friction is equal to F_f and the frictional force on each surface is exerted in the direction opposite to its motion relative to the other surface. Most dry materials in combination have friction coefficient values between 0.3 and 0.6. For example, silicone rubber or acrylic rubber-coated surfaces have a coefficient of friction that can be substantially larger than 1.

While it is often stated that the COF is a "material property," it is better categorized as a "system property." Unlike true material properties (such as conductivity, dielectric constant, yield strength), the COF for any two materials depends on system variables like temperature, velocity, atmosphere and also what are now popularly described as aging as well as on geometric properties of the interface between the materials, namely surface structure. For example, a copper pin sliding against a thick copper plate can have a COF that varies from 0.6 at low speeds (metal sliding against metal) to below 0.2 at high speeds when the copper surface begins to melt due to frictional heating. The latter speed, of course, does not determine the COF uniquely; if the pin diameter is increased so that the frictional heating is removed rapidly, the temperature drops, the pin remains solid and the COF rises to that of a 'low speed' test.

3.2.4.3 Static friction

Static friction is friction between two or more solid objects that are not moving relative to each other. For example, static friction can prevent an object from sliding down a sloped surface. The coefficient of static friction, typically denoted as μ_s, is usually higher than the coefficient of kinetic friction. Static friction is considered to arise as the result of surface roughness features across multiple length-scales at solid surfaces. These features, known as asperities are present

down to nano-scale dimensions and result in true solid to solid contact existing only at a limited number of points accounting for only a fraction of the apparent or nominal contact area. The linearity between applied load and true contact area, arising from asperity deformation, gives rise to the linearity between static frictional force and normal force, found for typical Amonton-Coulomb type friction.

The static friction force must be overcome by an applied force before an object can move. The maximum possible friction force between two surfaces before sliding begins is the product of the coefficient of static friction and the normal force: $F_{max} = \mu_S F_n$ When there is no sliding occurring, the friction force can have any value from zero up F_{max}. Any force smaller than F_{max}, attempting to slide one surface over the other is opposed by a frictional force of equal magnitude and opposite direction. Any force larger than F_{max}, overcomes the force of static friction and causes sliding to occur. The instant sliding occurs, static friction is no longer applicable. The friction between the two surfaces is then called kinetic friction. An example of static friction is the force that prevents a car wheel from slipping as it rolls on the ground. Even though the wheel is in motion, the patch of the tire in contact with the ground is stationary relative to the ground, so it is static rather than kinetic friction. The maximum value of static friction, when motion is impending, is sometimes referred to as limiting friction, although this term is not used universally.

3.2.4.4 Kinetic friction

Kinetic friction, also known as dynamic friction or sliding friction, occurs when two objects are moving relative to each other and rub together (like a sled on the ground). The coefficient of kinetic friction is typically denoted as μ_k, and is usually less than the coefficient of static friction for the same materials. The friction force between two surfaces after sliding begins is the product of the coefficient of kinetic

friction and the normal force $F_k = \mu_k F_n$ .New models are beginning to show how kinetic friction can be greater than static friction. Kinetic friction is now understood, in many cases, to be primarily caused by chemical bonding between the surfaces, rather than interlocking asperities; however, in many other cases roughness effects are dominant, for example in rubber to road friction. Surface roughness and contact area affect kinetic friction for micro- and nano-scale objects where surface area forces dominate inertial forces. The origin of kinetic friction at nanoscale can be explained by thermodynamics. Upon sliding, new surface forms at the back of a sliding true contact, and existing surface disappears at the front of it. Since all surfaces involve the thermodynamic surface energy, work must be spent in creating the new surface, and energy is released as heat in removing the surface. Thus, a force is required to move the back of the contact, and frictional heat is released at the front.

3.3 Existing estimation algorithms

An approach consists in using sensorial data given by video as proposed by Bhat et al. in [11]. It presents an optimization framework for estimating the motion and underlying physical parameters of a rigid body in free flight from video. The algorithm takes a video clip of a tumbling rigid body of known shape and generates a physical simulation of the object observed in the video clip. This solution is found by optimizing the simulation parameters to best match the motion observed in the video sequence. These simulation parameters include initial positions and velocities, environment parameters like gravity direction and parameters of the camera. A global objective function computes the sum squared difference between the silhouette of the object in simulation and the silhouette obtained from video at each frame. Applications include creating interesting rigid

body animations, tracking complex rigid body motions in video and estimating camera parameters from video.

Another method is proposed by Kaboli et al. in [14]. The paper proposes a solution for the problem of in-hand object recognition via surface textures. In this study, a robotic hand with an artificial skin on the fingertips was employed to explore the texture properties of various objects. This was conducted via the small sliding movements of the fingertips of the robot over the object surface as a human does. Using our proposed robust tactile descriptors, the robotic system is capable of extracting information-rich data from the raw tactile signals. These features then assist learning algorithms in the construction of robust object discrimination models. The experimental results show that the robotic hand distinguished between different in-hand objects through their texture properties (regardless of the shape of the in-hand objects). In this work, features without physical meanings are used for the object classification. Instead, the thesis proposes a method that uses as features for classification, the same physical properties used for manipulation.

Danfei Xu et al. in [15] propose a methodology in which they integrate multimodal tactile sensing (force, vibration and temperature). They program the robot to make exploratory movements similar to those humans make when identifying objects by their compliance, texture, and thermal properties. This process is called Bayesian exploration. When exploring a texture, the discrimination algorithm adaptively selects the optimal movement to make and property to measure based on previous experience to differentiate the texture from a set of plausible candidates.

3.3.1 Challenges in stiffness estimation

Stiffness estimation as a relevant and complex problem. This complexity arises from multiple factors, such as sensorial information with poor signal-to-noise ratio

(SNR) or limited accuracy of joint encoders. Uncertainty in the environment geometry is also relevant, as it obscures the relative positioning of the end-effector w.r.t. the environment, creating difficulties to detect free-space/contact transitions. This may be problematic, since most estimation techniques require accurate identification of the initial contact point. A position-based approach to obtain this information would be to contrast the end-effector position with known information describing the geometry of the environment. However, this may not be an option if the environment is poorly known or unknown, making it challenging to estimate the environment stiffness in these scenarios. Zero- crossing detection of force would allow such transitions to be detected, but force sensors are noisy, subjected to bias and may return non-null forces even in free-space operation, obscuring the exact instant of transition. This problem is exacerbated for low contact forces. These issues can be illustrated by considering a direct estimation approach using Hooke's Law along a single dimension. Hooke's Law is a linear model that relates applied force with deformation along an uncoupled dimension and stiffness coefficient. If $\mathrm{F_e}$ is the force applied by the robot on the environment, K_s the system stiffness coefficient, X_t is the current robot end-effector position and X_0 is the initial contact position, then Hooke's Law states that

$$Fe = K_s(X_t - X_0)$$

The estimation $\widehat{K_s}$ can be given by

$$\widehat{K_s} = \frac{F_e}{\widehat{X_t} - \widehat{X_0}}$$

where F_e is obtained by force sensing and $\widehat{X_t}$ comes from robot forward kinematics. The estimation $\widehat{X_0}$ could be obtained from geometric information if the relative positioning of the end-effector and object is well-known. However, if such information is not readily available, indirect contact position estimation approaches must be adopted. A viable approach entails the use of force

information to detect the free-space/contact transition and estimating X_0 with the end-effector position at that instant.

While the latter problem hardly occurs in soft environments, the former one still poses considerable practical difficulties. These can be overcome in tasks where the object geometry is static and well-known (as the case of many industrial machining tasks) or if it is possible to start the task with the end-effector already placed at the initial contact position (as assumed in many estimation works).

Chapter 4 Proposed algorithms for the estimation of physical properties

The following chapter explains the algorithms used to estimate some physical properties of unknown objects. In particular, the first section presents the algorithm used to estimate the mass and the center of mass, then the algorithm for the estimation of the stiffness and the friction coefficient are showed.

4.1 Mass and Center of Mass estimation

To estimate mass and center of mass of an object a robot KUKA IIWA LBR R800 has been used with a gripper WEISS ROBOTICS WSG 50 that has two fingers. A tactile sensor is attached on each finger as in *Fig. 21* that gives force and torque measurements, transferred to the PC through a wireless connection.

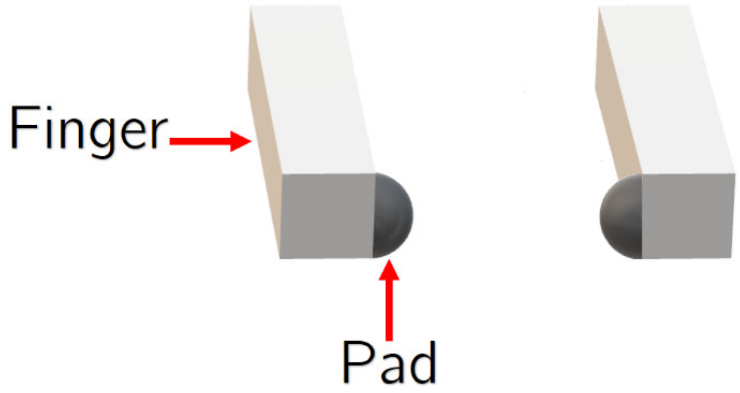

Figure 21:Scheme of the gripper and tactile sensors

The idea on which this algorithm is based, is to grasp the unknown object, rotate it several times and then read the force and torque measurements given by the tactile sensors. These data are then elaborated in order to obtain mass and center of mass estimates.

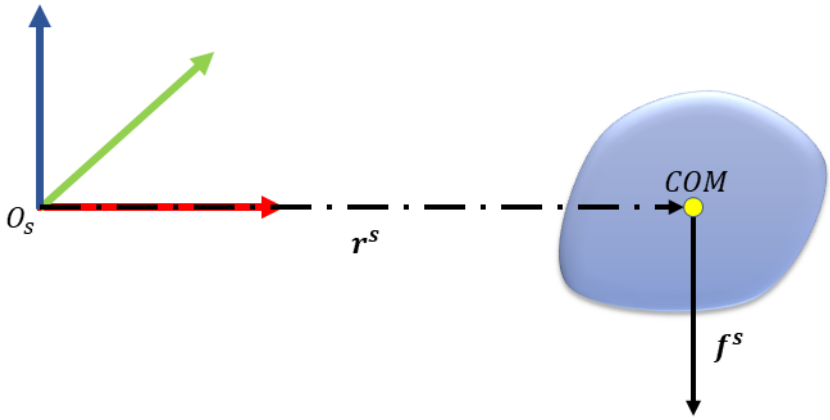

Figure 22: Center of mass of an object

Let Σ_{s1} and Σ_{s2} be the frame of each sensor, and Σ_{grasp} be the grasp frame that is located at the center of the fingers as shown in Figure 23. The relative position of these frames is not constant, but it depends on the position of the gripper fingers. Choosing the grasp frame aligned with the sensor 1 frame, the relative position of the frames is given by:

$$\boldsymbol{R}_{s1}^{grasp} = \boldsymbol{I}_3$$

$$\boldsymbol{R}_{s2}^{grasp} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

$$\boldsymbol{p}_{s1}^{grasp} = \begin{bmatrix} 0 \\ 0 \\ -\frac{d_f}{2} \end{bmatrix}$$

$$\boldsymbol{p}_{s2}^{grasp} = \begin{bmatrix} 0 \\ 0 \\ \frac{d_f}{2} \end{bmatrix}$$

where d_f is the distance between fingers which depends on gripper state. Let s_i be the ith sensor, the wrench of each finger can be expressed in the grasp frame as

$$\boldsymbol{f}_{s_i}^{grasp} = \boldsymbol{R}_{s_i}^{grasp} \boldsymbol{f}_{s_i}^{s_i}$$

$$\boldsymbol{\tau}_{s_i}^{grasp} = \boldsymbol{R}_{s_i}^{grasp} \boldsymbol{\tau}_{s_i}^{s_i} + \boldsymbol{p}_{s_i}^{grasp} \boldsymbol{f}_{s_i}^{grasp}$$

Finally, the external wrench in the grasp frame is simply the sum of the components of each finger in the grasp frame.

$$\begin{bmatrix} \boldsymbol{f}^{grasp} \\ \boldsymbol{\tau}^{grasp} \end{bmatrix} = [\boldsymbol{I}_6 \;\; \boldsymbol{I}_6] \begin{bmatrix} \boldsymbol{f}_{s_1}^{grasp} \\ \boldsymbol{\tau}_{s_1}^{grasp} \\ \boldsymbol{f}_{s_2}^{grasp} \\ \boldsymbol{\tau}_{s_2}^{grasp} \end{bmatrix}$$

Note that since the z-axes of the three frames are aligned to the same straight line, the variable d_f affects only the two moments τ_x and τ_y and not the other components of the wrench.

Σ_b is the base frame in which:

$$\boldsymbol{f}^{\boldsymbol{b}} = m\boldsymbol{g}^{\boldsymbol{b}}$$

$$\boldsymbol{\mu}^{\boldsymbol{b}} = \boldsymbol{S}(\boldsymbol{r}^{\boldsymbol{b}}) m\boldsymbol{g}^{\boldsymbol{b}}$$

where $\boldsymbol{f}^{\boldsymbol{b}}$ and $\boldsymbol{\mu}^{\boldsymbol{b}}$ are external force and torque applied to the object.

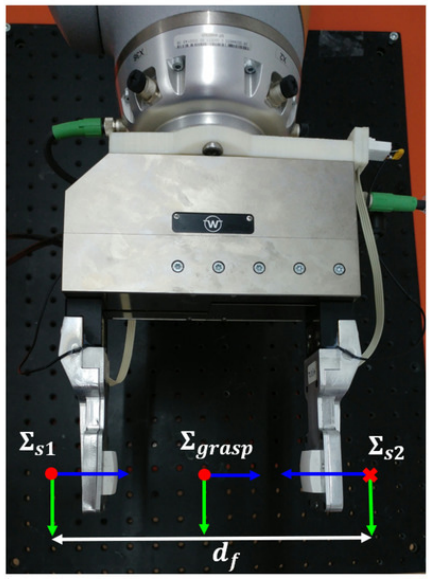

Figure 23: Sensor frame

Expressing $\boldsymbol{f}^b$ through the rotation matrix $\boldsymbol{R}_s^b$ it is possible to write $\boldsymbol{f}^b = \boldsymbol{R}_s^b \boldsymbol{f}^s$

$$\boldsymbol{R}_s^b \boldsymbol{f}^s = m\boldsymbol{g}^b$$
$$\boldsymbol{f}^s = m\boldsymbol{R}_b^s \boldsymbol{g}^b$$
$$\boldsymbol{f}^s = m\boldsymbol{g}^s$$

Similarly, it is possible to write $\boldsymbol{\mu}^b = \boldsymbol{R}_s^b \boldsymbol{\mu}^s$ and

$$\boldsymbol{R}_s^b \boldsymbol{\mu}^s = \boldsymbol{S}(\boldsymbol{r}^b)\, m\mathbf{g}^b$$
$$\boldsymbol{\mu}^s = \boldsymbol{R}_s^b \boldsymbol{S}(\boldsymbol{r}^b)\, m\boldsymbol{g}^b$$
$$\boldsymbol{\mu}^s = m\boldsymbol{S}(\boldsymbol{R}_b^s \boldsymbol{r}^b)\, \boldsymbol{R}_b^s \boldsymbol{g}^b$$
$$\boldsymbol{\mu}^s = m\boldsymbol{S}(\boldsymbol{r}^s)\, \boldsymbol{g}^s$$
$$\boldsymbol{\mu}^s = -\boldsymbol{S}(\boldsymbol{g}^s)\, m\boldsymbol{r}^s$$

where $\boldsymbol{g^s} = \boldsymbol{R_b^s}\, \boldsymbol{g^b}$ and $\boldsymbol{g^b} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}$

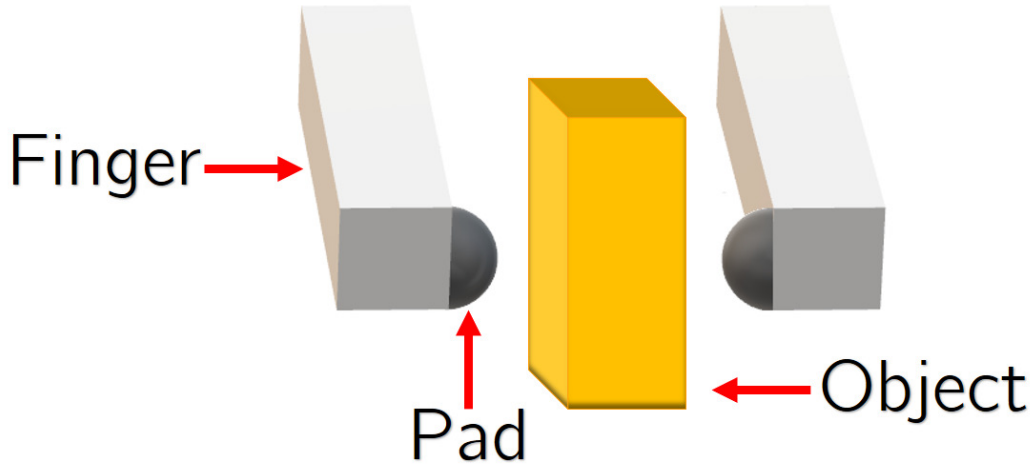

Figure 24: Fingers and object

More in detail, each object has been rotated different times and after each rotation, different force and torque measurements has been read and algebraically mediated to obtain more precise results. In particular, for each object five rotations have been executed in a range of angles from 0° to 10° with step of 2.5°. After N rotations it is:

$$\begin{bmatrix} \boldsymbol{f}_1^S \\ \boldsymbol{f}_2^S \\ . \\ . \\ . \\ \boldsymbol{f}_N^S \\ \boldsymbol{\mu}_1^S \\ \boldsymbol{\mu}_2^S \\ . \\ . \\ . \\ \boldsymbol{\mu}_N^S \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}_1^S & \mathbf{0}_{3x3} \\ \boldsymbol{g}_2^S & \mathbf{0}_{3x3} \\ . & . \\ . & . \\ . & . \\ \boldsymbol{g}_N^S & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x1} & -\boldsymbol{S}(\boldsymbol{g}_1^S) \\ \mathbf{0}_{3x1} & -\boldsymbol{S}(\boldsymbol{g}_2^S) \\ . & . \\ . & . \\ . & . \\ \mathbf{0}_{3x1} & -\boldsymbol{S}(\boldsymbol{g}_N^S) \end{bmatrix} \begin{bmatrix} m \\ m\boldsymbol{r}^S \end{bmatrix}$$

It is $\boldsymbol{b} = \boldsymbol{Ax}$. Dimensionally:

- $\boldsymbol{A}$ is 6N x 4 matrix
- $\boldsymbol{x}$ is 4x1 column array
- $\boldsymbol{b}$ is 6Nx1 column array

It can be solved as $\boldsymbol{x} = \boldsymbol{A}^+\boldsymbol{b}$ where $\boldsymbol{A}^+$ is the *pseudoinverse* matrix of A. Moreover, once calculated $\mathbf{x}$, obviously it is possible to calculate mass and center of mass of the object. Below, there is the pseudocode relative to the calculation of mass and center of mass.

Algorithm 1 Estimation of the mass and center of mass

input:
$\mathbf{f^b} \leftarrow$ external forces in base frame
$\mu^{\mathbf{b}} \leftarrow$ external torques in base frame
$\mathbf{g^b} \leftarrow$ gravity in base frame
$\mathbf{f^s} \leftarrow$ external forces in sensor frame
$\mu^{\mathbf{s}} \leftarrow$ external torques in sensor frame
$\mathbf{g^s} \leftarrow$ gravity in sensor frame
$\mathbf{g^s} \leftarrow$ gravity in sensor frame
$m \leftarrow$ mass of the unknown object
$\mathbf{R_b^s} \leftarrow$ Rotation matrix between base frame and sensor frame
$\mathbf{S}(\mathbf{g^s}) \leftarrow$ Skew symmetric matrix
$\dagger \leftarrow$ pseudo-inverse
procedure CALCULATION OF THE MASS AND CENTER OF MASS
  for $i = 1 : numRotations$ **do**
    $\mathbf{g_i^s} = \mathbf{R_{bi}^s g^b}$
    $\mathbf{f_i^s} = m\mathbf{g_i^s}$
    $\mu_{\mathbf{i}}^{\mathbf{s}} = -\mathbf{S}(\mathbf{g_i^s})m\mathbf{r^s}$
    Put $\mathbf{f_i^s}$ and $\mu_{\mathbf{i}}^{\mathbf{s}}$ into a vector $\mathbf{b}$
    Put $-\mathbf{S}(\mathbf{g_i^s})$ and $\mathbf{g_i^s}$ into a matrix $\mathbf{A}$
  end for
  return: $[m, m\mathbf{r^s}] = \mathbf{A}^\dagger\mathbf{b}$
end procedure

4.2 Stiffness estimation

The first step, in this procedure, is to calculate the nonlinear stiffness model of the sensor pad. With this purpose, a resin object has been used whose stiffness is supposed to be infinite and it has been positioned between the fingers of the gripper. The gripper has been gradually closed until the contact has been detected. Then an iterative routine begins. A position control approach has been adopted to move the gripper. Different small displacements have been imposed and they cause only pad deformations $\boldsymbol{\Delta_{pad}}$ since the object is supposed to be infinitely rigid. Therefore, reading the corresponding force given by the tactile sensors, it has been possible to calculate the sensor pad nonlinear model as in Fig. 25.

$$\boldsymbol{f} = k_{pad}\, \boldsymbol{\Delta}_{pad}^{\boldsymbol{2}}$$

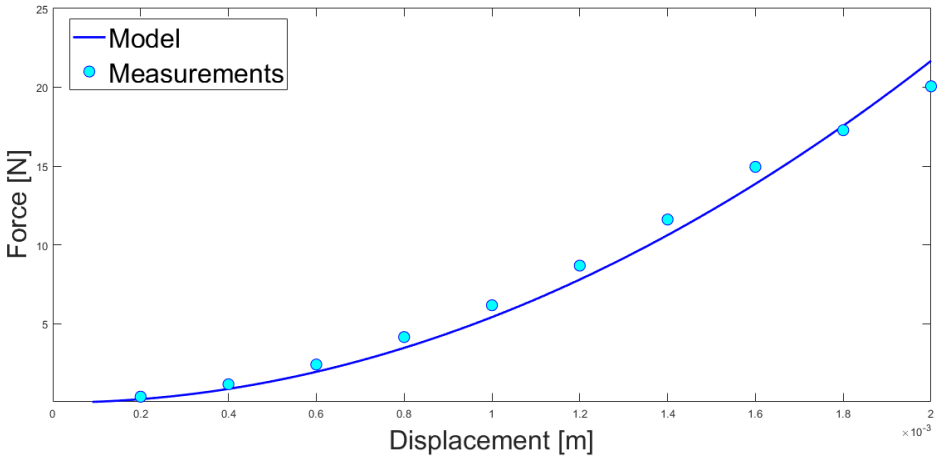

Figure 25: Sensor model

Once calculated the nonlinear stiffness of the pad, a similar procedure has been executed to calculate the stiffness of a deformable object. In particular, the object has been positioned between the fingers of the gripper and after the contact has been detected, through a position control approach, different small displacements $\boldsymbol{\Delta}$ have been imposed but in this case, since the object is deformable, the displacements $\boldsymbol{\Delta}$ have caused not only the pad deformation $\boldsymbol{\Delta}_{pad}$, but also the object deformation $\boldsymbol{\Delta}_{obj}$. Therefore, given the displacement, the corresponding

force has been read from the sensors and then this force has been used to obtain from the quadratic pad model, the pad deformation. At the end the $\boldsymbol{\Delta}_{\mathbf{obj}}$ has been trivially computed as

$$\boldsymbol{\Delta}_{\mathbf{obj}} = \boldsymbol{\Delta} - \boldsymbol{\Delta}_{pad}$$

Thus, the stiffness of the object has been calculated supposing a linear model, since small displacements have been applied:

$$\boldsymbol{f} = k_{obj}\boldsymbol{\Delta}_{obj}$$

Algorithm 1 Estimation of the stiffness

input:
$\mathbf{f} \leftarrow$ grasping force
$K_{pad} \leftarrow$ non linear stiffness of the pad
$K_{obj} \leftarrow$ stiffness of the object
$\boldsymbol{\Delta}_{\mathbf{pad}} \leftarrow$ pad deformation
$\boldsymbol{\Delta}_{\mathbf{tot}} \leftarrow$ total deformation
$\boldsymbol{\Delta}_{\mathbf{obj}} \leftarrow$ object deformation
$w \leftarrow$ initial distance between fingers
$\Delta_w \leftarrow$ gripper displacement before contact is detected
procedure CALCULATION OF THE STIFFNESS
    while *contactNotDetected* **do**
        $w - \Delta_w$
        for $i = 1 : numDisplacements$ **do**
            Apply Δ_{toti}
            Readf_i
            From the quadratic model of the pad $\Delta_{padi} = \sqrt{f_i/K_{pad}}$
            $\Delta_{obj_i} = \Delta_{toti} - \Delta_{padi}$
        end for
        return:$[K_{pad}] = LeastSquaresLinearFitting(\mathbf{f}, \boldsymbol{\Delta}_{\mathbf{obj}})$

Thus, in other words, in the first phase of this algorithm the task is to detect the contact between the object and the gripper. So, iteratively the gripper has been closed until the measurement of grasp force given by sensors has been higher than 0.10 N. Once the contact has been detected, the procedure for the stiffness calculation can start. The gripper is closed, iteratively, and each time the force has been read and then using the pad model, the K_{obj} has been calculated.

4.3 Friction coefficient estimation

Regardless of the specific continuous model chosen to represent the con-tact dynamics, several alternatives are available for representing the effect of frictional forces. Friction models provide only approximations of the outcome of a complex physical phenomenon. The simplest and more commonly used friction model is provided by the well known Coulomb's Law. Static friction (or dry, or traction) represents friction force between two sticking objects (i.e., in contact with no relative motion), while kinetic friction represents frictional force during sliding movement. Both kinetic and maximum static friction are proportional to the normal force and opposed to tangential movement. If $v_t$is the tangential velocity and μ_k and μ_s are the dimensionless kinetic and static friction coefficients, respectively, then the kinetic friction F_{fk} is described by

$$F_{fk} = -sgn(v_t)(\mu_k F_n),$$

while the maximum threshold for static friction is represented by

$$F_{fs} \leq \mu_s F_n.$$

To calculate the coefficient of friction, a gripper with its fingers slides along the surface of the object with constant velocity. In this configuration, measurements of normal and tangential force are taken.

Then, $\frac{F_t}{F_n}$ is plotted as function of time (Fig 26) and in the time interval from 10 to 12 seconds, the ratio $\frac{F_t}{F_n}$ increases as soon as the sensor pad starts deforming and the tangential force builds up. When the robot moves at constant speed between 12 and 27 seconds, F_t includes both the kinetic friction and the viscous friction, thus the data in this time interval are not considered for the friction

coefficient estimation. As soon as the robot stops at about 19 seconds, the tangential component decreases down to the static friction and thus the estimated friction coefficients is taken as the value of the ratio $\frac{F_t}{F_n}$ at t=30 s.

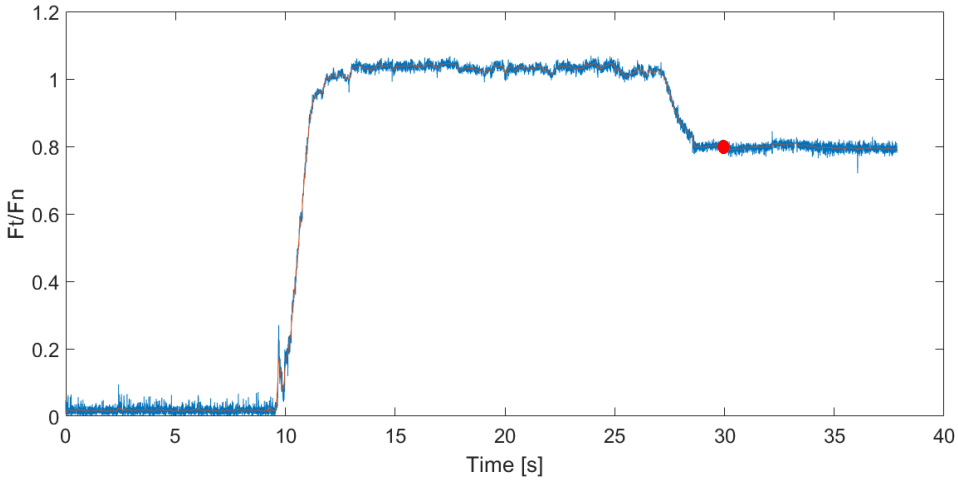

Figure 26: Friction estimation

Chapter 5 Classifier for object recognition

The final step is to create a Random Forest classifier able to recognize an object through its physical properties. After an introduction on the basic concepts of machine learning, the following chapter is focused on the explanation of the designed classifier.

5.1 Basic concepts of Machine Learning

Machine learning algorithms are a set of techniques that use computer programs to automatically extract models representing patterns from data and then evaluate those models.

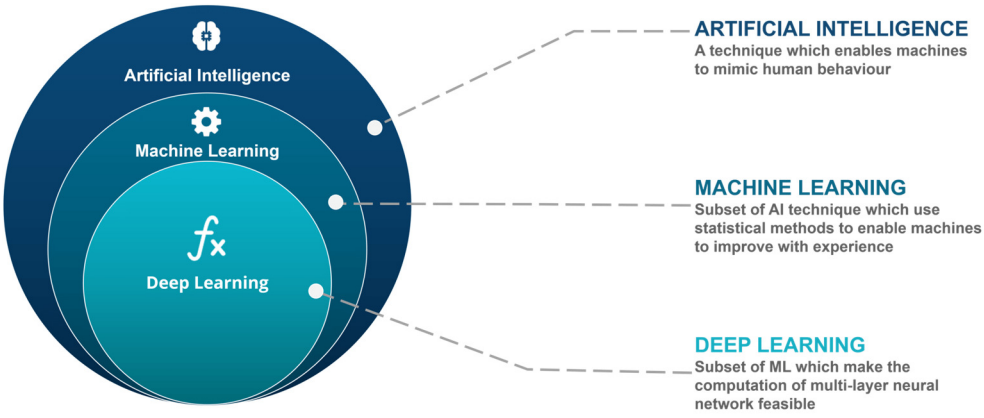

Figure 27: Artificial Intelligence vs Machine learning vs Deep Learning

Machine learning algorithms build a mathematical model of sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning tasks are classified into several broad categories. In supervised learning, the algorithm builds a mathematical model from a set of data that contains both the inputs and the desired outputs.

In unsupervised learning, the algorithm builds a mathematical model from a set of data which contains only inputs and no desired output labels. Unsupervised learning algorithms are used to find structure in the data, like grouping or clustering of data points. Unsupervised learning can discover patterns in the data and can group the inputs into categories.

Classification algorithms and regression algorithms are types of supervised learning. Classification is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing observations whose category membership is known.

An algorithm that implements classification, especially in a concrete implementation, is known as a classifier.

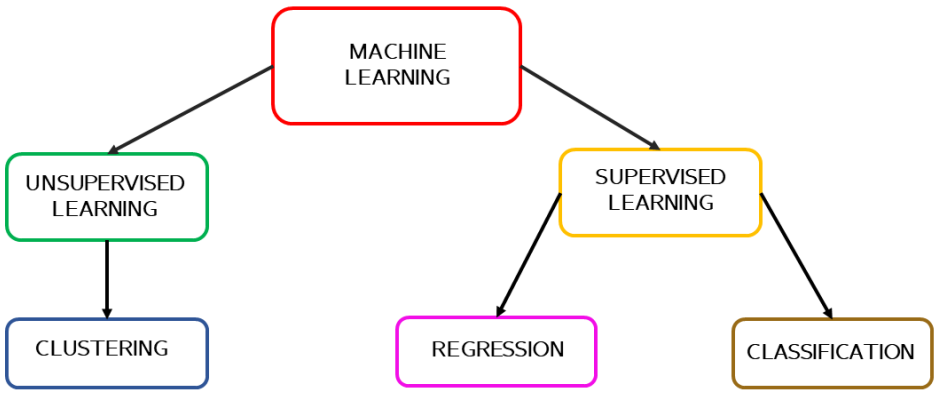

Figure 28: Categories of Machine learning algorithms

In a Machine Learning Algorithm one of the first phases is to create a training set, then the ML algorithm is trained on this data to create a model. This model, given new data, provides a prediction and then, comparing these results to the ground-truth could be possible to evaluate the model

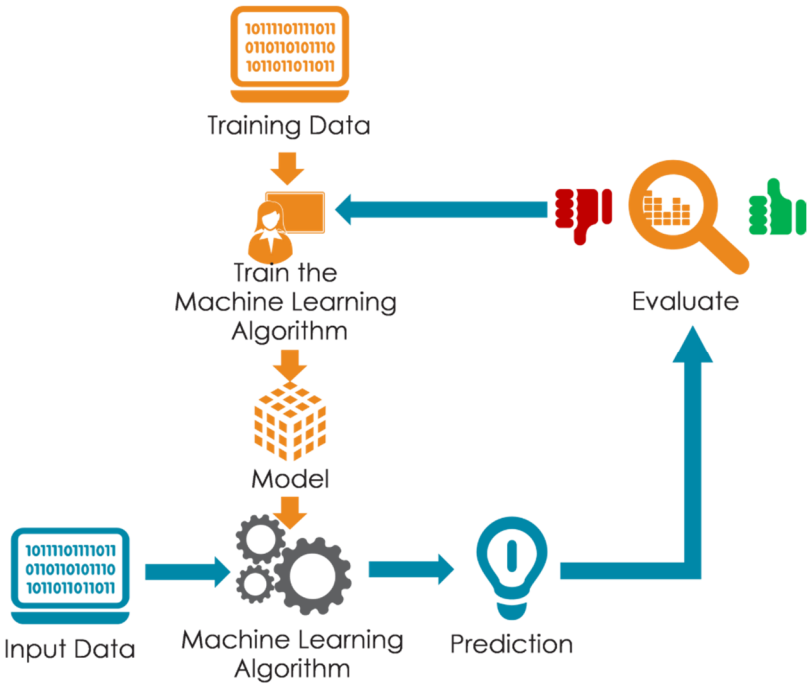

Figure 29: Machine learning algorithm

5.1.1 Decision tree learning

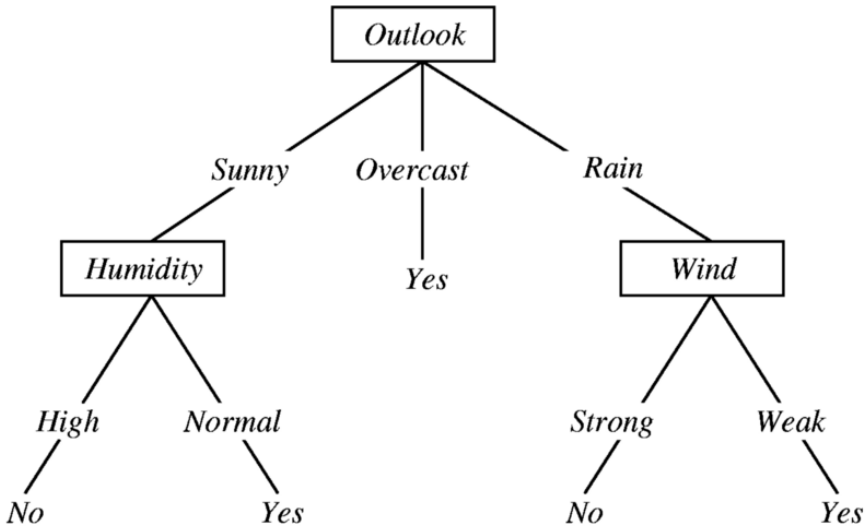

Figure 30:Example of decision tree stricture for rain forecasting

A decision tree consists of:

- **Nodes:** test for the value of a certain attribute

- **Edges**: correspond to the outcome of a test and connect to the next node or leaf
- **Leaves**: terminal nodes that predict the outcome

In Decision Tree Learning, a new example is classified by submitting it to a series of tests that determine the class label of the example. These tests are organized in a hierarchical structure called a decision tree.

5.1.1.1 How to build a decision tree

Decision Tree models are created using 2 steps: **Induction** and **Pruning**. Induction is the creation of the tree i.e. set all of the hierarchical decision boundaries based on data. Because of the nature of training decision trees, they can be prone to major overfitting. Pruning is the process of removing the unnecessary structure from a decision tree, effectively reducing the complexity to combat overfitting with the added bonus of making it even easier to interpret.

5.1.1.1.1 Induction

From a high level, decision tree induction goes through 4 main steps to build the tree:

1. Begin with your training dataset, which should have some feature variables and classification or regression output.
2. Determine the "best feature" in the dataset to split the data on;
3. Split the data into subsets that contain the possible values for this best feature. This splitting basically defines a node on the tree i.e., each node is a splitting point based on a certain feature from our data.
4. Recursively generate new tree nodes by using the subset of data created from step 3. We keep splitting until we reach a point where we have

optimised, by some measure, maximum accuracy while minimising the number of splits / nodes.

5.1.1.1.2Pruning

Because of the nature of training decision trees, they can be prone to major overfitting. Setting the correct value for minimum number of instances per node can be challenging. The key is that many of these splits will end up being redundant and unnecessary to increase the accuracy of our model.
Tree pruning is a technique that leverages this splitting redundancy to remove i.e. *prune* the unnecessary splits in our tree. From a high-level, pruning compresses part of the tree from strict and rigid decision boundaries into ones that are smoother and generalise better, effectively reducing the tree complexity. The complexity of a decision tree is defined as the number of splits in the tree.

5.1.1.1.3Advantages and Disadvantages

One of the most important advantages is that decision trees are **easy** to understand and interpret. At each node, it is possible to see exactly what decision our model is making. In practice it is easy to fully understand where accuracies and errors are coming from, what type of data the model would do well with, and how the output is influenced by the values of the features.
Moreover, they require **very little data preparation**. Many ML models may require heavy data pre-processing such as normalization and may require complex regularisation schemes. Decision trees on the other hand work quite well out of the box after tweaking a few of the parameters.
The **cost** of using the tree for inference is logarithmic in the number of data points used to train the tree. That is a huge plus since it means that having more data won't necessarily make a huge dent in our inference speed.

Instead, one of the principal disadvantages is that overfitting is quite common with decision trees simply due to the nature of their training. It is often recommended to perform some type of dimensionality reduction such as PCA so that the tree doesn't have to learn splits on so many features.

For similar reasons as the case of overfitting, decision trees are also vulnerable to becoming biased to the classes that have a majority in the dataset. It is always a good idea to do some kind of class balancing such as class weights, sampling, or a specialised loss function.

5.1.2 Random forest

Random forests or random decision forests are an ensemble of learning methods for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees habit of overfitting to their training set. Here are a few of the pro and cons of decision

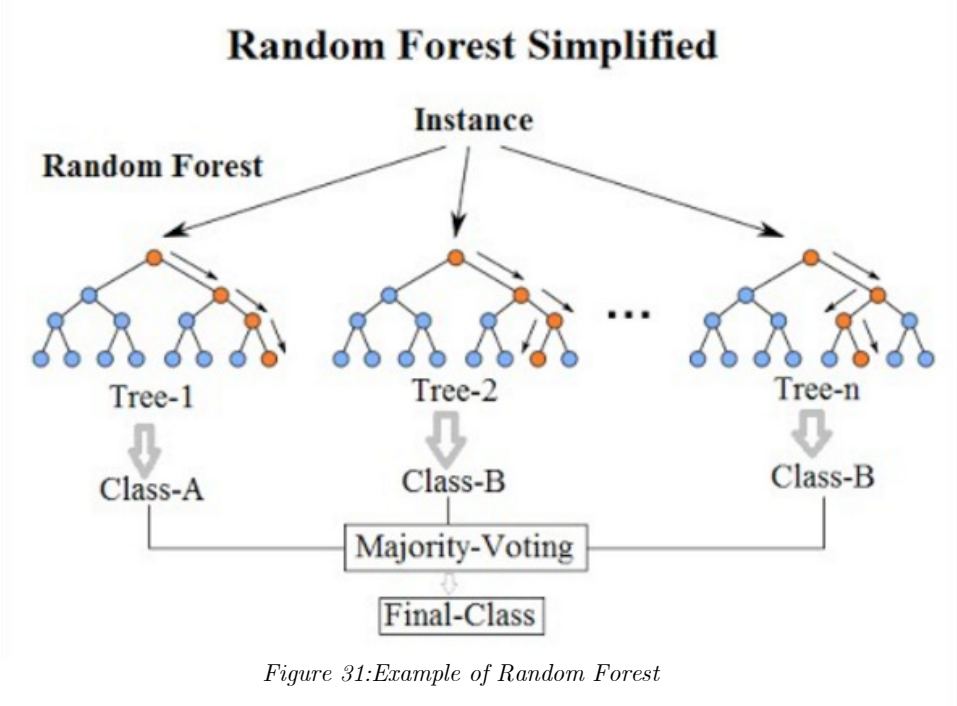

Figure 31:Example of Random Forest

trees that can help to decide on whether or not it is the right model for a certain problem.

Random Forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and the fact that it can be used for both classification and regression tasks. In this section it is explained how the random forest algorithm works and several other important aspects about it.

Random Forest is a **supervised** learning algorithm. Clearly, it creates a forest and makes it somehow random. The "forest" it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Random Forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, it is not necessary to combine a decision tree with a bagging classifier and can just easily use the classifier-class of Random Forest. Through Random Forest, it is also possible to deal with Regression tasks by using the Random Forest regressor.

Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random, by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

Another great quality of the random forest algorithm is that it is very easy to measure the relative importance of each feature on the prediction. Sklearn

provides a great tool for this, that measures a features importance by looking at how much the tree nodes, which use that feature, reduce impurity across all trees in the forest. It computes this score automatically for each feature after training and scales the results, so that the sum of all importance is equal to 1.
In a decision tree each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). A node that has no children is a leaf.
Through looking at the feature importance, it is possible to decide which features you may want to drop, because they don't contribute enough or nothing to the prediction process. This is important, because a general rule in machine learning is that the more features you have, the more likely your model will suffer from overfitting and vice versa.

5.1.2.1 Difference between Decision Trees and Random Forests

Random Forest is a collection of Decision Trees, but there are some differences.
If you input a training dataset with features and labels into a decision tree, it will formulate some set of rules, which will be used to make the predictions.
For example, if you want to predict whether a person will click on an online advertisement, you could collect the ad's the person clicked in the past and some features that describe his/her decision. If you put the features and labels into a decision tree, it will generate some rules. Then you can predict whether the advertisement will be clicked or not. In comparison, the Random Forest algorithm randomly selects observations and features to build several decision trees and then averages the results.
Another difference is that "deep" decision trees might suffer from overfitting. Random Forest prevents overfitting most of the time, by creating random subsets

of the features and building smaller trees using these subsets. Afterwards, it combines the subtrees. Note that this doesn't work every time and that it also makes the computation slower, depending on how many trees your random forest builds.

The Hyperparameters in Random Forest are either used to increase the predictive power of the model or to make the model faster.

5.1.2.2 Advantages and Disadvantages

An advantage of random forest is that it can be used for both regression and classification tasks and that it's easy to view the relative importance it assigns to the input features.

Random Forest is also considered as a very handy and easy to use algorithm, because its default hyperparameters often produce a good prediction result. The number of hyperparameters is also not that high and they are straightforward to understand.

One of the big problems in machine learning is overfitting, but most of the time this won't happen that easy to a random forest classifier. That's because if there are enough trees in the forest, the classifier won't overfit the model.

The main limitation of Random Forest is that a large number of trees can make the algorithm to slow and ineffective for real-time predictions. In general, these algorithms are fast to train, but quite slow to create predictions once they are trained. A more accurate prediction requires more trees, which results in a slower model. In most real-world applications the random forest algorithm is fast enough, but there can certainly be situations where run-time performance is important and other approaches would be preferred.

And of course, Random Forest is a predictive modelling tool and not a descriptive tool. That means, if you are looking for a description of the relationships in your data, other approaches would be preferred.

5.2 The proposed approach for object recognition

The last step was to a machine learning classifier able to identify an object based on its mass and stiffness previously estimated. The center of mass hasn't been used in order to avoid the ambiguity due to the grasping position that is not known in advance or, at least, can be very uncertain. The entire procedure has been realized using the algorithm for the physical properties estimation previously developed. Mass and stiffness estimations are published on a ROS topic.
Furthermore, a Random Forest classifier has been created using the *Scikit-learn* library in *Python*. Finally, a node rospy has been written to unify the procedure. It reads the estimations from the topic and uses the Random Forest model to do the prediction.

5.2.1 Dataset creation

To create a Random forest classifier, it has been necessary to build a training set. Data for training set have been collected through several experiments in which the mass and stiffness of the object are estimated, and then they have been structured in a matrix of dimension $N_{obj} \times 2$.
A small number of samples (7 objects), all of daily use, have been used only to test the procedure efficiency:

1. Finish
2. Denkmit 1
3. Denkmit 2
4. Box 1
5. Box2
6. Balea

7. Box 3

These objects are all used to train the model and tested to validate the procedure.

Figure 32:Objects in the training set

Due to the small number of samples, the training set has been built adding gaussian white noise with zero mean and variance equals to the $20\ \%$ of each value of mass and stiffness estimations. From each estimation, in this way, have been generated 10 training samples so the entire training set has dimensions $10N_{obj} \times 2$.

5.2.2 Training and testing model

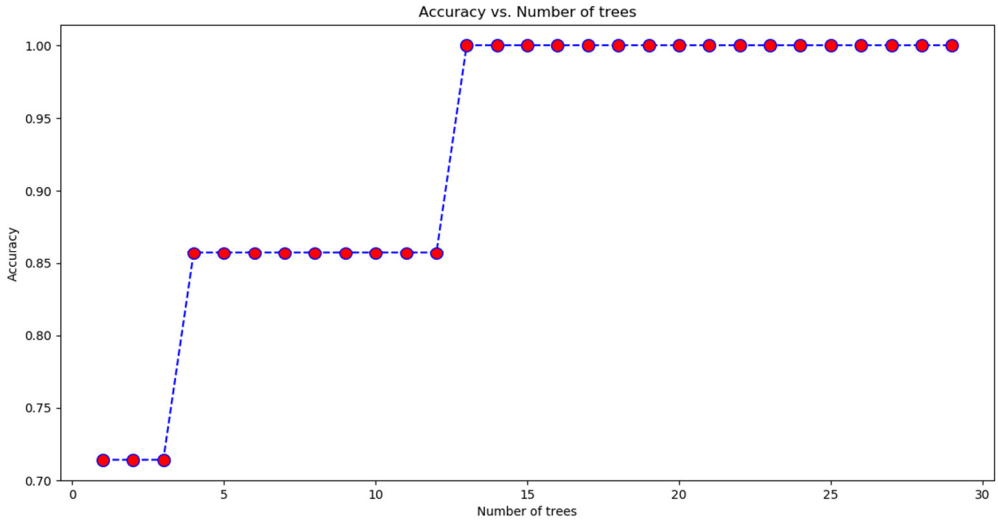

Figure 33:Accuracy of the Random forest

Once created the training set, the model has been created and trained, suitably tuning the hyperparameters. To do this, a Python library named *scikit learn* has been used. This library contains a function *randomForestClassifier* that allows to create a Random Forest.

Moreover, since this classification problem is quite simple, it is not necessary a complex tuning of the hyperparameters. Hence, only the number of trees has been tuned to improve the accuracy, choosing at the end a number of trees equals to 15 as showed in Fig. 33.

Using as ground truth the real mass and stiffness estimations, has been achieved in simulation an accuracy of 100%.

Chapter 6 Experimental results

This chapter shows the implementation details of the entire procedure in ROS and Python. Then, experimental results of the entire procedure, previously explained, are presented.

6.1 Implementation details

To realize the procedure in ROS, two nodes have been written. The first node is called *online_stiffness_com* and it reads from two different topics force measurements on each finger and the width between these fingers. These measurements are used to estimate mass and center of mass of the unknown object. The *online_stiffness_com* node subscribes also to another topic *wsg50/wrench* that, combining the forces on the two fingers, computes the grasping force used to estimate the stiffness of the object. The node *online_stiffness_com* publishes two topics, one called *pub_stiff* that contains the stiffness estimation and *pub_com*, that contains the center of mass estimation. The node *object_recognition_node*, highlighted in Fig. 34, is a *rospy* node and it has been created to use the Random Forest classifier with the online estimation data of mass and stiffness. The flow-chart in Figure 34 describes all the relations between the nodes and topics and it is extracted from the rqt_graph available in ROS.

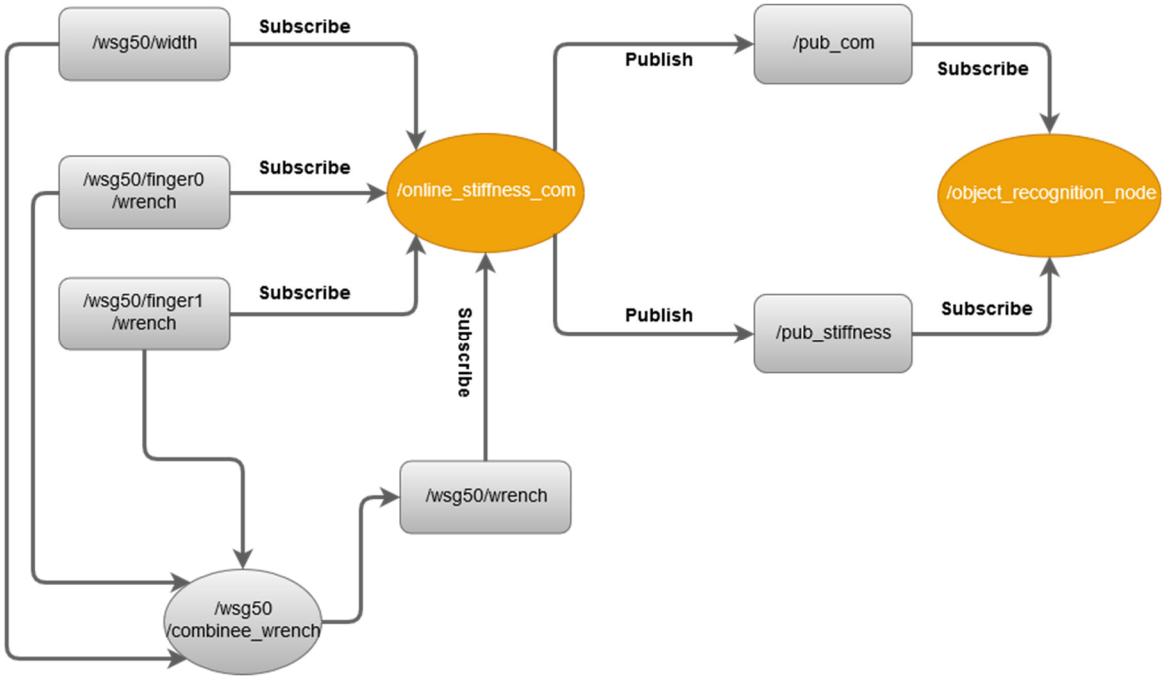

Figure 34:Flow-chart of the ROS implementation

6.2 Results

The first experiment has been the estimation of mass and center of mass of a resin object as in Fig 35.

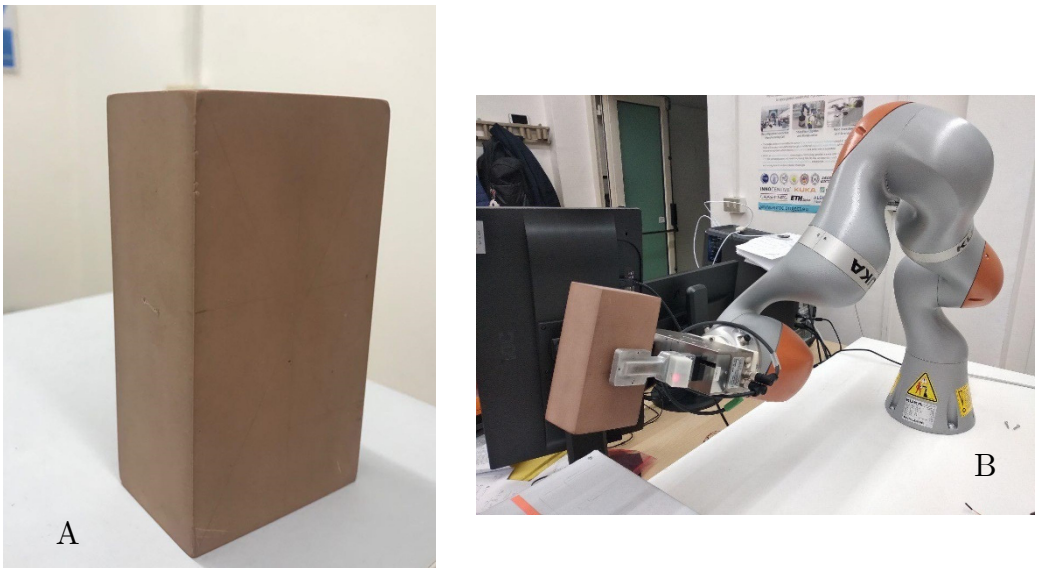

Figure 35:(A) Resin object (B) Center of mass estimation

Its stiffness has been supposed infinite, since the object is extremely rigid.

To calculate the mass and COM several rotations are computed along the x axis in sensor frame. In particular, rotations from 0° to 10° are done with steps of 2.5°. The mass estimated through the procedure is 0.325502 kg while the mass measured through the load cell is 0.348 kg. The estimated center of mass location is $x = 0.00814354\ y = 0.00337245\ z = 0.00835004$.

Since this object is rigid and not deformable, it has been used to estimate the stiffness of the deformable pad. The object has been positioned as in Fig 36, and small displacements have been applied Reading the force measurements from sensors, the parameter K_{pad} of the pad has model been calculated as $\boldsymbol{f} = k_{pad}\,\boldsymbol{\Delta}_{pad}^{\boldsymbol{2}}$. It is $K_{pad} = 5.32759 \times 10^3 \frac{N}{m^2}$

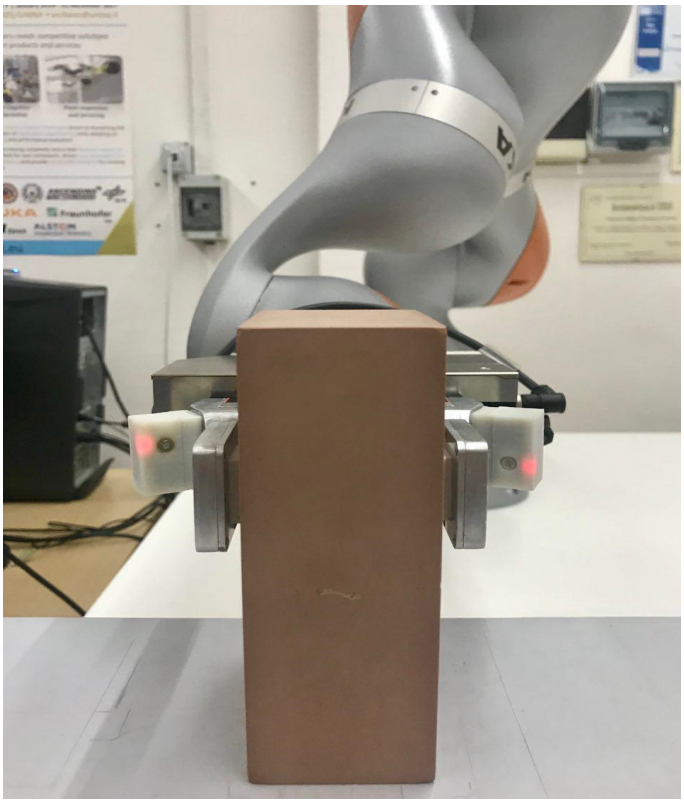

Figure 36: Nonlinear stiffness of the sensor pad estimation

To validate the stiffness estimation algorithm, another measurement system has been used: a load cell with a deformable pad and a micro-positioning stage, also with the same deformable pad as in Fig. 37.

Through the micro-positioning stage, small displacements are applied to the object and then the corresponding measurements given by the load cell are taken in order to calculate the stiffness of the object. This experiment shows the validity of the algorithm since the results obtained are almost the same with errors of 15%.

Similarly, the load cell has been used also to validate the mass estimation algorithm and error has been estimated of about 10-15%.

The following section shows the experimental results obtained for different objects. Initially, the procedure has been tested in MATLAB and then developed in ROS.

Considering an object in Fig. 38, the properties estimated are resumed in the Table 3 and Figure 38 shows the force/displacement samples and the corresponding linear model. The object stiffness model is linear, and the slope of the least square linear regression is the estimated stiffness.

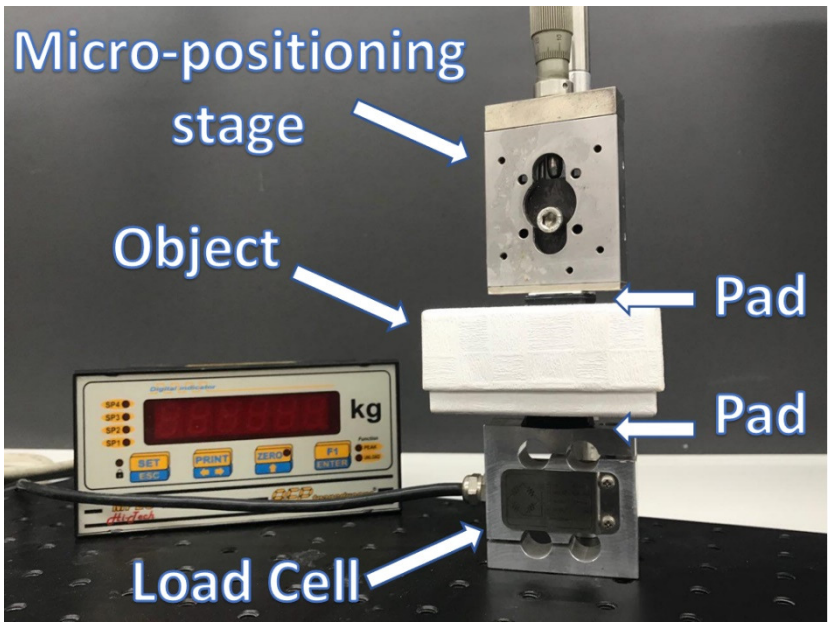

Figure 37: Load cell and micro-positioning stage

Figure 38: Denkmit MASCHINENPFLEGER

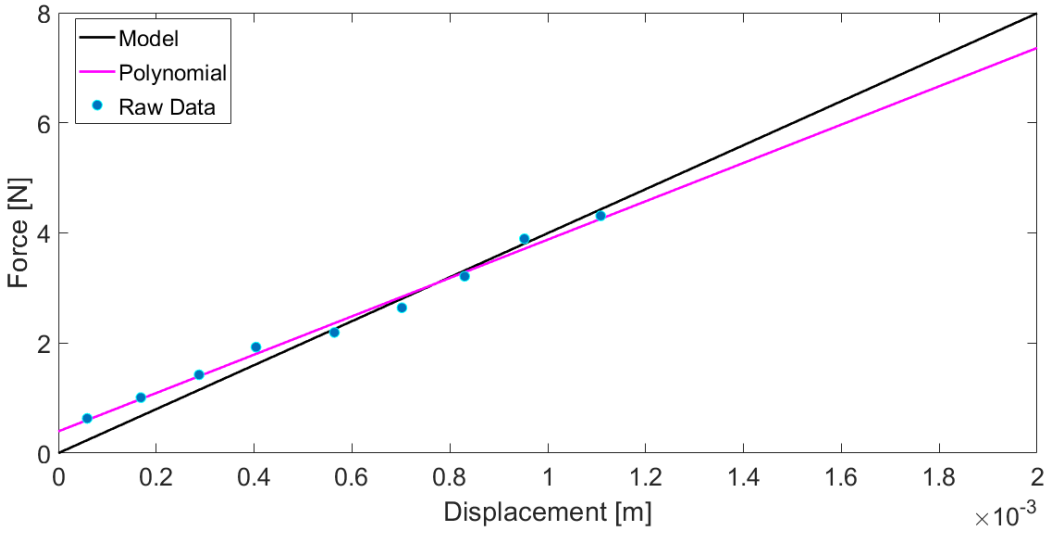

Figure 39: Stiffness for Denkmit MASCHINENPFLEGER

Mass $[kg]$	COM x$[m]$	COM y $[m]$	COM Z $[m]$	Stiffness $[N/m]$
0.280174	0.00642419	0.00345338	0.00158360	3418

Table 3: Estimated properties for Denkmit MASCHINENPFLEGER

Another experiment has been conducted on the object in Fig. 40. The properties estimated are shown in Table 4 and the Fig. 41 shows the estimated stiffness model.

Figure 40: Denkmit EDELSTAHL-REINIGER

Mass $[kg]$	COM x$[m]$	COM y $[m]$	COM Z $[m]$	Stiffness $\left[\frac{N}{m}\right]$
0.30296	0.00743418	0.00247369	0.00138679	3639

Table 4: Estimated properties for Denkmit EDELSTAHL-REINIGER

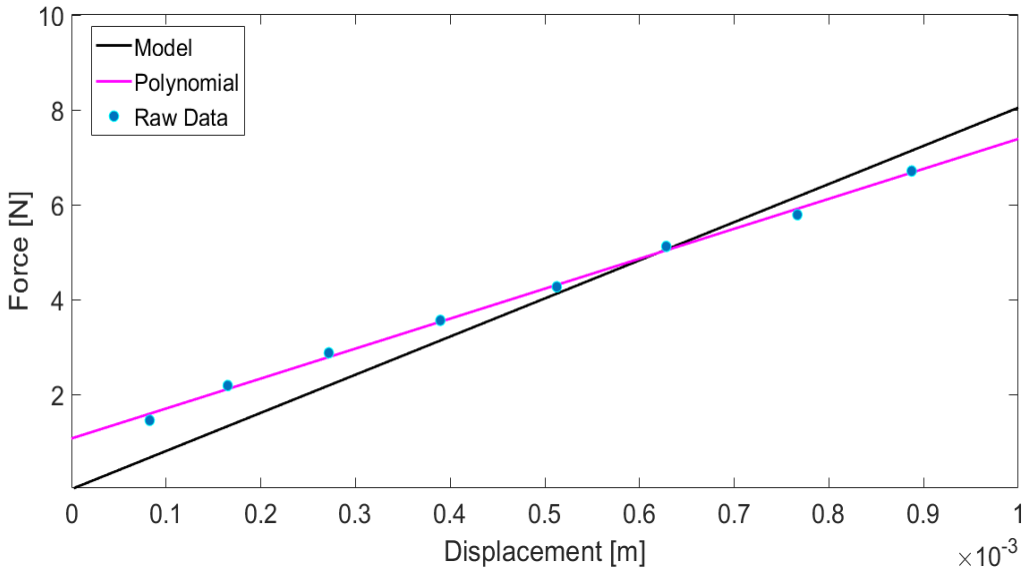

Figure 41: Stiffness for Denkmit EDELSTAHL-REINIGER

Once the procedure for physical properties estimation has been completed, the classifier has been built. The confusion matrix in Table 5 shows the results of the classification. Each object has been tested five times and, obviously, the error rate increases for objects with similar properties. This is the principal limitation of a tactile approach. A further improvement could be to also use the friction coefficient to recognize the objects or to adopt a visual approach.

ACTUAL OBJECT \ IDENTIFIED AS	Denkmit M	Box 1	Balea	Box 2	Denkmit ER	Box 3	Finish
Denkmit M	80						20
Box 1		80				20	
Balea			100				
Box 2				100			
Denkmit ER					100		
Box 3		20				80	
Finish	40						60

Table 5: Confusion Matrix of the success recognition rates (percentage)

Chapter 7 Conclusions

Comparing the results obtained through the proposed algorithms for the estimation, with those obtained from the load cell and micro-positioning stage, it is clear that this procedure is quite reliable except small measurement noise. Hence, tactile sensors provide important information for the estimation of physical properties. The entire procedure has been built without a vision system, so just using the tactile measurements it is possible not only to estimate the physical properties but also recognize an object, based on its properties.

Furthermore, the procedure does not require any hypothesis on the unknown object, but only information about the location of the novel object. For this reason, the presented approach is quite general and always applicable.

Using Machine Learning, it was also possible to classify different objects without any visual information.

One of the future improvements could be to add a vision system able to localize the object in the environment and also to give information about its shape.

Another sensor fusion approach could be to use vision system and applying a Convolutional Neural Network to understand how to grasp the object and then to control the grasping force based on the physical properties estimated.

Finally, increasing the number of samples in the dataset could be useful to not only recognize an object, but also to clusterize different objects not present in the training set, in other words to learn object classes with common physical properties then trying to apply the same manipulation strategies devised for already known objects of the same class.

References

[1] Johansson R. S., Westling G. - *Influences of cutaneous sensory input on the motor coordination during precision manipulation.* In C. von Euler, O. Franzen, U. Lindblom & D. Ottoson (Eds.), Somatosensory Mechanisms, pages 249-260.

[2] Martinez A., Fernàndez E. - *Learning ROS for Robotics Programming*, Birmingham, Packt Publishing, 2013

[3] Python Random forest classifier documentation URL - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[4] KUKA Roboter GmbH, LBR iiwa 7 R800 specification URL - https://www.kuka.com/en-de/products/robot-systems/industrial-robots/lbr-iiwa

[5] WSG 50 Network-enabled Universal Gripper URL - https://www.weiss-robotics.com/en/produkte/gripping-systems/performance-line-en/wsg-50-en/

[6] M.H. Lee, H.R. Nicholls - *Review article tactile sensing for mechatronics – a state of the art survey*, Mechatronics 9, 1999, pages 1–31.

[7] W. Goldsmith. Impact - *The Theory and Physical Behaviour of Colliding Solids.* Edward Arnold Publishers Ltd, London, 1960.

[8] G. Gilardi and I. Sharf - *Literature survey of contact dynamics modelling. Mechanism and Machine Theor*y, 37(10), 2002, pages 1213–1239.

[9] S. Dubowsky and F. Freudenstein - *Dynamic analysis of mechanical systems with clearances - part 1: formation of dynamical model.* Journal of Engineering for Industry, 1971.

[10] T. Lee and A. Wang - *On the dynamics of intermittent-motion mechanism, part I:* Dynamic model and response. Journal of Mechanism, Transmissions and Automation in Design, 1983.

[11] Bhat K., Seitz S., Popovic J, Khosla P. - *Computing the Physical Parameters of Rigid-body Motion from Video,* European Conference on Computer Vision, April 2002.

[12] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo - *Robotics: Modelling, Planning and Control*, Springer, 2010.

[13] G. De Maria, C. Natale, S. Pirozzi. - Force/tactile sensor for robotic applications, *Sensors and Actuators A: Physical 175*, 2012, pages 60–72.

[14] Mohsen Kaboli, Armando De La Rosa, Rich Walker, Gordon Cheng - *In-Hand Object Recognition via Texture Properties with Robotic Hands, Artificial Skin, and Novel Tactile Descriptors,* IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), Nov. 2015.

[15] Danfei Xu, Gerald E. Loeb, Jeremy A. Fishel - *Tactile identification of objects using Bayesian exploration,* IEEE International Conference on Robotics and Automation, 2013.

[16] Christopher M Bishop. - *Pattern recognition and machine learning*, Springer, 2006.

[17] K. Hunt and F. Crossley. - *Coefficient of restitution interpreted as damping in vibroimpac,* Journal of Applied Mechanics, 42:440–445, 1975.

[18] Marco Costanzo , Giuseppe De Maria, Ciro Natale, Salvatore Pirozzi - *Design and Calibration of a Force/Tactile Sensor for Dexterous Manipulation* In Sensors 2019, 2019.