UNIVERSITÀ DEGLI STUDI DI VERONA

Dipartimento di Informatica

Master's Degree in

Computer Engineering for Robotics and Smart Industry

Master's Thesis

DEVELOPMENT AND CHARACTERIZATION OF A 3D PRINTED HEMISPHERICAL SOFT TACTILE SENSOR FOR AGRICULTURAL APPLICATIONS

Supervisor:

Prof. RICCARDO MURADORE

Co-supervisor:

FRANCESCO VISENTIN, PhD

Candidate:

FABIO CASTELLINI

Student ID:

VR464639

ACADEMIC YEAR 2021-2022

List of Figures

List of Tables

Preface

In this space, I'd like to thank all the people that supported my university studies. Starting from Professors Riccardo Muradore and Francesco Visentin, they both provided valuable feedback as supervisors of this Master's Thesis. They kindly shared their knowledge and expertise contributing in a significant way to this work. Particularly, Francesco dealt with the design and 3D printing of the characterized sensor and of the cases to hold it in place. I'd like to also thank Post-Doc Researchers Giacomo De Rossi and Nicola Piccinelli that helped us with the robot's motion planning during the final evaluation phase. Last but not least, a thank goes to all my friends and family that supported my journey from all perspectives. Particularly I'd like to show gratitude to my parents Mariangela and Livio, my girlfriend Lucia, my brother Matteo and my closest friends.

Abstract

Soft robotics, and particularly soft gripping technology, faces many challenges. Due to several strict requirements such as small dimensions, low cost and efficient manufacturing process, accurate sensing capabilities, the development of soft grippers is still an open research problem. In this work a hemispherical deformable and cheap to manufacture tactile sensor is proposed and characterized. The device is 3D printed using a stereolithography (SLA) 3D printer and is made of a semi-transparent elastic polymer resin that is properly cured afterwards. The overall aim is to sense normal and tangential forces applied to the gripper. The gripper is designed and thought for agricultural applications such as grasping delicate fruits and vegetables.

Contents

Chapter 1

Introduction

In this chapter the Master's Thesis' work is motivated. Agriculture 4.0, industrial automation and soft robotics can help to transition towards a more sustainable and efficient harvesting process. At the end of Chapter 1 will be briefly explained how the rest of the work is structured.

1.1 Problem statement

1.1.1 How to cope with the increasing food demand

As stated by several papers [7, 47, 65, 30] and well respected institutions, agriculture is required to significantly grow its productivity to keep up with the rising global food demand. The United Nations Food and Agricultural Organization (FAO) foresees that "food and feed production will need to increase by 70% by 2050 in order to meet the world's food needs" [65]. Making harder to accomplish such a result is the shortage of workers in this field, due to the time consuming and labour intensive activities they're asked to stand up to, while being often exploited and underpaid [11, 8]. The main reasons of the declining trend in agricultural interest are: high land, real estate, machinery and

agrotechnology prices; unequal work-life balance; lack of government incentives and, in most cases, poor working conditions. Also, the agricultural industry is having troubles competing with corporate jobs that offer higher pay and smart-working options. All of this translates in fewer young farmers coming in to fill the shoes of retired ones. In addition, the global COVID-19 pandemic has "increased the need of industrial automation for relieving workforce challenges and increasing operational and food safety in factory environments" [30].

1.1.2 Agriculture 4.0 and the role of soft robotics

It's known that from roman and greek times, if not even before, humans are trying to automate as much as possible laborious and repetitive tasks, making use of the available technology. As an example, romans used a so called "Gallo-Roman harvester" shown in Figure 1.1 to speed up the grain harvesting process.

Figure 1.1: *On the left a sculpture depicting an ancient roman harvester called Gallic Vallus; on the right a modern combine harvester designed for grain, potatoes, carrots, beets.*

Nowadays automation plays a big role in the farming industry minimizing waste of products, time and optimizing the crop production cycle. An increasing number of companies are working on robotics innovation to develop drones, autonomous tractors, robotic harvesters, automatic watering, and seeding robots [30]. The main goal is to address or at least help with the previously mentioned issues that affect the agriculture sector. ***Agriculture***

4.0, also called *precision agriculture* refers to the use of Internet of Things (IoT), big data, Artificial Intelligence (AI) and robotics to make the entire production chain more efficient. Technological innovation is exploited to collect, transmit and precisely analyze data from the field. Data gathered from sensors is then elaborated with the aim of supporting farmers in the decision-making process related to their activities. The ultimate goals are "increasing economic, environmental, and social sustainability - as well as profitability - of agricultural processes" [2].

The main benefits of Industry 4.0 in the context of agriculture are:

- avoiding unnecessary waste (e.g. computing the exact water requirements of the crop);
- minimizing costs by planning and predicting all stages of cultivation, from land preparation and sowing to harvesting;
- improving the traceability of the supply chain thus the food quality in a sustainable manner.

Although more and more companies are now using high-tech devices and sensors to ease the farmers' work, making them able to concentrate on higher level tasks, there's still plenty to be discovered in this field. In this regard, a relatively new research branch is focusing on *soft robotics* and in particular *soft grippers*. ***Soft robotics*** is a subfield of robotics concerning the design, control, and fabrication of robots composed of compliant materials, instead of rigid links. Figure 1.2 shows two examples of soft grippers designed by the U.S. company Soft Robotics [66]. The depicted *mGrip* modular gripping system is fully configurable and is thought for safely and efficiently picking and packing delicate products such as food.

Figure 1.2: *Soft Robotics' mGrip: a modular gripping system that enables reliable, high-speed picking of traditionally hard-to-grasp single items.*

In the last decades the ongoing trend in robotics points towards *collaborative robots* that can operate outside of a cage, interacting with humans and the surrounding environment. Soft robots, not only aim at being safer for humans but may solve many open issues in robotics. In fact, being ***compliant***, they're flexible, harder to break or damage, adaptable to unstructured and dynamic environments. They can meet hygiene and strict manipulation requirements while operating with delicate or fragile products, making them desirable in the agriculture and food industries. Also, in a pick-and-place task they don't require an in-depth characterization of the object to handle, ensuring good performance even without any force feedback. In general, soft robotic grippers allow a simpler control architecture than traditional rigid robots. In fact, most soft grippers are ***underactuated***, meaning that the control inputs are less than the achievable degrees of freedom. By replacing the intricate rigid body joint mechanics with simple compliant mechanisms, the number of parts required is significantly reduced, leading to lower costs for maintenance and assembly.

1.1.3 Do all have the means to face a transition?

Global agriculture has been constrained by many factors, such as socioeconomic issues, climate change, desertification and diminished crop yields, attributed to the decrease of vital nutrients in agricultural lands [39]. Despite the compelling evidence supporting agricultural automation, critics have argued that developing countries, including those in Sub-Sahara Africa, are less equipped for the transition to Agriculture 4.0 [39]. In fact, it's not clear how those countries would be supplied with new technology, especially soft robots. As mentioned, Europe is experiencing an intense labour shortage in this field, thus soft robotics and seeding/planting equipments would be a valid solution. On the other hand, developing nations in Africa have a critical mass of unemployed youth [39] but this is not enough to transition to a more automated harvesting process. According to Food and Agriculture Organization of the United Nations (FAO) [73], "digital innovations in mechanization technologies can make agriculture more attractive to rural youth, especially in developing countries". Such a high unemployed youth offers the opportunity to create new and more attractive jobs to leave behind rudimentary hand tools. According to FAO, governments should be provided with the necessary technical support to transform agriculture in a sustainable way; the initiative is aligned with the Framework for Sustainable Agricultural Mechanization in Africa (SAMA) and Asia (SAM).

Over the years, agricultural mechanization has evolved from basic hand tools and animal-driven implements to engine-powered equipments, but not uniformly all over the world. In fact, manual tools and animal power are still commonly used in developing countries, negatively affecting the livelihoods of small-scale farmers and their productivity [73].

Economic incentives to help with the transition to Agriculture 4.0 are tak-

ing place also in Italy for several years. As reported by a recent article on this topic [51], it's important not only to offer incentives but also to communicate them effectively in order to reach most of the farmers and small-medium enterprises. Supporting this statement, 87% of big companies know and use some of the current concessions, but only 59% of the smaller companies know about them. To be able to adopt new technology such as ***agrobots*** (robots for agriculture), a certain level of understanding of the robotic device is required: a good farmer is not necessarily expert in digital technologies and automation. Some of the farmers' reluctance is due to: a not so straightforward process, lack of continuous support and training, and absence of external incentives (e.g. policies or market prices) [39]. Also, to achieve the best results, the farm system and farmers' workflow itself must adapt to the robots. For instance, spacing between crops and crop structures needs to match the operational parameters of the agrobot as it moves among the cultivated crops. Of course, purchase price of the device has to be taken into account as it could be unsustainable for medium-small farms. This can become less of a problem for large commercially oriented farms where high labour costs during harvest season can be attenuated through automation. FAO presents the need to find profitable business models where the farmer does not necessarily own the robot but can benefit from the technology. Two possible solutions, already in place in many farming systems, are service provision and cooperative ownership. To conclude, a recent study [73] summarizes the main benefits for Agriculture 4.0 in developing countries and particularly in Vietnam. It reports that at this stage "the agriculture of Vietnam is still dominated by individual households with small scale production and low skill techniques. However, there is a growing trend of private investment in agriculture, which apply modern techniques, from both foreign and domestic investors. More interestingly, there are

companies now specializing in technical solutions for agriculture".

1.2 Thesis structure

This Master's Thesis focuses on the research, development and characterization of a marker-based hemispherical soft gripper capable of sensing forces when in contact with the external environment.

The work is structured as follows:

- ***Chapter 2*** presents an overview about soft robotics and soft grippers as a possible solution to the problems the agricultural industry faces nowadays. The main state-of-the-art soft gripping approaches are discussed and an overview of the open issues in this field is reported;

- ***Chapter 3*** addresses the prototyping and manufacturing phase explaining the main steps to develop the sensing device. The exploited experimental setups are briefly presented, while the final part of the Chapter explains the implemented offline pipeline to create the dataset. Also, some approaches for 3D shape reconstruction inspired from state-of-the-art similar or somehow related work, that turned out to be not very accurate, are nonetheless shown;

- ***Chapter 4*** is dedicated to the experimental results. The developed offline and online pipelines' outputs are shown and commented. In this Chapter the mainly faced problems and solutions are discussed. Moreover, all the exploited algorithms, Machine Learning and Deep Learning estimation approaches are cited and briefly explained. Finally, using the developed setup mounted on the Franka Emika Panda robot a simple picking task is attempted and the qualitative results are reported.

Chapter 2

Overview of soft robotics in agriculture

In this Chapter the soft-gripping technology is introduced and presented as a possible solution to many problems the agricultural industry is currently facing. An overview about the available technologies on the market is presented and the publications related to the case study are briefly summarized.

2.1 Soft grippers

2.1.1 A brief introduction to robotic grippers

Grasping and manipulation are fundamental functions that require interaction with the surrounding environment. ***Grasping*** can be described as the "ability to pick up and hold an object against external disturbances" [63], while ***manipulation*** is the ability to exert forces on an object, causing its rotation and displacement with respect to the manipulator's reference frame. A ***robotic gripper*** is a robotic end-effector that can be mounted on a robotic

arm, acting like a tool or, specifically, a hand for grasping, picking and placing objects. Traditionally, robotic grippers are made of rigid joints and links and they're actuated through electric motors inside the structure. In alternative they can be actuated through cables or tendons, as shown in Figure 2.1. Gripper designs range from two-fingered grippers to anthropomorphic hands with articulated fingers and palm (Figure 2.1) [63]. In fact, their design is often inspired to human or animal features to achieve *dexterity* (the ability to perform non-trivial action quickly and skilfully with the hands) and *compliance* (flexibility and elastic deformability). In Figure 2.2 three grippers from the ROBOTIQ company [56] are shown. They're all designed to be mounted on collaborative robots for precision assembly tasks. Moreover, in Figure 2.3 there are vacuum grippers produced by different manufacturers, such as Universal Robots [72], OnRobot [49] and Joulin [35]. The latter proposes "The Foam Gripper" which is characterized by a foam suction cup that is insensitive to porosities.

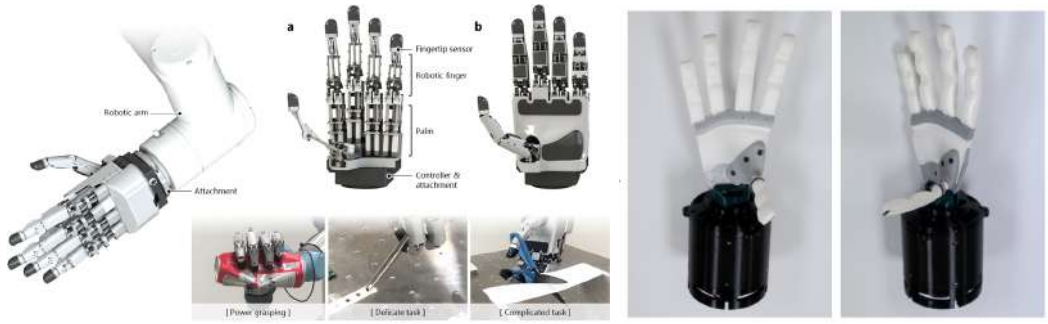

Figure 2.1: *On the left an "integrated linkage-driven dexterous anthropomorphic robotic hand" [38] while on the right a soft anthropomorphic hand [7].*

One of the main challenges of rigid anthropomorphic grippers is handling soft and deformable objects like fruits or vegetables that require an additional care during manipulation. The so called ***soft grippers*** are trying to address some of those problems avoiding the high mechanical and control complexity

Figure 2.2: *From left to right: ROBOTIQ Hand-E Adaptive Gripper, 2F-85 Gripper and 3-Finger Adaptive Robot Gripper.*

Figure 2.3: *From left to right: Universal Robots ZXP7*01 Vacuum Unit, OnRobot VG10 Vacuum gripper and Joulin Foam Gripper.*

of classical grippers required to achieve *"software compliance"*. In fact, they allow a simpler controllability and adaptability to dynamic environments, while being robust, durable, versatile and ***inherently compliant*** thanks to the soft materials. ***Underactuation***, that denotes a lower number of actuators than degrees of freedom, is fundamental to have a simpler controllability: as an example, human fingers can be seen as the composition of one tendon and three links, meaning two degrees of freedom given a single control input.

Also, robotic grippers can be equipped with sensors to estimate position and velocity of the gripper elements (e.g. with Hall-effect sensors, encoders, torque sensors) and with sensors to retrieve information about the in-contact

objects or applied external forces (e.g. pressure, force, torque sensors, optical sensors, resistive, conductive and electromagnetic sensors).

2.1.2 Soft gripping technologies for agriculture

According to [63], soft gripping technologies can be classified in three macro categories, even though they're not exclusive and many devices make use of combinations of two technology classes to reach higher performance:

- ***Actuation:*** passive structure with external motors, fluidic elastomer actuators (FEAs), electroactive polymers, shape memory alloys (SMAs);
- ***Controlled stiffness:*** granular jamming, low melting point alloys (LMPAs), electro-rheological (ER) and magneto-rheological (MR) fluids, shape memory polymers (SMPs);
- ***Controlled adhesion:*** electro-adhesion, geckoadhesion (dry adhesion).

Gripping by ***actuation*** consists of bending gripper fingers or elements around the object, as we do with our fingers when picking up an egg or a glass of water. The bending shape can be actively controlled, otherwise contact with the object can be exploited to induce deformation [63].

Gripping using ***controlled stiffness*** exploits the large change in rigidity of some materials to hold the target object. An actuator is needed to envelop the object with part of the gripper and while it's soft the applied force can be very low, allowing the manipulation of delicate objects. Such grippers are fast and allow tuning of the stiffness to a desired level but its range can be limiting.

Gripping using ***controlled adhesion***, similarly to variable stiffness, requires an actuation method to partially envelop the object. Controlled adhesion relies on surface forces at the interface between gripper and object.

This operating principle is a major advantage when manipulating very delicate objects, as it avoids the high compression forces required in gripping by actuation. Also, it's an ideal method for flat objects or objects that can't be enveloped but requires clean, relatively smooth and dry surfaces.

As described in [47] additional criteria to choose the gripper's technology could be: target object size, gripper size, lifting capabilities and ratio between gripper's and object's masses. Also, power consumption, controllability ease (open loop), scalability, modularity, adaptability to various target objects. Response time, surface-related requirements, bio-compatibility, robustness in unstructured environments, compliance, lifetime can all affect the efficiency of the agriculture task. Figure 2.4 shows a brief timeline of milestones in the development of soft gripper technologies according to [63], starting from the late 70s' tendon driven grippers to 2017's FEAs using thermo-reversible Diels-Alder polymers.

According to the mentioned requirements, the most suitable and commonly used technologies are:

- ***Granular jamming:*** reacting to external variables such as chemical concentration, humidity, or light, they achieve good lifting ratio, response time and ability to lift medium-size fruits;
- ***Passive structures with external motors and FEA actuators:*** ideal for fruit harvesting grippers, high lifting ratio, wide object size range, good response time, ability to grasp any object.

Soft components typically used in the grippers' actuators include urethanes, hydrogels (invisible in aqueous environments), hydraulic fluids and polymers, such as ***silicone elastomers*** [47, 63]. Actuators based on silicone elastomers have attracted strong interest due to their low cost and ease of manufacture;

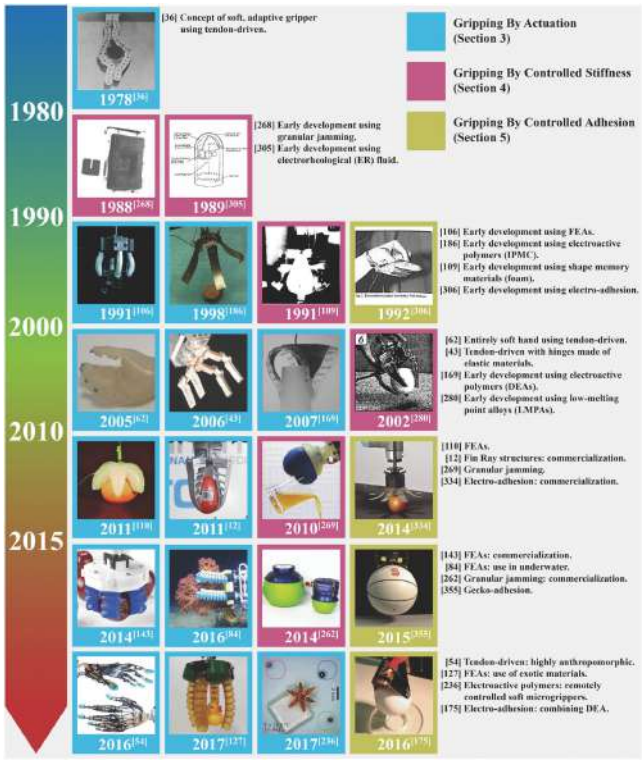

Figure 2.4: *A brief timeline of milestones in the development of soft gripper technologies as presented in [63].*

they do not require the use of complex machinery or skilled labour. In addition, these compliant materials are also advantageous when considering the safety of interaction with biological products, making them appropriate candidates for agricultural applications.

2.1.3 State-of-the-art of soft grippers

In the field of soft robotics, there's still plenty of room for improvement of soft actuators designed for picking, placing and harvesting fruits and vegetables. Handling this type of products requires precise control of the gripper to successfully follow the *picking pattern*'s movements without causing any damage to the fruit. In literature [47] the main capabilities of an ideal picking robot

would be:

- 3D localization of fruits inside the plant;
- path and trajectory planning;
- application of the suited fruit detachment method;
- adequate storage of the fruit.

All of this should be carried out with the aim of increasing the harvest ratio between robotic and manual picking, increasing the harvested fruit quality, being economically justified. End-effectors are required to appropriately handle fruits to preserve their quality, meaning their value on the market.

Soft grippers are considered to be one of the best solutions for harvesting crops, thanks to their adaptability and delicacy when grasping and manipulating the target products. By using materials with a module of elasticity similar to biological materials, soft grippers ensure safe interaction with humans and the working environment. Table 2.1 summarizes the most recent proposals for food soft grippers [47].

Regarding commercially available soft grippers, in 2015 the company Soft Robotics [66] introduced ***mGrip***: a pneumatically powered gripper made of soft elastomers. As shown in Figure 2.5, it consists of a network of parallel air chambers embedded in the elastomer thanks to which a single pneumatic source can control the device. So, compliance is achieved without hard linkages, additional sensors or a vision system. This modular gripper can be set up as two opposing fingers or multiple fingers placed in a circular pattern.

Festo [19] is an industrial automation company that produces collaborative robots with soft grippers attached (examples in Figure 2.6 and 2.7). ***BionicSoftArm*** is a robotic arm that in its largest version has seven pneumatic

Soft technology and year	Grasped object	Object size or weight	Gripper type	Controllability
FEAs (2020)	Lettuce	50x250mm	2 pneumatic actuators and a blade (8kg, 450x450x300mm)	Close-loop with force feedback
FEAs (2010)	Apple, tomato, strawberry, carrot	69mm; 5-150g	Magnetorheological gripper (finger size: 82x16x15mm)	PID
FEAs (2017)	Cupcake	75.2g	Soft fingers (finger length: 97mm)	Open-loop
FEAs (2020)	Orange	1kg	Soft fingers (finger size: 95x20x18mm)	Open-loop
FEAs (2020)	Tomato, kiwifruit	46-76mm	4 soft chambers in circular shell (diameter: 46mm; height: 30mm)	Open-loop
Tendon-driven (2020)	Tomato	500g	3 soft finger design	Pre-programmed motors' rotation
FEA-tendon driven (2019)	Banana, apple, grapes	2.7kg	3 soft finger design with a suction cup (390g)	Teleoperation
Topology optimized soft actuators	Apple, grapefruit, guava, orange, kiwi	1.4kg	2 compliant fingers	Open-loop

Table 2.1: *A brief summary of Table 4 reported in "Soft Grippers for Automatic Crop Harvesting: A Review" [47] as a literature review of food soft grippers.*

actuators and as much degrees of freedom. It can be equipped with various adaptive grippers for pick and place tasks such as ***FlexShapeGripper***, inspired to the behaviours of a chameleon; ***MultiChoiceGripper***, an adaptive, flexible handling system inspired to the opposable thumb; ***TentacleGripper***, an octopus-inspired gripper which wraps around objects like an octopus's arm and then uses vacuum suction cups to hold it firmly in place.

Hank soft gripper from Cambridge Consultants [10] attempts to emulate the human hands' four fingers and opposable thumb that allow a sophisticated sense of touch and slip using sensors embedded in its individual pneumatic fingers (Figure 2.8 shows the Hank soft gripper). These sensors are embedded during the molding process inside its hollow silicone fingers, that are actuated pneumatically. Based on the deformation of the fingers, the applied force is measured and the force feedback closure is provided.

Finally, the increasing pressure for environmentally friendly technologies

Figure 2.5: *Soft Robotics' mGrip modular soft gripper.*

Figure 2.6: *On the left Festo FlexShapeGripper and on the right Festo TentacleGripper.*

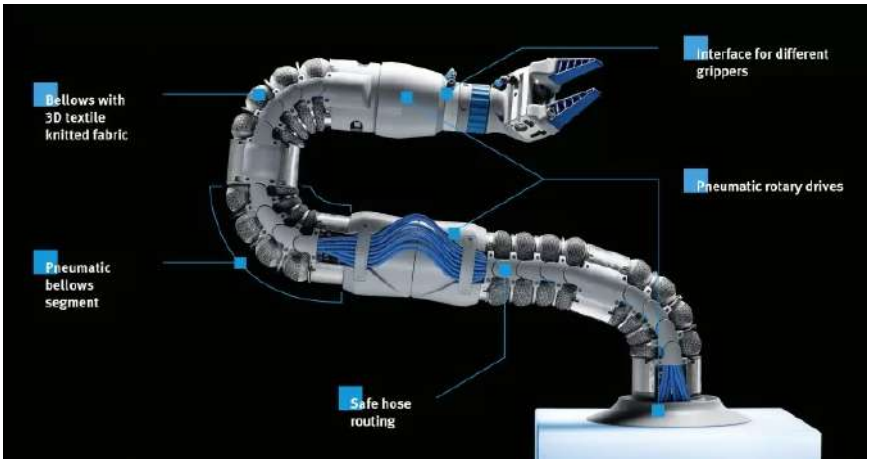

Figure 2.7: *Festo BionicSoftArm equipped with a flexible MultiChoiceGripper.*

has induced researchers to explore soft grippers made of biodegradable, and even edible, materials [63].

Figure 2.8: *Cambridge Consultants' Hank soft gripper.*

2.1.4 Soft grippers' materials and manufacturing process

According to [63], soft grippers are made of urethanes, hydrogels, braided fabrics, hydraulic fluidics and polymers, such as silicone elastomers, which became very desirable thanks to their low cost and simplicity to manufacture. The most commonly used soft materials that reside in the silicone elastomers category are: Dragon Skin, Ecoflex, polydimethylsiloxane (PDMS), Elastosil M4601 and Smooth-Sil. Other polymers are Agilus30/VeroClear, ultra-high molecular weight polyethylene, electrostatic discharge (ESD) plastic sheet, thermoplastic elastomers (TPEs) and thermoplastic polyurethane (TPU). An important aspect for the suitability of soft grippers in the agricultural sector, as suggested by [63], is that the materials they're made of mustn't contaminate the food. This topic should be investigated more, to understand if soft grippers' degradation may leave particles on the manipulated crops. Table 2.2 summarizes the main advantages of the mentioned materials.

Regarding the **manufacturing process**, several approaches can be mentioned:

- ***Moulding***: fused material is placed inside a (typically 3D printed) mold

Soft material	Main specifications	Shore hardness
Dragon Skin, Ecoflex, Smooth-Sil	Versatile, easy to use and handle, low cost	10 to 50 Shore A
Elastosil M4601	Highly resistant to bending and elongation; low viscosity in its uncured form; easy to mold	Approximately 28 Shore A
PDMS	High elasticity; it is a thermosetting polymer, obtained by irreversibly hardening (curing) a soft solid or viscous liquid prepolymer (resin). Precisely mathematically modellable through Finite Element Method (FEM) analysis. The variation in its hardness through several mixing ratios has been extensively studied in the literature.	Approximately 50 Shore A
TPU and TPE	Can be 3D printed. Also, TPU-95 is very durable suitable for agricultural environments, where harmful collisions with objects are frequent.	85 Shore A

Table 2.2: *A summary of the main materials' characteristics used in soft grippers mentioned in [47].*

and removed after hardening. It can be done manually or through Fused Deposition Modelling (FDM) printers;

- ***Shape Deposition Manufacturing (SDM)***: suitable for 3D soft actuators made of multiple materials with different properties;
- ***Soft lithography***: suited for developing multichannel soft actuators;
- ***Virtual lost-wax casting***: a variant of a technique normally applied to cast metal. In this case, the final part to be obtained is virtually designed (CAD) and a virtual mold is created by inverting the part design. This mold is then 3D printed and filled with uncured silicone. After curing, the mold is destroyed using a solvent;
- ***Soft 3-D printing***: the most promising technology due to the elimination of several moulding stages, which makes the manufacturing process easier and allows the design of more complex inner chambers or pneumatic networks.

In [29] a multi-fingered soft gripper design that comprises hydraulic-driven and sheet-shaped fabric bending actuators is proposed. In [44] a bioinspired soft robotic gripper for adaptable grasping is proposed. The manufacturing process involves molding and casting of the Dragon Skin 30 silicone inside the molds. In [25] both hybrid robotic gripper and a complete soft robotic gripper are proposed. They are characterized by retractable telescopic inflatable fingers. This design is thought to be exploited in unknown environments due to their high conformability and compactness. As shown in Figure 2.9, telescopic mechanisms are made of urethane rubber (Smooth-On Vytaflex 40) while the claws are 3D printed with PLA (PolyLactic Acid) material, and driven via rack and gear couplings connected to three Robotics Dynamixel XM430-W350 smart actuators.

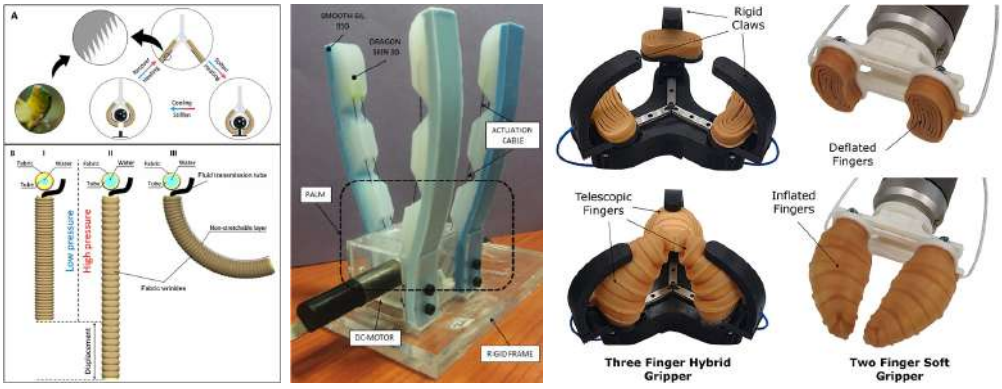

Figure 2.9: *From left to right: a soft fabric gripper with gecko adhesion; a bioinspired soft gripper made of Dragon Skin 30 silicon; a hybrid/soft robotic gripper made of urathen rubber.*

2.1.5 Controlling a soft gripper

As previously mentioned, soft actuators are deformable and compliant, which translates into a large intrinsic number of degrees of freedom. How a soft

actuator is controlled highly depends on the chosen materials and the control complexity can be reduced based on the design. Soft grippers are often cited as an example of ***morphological computation*** meaning that control complexity is reduced by material softness and mechanical compliance [63]. Several control strategies have been proposed for FEA-type actuator technology such as Proportional-Integral-Derivative (PID) control, closed loop curvature control, real-time Artificial Neural Network (ANN) control. However, open-loop control is one of the most frequently used. According to a recent review [47], difficulties can be encountered while controlling certain types of FEA soft actuators and passive structures actuated by external motors or tendon motors, due to their deflection around the object.

2.2 Agricultural practices' automation

2.2.1 Harvesting process classification

Harvesting is a process that comes into play right at the final stage of fruit development and determines the fruit quality. It is important to harvest fruits and vegetables at the proper stage of maturity in order to maintain their nutrient quality and freshness for prolonged period of time [30]. Nowadays, the majority of fruits used for fresh consumption are harvested by hand, and a mechanical harvester may take care of those used for processing. Hand harvesting requires quite a long time and excessive labour use, while mechanical harvesting has a greater efficiency. According to a recently published (2021) review paper [47], mechanical harvesting methods can be divided into:

- ***Indirect harvesting:*** a force is applied to the plant without making a direct contact with fruits; involves methods such as air blasting, limb

shaking, trunk shaking and canopy shaking (typically used for olives, almonds, pistachio nuts).

- ***Direct harvesting:*** used whenever a plant due to its structure can't be shaken, requiring a direct application of a mechanical force on the fruit or its peduncle. In this case picking techniques (or patterns) such as twisting, pulling, bending, lifting or a combination of them, are chosen to effectively detach fruits from the stem (e.g. strawberries, apples, tomatoes).

- ***Direct harvesting with an actuation force on the peduncle:*** applied when a cutting tool is required to properly detach the fruit because of its hard peduncle connection to the plant (e.g. oranges, cucumbers, peppers).

In figure 2.10 a classification of the most commonly used harvesting methods as presented in [47] is shown.

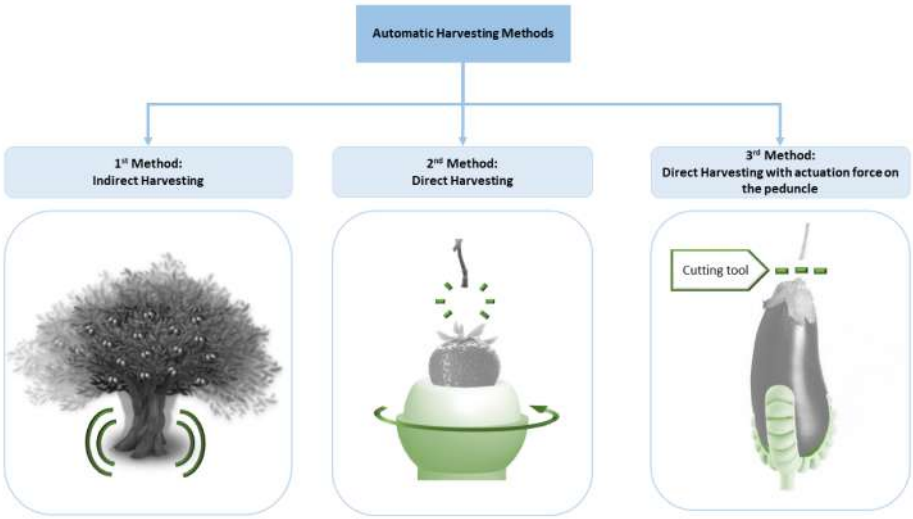

Figure 2.10: *Classification of automatic harvesting methods, according to [47].*

Depending on the crop, more than one harvesting technique could be used and several factors such as size, shape, fragility of the tree, maturity stage of the fruits, the will to risk damaging fruit or plant, financial profitability, determine the choice of the most suitable one.

2.2.2 Harvesting picking patterns

Regarding the second mentioned harvesting method (direct harvesting), further considerations can be made. A research branch in robotics focuses on studying the human movements performed during the harvesting of crops, with the objective of replicating them using robotic grippers. These movements are the so called ***picking patterns***, which include bending, lifting, twisting, and pulling or a combination of them (shown in Figure 4.34).

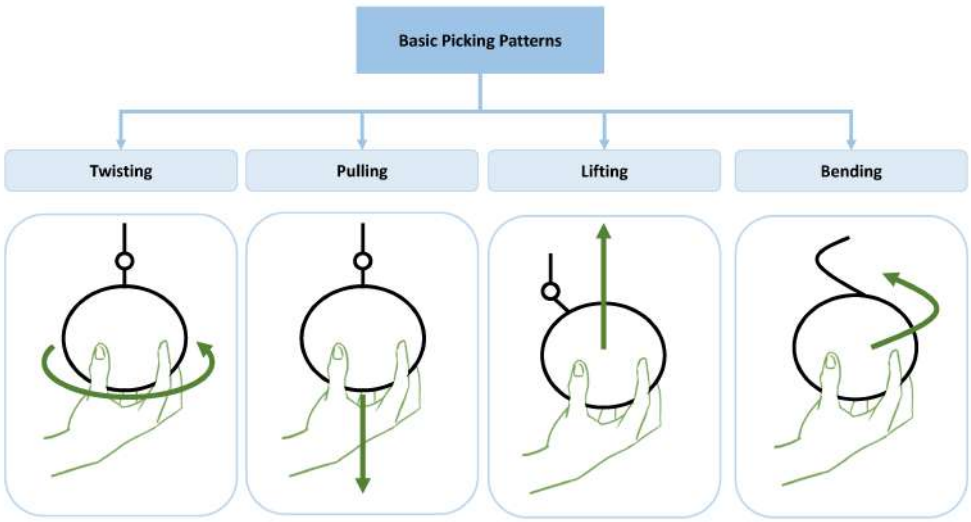

Figure 2.11: *A simplified scheme of basic picking techniques, according to [47].*

In literature, several studies have been conducted to understand the most suitable picking pattern and therefore gripper design, for each fruit such as tomatoes, apples, kiwis, strawberries. In particular, soft grippers are being

developed because of the compliance characteristics that allow delicate manipulation of the fruit, since direct contact is required while harvesting. Also, direct harvesting with an additional actuation force can be solved using soft gripper technology and a suitable cutting tool such as saw, hot wire, scissors or a knife [63].

This Master's Thesis' work, mainly due to the dimensions of the manufactured sensing device, focuses on the harvesting of small-size fruits such as strawberries and tomatoes that can be harvested following the second method. The picking pattern usually includes twisting and pulling once the fruit is grasped. Instead, many other fruits such as olives, raspberries and blueberries that would be directly harvested by hand, they're far easier to harvest with the first method (e.g. shaking the plant), if automation is involved.

2.2.3 Automation level of agricultural processes

As often happens, tasks that can be easily carried out by humans become very challenging for robots. In the field of crop harvesting, an experienced farmer can firmly and rapidly distinguish maturity stage of a crop not only by the color but also by the size, shape, surface texture, softness and resonance (sound it creates when tapped). If tasted, the fruit or vegetable can be harvested considering its aroma, sweetness, sourness, bitterness. Most of this properties are difficult to be sensed by a robot and this is the reason why most of the agrobots are only suited for harvesting crops that will be processed before sale. Also, the farmland environment is typically non-structured, highly dynamic and full of obstacles. It's characterized by dumpy and uneven ground that can only get worse if bad weather presents. A brief review of the currently present on the market solutions is presented.

According to a recently published (2021) review paper "Advances in Agri-

culture Robotics: A State-of-the-Art Review and Challenges Ahead" [48], AI, IoT technologies and computer vision algorithms can be successfully used for soil and weed management, fruit classification and weed detection in complex environments. Regarding the land preparation task, the German company Raussendorf [55] developed in 2014 "***Cäsar***" (shown in Figure 2.12), a mobile four-wheel drive remote-controlled/temporarly-autonomous robot for soil fertilization. In order to perform such a task Real-Time Kinematic (RTK) technology for the Global Navigation Satellite System (GNSS) is used, allowing to improve the robot's location accuracy up to 3cm. It is designed to work in conjunction with farmers featuring a collision detection system with a maximum detection distance of 5m. On the other hand, the Chinese company DJI [15] developed a flying drone equipped with 8 rotors (***AGRAS MG-1P***) to perform agricultural activities such as applying liquid fertilizers, pesticides and herbicides (shown in Figure 2.12). It has a transporting capacity up to 10 liters over a maximum distance of up to 3 km, ensuring a spraying capacity of 6 ha/h. To avoid collisions with high voltage wires or high vegetation, an omnidirectional radar anti-collision system is embedded, allowing to detect obstacles up to 15 meters. This approach can be useful whenever direct contact with the soil is not required, as it accelerates the process and avoids terrestrial obstacles and navigation in a rough environment.

Ascending Technologies [5] proposed ***AscTec Falcon 8***: a remotly controlled multicopter designed for inspection and monitoring, survey and mapping applications. It can be used in agriculture for monitoring the amount of chlorophyll present in the vines, preventing or highlighting any possible disease. ***Tevel Aerobotics Technologies*** [70] proposed a flying autonomous robot aimed at automating the fruit picking task, selecting the ripe crops and gently grasping them. Figure 2.13 includes the mentioned flying drones.

Figure 2.12: *On the left Raussendorf autonomous system for agricultural purposes Cäsar; on the right DJI AGRAS MG-1P Series agriculture drone.*

Figure 2.13: *On the left AscTec Falcon 8 flying drone; on the right Tevel Aerobotics automated fruit picker.*

Another key task in agriculture is weed control: a type of pest control that aims at reducing the growth of noxious weeds that compete with crops for space, nutrients, water and light. Several companies developed autonomous robots to remove those undesirable weeds: the french company Naïo Technologies [46] proposed "***Oz***", "***Dino***" and "***Ted***" for large scale vegetable farms and wine growers. According to Naïo Technologies, 70 Oz robots were sold in 2018 alone, being 80% of sales to the French internal market, 15% to European countries and 5% to the rest of the world. The three Naïo Technologies robots are shown in Figure 2.14.

The french company ***VITIROVER Solutions*** [74] developed a compact, lightweight mobile robot for weeds removal (shown in Figure 2.15). It's able

Figure 2.14: *Autonomous robots proposed by Naïo Technologies, from left to right: Oz, Dino and Ted*

to operate under various weather conditions, it is equipped with photovoltaic panels and allows control and monitoring through a mobile application, putting into practice the IoT concept. Also, Tertill Corporation [69], based in Massachusetts, proposed ***Tertill***: a cheap (349.00 USD), light and small wheeled autonomous robot designed to remove weeds from residential gardens (shown in Figure 2.15).

Figure 2.15: *On the left the completely autonomous mobile robot for precisely controlling soil grassing proposed by VITIROVER Solutions; on the right Tertill Weeding Robot.*

Once crops are mature, the harvesting process takes place. The spanish company Agrobot [3] proposes ***Agrobot E-Series***: a robot that consists of (up to) 24 independent cartesian robotic arms able to work together for gently

harvesting delicate fruits such as strawberries. As can be seen in Figure 2.16, it has three wheels and its mechanical structure can be adjusted to suit the crop dimensions.

Figure 2.16: *Agrobot E-Series stainless steel and military-grade aluminum robot.*

Moreover, the american company Harvest CROO Robotics [28], created ***Berry 5***: a robotic picker that exploits AI to determine if a strawberry is ripe or not, before harvesting (shown in Figure 2.17). It has a picking speed of 8 seconds per fruit, moving through strawberries beds at a speed of 1.6 km/h, resulting to an equivalent yield of 25 to 30 human harvesters [48]. Like the Agrobot E-Series, the various mechanisms of the Berry 5 robot are protected by patents, making its scientific analysis difficult.

Other companies such as Augean Robotics [9] and Harvest Automation [52] are focusing on robots that can cooperate with humans for carrying and organizing products with the objective of increasing industrial productivity. Augean Robotics developed ***Burro*** (shown in Figure 2.18): a mobile collaborative robot that exploits computer vision, high precision GPS, and AI to follow people and navigate autonomously while carrying or towing objects. It has a maximum carrying payload of 226Kg, depending on the terrain and a maximum towing capacity of 907Kg. Harvest Automation proposed ***HV-***

Figure 2.17: *Harvest CROO Robotics Berry 5 fruit picking robot.*

100 (shown in Figure 2.18): a mobile autonomous robot designed to perform material handling tasks in unstructured, outdoor environments such as those typically found in commercial growing operations. The robot can safely collaborate with workers and require minimal training to operate, with a maximum payload of 10Kg.

Figure 2.18: *On the left Augean Robotics' Burro self-driving robot; on the right Harvest Automation's HV-100 robot.*

Interestingly, the review paper [48] shows that among the 62 considered projects/available products, 80% of them are in the research stage. Also, most of them consists of four-wheel drive (4WD) mobile robots and almost 70% of

them do not include computer vision algorithms.

Despite the constant technological advances, many challenges are still to be overcome such as fruit occlusions, changes in ambient lighting, simplicity of construction and efficiency.

2.2.4 Open issues in agricultural automation and proposed solutions

According to [48], most agricultural robots are 4WD, but the agricultural environment is classified as semi-structured and this kind of locomotion is strongly affected by soil characteristics. Also, a trade-off between quality and cost of embedded electronic devices (sensors, cameras, IoT components) must be taken into account.

Wheeled robots not only struggle to move in an agricultural environment but can also cause undesired soil compaction. As mentioned, ***UAV*** devices can be a valid alternative if the task allows it. ***Legged robots*** are also proposed by [48] requiring less contact with the ground while moving and being able to adjust their posture depending on terrain's slope. Legged robots such as the ones shown in Figure 2.19 are relatively light, small, autonomous and have locomotion patterns that adapt to the environment. One drawback is that their small feet imply a small contact area that creates a considerable amount of pressure on the foot placement region. So, to prevent robots' feet from penetrating soft soils and trapping themselves, they require a customized feet design.

Also, embedded sensors can have a significant impact on the final product's cost. An idea can be to estimate variables instead of measuring them through a smart design, like suggested by the recent soft gripping and tactile sensing publications (discussed in Subsection 2.3.2). If budget allows it, by

Figure 2.19: *On the left ANYmal proposed by ANYbotics; on the right Unitree Go1 proposed by Unitree Robotics.*

investing in sensors with a high Ingress Protection (IP), meaning they can operate with high temperature and humidity ranges, at a small price increment, the robotic system can benefit in terms of lifetime. Also, computer vision and machine learning algorithms can furthermore improve the efficiency of automated tasks such as diseases' identification, detection of weeds, selective application of pesticides, location of crops, classification of ripeness and yield estimation. However, those algorithms need improvements in terms of robustness making them independent of weather, temperature, humidity and lighting changes. The use of AI algorithms, such as MLP (MultiLayer Perceptron), CNN (Convolutional Neural Network), R-CNN (Region-based CNN), and SVM (Support-Vector Machines) proved to be adaptable to rapid variation in natural lighting, changing seasons and crop growing [48].

2.3 Open issues in soft robotics

2.3.1 Soft grippers' limitations and required improvements

As seen, soft grippers have lots of advantages but still need some improvements. A common complain is that soft robots and grippers require a ***time-consuming multi-step fabrication process*** that involves mold making, casting, curing and support removal [7]. As can be noticed by the previously mentioned state-of-the-art solutions, the manufacturing of most soft grippers is very "handmade" and still far to be optimized for production. Therefore, repeatability can be hard to achieve, even though processes based on 3D printing and lost wax manufacturing can be valid options for standardizing the manufacturing.

Also, soft end-effectors are not easy to model often requiring technical expertise to account for the continuous deformations given by soft materials. Although their design, placement and testing is not trivial, many researches propose easier to develop end-effectors exploiting 3D printing technology [7, 47, 63]. As an example, at Carnegie Mellon University, Pennsylvania, some researchers proposed a fully printable low-cost dexterous soft manipulator that was designed through a framework they developed [7]. They were able to use the classical rigid-linked *Unified Robot Description Format* (URDF), generally not capable of describing continuous deformations of soft materials, exploiting quasi-rigid approximations. This way the end-effector's behaviour can be quickly evaluated in simulation. Depending on the type of gripping method there are advantages but also limitations: an apple can be firmly grasped while a strawberry would need a more gentle approach. Some grippers are easier to maintain and clean, others are only able to grip smooth

and dry surfaces, while others have a limited adaptive grasping. Features like modularity, ease of repair and the ability to handle food and multiple crops are desired for agricultural applications.

Another problem that is typically not addressed by researches is the ***energy source system*** of the soft grippers, that should be tailored to the agricultural unstructured environment. In fact, the proposed electrical, pneumatic or chemical energy sources are typically only suitable for a laboratory or a very structured industrial context.

In general, challenges for soft grippers include miniaturization, robustness, speed, integration of sensing, and control. Improved materials (elastomers, phase change materials) and processing methods play a large role in future improvements [63]. Finally, no matter how much the technology is advanced and reliable, the transition of soft grippers from the research stage to the industrial context needs to be economically competitive with respect to older methodologies and semi-automatic or manual approaches. This is a not so trivial point to address, because without political manoeuvres to incentive technology innovation, it's hard to justify huge investments for most of the small-medium sized enterprises.

2.3.2 Case study-related state-of-the-art

This Subsection is dedicated to a roundup of the main papers that were taken as inspiration for this Master's Thesis. In the current state-of-the-art there's a lack of information about spherical and hemispherical soft tactile sensors and grippers, so the few available researches on this topic are considered to be worth mentioning. The following case study-related works will be useful to better understand the next chapter.

One of the papers that inspired this work is ***"Rapid manufacturing of***

color-based hemispherical soft tactile fingertips", published in 2022 [58]. In this paper the authors present a 3D printed tactile sensor called ChromaTouch that exploits hue, centroid and apparent size of the markers to estimate normal and lateral forces. The device, shown in Figure 2.20 is made with a Stratasys J735 multi-material additive manufacturing system that allows the precise alignment of up to 400 markers on a 21mm radius hemisphere. The sensing principle is based on the relative displacement between differently colored markers that lie on two separate layers. In particular, the subtractive color mixing encodes the normal deformation of the membrane, and the lateral deformation is found by centroid detection. This approach stands out because

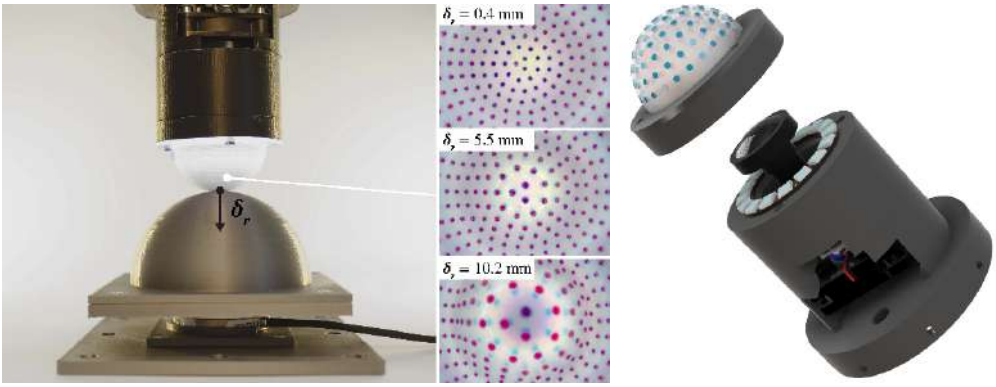

Figure 2.20: *On the left the 3D printed ChromaTouch tactile sensor and the transduced forces into marker appearance changes; on the right a render of the fingertip assembly.*

most of the existing marker-based solutions fail to directly encode the distance between markers and monocular camera, forcing the normal deformation to be estimated from the lateral displacement of the markers. In this case, the ChromaTouch sensor encodes normal deformation in the hue-value of the markers. Also, using subtractive color mixing (the color of the translucent markers on the inner layer is mixed with the color of the opaque markers behind them) allows a higher sensing resolution with respect to older proposals such as the GelForce sensor. However, the main objective of this work is to perform accu-

rate curvature estimations when the sensor is pressing against a positively or negatively curved object but contacting forces and torques are not estimated.

Another research work developed at the University of Madrid entitled ***"A universal gripper using optical sensing to acquire tactile information and membrane deformation"*** [57], proposed a granular-jamming based gripper with semi-transparent filling that allows to detect the membrane's deformation and the object being grasped. The prototype, shown in Figure 2.21, is able to grasp cylindrical and rectangular objects (10 to 70 mm length) while tracking the gripper's deformation so that object classification through the reconstructed pointcloud could be performed. Also, grasping success is detected estimating shear forces. The fabrication process consists in 3D printing the molds that are later filled with silicone or epoxy resin to respectively create the soft gripper's membrane and bulkhead. Also, a thickness of 1mm and a Shore Hardness A-20 were chosen, while the embedded circular markers have a diameter of 6mm and a thickness of 1.5mm. Even though the sizing and manufacturing of this prototype are different from ours, a very similar marker tracking approach was used and the suggested 3D position estimation algorithm was implemented but did not achieve acceptable results.

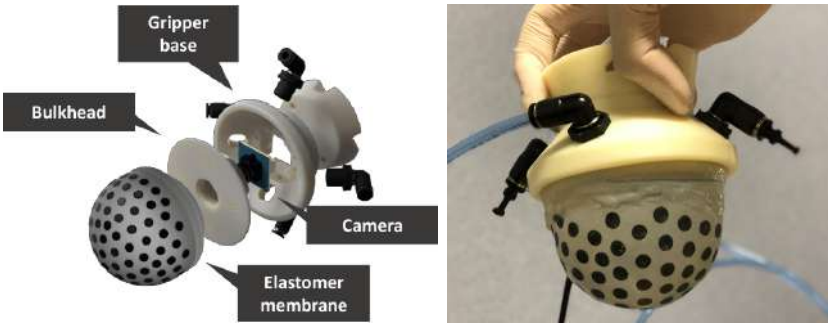

Figure 2.21: *On the left the rendered structure of the Universal Gripper; on the right the manufactured prototype.*

The paper ***"Soft-bubble: A highly compliant dense geometry tactile sensor for robot manipulation"*** [4] proposes a dense geometry sensor and end-effector for tactile-object classification, pose estimation and tracking (shown in Figure 2.22). It measures deformation of a thin, flexible air-filled membrane using a depth camera. The sensed features are then exploited to perform object shape and texture classification (using a Deep Neural network), object sorting, object pose estimation and tracking. As shown by the following scheme, the dimensions of the proposed device allow the use of a Time Of Flight depth camera (PMD pico flexx) and the design follows its minimum sensing distance of 100mm. As mentioned by the authors, this choice allows avoiding non-trivial and robust algorithms for 3D shape reconstruction (e.g. structured lighting, photometric stereo algorithms). To our knowledge, a similar approach could not be adopted due to the currently available on the marker depth sensors' limitations (size and sensing range), while keeping a compact design.

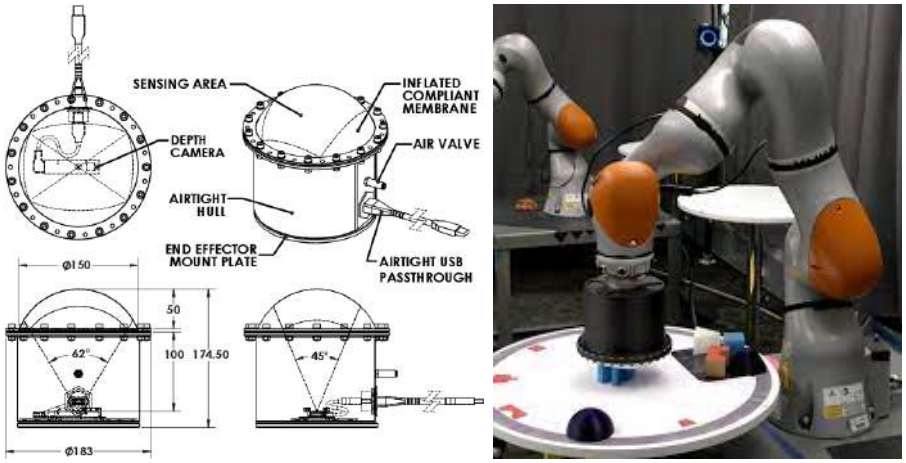

Figure 2.22: *On the left the dimensioned (mm units) sensor assembly; on the right the soft-bubble mounted on a KUKA iiwa robot.*

Similarly to the paper which has just been discussed, in ***"Soft-bubble***

grippers for robust and perceptive manipulation" [40], a soft-bubble gripper system is presented (shown in Figure 2.23). In this work the main contributions to this technology are the design improvements with a smaller parallel gripper form factor, the introduction of high-density markers on the internal bubble surface, used for estimating shear forces, a proximity pose estimation framework and integrated tactile classification. As in the previous work, a ToF camera is used to sense depth but in this case a prototype camera from PMD technologies with a working range of 4-11cm was employed (and placed at an angle to reduce the overall gripper width). Also, markers were added not to infer depth but to estimate slippage and grasp quality from shear-induced displacements.

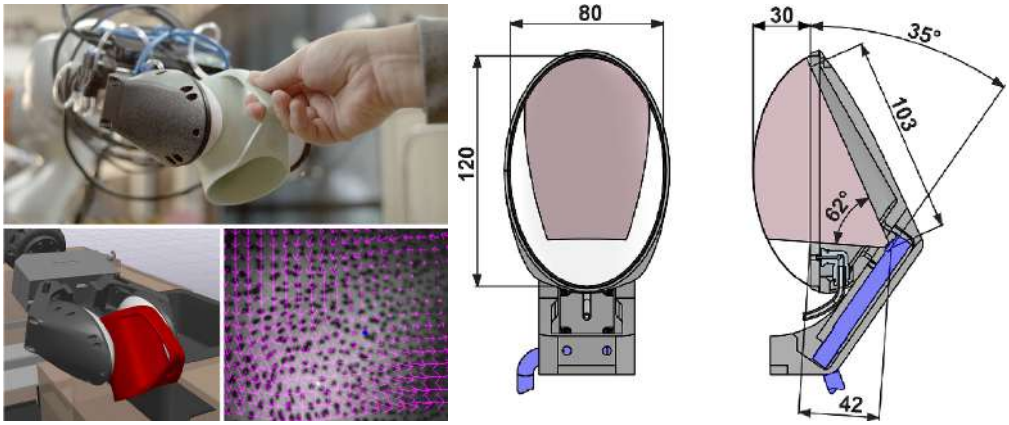

Figure 2.23: *On the left the soft-bubble parallel gripper that estimates in-hand pose and tracks shear-induced displacements; on the right the dimensioned scheme where the ToF depth sensor is depicted in blue.*

In ***"GelSight Fin Ray: Incorporating Tactile Sensing into a Soft Compliant Robotic Gripper"*** [43] a soft gripper with two sensorized fingers for retrieving tactile information is proposed. The Fin Ray design has the advantage of not requiring actuation for securely grasping objects unlike many soft and rigid grippers. Like mentioned by other studies, external ambient

lighting can interfere with a vision-based system: in this case a dark cloth was applied on the sensing device, obstructing outside lighting. As shown in Figure 2.24, the proposed Fin Ray finger is equipped with markers for slip and twist detection, it can measure the orientation of the in-contact object and through an RGB illumination and pre-collected reference images can perform 3D reconstruction.

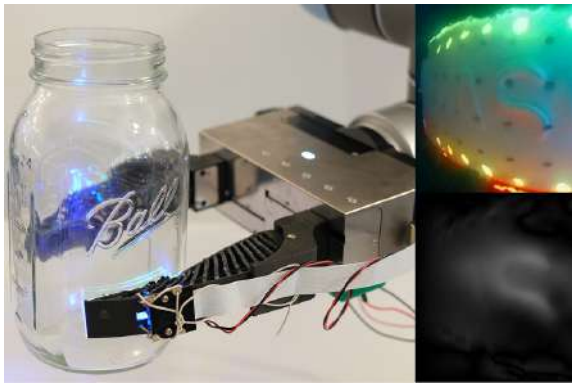

Figure 2.24: *The GelSight Fin Ray gripper.*

Regarding the force estimation problem, in ***"HiVTac: A High-Speed Vision-Based Tactile Sensor for Precise and Real-Time Force Reconstruction with Fewer Markers"*** [53] a prototype for force reconstruction is proposed. The developed algorithm allows real-time estimation of the direction and intensity of the external force. The HiVTac tactile sensor shown in Figure 2.25 is made of a square sheet of PDMS (polydimethylsiloxane) with dimensions of $40mm \times 40mm$. It has 4 markers that are tracked through a wide-angle camera. The main problem when re-adapting the obtained results on the case-study are the strong geometrical assumptions that are made thanks to a planar design, that are not suited for a hemispherical dome.

In ***"FingerVision for Tactile Behaviors, Manipulation, and Haptic Feedback Teleoperation"*** [78] and ***"Implementing Tactile Behav-***

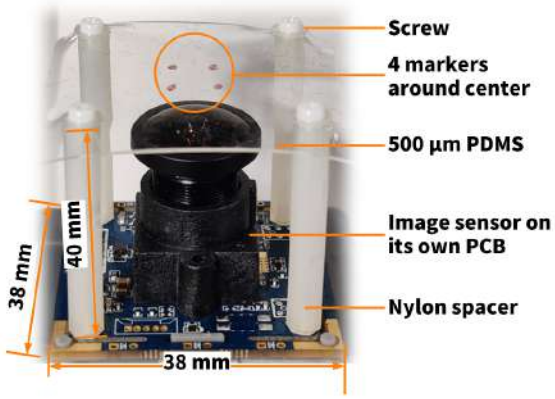

Figure 2.25: *The HiVTac tactile sensor prototype.*

iors Using FingerVision" [77] by Yamaguchi, the same force estimation approach is used. Also in these papers, the vision-based tactile sensor (shown in Figure 2.26) is almost planar, making it different from our design. Nonetheless, the same marker tracking and force estimation approach has been implemented and tested to see if a simple linearization would be suitable for small deformations at least. In particular, tangential forces are estimated considering the horizontal displacements of the markers while normal forces, due to the unstable marker's width reading of the detection algorithm, is estimated through the norm of the markers' position change. The noisy radius reading given by the BlobDetection algorithm is also confirmed by our work; the accuracy of this method to a hemispherical surface will be later discussed.

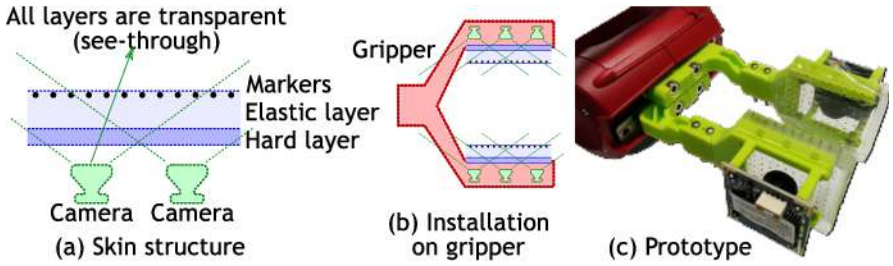

Figure 2.26: *Design of the FingerVision and its prototype installed on the Baxter gripper.*

In ***"Visiflex: A Low-Cost Compliant Tactile Fingertip for Force, Torque, and Contact Sensing"*** [18] a cheap compliant tactile fingertip is proposed. The sensor, shown in Figure 2.27, is capable of contact localization and force/torque estimation. According to the paper, tests indicate that typical errors in contact location detection are less than 1mm and typical errors in force sensing are less than 0.3N. At a first glance, the design of the Visiflex is very similar to ours, even though different approaches for force and application point estimation are used. In this work the american researchers used a dome-shaped acrylic waveguide covered by a silicone cap. Eight LEDs act as fiducial markers and the light injected into the waveguide is totally internally reflected except where the cap contacts the waveguide. This behaviour is exploited to easily sense either single or multi-contact with the external environment, as shown in Figure 2.28. Another difference in the design with respect to ours is due to a 6 degrees of freedom (DoF) fingertip that is accomplished using a 6-DoF flexure system. Wrench sensing is performed using several techniques such as stiffness matrix estimation, linear approximation and nonlinear approximation starting from the experimental data (based on Neural Networks).

Figure 2.27: *A rendering of the Visiflex [18] tactile sensor and its exploded view.*

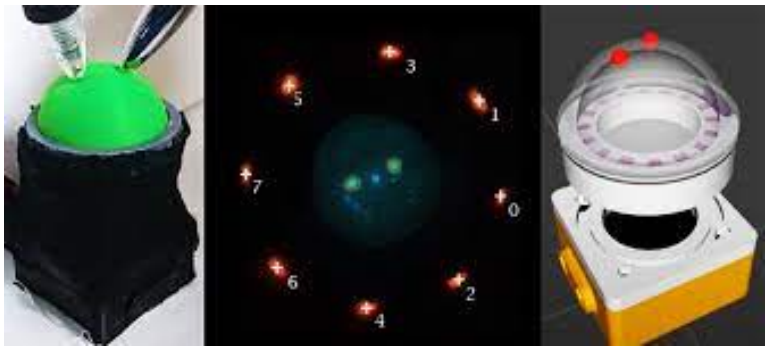

Figure 2.28: *The Visiflex sensor contacted in multiple points; red LEDs represent are the fiducial markers, while green LEDs are the contact points that can be seen by the camera, because of the waveguide.*

"The TacTip Family: Soft Optical Tactile Sensors with 3D-Printed Biomimetic Morphologies" [75] is a slightly older paper (published in 2018) that compares several proposed devices with the same biomimetic design principle. In particular, they all exploit deformation of the fingertip sensing the displacement of pins or markers using a camera. Several patterns are considered and compared, based on the in-hand manipulation and object exploration capabilities. In Figure 2.29 three iterations of the design are shown.

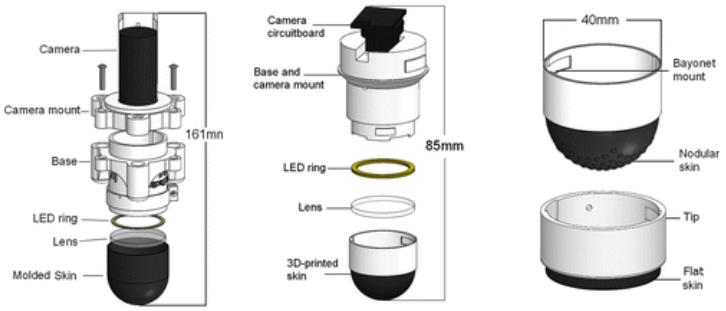

Figure 2.29: *As reported in [75]: Open-TacTip (left): the original version of the sensor comprises a 3D-printed camera mount and base and a cast silicone skin. Improved TacTip (center): the redesigned base houses a webcam, and modular tips with 3D-printed rubber skin. Modular tips (right): separate modular tips with a nodular fingerprint (above) and flat tip (below).*

In ***"DenseTact: Optical Tactile Sensor for Dense Shape Reconstruction"*** [16] a compact tactile sensor with high-resolution surface deformation modeling for surface reconstruction of the 3D sensor surface is presented. In this work force estimation is not addressed and the design doesn't comprise fiducial markers. As shown in Figure 2.30, using a 3-colored lighting inside the dome and an RGB camera, the surface deformation is estimated and the contacting object's shape is reconstructed. However, this estimates are highly dependent on 3D shape calibration process (using 3D printed objects and inferring CAD models) and Deep Neural Network's accuracy.

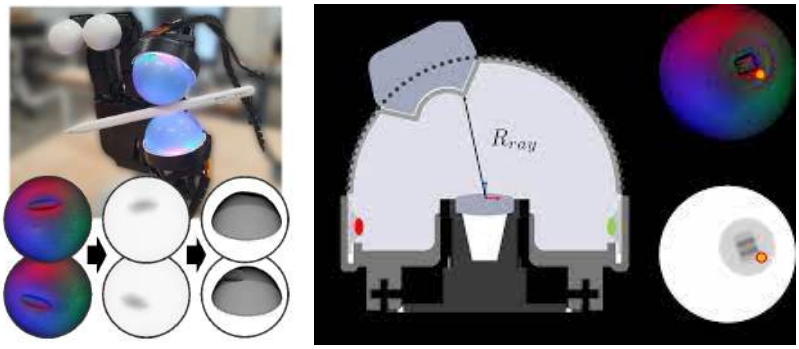

Figure 2.30: *On the left the DenseTact sensor mounted on the Allegro hand and its 3D reconstruction results; on the right a visualization of the ray casting algorithm, used to determine the radial depth from the 3D calibration surface which is then projected into the image plane.*

Similarly to the previously cited paper, in ***"GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force"*** [79] 3D shape reconstruction is performed using a structured light setup and a photometric stereo algorithm. In particular, 3 differently colored LEDs are arranged at different directions and combining the shading from three or more directions, surface normals on each pixel of the shaded image are estimated (as shown in Figure 2.31). Afterwards, surface normal is integrated to get the 3D shape of the surface. Also, force estimation is carried out through

marker tracking. The proposed sensor has a planar geometry, so the mentioned approaches were not easily applicable to our case study.

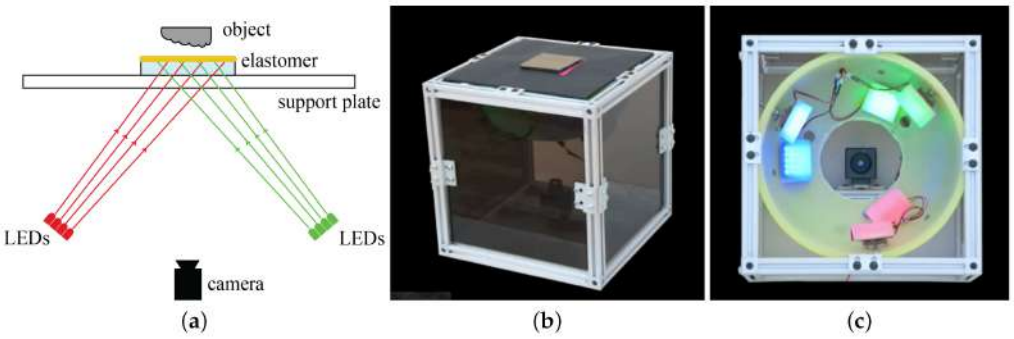

Figure 2.31: *As reported in [79]: (a) basic principle of the Gelsight design that consists of a sensing elastomer piece with the opaque reflective membrane on top, supporting plate, LEDs and camera to capture the shaded images with different lightings; (b) picture of the sensor; (c) arrangement of the LEDs and camera when viewing from the top.*

Chapter 3

The design of the prototype

In this chapter the specific case study is presented, analyzing step by step the methodologies used to manufacture the prototype. Also, the experimental setups and the main Computer Vision algorithms are described in detail.

3.1 Specifications and goals

As discussed in Chapter 2, there are several reasons why soft robotics is drawing attention of companies and researchers. Several recently published papers focus their attention on tactile sensing, that can provide information about the contact such as friction, slippage, surface features (curvature, texture), applied reaction forces and torques. Moreover, it can be a valid option for describing the object's shape, orientation, rigidity, in situations where vision is out of reach. As mentioned in [58] most of the research on this topic has been focusing on soft but flat tactile sensors. On the other hand, extending this technologies to freeform or hemispherical surfaces is not trivial. In order to sense the mechanical deformation the common strategy is to apply a pattern of markers inside the fingertip and track its motion using an RGB or RGBD

camera. This way, there's no need for direct wiring to the soft material, guaranteeing higher compliance than embedding capacitive or resistive materials. Based on the chosen pattern and marker density, the sensing resolution can be fairly high. As discussed in Chapter 2, since the fingertip design has to be compact, there are not available depth cameras (RGBD nor ToF) on the market that are able to precisely sense depth, requiring a minimum distance of around 10cm. So, if a monocular camera is used some other techniques have to be exploited to perform depth estimation. In [58] 2 layers of differently colored markers are exploited; in [57] they rely on camera calibration and the sensed markers' dimensions; in [43], [16] and [79] deep neural network approaches are used to calibrate the sensor and perform 3D reconstruction.

In our work we focused on the development of a ***cheap, 3D printed, marker-based hemispherical soft gripper*** capable of sensing forces when in contact with the external environment. Moreover, the 3D shape estimation approach proposed in [57] has been explored.

The final goals of our work are to:

- design and manufacture a soft tactile sensor with a suited pattern of fiducial markers to be tracked;
- calibrate the sensing device in order to perform online estimation of both shear and normal forces;
- design a structure that can hold the sensing device in place and mount it on the "Franka Emika Panda" [23] robot's end-effector (shown in Figure 3.1);
- attempt a simple harvesting task exploiting the developed device within a force control loop on the gripper and an external depth camera mounted on the robotic arm.

Figure 3.1: *The Franka Emika Panda robotic arm equipped with the Franka Hand 2-fingers gripper.*

3.2 The manufacturing process

3.2.1 Prototyping and manufacturing the hemispherical dome

First of all, we focused on the manufacturing of the tactile sensor and the pattern selection. Prioritizing ease of manufacturing and reproducibility, we decided to avoid as much as possible multi-material additive manufacturing and molding, focusing on 3D printing. Our hemispherical sensing device was 3D printed using the Formlabs Form 2 stereolithography (SLA) 3D printer [21] shown in Figure 3.2. It costs around 3000$ and it's characterized by a layer thickness or axis resolution of respectively 25, 50 and 100 microns. The compatible materials are also provided by Formlabs [22] and they range from 150$/l to 400$/l depending on mechanical properties such as stiffness, elasticity, thermal resistence and achievable resolution (up to 0.005mm). To 3D print the hemispherical dome we used the *Elastic 50A Resin*, a polymer resin that supports a maximum resolution of 100 microns and 50A of Shore Hardness. The required printing time depends on the geometry and height

of the model; in our case it was around 4h at a 0.10mm resolution for the single dome, while 5h at a 0.10 resolution for two domes. SLA resins are ***photocurable*** through light that lies in the ultraviolet spectrum.

Figure 3.2: *From left to right: the Formlabs Form 2 stereolithography 3D printer; the Elastic 50A Resin; a 3D printed sample as shown on the website [22].*

After the ***3D printing process***, the elastomer looks like in Figure 3.3, so all the unnecessary material has to be properly removed to make the surface smooth. After 3D printing, the models have to be cleaned either with IPA (Isopropyl Alcohol), with a proper washing machine or by hand, like in our case, due to the fragility of the material. In particular the 3D printed domes need to be dipped in IPA for about 10 minutes; then it has to be gently shaken while inside the washing tub and left to soak for another 10 minutes. This process allows to remove all the resin residue from the polymerized structure, due to the fact that the printer continuously deposits resin on the model and the laser beam causes its polymerization just above the printing plane (the laser loses power after the 0.10mm length). The cleaning process is very important to avoid that all the residues on the surface polymerize during the next step.

Afterwards, the ***UV curing process*** takes place inside a special ultraviolet oven for about 20 minutes at 60°. Without entering into too much detail, *curing* is a chemical process that causes the toughening or hardening of a poly-

Figure 3.3: *From left to right: the just printed dome; the washed dome; how the washed dome presents on the inside.*

mer material by cross-linking of polymer chains and in our case it was induced by heat. The amount of required curing time is computed considering the goal Shore Hardness of the final material that is about 50A. To better understand the meaning of this value, Figure 3.4 shows a Shore Hardness scale of general purpose items. Shore Hardness is in fact a measurement unit that indicates the resistance of a material to indentation. As shown in Figure 3.4, there are different scales, depending on the application field and material type, that can overlap each other.

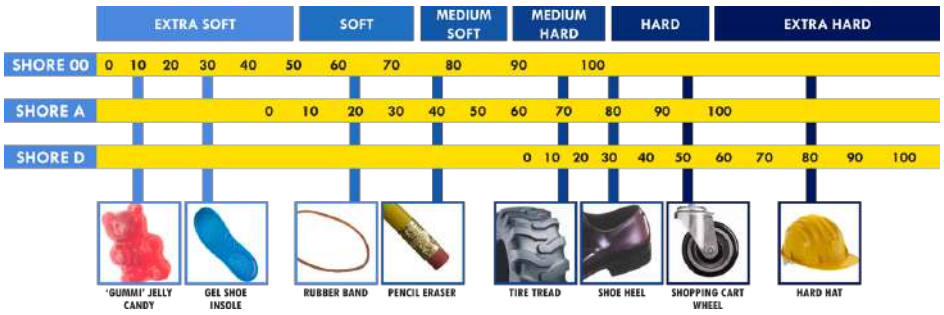

Figure 3.4: *Shore Hardness scale of general purpose items.*

Figure 3.5 shows a summary of the most common materials' properties and curing times, according to [33], used in soft robotics.

Figure 3.6 shows how the polymer dome presents before and after the curing process.

		Shore Hardness	Tensile Strength	Elongation at Break	100% Modulus	Cure Time	Shrinkage
Latex	Trylon Latex Liquid Rubber						0.1 in. / in. 10%
Silicone rubber	Smooth-On Ecoflex® 00-30	00-30	200 psi	900%	10 psi	4 hours	<.001 in. / in. 0.1%
	Smooth-On Ecoflex® 00-35 Fast	00-35	200 psi	900%	10 psi	5 minutes	<.001 in. / in. 0.1%
	Smooth-On Ecoflex® 00-50	00-50	312 psi	980%	12 psi	3 hours	<.001 in. / in. 0.1%
	Smooth-On Dragon Skin® 10 Fast	10A	475 psi	1000%	22 psi	75 minutes	<.001 in. / in. 0.1%
	Smooth-On Mold Star® 20T	20A	420 psi	470%	47 psi	30 minuts	<.001 in. / in. 0.1%
	Smooth-On SORTA Clear® 37	37A	600 psi	400%	90 psi	4 hours	<.001 in. / in. 0.1%
	Smooth-On SORTA Clear® 40	40A	800 psi	400%	90 psi	16 hours	<.001 in. / in. 0.1%
Polyurethane rubber	Smooth-On Econ® 60	60A	350 psi	500%	82 psi	16 hours	<.001 in. / in. 0.1%

Figure 3.5: *Chart of the most common soft materials' properties and curing times, according to [33].*

Figure 3.6: *On the left the hemispherical dome before being cured; on the right the hemispherical dome after being cured.*

Once the material is cured, it becomes semi-transparent allowing external light to partially pass through. This behaviour, that initially was thought to be an added value is actually something to deal with in order to achieve an accurate tracking algorithm. In fact, the shadows generated by the in-contact object tend to interfere with the markers' detection and tracking algorithms. To avoid this problem, a temporary light-colored cloth was placed over the dome during testing and calibration, while a thin layer of latex was used as a cover on the final sensing device. Also, it's worth mentioning that the higher the curing time the greater the opacity but also the material's stiffness.

The prototyping process included multiple iterations of the dome design, in particular:

- initially, the markers' diameter was 0.9mm (inside view of the dome shown in Figure 3.7) and they were spaced by 1.5mm intervals. To improve markers' detection and tracking they were increased to a diameter of 2mm and a spacing of 2mm;

- four different patterns were printed and evaluated based on their sensitivity to applied forces and robustness during markers' tracking. As mentioned by state-of-the-art related papers, a tradeoff between sensing resolution and signal-to-noise ratio must be found. In our case, we decided to adopt the "double cross" design shown in Figure 3.8. We led to this decision due to the markers' detection and tracking reliability; also, the pattern was designed in such a way that diagonal crosses can be distinguished from orthogonal crosses by the number of markers and have a slightly different spacing. The total number of markers on the chosen design is 29 and they were filled with black resin to maximize the contrast (this operation could be easily integrated during 3D printing, if the printer allows it).

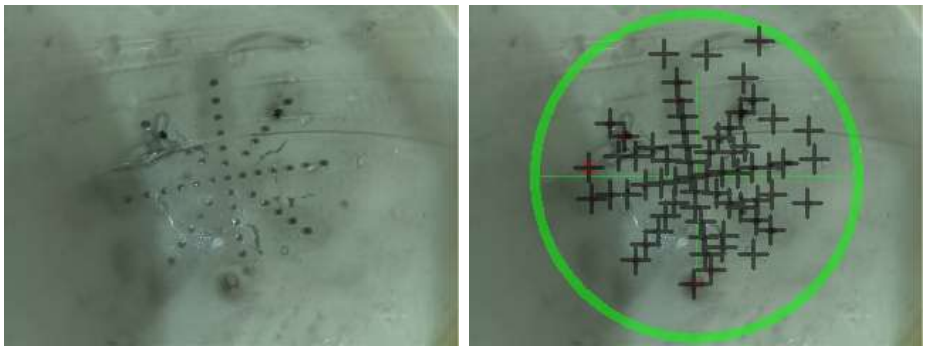

Figure 3.7: *On the left the initial "double cross" design consisting of 0.9mm diameter markers with a spacing of 1.5mm; on the right the superimposed detected blobs which are clearly noisy.*

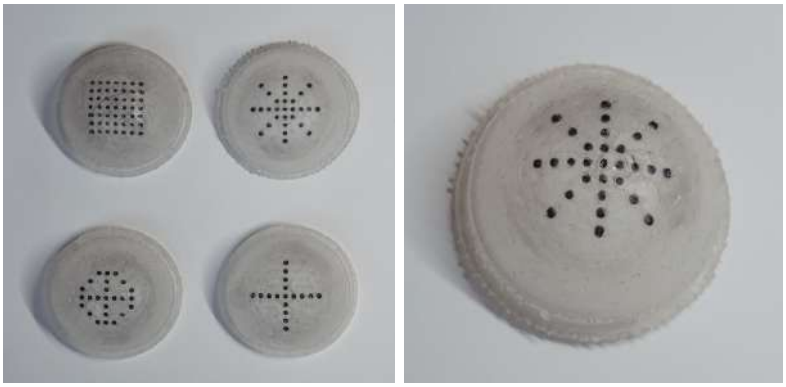

Figure 3.8: *On the left the four designed patterns; on the right the chosen "double cross" pattern.*

3.2.2 Design and manufacturing of the case

The case was 3D printed using a cheap FDM (Fuse Deposition Modelling) printer and it's made of PLA (Polylactic Acid). Initially, it was thought as a 4 parts design to decouple as much as possible the different layers, as shown in Figure 3.9. In fact, starting from top to bottom:

- a square piece constrains the hemispherical dome at its base;

- a thinner plate with a circular indentation holds the Adafruit NeoPixel Ring (12 LED) [1];
- a "box-like" shaped piece holds the 5MP fish-eye RGB camera for Raspberry Pi [20];
- the base completes the design and includes several holes for positioning screws and cable management.

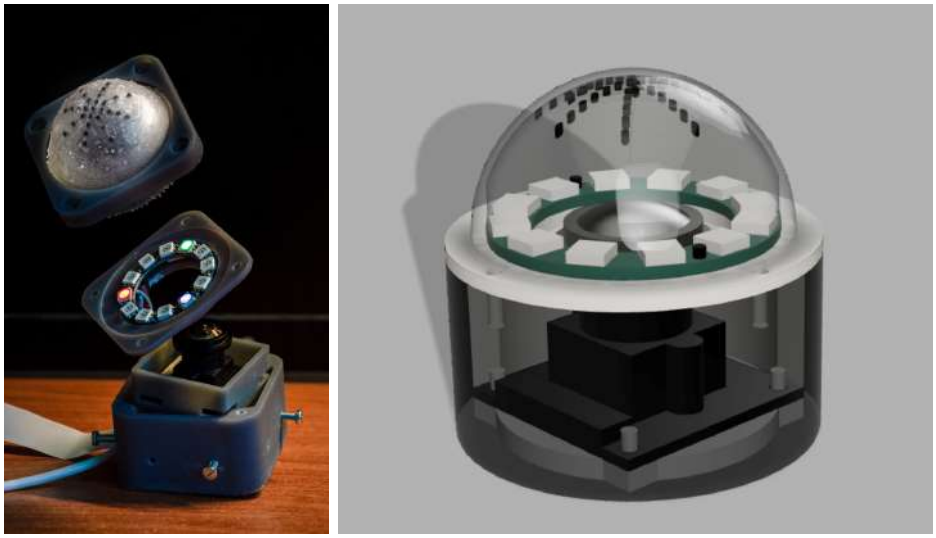

Figure 3.9: *On the left a photograph of an exploded view of the initial prototype; on the right a render of the prototype.*

After several iterations, mainly due to a bad centering of the camera that led to poor force estimation results and a non-symmetric image, the design was further improved. Having a modular design allows to re-think and print a single part instead of the whole prototype, making the overall process faster and minimizing products' waste. In fact the final design consists in 4 flat elements that are fast to print (around 40 minutes each) and the main through hole box that requires 2h, for a total of about 5h at 0.18mm resolution for the

whole case. Considering the amount of required material, the production cost of the case is around 4€.

Figure 3.10 shows a render of the final design that was mounted on the Franka Emika Panda's gripper.

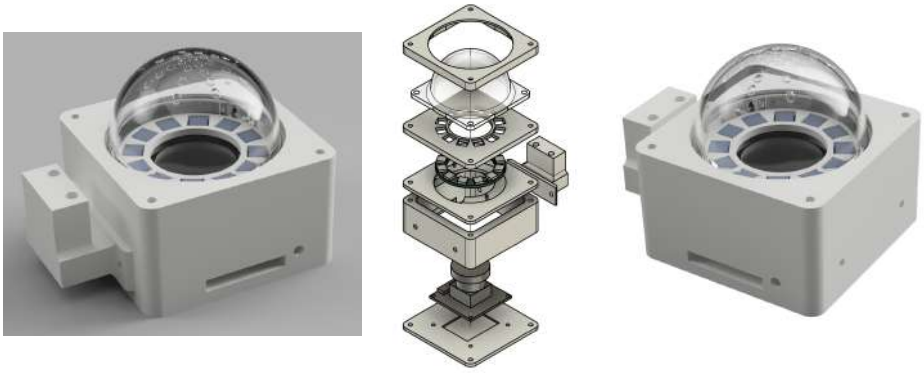

Figure 3.10: *Multiple rendered views of proposed 3D printed modular design.*

3.3 Experimental setups

In this section the experimental setups are described and divided into three categories: testing setup used during software implementation; temporary setup for data acquisition during sensor's calibration; definitive setup mounted on the robot.

3.3.1 Testing setup used during software implementation

The most frequently used setup during our work is the one shown in Figures 3.11 and 3.12. It includes the following hardware components:

- Raspberry Pi 3 Model B [54], its power supply and USB WiFi Adapter;

- Arduino Uno [68];
- 3D printed hemispherical dome mounted on the 3D printed case;
- Adafruit LED NeoPixel Ring 12 RGB [1] (inside the case);
- IR 1.7mm focal length 5MP resolution 175° field of view fish-eye camera (Chip OV5647) for Raspberry Pi [20] (inside the case);
- connection cables for the LED Ring and the camera;
- power supply connector and splitter for charging the Arduino Uno board and the LED Ring;

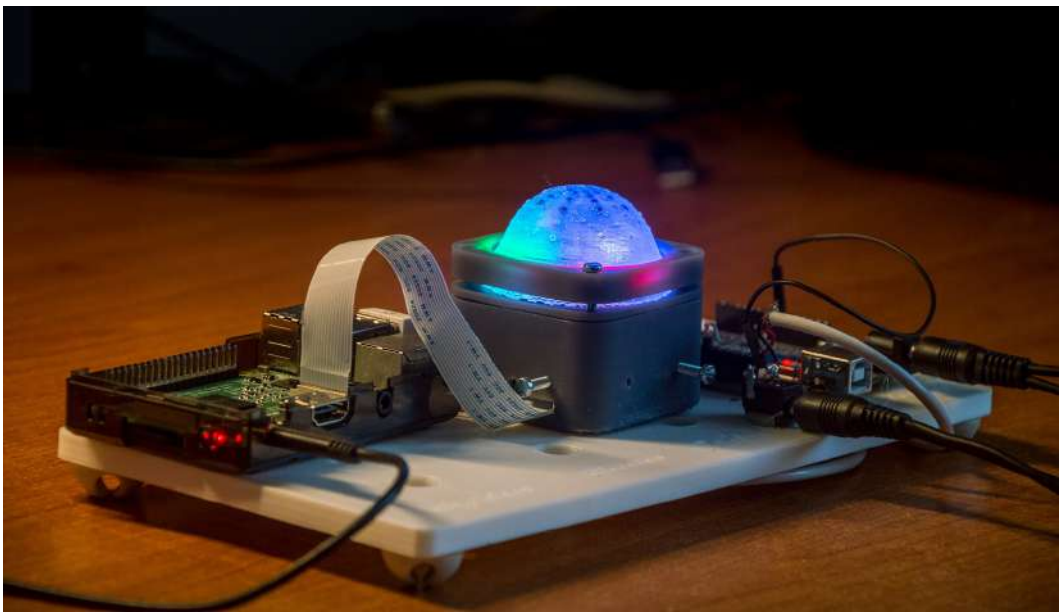

Figure 3.11: *Overall view of the testing experimental setup.*

On the Raspberry Pi 3 board, "Raspberry Pi OS" (previously called Raspbian) was installed. It acts like a server establishing a connection with the main computer and sends to it the real-time images collected by the camera. The Arduino Uno board was used to easily control the Adafruit LED

NeoPixel Ring with the dedicated library [17]. The main computer connects to the Raspberry server as a client (they must be connected to the same Local Area Network) and processes the received raw frames.

Figure 3.12: *Closer views of the experimental setup.*

3.3.2 Temporary setup for data acquisition during sensor's calibration

Once the markers were properly detected and tracked by the Computer Vision algorithms (that will be discussed in Section 3.4), a temporary setup for data acquisition was created. In addition to all the mentioned components of the "testing setup", an ATI Nano17-E force-torque sensor [6] and a 3-axis DoF Cartesian robot were exploited. Figure 3.13 shows a 3-axis Cartesian robot similar to the one we used and the ATI Nano17-E sensor. The "testing setup" was placed on a horizontal surface while the 3-axis robot holding the ATI Nano sensor was pressed against the hemispherical dome. In particular, we measured normal forces applied to the same point that was, approximately, the dome's center. Data acquisition and processing will be discussed more in Subsection 3.4.3.

Figure 3.13: *On the left an example of Cartesian robot; on the right the ATI Nano Series 6-axis force/torque sensor.*

3.3.3 Definitive setup mounted on the robot

As mentioned before, the Franka Emika Panda robotic arm was used. According to the datasheet [24], it has a maximum payload of 3kg, 855mm of reach and 7 degrees of freedom (Figure 3.14 shows the robot's workspace from two different points of view).

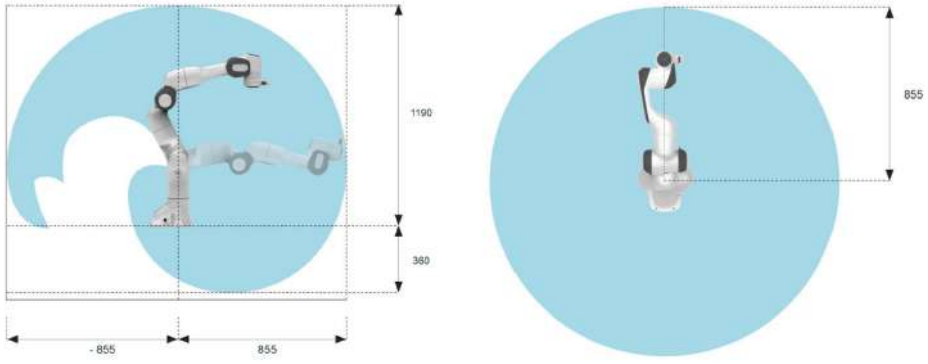

Figure 3.14: *On the left a side view of the arm's workspace, on the right a top view of the arm's workspace.*

Also, it is equipped with a hand parallel gripper with exchangeable fingers (shown in Figure 3.15), that we unscrewed to install the developed sensing

device. Regarding the applicable force, it ensures a continuous force of 70N and a maximum force of 140N. The robot was programmed with C++ and Python languages using ROS (Robot Operating System).

Figure 3.15: *Original Franka Emika Hand gripper.*

As shown in Figure 3.16, the setup that was mounted on the robot arm in order to perform a grasping task includes:

- the Franka Emika Hand gripper with 2 sensing devices (of which only one was used for real-time force sensing, assuming that the force is equally distributed among the two);
- an Intel RealSense D435i [32] depth camera;
- a Raspberry Pi 3 board;
- a series of 3D printed suitable holders for each component.

Figure 3.16 shows two renders of the Panda's gripper with the custom sensing device that replaced the original fingers and the Real Sense depth camera. Moreover, Figure 3.17 shows the 3D printed design mounted on the real robot.

Figure 3.16: *Renders of the definitive setup mounted on the Franka Emika Panda robot.*

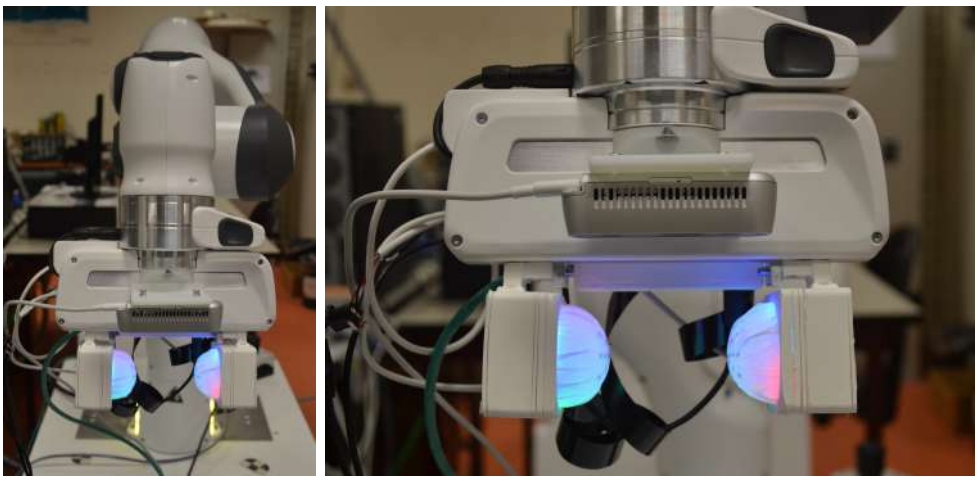

Figure 3.17: *Photographs showing the Franka Emika Panda robot with the custom gripper installed on the Franka Hand.*

As can be noticed looking at Figure 3.16, the Arduino Uno was removed to improve the design, by simply installing the dedicated library [26] on Raspberry Pi to manage the Adafruit LED Ring. The Intel RealSense D435i shown in Figure 3.18 is a depth camera that was exploited to autonomously perform a grasping/picking task. In fact, RGBD cameras allow to retrieve not only a color image but also 3D coordinates (with respect to its own reference frame) associated to every pixel in the image. Future developments may further take advantage of it offering: collision avoidance and consequent trajectory plan-

ning, safer human-robot interaction, more effective picking pattern selection, more robust and reliable object detection.

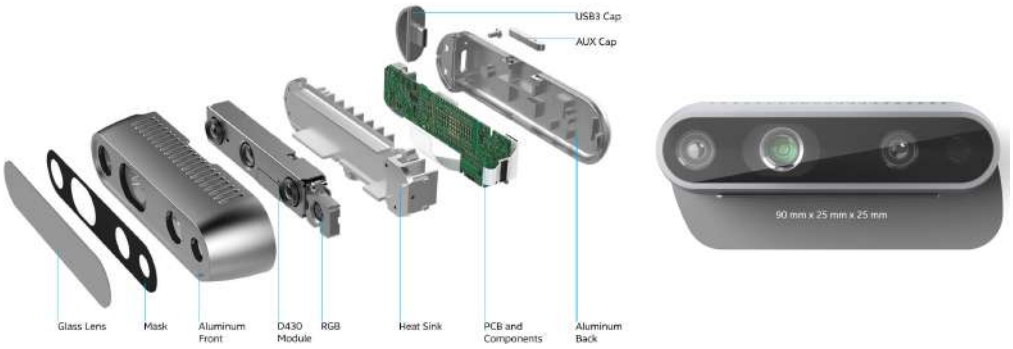

Figure 3.18: *The Intel RealSense D435i RGBD camera.*

3.4 Implemented software algorithms and approaches

This section explores the main implemented algorithms and pipelines, ranging from marker detection to online force estimation.

3.4.1 Marker detection and tracking

The individual markers inside the dome are detected using OpenCV Blob Detection algorithm. The input image that is sent from the Raspberry Pi board to the main laptop is then converted to grayscale used as input of the OpenCV *SimpleBlobDetector*. The detector's parameters are fine-tuned for our specific case; in particular, blobs are filtered by area and a maximum threshold is applied. According to "LearnOpenCV"'s guide [41], the main steps of the SimpleBlobDetector are:

- ***Thresholding:*** the source image is converted to several binary images by thresholding it with thresholds starting at *minThreshold*. These thresholds are incremented by *thresholdStep* until *maxThreshold*;
- ***Grouping:*** in each binary image, connected white pixels are grouped and they're called *binary blobs*.
- ***Merging:*** the binary blobs' centers in the binary images are computed, and blobs located closer than *minDistBetweenBlobs* are merged;
- ***Center & Radius Calculation:*** the centers and radii of the newly merged blobs are computed and returned.

As shown in Figure 3.19, further thresholding options can be set as parameters of the SimpleBlobDetector. In our application, markers can deform to ellipses but still need to be tracked, so filtering was applied by area and a manual filter was introduced to consider only markers detected within a pre-defined radius from the frame's center (assuming the dome to be centered with respect to the camera).

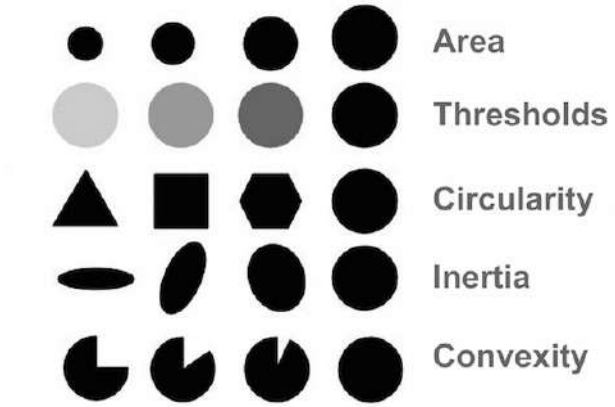

Figure 3.19: *SimpleBlobDetector's thresholding options.*

The Listing 3.1 shows how the SimpleBlobDetector was created and initialized specifically to be reliable during marker detection with the 29 markers "double cross" design.

```
import cv2
def set_parameters():
    # Initialize parameters of the SimpleBlobDetector
    params = cv2.SimpleBlobDetector_Params()

    # Filter by Area
    params.filterByArea = True
    params.minArea = 20   # ideal for "double cross" pattern
    params.maxArea = 200  # ideal for "double cross" pattern

    # Maximum threshold
    params.maxThreshold = 125

    # Create a SimpleBlobDetector with the chosen parameters
    detector = cv2.SimpleBlobDetector_create(params)

    return detector
```

Listing 3.1: *Python snippet of the SimpleBlobDetector's initialization*

Also, in Figure 3.20 both the raw frames (before and after deformation) sensed by the fish-eye camera and the superimposed detected markers are shown. The green circle represents the manually defined area of interest, so if a blob is detected outside of it, it is ignored (sometimes the borders of the image tend to create problems), while the green lines mark the center of the camera image, allowing a better manual alignment. The Blob Detection algorithm is fairly robust even when high deformations are involved; on the other hand it considers blobs as circles and not ellipses. This aspect is highlighted and properly managed by [57] where the diameter of the circular marker is com-

puted as twice the semi-major axis of the ellipse shown in the camera image. The semi-major and semi-minor axes are computed using the ellipse fitting algorithm proposed by Fitzgibbon et al., also integrated in the *cv2.fitEllipse()* function. In our work this aspect was investigated, even though our markers' size is about one order of magnitude smaller than the Universal Gripper's, resulting in a less pronounced shape transition from circle to ellipse.

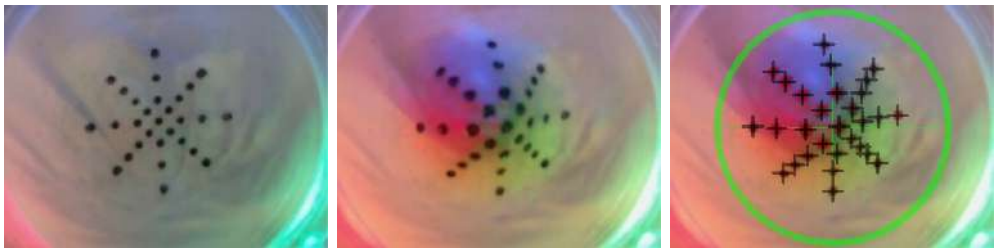

Figure 3.20: *From left to right: the raw frame when no forces are applied; the raw frame after applying a force; the frame after marker detection.*

Looking at Figure 3.21 we can see that after applying a normal force of around 4N, the markers' shape tend to be slightly elliptic depending on the application point and marker location, but the standard Blob Detection algorithm returns circles that approximate their shape. The central image shows the Blob Detection's output, highlighting in white the circle and the center with black crosses. The image on the right shows the fitted ellipses that can't be obtained directly from the Simple Blob Detector (even though one could exploit circularity and inertia filters to retrieve the amount of deformation). In fact, a simple algorithm was developed and can be summarized in the following steps:

- convert the input image to gray-scale (*cv2.cvtColor()* function);
- blur the gray-scale image with a 3×3 filter (*cv2.blur()* function);
- detect edges using the Canny algorithm (*cv2.Canny()* function);

- find contours on a binary image (*cv2.findContours(canny_output, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)* function);
- for each contour (that represents the marker's border) if it consists of at least 5 points, fit an ellipse using the Fitzgibbon95's algorithm implemented in the *cv2.fitEllipse()* function;
- return the ellipses' center and major-axis.

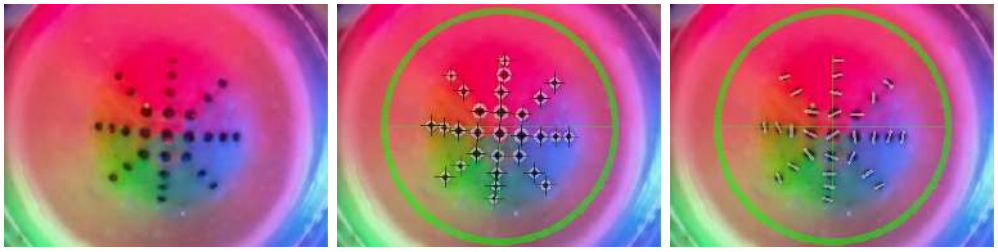

Figure 3.21: *Left: the raw frame when a normal displacement of 10mm is caused by the external force. Center: the frame after marker detection, fitting ellipses. Right: the frame after marker detection fitting circles.*

Figure 3.22 shows the horizontal, vertical and radius displacements of the detected circular markers in pixel units, while Figure 3.23 refers to the detected elliptic markers. This results come from one of the measured samples during the data acquisition phase, in particular the centered normal force from rest position to a maximum Z-deformation of around 10mm, corresponding to about 4N of vertical force.

As can be seen by the two graphs, the sensed displacements are pretty similar but the radii measurements become even more noisy when considering markers as ellipses. Considering the not so straightforward ellipse fitting algorithm that includes several thresholding steps, the rest of the work uses the standard Blob Detection's output that is accurate enough for our case study.

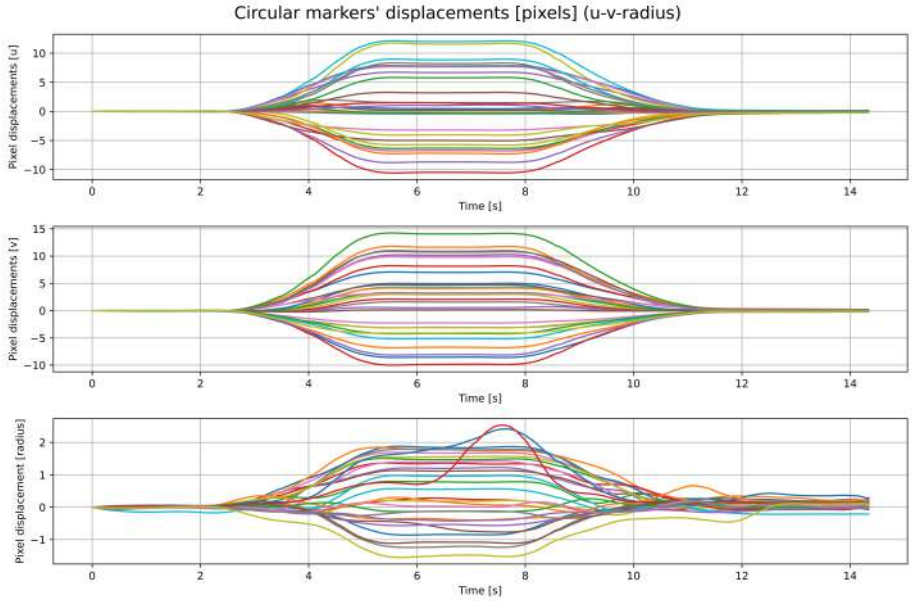

Figure 3.22: *Circular markers' displacements (horizontal, vertical and radius) in pixel units. Each of the 29 markers is represented by a different hue.*

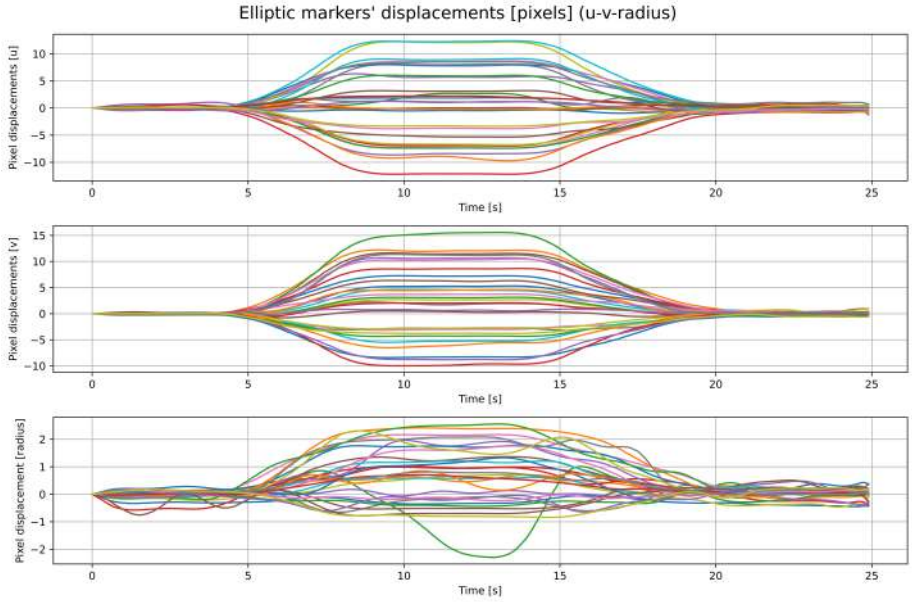

Figure 3.23: *Elliptic markers' displacements (horizontal, vertical and radius) in pixel units.*

The main reason why this method was implemented is because the 3D reconstruction formulas returned sometimes noisy and non-realistic results; unfortunately considering ellipses' centers and widths didn't improve the outcomes by much.

Once the markers are properly detected, ***tracking*** comes into play. The tracking approach is fairly simple: after an initialization step that is passed if all the 29 markers are detected, for every received frame, marker detection is applied and the new coordinates are associated to the closer marker of the previous frame. Using as a metric to sort markers the Euclidean distance, we found the approach robust enough: the frame rate is about *20fps*, allowing it to correctly perform tracking. In order to make the algorithm a bit more reliable during the offline dataset creation, a *Boolean interpolation matrix* has been created. In each row of this matrix there are 29 Boolean instances that indicate the tracking state ("True" means currently tracking; "False" means tracking was lost) of each marker corresponding to the considered frame. Once all the states are collected, the DataFrame data structure (*pandas* library) is exploited to perform linear interpolation of the u, v and $radius$ measurements, when tracking was lost (a code snippet is shown in Listing 3.2).

```
import numpy as np
import pandas as pd
# [...]
# Linear interpolation where interpolation matrix value is True
for it, interp_arr in enumerate(interpolation_matrix):
  interp_arr = np.array(interp_arr)
  ind = list(np.where(interp_arr == True)[0])
  if len(ind) > 0:
    for marker_id in ind:
      all_traj_markers[it][marker_id] = [np.nan,np.nan,np.nan]

```

```
# Interpolation step: if "np.nan", means tracking was lost,
# so interpolate linearly between positions
trajs = np.array(all_traj_markers)
for marker_id in range(num_markers_gripper):
  interp_df = pd.DataFrame(trajs[:,marker_id,:]).interpolate()
  for it, el in enumerate(interp_df.values.tolist()):
    all_traj_markers[it][marker_id] = el
# [...]
```

Listing 3.2: *Python snippet of how linear interpolation is performed when tracking is lost.*

Figure 3.24 shows a visual representation of the markers being independently tracked (subsequent coordinates of the same marker are scattered with the same color) and sorted (the sorting algorithm will be discussed in Subsection 3.4.4); while Figure 3.25 shows the effects of the same deformation directly on the frame sent by Raspberry Pi.

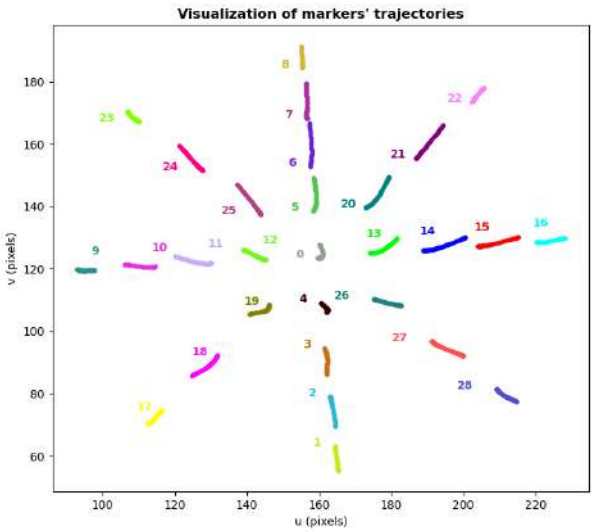

Figure 3.24: *Plot of the subsequent markers' coordinates in pixel units during deformation.*

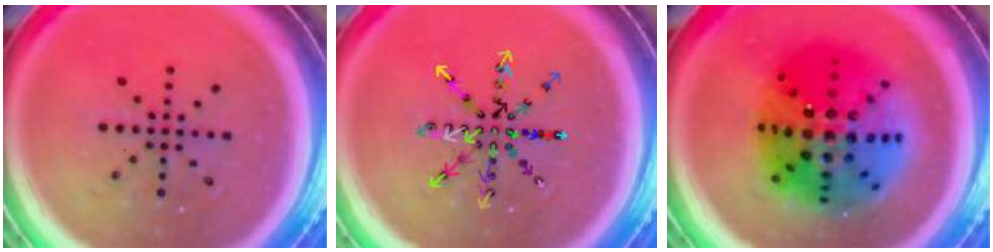

Figure 3.25: *From left to right: the raw frame in rest position; the superimposed arrows that show the displacement's direction of each marker (the arrows were lengthened by a factor of 8 to make them more visible); the raw frame under a force of around 4N.*

3.4.2 From pixel to metric units with a monocular setup

We believe that it's worth mentioning all the tried approaches to estimate the markers' 3D coordinates, because at first we wanted to exploit those results for force estimation. Despite the efforts, in the end we decided to directly train the force estimation models on the coordinates expressed in pixel units, being them far more reliable.

As discussed in Subsection 2.3.2, most of the proposed methods to reconstruct the grasped object's (or dome's) 3D shape that don't imply RGBD cameras, rely on Neural Networks or on a preliminary training process. In fact, obtaining depth information with a single camera (monocular setup) is not a trivial task. In our work we tried to reproduce the results obtained in [57] using the suggested formulas. In particular 3D marker position estimation is performed as:

$$X = \frac{(u - c_x)O_{mm}}{O_{px}} \tag{3.1}$$

$$Y = \frac{(c_y - v)O_{mm}}{O_{px}} \tag{3.2}$$

$$Z = \sqrt{\frac{(f_x O_{mm} I_{px})}{O_{px} I_{mm}} - X^2} \tag{3.3}$$

where (X, Y, Z) are the 3D estimated coordinate of the marker; (u, v) are

the coordinates of the marker in pixel units; (c_x, c_y) are the principal point coordinates (obtained through camera calibration), usually at the image center; (f_x, f_y) are focal lengths in metric units; O_{mm} is the marker's width in metric units, while O_{px} is the marker's width in pixel units, both corresponding to the ellipse's major-axis; I_{mm} is the image sensor width in metric units while I_{px} is the image sensor width in pixel units. Figure 3.26, [57], reports a visual representation of the markers' width and the resulting 3D shape estimation.

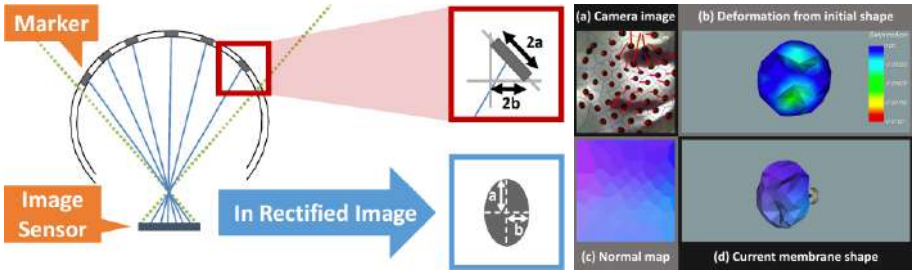

Figure 3.26: *On the left a representation of how circular markers embedded in the membrane are searched as ellipses; on the right the developed viewer tool illustrating the gripper's deformation [57].*

First of all, we focused on this approach. After applying the standard Computer Vision calibration process with a 9×6 chessboard and the use of OpenCV, the camera matrix and distortion parameters were computed. After that, the pixel to meters conversion takes place to retrieve the 3D position of the 29 markers. In this regard, we tried to exploit both the ***circular*** and ***elliptic*** marker detections to compare the results and look for improvements. From a qualitative point of view, the retrieved 3D coordinates are reasonable but they don't measure up to the CAD model.

As can be noticed by looking at Figure 3.27, when the sensor is in its rest position, some of the central markers are actually lower than the external ones. To better understand the graphs: in Figure 3.27 and in the left image of Figure 3.28 the cyan coordinates represent the 3D ground truths obtained from

the CAD model in rest position; the green and red coordinates were computed using the Equations (3.1) - (3.3) and they respectively refer to the rest positions (that should coincide with cyan coordinates) and deformed positions (for which we don't have a reliable ground truth). Moreover, Figure 3.28 shows a 3D mesh representing the deformed dome, that was obtained performing a Boolean difference between a sphere of the same diameter of the dome and the 3D triangulated volume from the markers' coordinates after deformation, using the *pyvista "delaunay_3d()"* function.

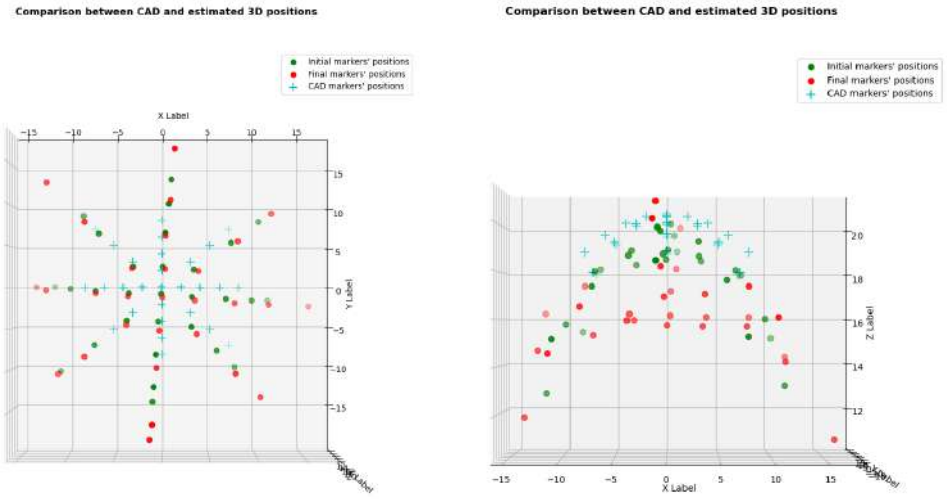

Figure 3.27: *From left to right: top and front views of the obtained 3D coordinates of the markers detected as* ***circles****, using the formulas described in [57].*

Also, this method heavily relies on camera calibration, image rectification and a correct camera placement that ensures the validity of the geometric assumptions. In fact, with the initial temporary setup, we experimented that moving and tilting the camera in different directions would drastically change the obtained 3D coordinates.

Trying to improve the 3D estimation, that could open up the possibility of performing object in-hand localization, classification and surface estimation,

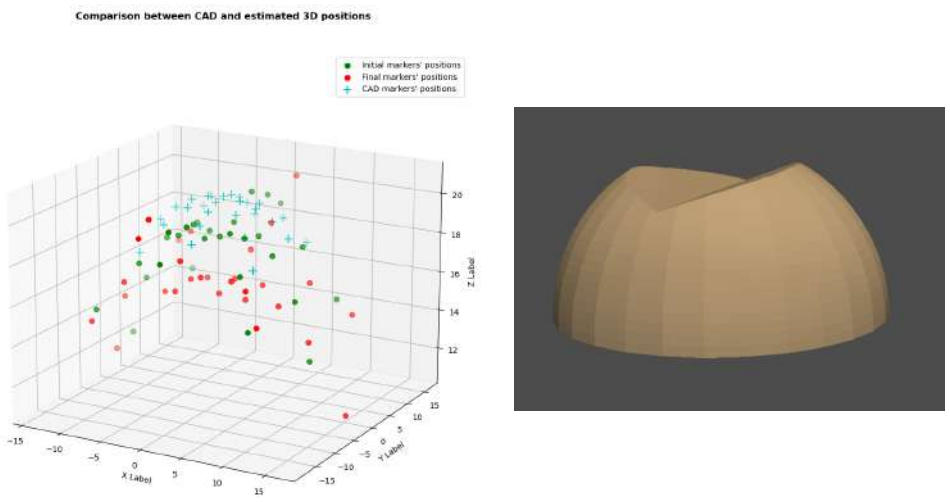

Figure 3.28: *Left: side view of the obtained 3D coordinates of the markers detected as* ***circles****, using the formulas described in [57]. Right:3D mesh obtained using the "pyvista" library.*

we tried to exploit the discussed ***ellipse fitting algorithm***. So, each marker is represented by the (u, v) coordinates that define the ellipse's center and by twice the ellipse's semi-major axis in place of the circle's radius. As shown in Figure 3.29, this approach leads to slightly better results. Moreover, Figure 3.30 shows a comparison of the obtained 3D coordinates between fitting circles and ellipses.

Unlike the previous two, ***another approach to estimate 3D coordinates*** that was developed from scratch is presented. It relies on the measured ***circles' radii*** and on the 3D initial position of the markers extracted from the ***CAD model***. In fact, despite being fairly noisy, the radius of each marker is the only "measurement" of depth that can be retrieved. So, the idea is to estimate the 3D position of each marker through a linear (for X and Y) or inverse (for Z) proportionality between the CAD coordinates and the $(\Delta u, \Delta v, \Delta radius)$ displacements. Listing 3.3 shows the Python code used to

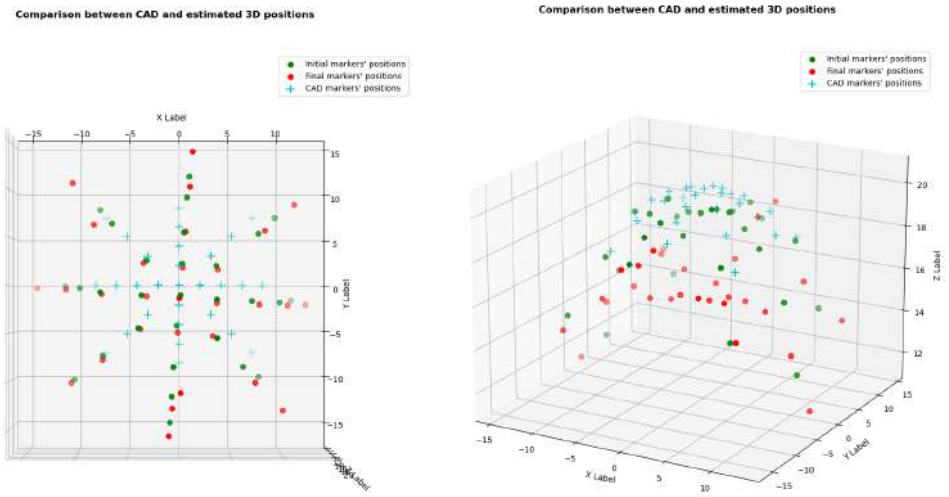

Figure 3.29: *Top and front views of the obtained 3D coordinates of the markers detected as* ***ellipses****, using the formulas described in [57].*

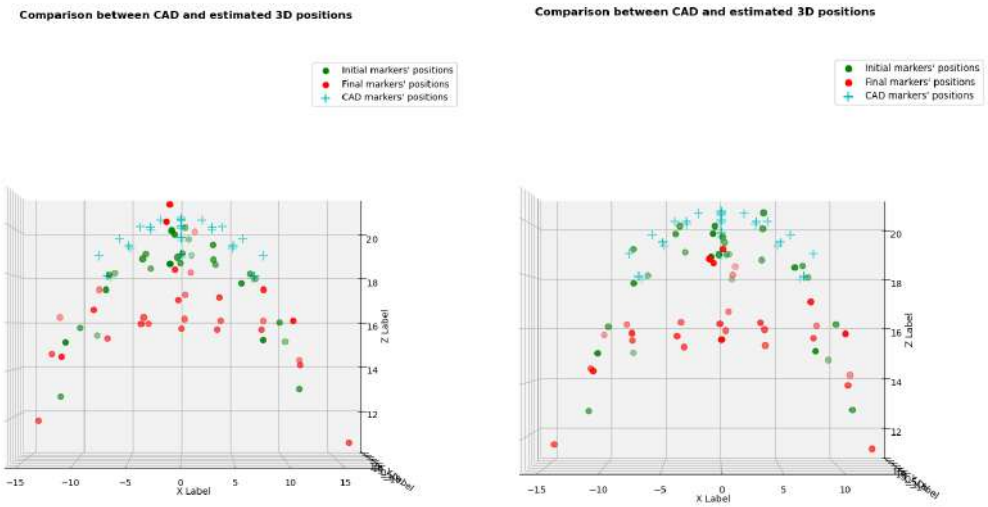

Figure 3.30: *On the left a front view of the obtained 3D coordinates of the markers detected as* ***circles****, on the right a front view of the obtained 3D coordinates of the markers detected as* ***ellipses****.*

perform the 3D estimation. As clearly stated by the comments, there are some manually defined normalizing factors that make the estimation reasonable, but to be automatically computed, 3D ground truths of the deformed dome would be required. Regarding the "*Z_reduction*" array, the boundaries were set considering the normal displacement that was acquired during the dataset creation with a 3-axis robot. So, having recorded a maximum deformation of negative 10mm, we accordingly set the boundaries so that the Z-coordinate estimation would be correct for the central marker, then we assumed that a linearly distributed normalizing factor could be suitable for the other markers. Also, the normalizing factors would need to be kept consistent even with less pronounced deformations. On the other hand, the "*norm_factor_u_v*" normalizing factor was manually defined so that the X and Y estimations would be reasonable even though (X, Y) ground truths are missing. Future developments could retrieve ground truths 3D coordinates simulating force application with a software such as SOLIDWORKS [67].

```
# Normalize radius displacements between -1 and 1
displacements_px_c = final_markers_c - initial_markers_c
norm_radius_disp = displacements_px_c[:,2].copy()
norm_radius_disp /= max(abs(norm_radius_disp))

# Manually define X,Y,Z reduction factors
num_steps = 30
Z_reduction = np.linspace(1, 0.6, num_steps)
disp_steps  = np.linspace(0.1, 1, num_steps)
norm_factor_u_v = 40

# Compute the final 3D markers' coordinates
final_markers_3d = []
for m_id,(rad_disp,mark) in enumerate(zip(norm_radius_disp,
                                          final_markers_c)):
```

```
16  u,v,radius = mark
17  X,Y,Z = markers_XYZ[m_id,:] # get ground truths from CAD
18  u_disp,v_disp = displacements_px_c[m_id,0:2]/norm_factor_u_v
19    for k, factor in enumerate(disp_steps):
20      if rad_disp < 0.3 and [X,Y,Z] not in final_markers_3d:
21        final_markers_3d.append([X*(1+u_disp),Y*(1+v_disp),Z])
22        break
23      elif 0.3 <= rad_disp <= factor:
24        final_markers_3d.append([X*(1+u_disp),Y*(1+v_disp),
25                                 Z*(Z_reduction[k])])
26        break
```

Listing 3.3: *Python snippet of the proposed 3D markers' positions estimation*

Figure 3.32 shows the obtained 3D coordinates with the proposed method, while Figure 3.31 shows a heatmap representing radius displacement (from black to yellow) superimposed on the deformed frame, on which is based the Z-coordinate estimation. Finally, Figure 3.33 presents a 3D visual representation of the deformed dome obtained through the *pyvista* library.

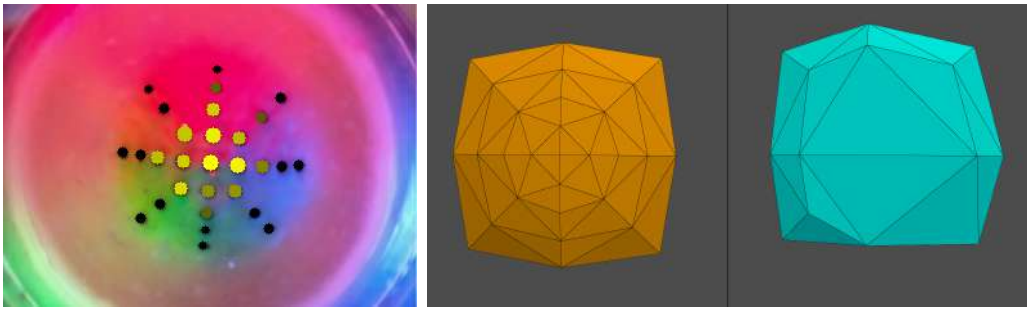

Figure 3.31: *On the left a radius displacement heatmap superimposed on the deformed frame; on the right the before (coinciding with the CAD ground truths) and after deformation meshes obtained triangulating the 3D markers' coordinates.*

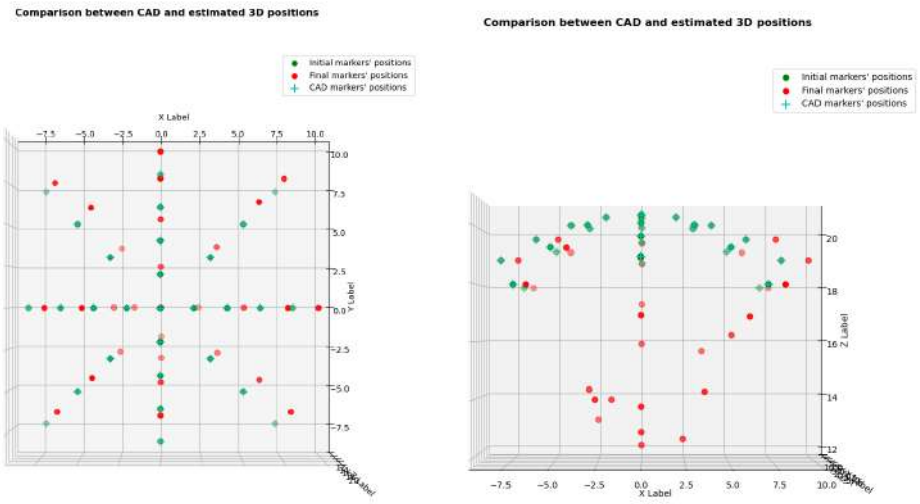

Figure 3.32: *Top and front views of the 3D markers' coordinates obtained with the* ***proposed method****.*

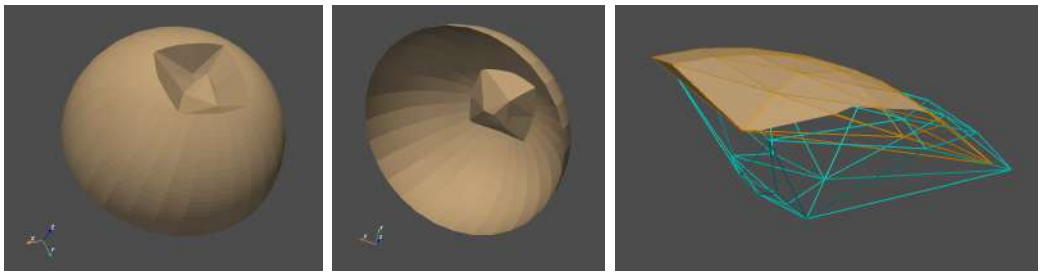

Figure 3.33: *From left to right: a top and a bottom view of the 3D mesh obtained through boolean difference between dome and triangulated markers' coordinates; a side view of the 3D mesh obtained through boolean difference between rest position and deformed position triangulated markers' coordinates.*

3.4.3 Raw data acquisition for sensor calibration

To properly calibrate the sensing device, ground truth forces and tracking data have been acquired. In particular, as mentioned in Subsection 3.3.2 an ATI force/torque sensor was used to measure ground truth forces while pressing against the dome's surface. To log the force measurements we used ROS (Robot Operating System), so they were saved inside the so called "rosbags". After positioning and centering the robot above the dome and resetting the sensors' bias caused by its orientation, we collected force measurements and raw frames slowly moving down the Z-axis of the robot to generate pressure on the dome. In this fashion we acquired data from 9 different experiments ranging from 3mm to 12mm (3,4,5,6,7,8,9,10,12 mm) of vertical displacement, generating a normal force approximately centered on the central marker. Figure 3.34 shows the force and torque components measured by the ATI Nano sensor.

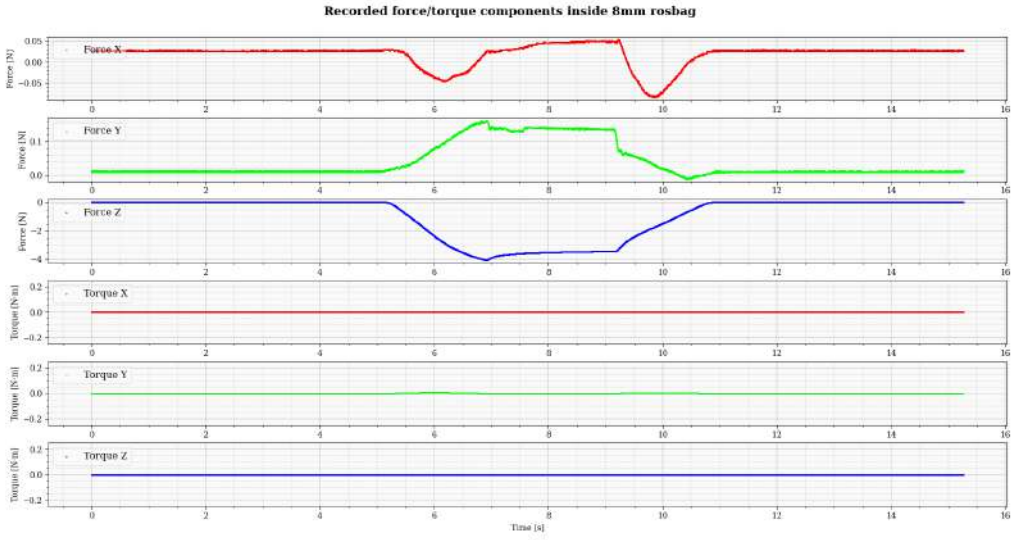

Figure 3.34: *Force and torque components measured by the ATI Nano sensor.*

3.4.4 Offline semi-automated dataset creation pipeline

After collecting all the images and force measurements during the data acquisition process, the raw data has to be pre-processed and properly stored. In particular, after organizing in a proper folder structure the raw data, a ***semi-automated offline pipeline*** was developed in order to perform the following tasks:

1. automatic offline detection and tracking of the markers, starting from the pre-collected images during each trial. The low-pass filtered and interpolated (if needed) tracking data (timestamp, marker trajectory, marker displacement) is than written on a .json file;

2. semi-automatic synchronization between ground truth forces and marker displacements. The synchronized ground truths and displacements are written into two separate .json files, while the corresponding images are correctly placed inside each folder.

Starting from the ***first task***, the raw data was stored and organized in the following manner:

```
raw_data
├── 3mm
│   ├── out_img
│   │   ├── *_1frame.png
│   │   ├── *_2frame.png
│   │   └── *_3frame.png
│   └── *.bag
├── 4mm
│   ├── out_img
│   │   └── ..
│   └── *.bag
└── ..
```

Once the code is executed, all the folders contained inside “raw_data” are browsed one by one. If a “.bag” file and the “out_img” folder are found,

the stored frames are read based on the index order, then marker detection and tracking is performed. After having tracked the 29 markers, they are linearly interpolated as explained in Subsection 3.4.1 and filtered with a *butter low-pass filter* implemented in the *scipy.signal* library. It's worth mentioning that the detected markers are not sorted between different trials, meaning that "marker 1" of experiment "3mm" could correspond to "marker 5" of experiment "4mm".

Next, the ***second task*** comes into play. Due to the absence of a unique triggering signal to synchronize the ATI measurements with the images or online tracking data during acquisition, a semi-automatic and fast synchronization step was introduced. After the first step, Figure 3.35 pops up and the user is asked to click on the figure six times. In particular, the user needs to specify the start and end of the ground truth portion (2 clicks on the first subplot) and only then the boundaries of the corresponding markers' displacements (2 clicks on the second subplot). This process has to be done 3 times, in order to automatically retrieve synchronized pixel displacements and trajectories, force ground truths and raw images. In fact, after six clicks, the three obtained segments are shown to the user that needs to supervise the results (this pipeline could be easily automated but we think that this task is crucial enough to require human supervision). Figures 3.36, 3.37 and 3.38 show how the first, second an third segments present after synchronization.

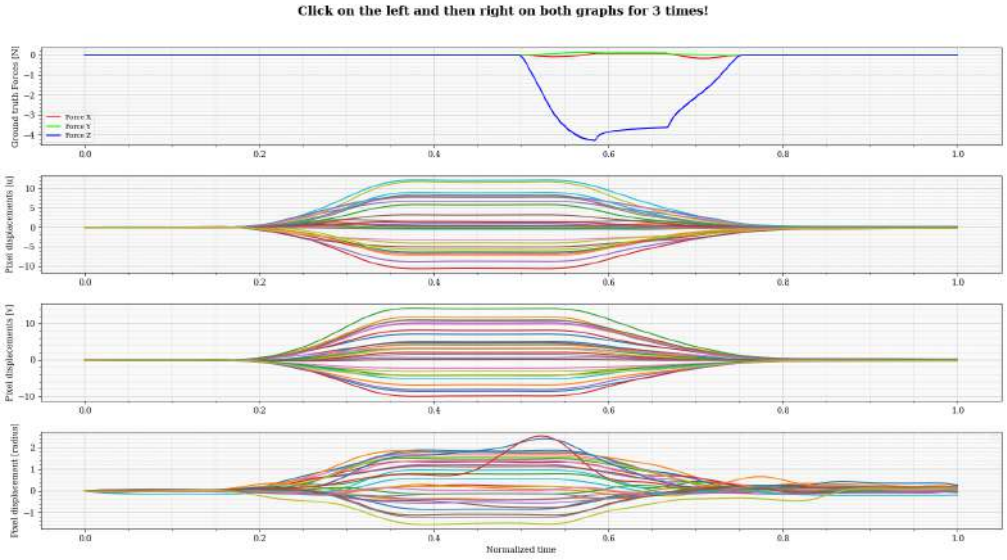

Figure 3.35: *Figure showing from top to bottom: the 3 force components' ground truths, the u, v and radius displacements of each marker.*

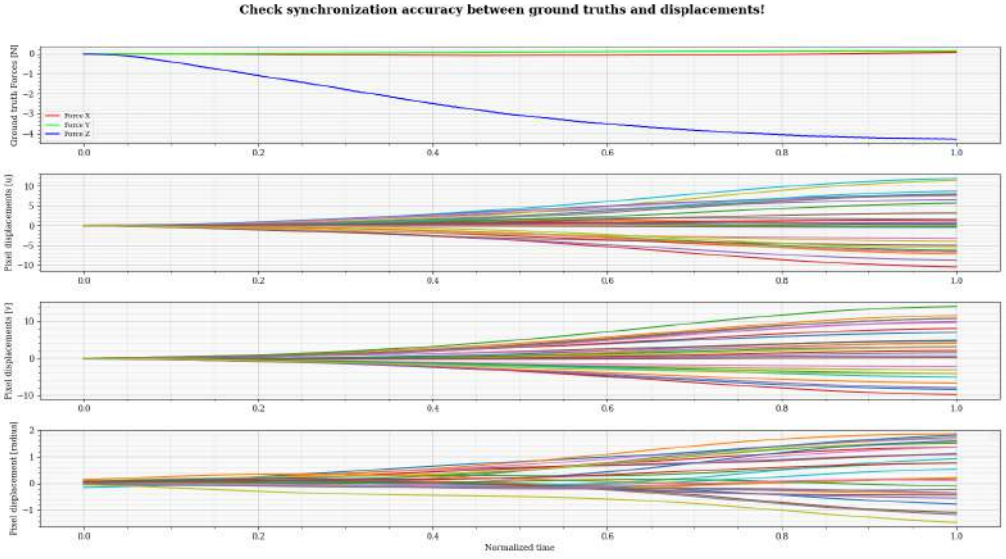

Figure 3.36: *Plot showing the first segment of the synchronized ground truths and markers' displacements.*

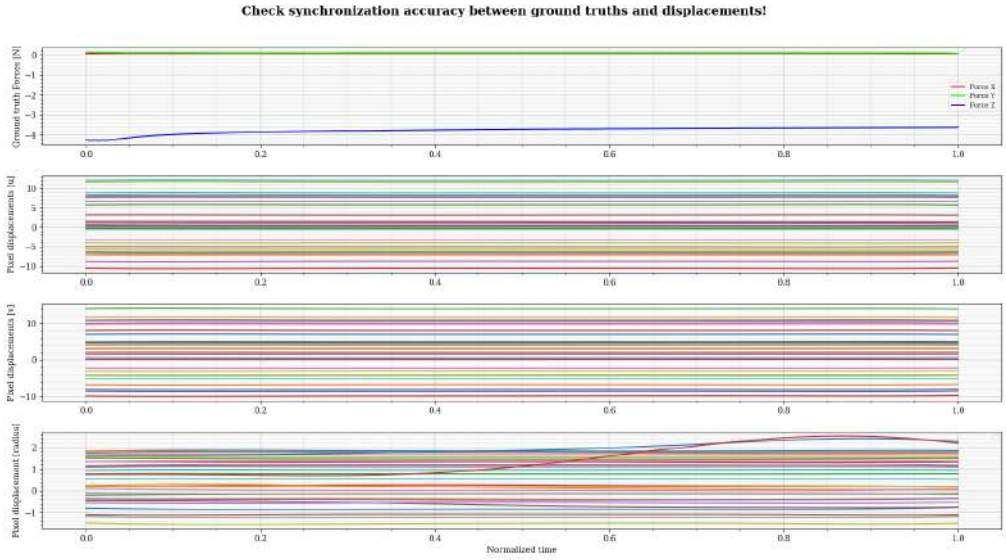

Figure 3.37: *Plot showing the second segment of the synchronized ground truths and markers' displacements.*

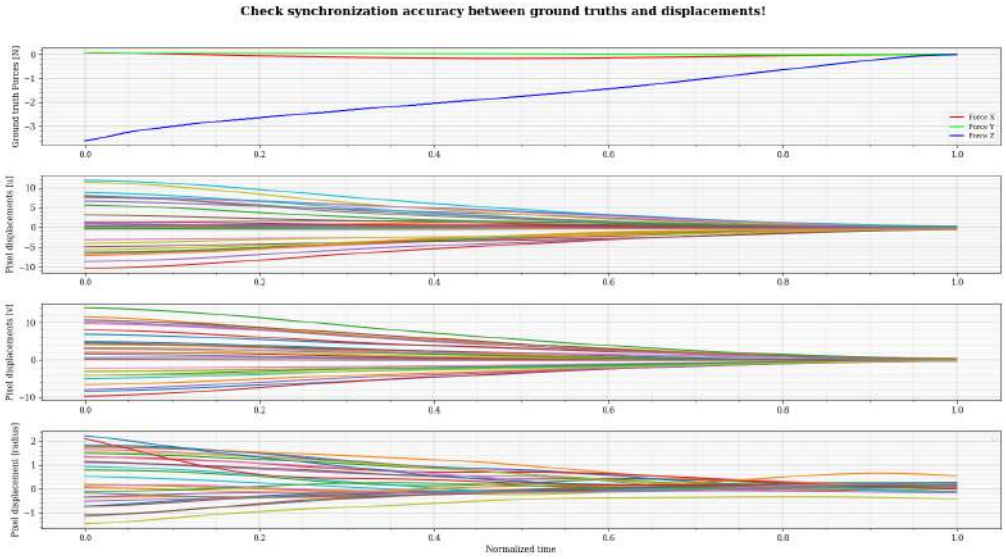

Figure 3.38: *Plot showing the third segment of the synchronized ground truths and markers' displacements.*

Once all the data stored in the "raw_data" folder has been synchronized, the "clean_data" folder contains:

```
clean_data
├── 3mm_synchronized_partial1
│   ├── synchronized_rgb
│   │   ├── *_click1frame.png
│   │   ├── *_click1 +1frame.png
│   │   ├── ..
│   │   └── *_click2frame.png
│   ├── synchronized_pixel.json
│   └── synchronized_rosbag.json
├── 3mm_synchronized_partial2
│   ├── synchronized_rgb
│   │   ├── *_click3frame.png
│   │   ├── *_click3 +1frame.png
│   │   ├── ..
│   │   └── *_click4frame.png
├── 3mm_synchronized_partial3
│   ├── synchronized_pixel.json
│   ├── synchronized_rosbag.json
│   ├── synchronized_rgb
│   │   ├── *_click5frame.png
│   │   ├── *_click5 +1frame.png
│   │   ├── ..
│   │   └── *_click6frame.png
└── ..
```

To make the mentioned folder representation clear, *click1* refers to the index of the image that corresponds to the associated ground truth and displacements values. Also, ***time normalization*** is performed due to the lack of a common trigger and ***ground truths' downsampling*** is carried out because the ATI Nano sensor's sampling frequency is far greater than the received frames per second (since the marker tracking process is performed offline, the number of displacements - or markers' coordinates - corresponds to the number of acquired images). Some experiments were conducted to see if introducing a linear upsampling of the displacements would lead to better results, thanks to

a higher number of training samples. In some cases this approach seemed to worsen the force estimates, maybe due to a linearization of highly non linear data, so it was not adopted to create the final dataset.

Once all the raw data was cleaned and synchronized, the dataset has been split between 80% train and 20% test both manually and using the "*sklearn.model_selection.train_test_split()*" function. Also, in addition to the mentioned trials, some dynamic tests were manually performed and synchronized to check the models' robustness and ability to generalize.

Chapter 4

Experimental results

In this Chapter all the utilized force estimation approaches are presented, briefly explaining the exploited algorithms, ranging from Linear approximation to Machine Learning and Deep Learning techniques. After that, the achieved results are compared based on the Mean Squared Error metric.

4.1 Force estimation approaches

Several approaches were exploited and compared to perform force estimation, ranging from simple linear approximations to Machine Learning and Deep Learning algorithms. In particular, we used:

- a linear elastic ***force approximation*** based on [78];
- a ***non-linear*** and more sophisticated ***variant*** of [78];
- a properly tuned ***Linear Regression*** model implemented by *sklearn* [60]
- a properly tuned ***K-Neighbors Regressor*** model implemented by *sklearn* [59]

- a properly tuned ***Support Vector Regression*** model implemented by *sklearn* [62]
- a ***Neural Network Sequential*** model implemented through *keras* [37]
- a ***Deep Convolutional Neural Network*** model implemented through *keras* [36]

4.1.1 Machine Learning vs Deep Learning

In this Subsection as brief introduction to the Machine Learning pipeline is presented. Then, we discuss the main differences between Machine Learning and Deep Learning approaches, rapidly explaining what a Neural Network is.

- ***Data preparation***: collection, "cleaning" and pre-processing of the required input data (images or numerical data);
- ***Feature extraction***: retrieving more manageable information that still describes the raw data and is suitable for modelling;
- ***Feature selection***: reducing the features' space dimensionality, keeping only the most relevant features to train the model. Feature selection is a prior knowledge in Machine Learning approaches;
- ***Model selection***: choosing a statistical model and tuning its hyperparameters to solve the regression problem. Model selection is also a prior knowledge in Machine Learning approaches;
- ***Model training***: process that uses the training dataset to build a model that will be able to classify new samples, belonging to the test set;
- ***Prediction***: model output decision based on its acquired knowledge;

- ***Model testing***: testing the previously trained model with unseen input data to check its performance. The aim of model training is building a general model that can classify new samples, avoiding overfitting the train set.

The main difference between *Machine Learning* and *Deep Learning* approaches is the prior knowledge. In fact, if Neural Networks are used, not only feature selection and model selection, but also feature extraction and prediction processes are considerable prior knowledge. Features are constantly learned during model training, through layers of Convolutional filters, core of CNN architectures. Another important aspect that we considered when testing the real-time implementation was the required prediction time. In fact, Machine Learning algorithms tend to be faster to train and evaluate than Deep Learning architectures. Figure 4.1 schematizes the differences between Machine Learning and Deep Learning, while Figure 4.2 shows a typical Deep-CNN structure.

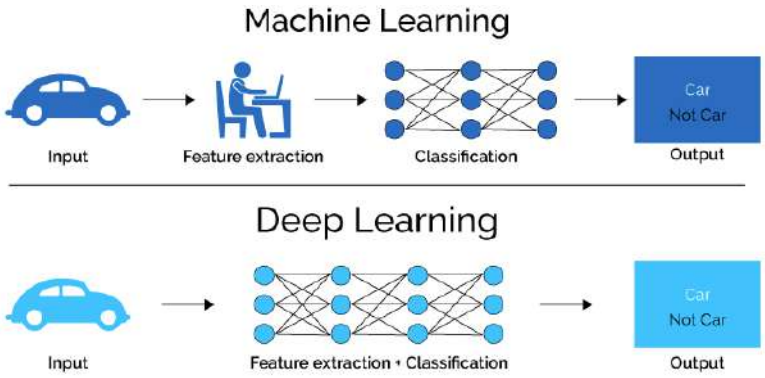

Figure 4.1: *The main differences between Machine Learning and Deep Learning and an example of Deep-CNN [42].*

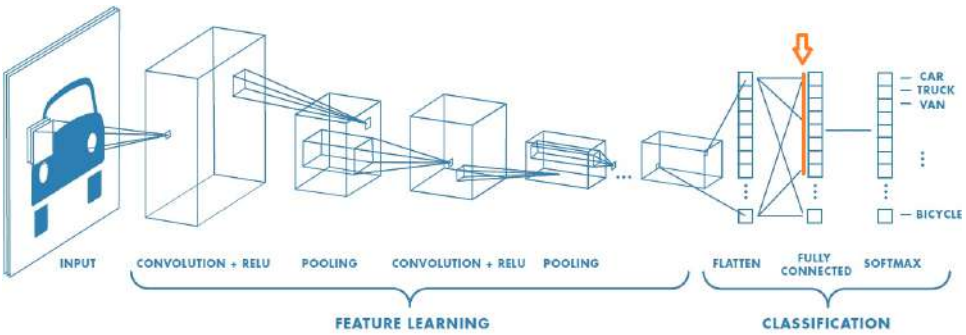

Figure 4.2: *The typical architecture of Deep-CNN [71].*

4.1.2 Linear estimation

The first method we focused on was a linear approximation based on the elastic force formulation, discussed in "*Implementing Tactile Behaviors Using FingerVision*" [78] and "*FingerVision for Tactile Behaviors, Manipulation, and Haptic Feedback Teleoperation*" [77]. Figure 4.3 extracted from paper [78] shows the detected markers' trajectories when the almost planar surface is deformed.

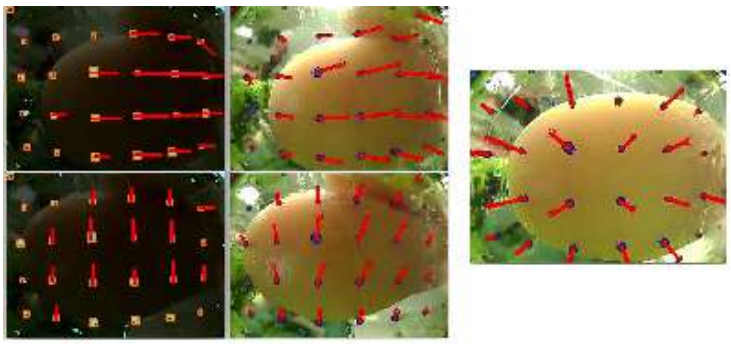

Figure 4.3: *On the left the detected markers' movement using SimpleBlobDetector; on the right an example of marker movements when a normal force is applied [78].*

Let (d_x, d_y) be the horizontal displacement of each marker from its initial position and let (c_x, c_y, c_z) be the constant elastic coefficients, a force estimate

applied at each marker is given by

$$[f_x, f_y, f_z] = [c_x d_x, c_y d_y, c_z \sqrt{d_x^2 + d_y^2}]. \tag{4.1}$$

So, the overall force estimates of the sensing device are defined as average of the single forces:

$$[F_x, F_y, F_z] = [\frac{1}{29} \sum_{i=0}^{28} f_x, \frac{1}{29} \sum_{i=0}^{28} f_y, \frac{1}{29} \sum_{i=0}^{28} f_z] \tag{4.2}$$

This approach assumes that every marker has the same impact on the force estimates and that there's a linear relation between horizontal/vertical displacements and forces. Regarding the normal force estimation, they use the norm of markers' position change being it more stable and less affected by noise than the radius reading of the Blob Detector. As mentioned in Chapter 2, this approximation is really put to the test in our case study, being the surface hemispherical and not planar like in [78].

In this case the model is fairly simple, being it linear and composed by 3 constant parameters. To retrieve the average stiffness coefficients along the 3 main directions, we used the following approach (a code snippet is shown in Listing 4.1):

- consider the train dataset's (F_x, F_y, F_z) ground truths and pixel horizontal/vertical displacements;
- if the ground truths are greater than a threshold, compute the three stiffness coefficients for each of the 29 markers by inverting Equation 4.1 (avoiding the division-by-zero exception);
- once the previous steps are repeated over all the training samples (among the several recorded experiments), the final (C_x, C_y, C_z) coefficients are computed as the median (instead of the mean) of all the collected values.

```
# [...]
# Avoid having high C_hat coeffs. because of
# near to zero displacements
th_from_zero = 0.1
if dx == 0:
  dx = th_from_zero
elif abs(dx) < th_from_zero:
  dx = np.sign(dx) * th_from_zero
if dy == 0:
  dy = th_from_zero
elif abs(dy) < th_from_zero:
  dy = np.sign(dy) * th_from_zero
# [...]
# Linear formulas from the paper:
# compute coeffs. only if the ground truth is meaningful
force_th = 0.1
if abs(force_gt_x) > force_th:
  Cx_hat_final.append( force_gt_x/(dx) )
if abs(force_gt_y) > force_th:
  Cy_hat_final.append( force_gt_y/(dy) )
if abs(force_gt_z) > force_th:
  Cz_hat_final.append( force_gt_z/(np.sqrt(dx**2 + dy**2)) )
# [...]
# Compute the final stiffness coefficients considering
# the median value
Cx_hat_final = statistics.median(Cx_hat_final)
Cy_hat_final = statistics.median(Cy_hat_final)
Cz_hat_final = statistics.median(Cz_hat_final)
```

Listing 4.1: *Python snippet of the stiffness coefficients' estimation as suggested by [78]*

This linear model has the advantage of being easy to understand, implement and improve by complicating the force/displacement relations. In par-

ticular, we decided to compute the 3 stiffness coefficients using the median operator instead of the mean, being it more robust to outliers if data is not normally distributed, as graphically shown in Figure 4.4 [45].

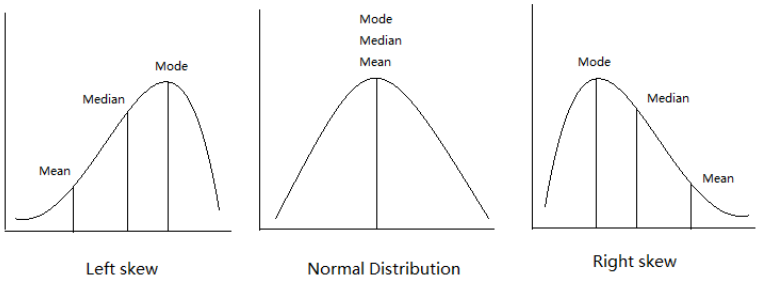

Figure 4.4: *Mode, mean and median in 3 different data distribution scenarios [45].*

To further motivate this decision, Figure 4.5 represents a histogram of the sum of the stiffness coefficients computed during each iteration. All these numbers are stored and, in the end, the 3 final coefficients are retrieved by either selecting the mean, mode or median value. Considering that the created dataset is balanced and wide regarding normal forces but pretty limited in the X and Y components' range, the $\sum C_x$ and $\sum C_y$ coefficients seem normally distributed but the $\sum C_z$ are fairly skewed.

After the training process, the 3 constant coefficients are estimated and testing can be done directly applying Equation 4.1 and Equation 4.2 together, to retrieve the total force components.

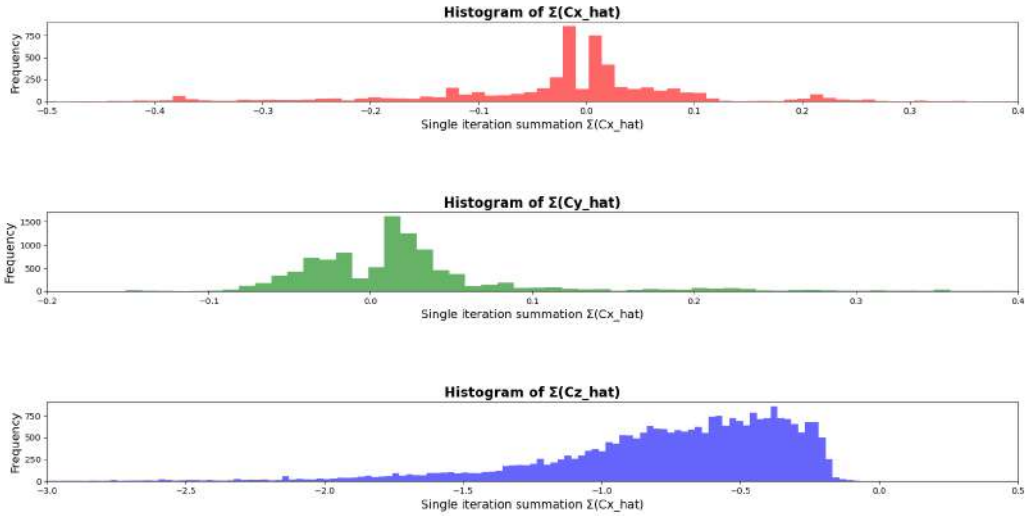

Figure 4.5: *Histograms of the computed* $\sum C_x$, $\sum C_y$, $\sum C_z$ *coefficients during each iteration.*

4.1.3 Non-linearly compensated and marker-location based estimation

As mentioned, this method takes inspiration from the previous one but tries to improve some of its limitations due to the non-planar shape of our dome. In fact, it's more feasible to approximate a quasi-planar surface to a linear elastic behavior than a hemispherical surface. Another assumption is that every marker, meaning every region of the material, behaves the same in terms of stiffness, deformation and elastic properties. Due to the several steps that are required to manufacture the elastic dome, we experienced that many variables can affect the model's stiffness and therefore it's response to external forces. We think that the previous method can rapidly give a good indication of how stiff the material is, but could also lack of generalization and robustness capabilities. For these reasons, we tried to improve this method by:

- considering the 29 markers independent of each other;

- non-equally distributing the ground truth forces;
- creating a 29×3 stiffness matrix (instead of a 1×3 stiffness vector) so that every marker has an elastic coefficient for every direction;
- applying force estimation using the same equations as before (Equations 4.1 and 4.2) but using the estimated stiffness corresponding to the $k-th$ marker.

How the 29 markers are properly sorted is explained in Subsection 4.1.9, for now let's assume that the input marker trajectories are sorted as shown in Figure 4.6.

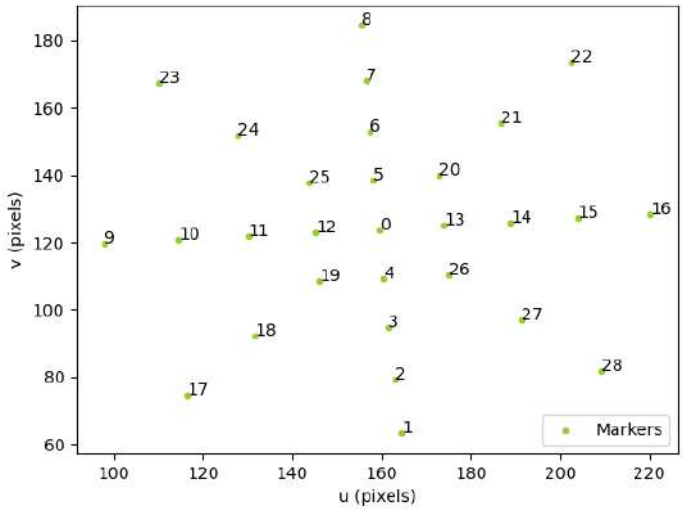

Figure 4.6: *Sorting order of the 29 fiducial markers.*

As Listing 4.2 shows, the ground truth forces are compensated with coefficients that were tuned based on the outcomes. In particular, the logic behind their tuning is based on the percentage of the force that should be "absorbed" by the region around the considered marker. Assuming that a normal force is applied to the center of the dome, the central marker should be still no matter

the intensity of the normal force, while on the farthest markers should be exterted the least amount of force. If needed, this reasoning could be generalized estimating the application point of the external force.

```
Cx_hat, Cy_hat, Cz_hat = [],[],[]
for k_it in range(num_iterations):
  # Ground truth forces corresponding to the k-th iteration
  force_gt_x, force_gt_y, force_gt_z = train_labels[k_it, :]

Cx_hat_it, Cy_hat_it, Cz_hat_it = [], [], []
for marker_id in range(num_markers):
  dx = all_dx[k_it, marker_id]
  dy = all_dy[k_it, marker_id]
  #-------------------------------------#
  if marker_id == 0:  # central marker
    compensation_fact = 0
  elif marker_id in [4, 5, 12, 13]:
    compensation_fact = 0.2/4
  elif marker_id in [19, 20, 25, 26]:
    compensation_fact = 0.3/4
  elif marker_id in [3, 6, 11, 14]:
    compensation_fact = 0.2/4
  elif marker_id in [2, 7, 18, 27, 21, 24, 10, 15]:
    compensation_fact = 0.15/8
  elif marker_id in [1, 8, 9, 16, 17, 22, 23, 28]:
    compensation_fact = 0.15/8
  force_gt_x_new = force_gt_x * compensation_fact
  force_gt_y_new = force_gt_y * compensation_fact
  force_gt_z_new = force_gt_z * compensation_fact
```

Listing 4.2: *Python snippet of the proposed ground truths' compensation based on the marker location*

On this subject, a simple estimation of the force's application point is pro-

posed (but not integrated with the mentioned ground truths' compensation). To retrieve the force's application point, the idea is to compute a weighted average of the initial markers' coordinates, depending on their radius increment. This way, we consider the correlation between size increase of the detected blob and where the cause of deformation comes from. Figure 4.7 shows a sequence of images collected by the fish-eye camera, with the superimposed estimate of the application point (green) and area (blue), depending on the force's magnitude.

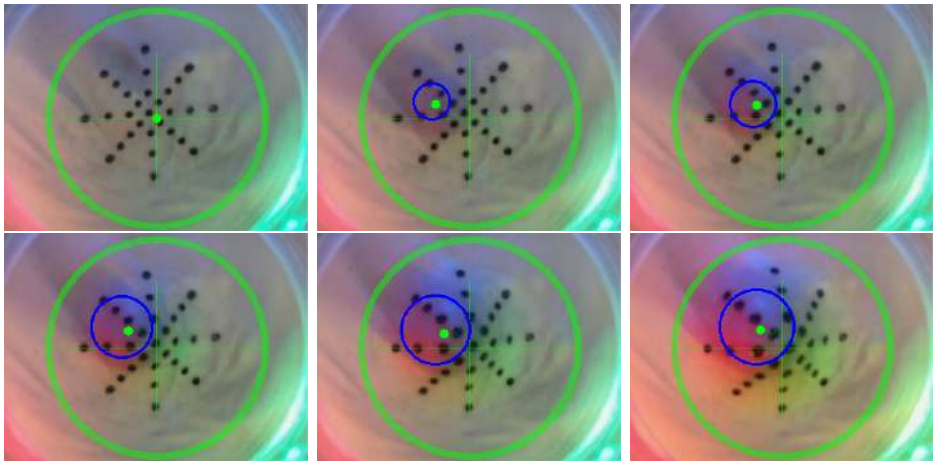

Figure 4.7: *A sequence of images showing the estimated application point (green) and area (blue), depending on the applied force's magnitude.*

Once the ground truth forces are compensated, the implementation is similar to Listing 4.1, with the main difference that in this case 29 stiffness coefficients are estimated for each direction. Finally, after training, two testing approaches can be exploited: compute the (F_x, F_y, F_z) forces using as a stiffness scalar the mean or median value for every direction (Equation 4.6); compute the resultant forces as a summation of the single components that every marker absorbs (Equation 4.7), i.e.

$$C_x = [c_{x_0}, c_{x_1}, ..., c_{x_{28}}] \tag{4.3}$$

$$C_y = [c_{y_0}, c_{y_1}, ..., c_{y_{28}}] \tag{4.4}$$

$$C_z = [c_{z_0}, c_{z_1}, ..., c_{z_{28}}] \tag{4.5}$$

$$[F_x, F_y, F_z] = [\frac{1}{29}\sum_{i=0}^{28}\overline{C_x}d_{x_i}, \frac{1}{29}\sum_{i=0}^{28}\overline{C_y}d_{y_i}, \frac{1}{29}\sum_{i=0}^{28}\overline{C_z}\sqrt{d_{x_i}^2 + d_{y_i}^2}] \tag{4.6}$$

$$[F_x, F_y, F_z] = [\sum_{i=0}^{28}c_{x_i}d_{x_i}, \sum_{i=0}^{28}c_{y_i}d_{y_i}, \sum_{i=0}^{28}c_{z_i}\sqrt{d_{x_i}^2 + d_{y_i}^2}] \tag{4.7}$$

4.1.4 Linear Regression model

The ordinary least square Linear Regression model has been used, exploiting the *sklearn* [60] library. As mentioned by the *scikit-learn* documentation, this model is fitted to the training data in order to minimize the residual sum of squares between the observed targets in the dataset and predictions made by the linear approximation. In this case there aren't hyperparameters to be tuned.

4.1.5 K-Neighbors Regressor model

The *sklearn* [59] library implements a regressor model based on the ***K-Nearest Neighbors*** model, typically used for classification tasks. As graphically shown in Figure 4.8, given a set of X training samples and a new sample to classify, the distance between the new point and all the other points is computed; the new sample is then classified as belonging to the most frequent class that appears inside the new point's "neighbourhood".

On the other hand, the ***K-Neighbors Regressor*** model can be used when the dataset's labels are continuous rather than discrete variables. The label

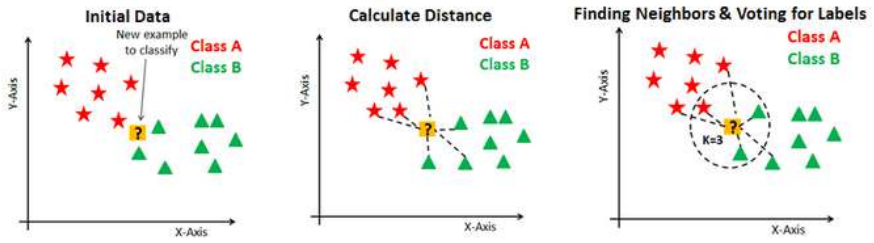

Figure 4.8: *Graphical representation of the KNN algorithm [13].*

assigned to a query point is computed based on the mean of the labels of its nearest neighbors. The main hyperparameters to be tuned are:

- ***n_neighbors***: the number of samples to be considered as neighbors (K);
- ***metric***: the distance metric to be used for detecting the closest neighbors (minkowski, cityblock, cosine, euclidean);
- ***weights***: the weighting policy that is applied to the neighborhood (uniform, distance); Figure 4.9 shows how the "weights" parameter can affect the estimation results.

To choose the best parameters, the *sklearn.model_selection.GridSearchCV()* function is used (not only for this, but also for the following algorithms). To briefly explain this function, given the hyperparameters to try, it iterates among all the possible options, returning the best combination (based on the achieved accuracy on the trainset). Once GridSearch is performed, the model with the best parameters is created and fitted on the training data.

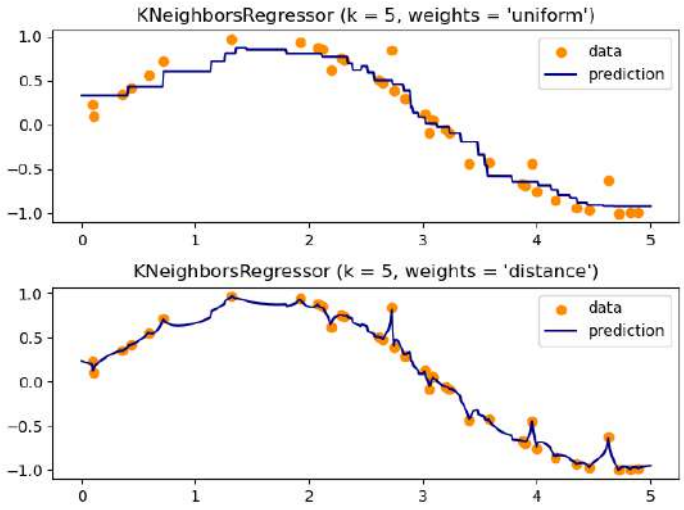

Figure 4.9: *The effect of the "weights" parameter on the estimates. The default value is "uniform" and assigns equal weights to all points; "distance" assigns weights proportional to the inverse of the distance from the query point [59].*

4.1.6 Support Vector Regression model

The *sklearn* [59] library implements a regressor model based on the ***Support Vector Machine*** model, typically used for classification tasks. SVM is a binary classification technique that uses the training dataset to predict an optimal hyperplane in an N-dimensional space, to separate data into two classes. The identified hyperplane is called *decision boundary* and by definition, it will always have one less dimension than the data space it is built in (e.g. a line in 2D space is a hyperplane of dimension 1). To identify the optimal decision boundary that will clearly separate the different classes, SVM uses Support Vectors, which are the closest data points to the edge of each class and are the most difficult points to correctly classify. The other, more generic data points are ignored for determining the boundary. The distance between the hyperplane and the support vectors are called *margins* and those need to be maximized by the model to retrieve the optimal decision boundary. Support

Vector Machines are able to separate both linear and non-linear distributed data, using the so called ***kernel trick***. A *kernel* is a function that can be used to transform a dataset into higher-dimensional space so that the data becomes linearly separable. The kernel trick effectively converts a non-separable problem into a separable one by increasing the number of dimensions in the problem space and mapping the data points to the new problem space. Figure 4.10 shows how SVM can be applied to linearly separable data, using the "linear" kernel; Figure 4.11 shows some examples of non-linearly separable data and how can be separated using non-linear decision boundaries.

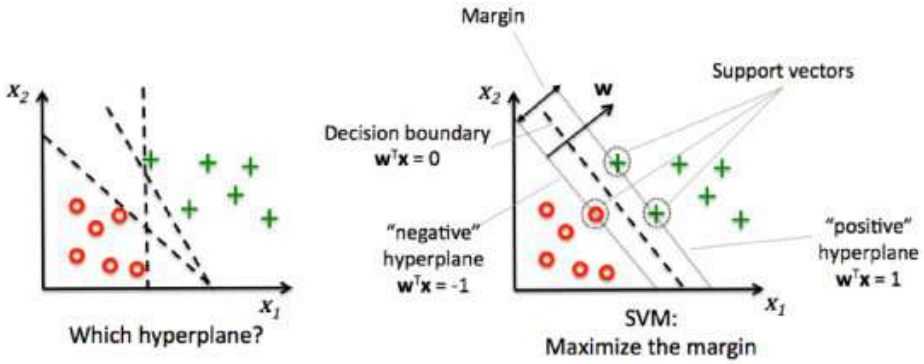

Figure 4.10: *Application of SVM in case of linearly distributed data [50].*

Moreover, SVM and, consequently SVR doesn't support multiclass classification natively, so we trained 3 different models to estimate each component of the force. In particular, we used the *sklearn.svm.SVR()* function to exploit the so called ***Support Vector Regression*** model. The implementation is based on the *libsvm* library and it exploits the concept of Support Vector Machines and re-adapts it to a continuous classification problem, that is regression. The main hyperparameters we tuned are:

- ***kernel***: the type of kernel (linear, poly, rbf, sigmoid);
- ***C***: the cost, used to regularize data with an inverse proportionality.

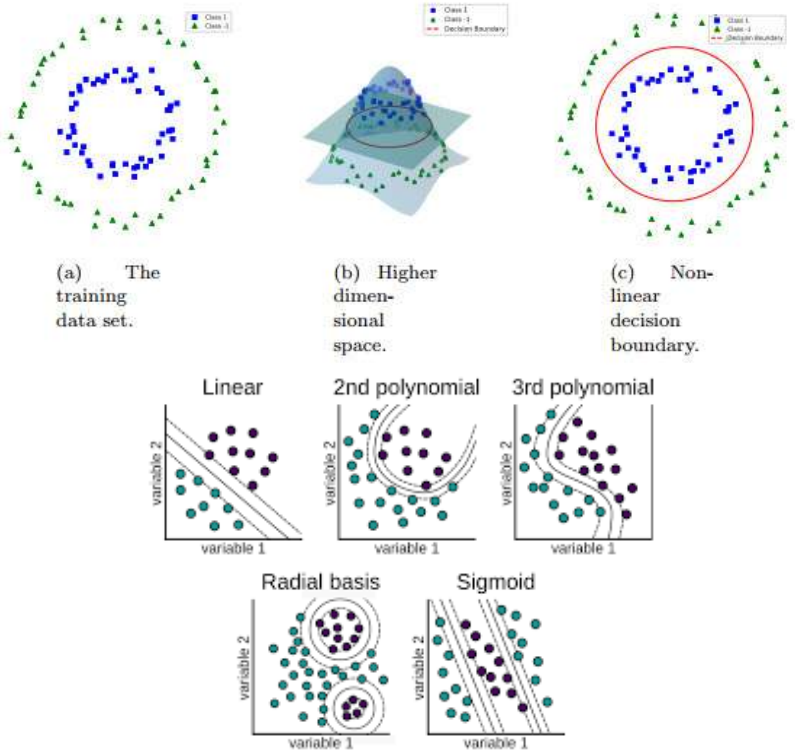

Figure 4.11: *Application of SVM in case of non-linearly distributed data [50].*

4.1.7 Neural Network Sequential model

We used the *keras* [37] library to create a simple Neural Network Sequential model. In particular, Figure 4.12 shows the Neural Network's structure that is composed of a normalization layer fitted on the training dataset, two hidden layers (*Dense*) with a depth of 64 and *Rectified Linear Unit (ReLu)* and a final *Dense* linear layer that outputs the three estimated force components. The model was compiled using a standard *Adam* optimizer to minimize the *Mean Squared Error (MSE)*. This rather simple model has a total of 8248 parameters of which 8131 are trainable.

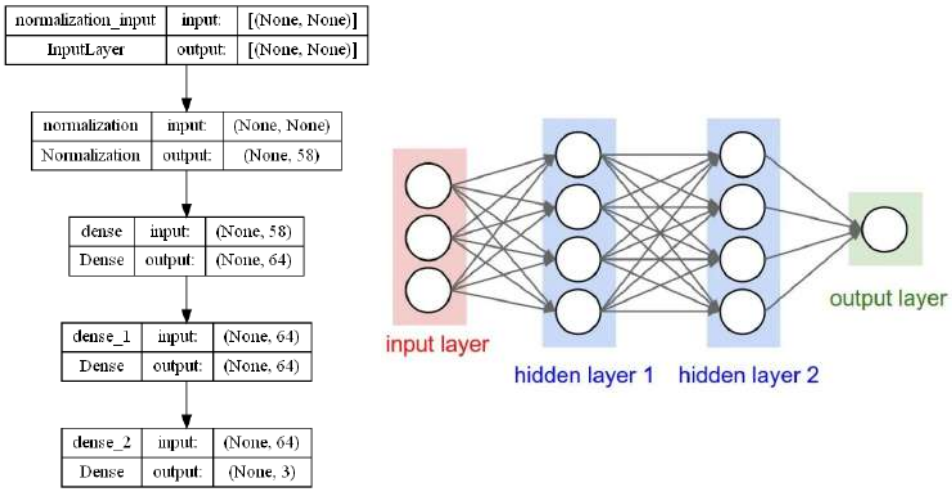

Figure 4.12: *Left: a graphical representation of the chosen Sequential model using the plot_model() function; Right: a generic example of 2-hidden layers Neural Network [12].*

4.1.8 Deep Convolutional Neural Network model

Finally, we exploited a ***Deep-CNN model*** implemented by the *keras* [36] library. In particular, several models were tested and based on the Mean Squared Error metric we chose the best performing architecture and respective hyperparameters. In this case the models are "Deep" in the sense that the number of trainable parameters is in the order of millions, while the number of hidden layers (that determine the depth) goes from 16 (e.g. VGG16) all the way to 152 (e.g. ResNet152). Those models tend to be time consuming to train and they're hardly interpretable, meaning that it's difficult to understand why certain decisions or predictions have been made. Nonetheless the achievable performance due to the deep extracted features can be outstanding. This is true especially when the tasks to perform are non-trivial; later in the Chapter we'll be discussing if this is required in our case study. Moreover, this is the only case in which we didn't input time series to the model, but images instead. As mentioned, CNNs are built to work with images through the use

of Convolutions. So, we tried to give the model both binarized images with a white background and black dots, representing the 29 fiducial markers and (during another training istance) raw RGB images recorded by the fish-eye camera (examples are shown in Figure 4.13).

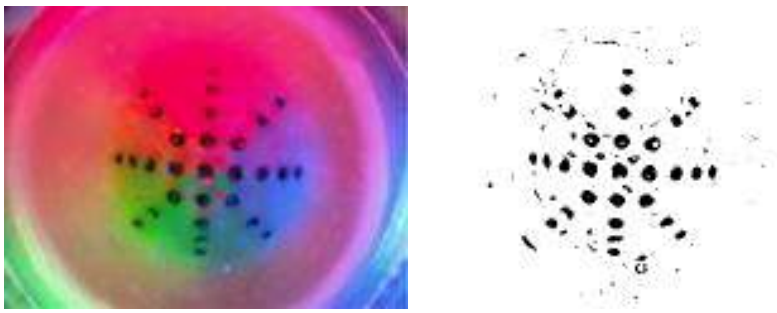

Figure 4.13: *On the left the raw image sensed by the fish-eye camera; on the right the binarized image.*

4.1.9 Feature extraction

Important aspects when training a Machine Learning or Deep Learning model is ***feature selection*** and ***extraction***. After collecting and pre-processing the dataset, we tried to extract features without mixing the samples and noticed that every algorithm didn't perform good enough. To make the models more robust and to make them converge faster, the *sklearn.model_selection.train_test_split()* function was found to be essential. As shown in Listing 4.3, the whole dataset is divided in 80% train of which a 10% used for validation during training, and 20% test. To give a quantitative measure the training set is composed by 897 samples, following the validation set with 100 samples and the testing set with 250 samples.

```
1 from sklearn.model_selection import train_test_split
2 # Split dataset between train and test
3 train_features, test_features, train_labels, test_labels =
```

```
train_test_split(all_features, all_labels, test_size=0.2,
                 random_state=42)
# Split trainset between train and validation
train_features, valid_features, train_labels, valid_labels =
train_test_split(train_features, train_labels, test_size=0.1,
                 random_state=42)
```

Listing 4.3: *Python snippet showing how the dataset is divided between train, test and validation.*

Features were extracted with several different methods to compare the results and understand the amount of required information and depth to properly characterize the sensing device. In fact, using 2 or 3 features instead of millions like in Deep Neural Network approaches can seem a huge loss of information, but on the other hand the DCNN could easily overfit data while a simple linear estimation could generalize more due to a feature extraction with limited depth. We implemented and tested 8 different methods for feature extraction (on top of which there's the possibility of scaling the features through a *MinMaxScaler*):

1. the average displacement of the 29 markers along the horizontal and vertical axis (***2 features***);
2. the average displacement of the 29 markers along the horizontal and vertical axis, the average radius increment and decrement (***4 features***);
3. the average absolute displacement of the 29 markers along the horizontal and vertical axis (***2 features***);
4. the average absolute displacement of the 29 markers along the horizontal, vertical axis and radius measurement (***3 features***);

5. the 29 sorted horizontal and vertical coordinates of the markers (***58 features***);
6. the 29 sorted coordinates and radii of the markers (***87 features***);
7. the 29 sorted horizontal and vertical displacements of the markers (***58 features***);
8. the 29 sorted horizontal, vertical and radius displacements of the markers (***87 features***).

To briefly explain how the 29 markers were sorted, the developed algorithm can be summed up with the following steps (assuming that the dome is installed in a fixed position with respect to the inner camera):

- detect the 29 markers in the first frame;
- given the markers' pattern as a prior information, use the RANSAC (RANdom SAmple Consensus) [61] algorithm to fit the vertical line;
- sort the vertical inliers according to the indexing shown in Figure 4.14;
- use the RANSAC algorithm to fit the horizontal line on the remaining 20 markers (considered to be outliers in the previous step because they don't lie on the vertical line);
- sort the horizontal inliers according to Figure 4.14;
- repeat the same process for the remaining markers placed on the two diagonals;
- whenever a new array of 29 coordinates is given, it will be automatically sorted (using the Euclidean distance metric) according to the initial sorting, that in this case doesn't change between different trials.

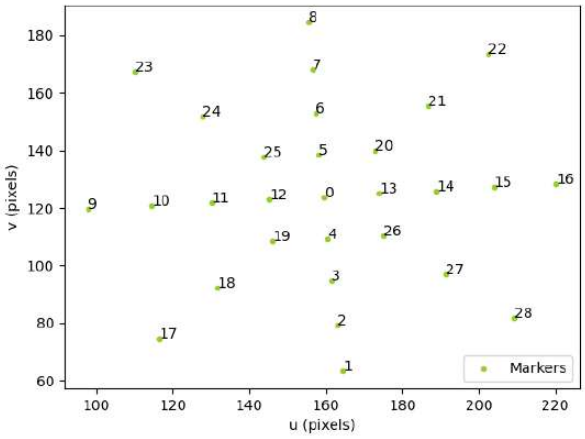

Figure 4.14: *Sorting order of the 29 fiducial markers.*

4.2 Comparison of the results

In this Subsection the obtained results are presented, starting from the evaluation on the testing portion of the collected dataset to the online validation with the robotic gripper.

4.2.1 Evaluation on the testset

Figures 4.18, 4.19, 4.20 and 4.21 show the best achieved results while estimating the (F_X, F_y, F_z) force components, with all the models that require numerical data (KNN, SVR, LR, Sequential NN, linear and non-linear methods). Moreover, Figures 4.15, 4.16 and 4.17 show a comparison of the achieved MSE values for each feature type/scaling combination (a total of 16 possibilities). To better understand the following graphs, some observations should be done:

- ***Mean Squared Error*** is the evaluation metric that is used to compare the results, even though the code allows to compute other metrics (such as Root Mean Squared Error, R^2 score, Mean Absolute Error);

- the force estimates that overcome 100% MSE are not plotted, this keeps the graphs more readable by removing unnecessary information; the same concept applies for histograms (Figures 4.15, 4.16 and 4.17) where the unsignificant results are represented by negative MSE values;

- all the models were trained, validated and tested on the same portion of dataset, randomly splitted by the mentioned *train_test_split()* function;

- the histograms shown in Figures 4.15, 4.16 and 4.17 depict the 16 feature type/scaling combinations' MSE. This comparison is useful to understand if any improvement is achieved when increasing the number and complexity of features and to visualize the effect of feature scaling.

Among all the possible combinations, the best performing options' results are shown. Particularly, Figure 4.18 shows force estimates when feature option number 2 is selected (Option **2.** in the list); due to the low number of features (only 2) the best result is achieved by KNN with a MSE of 14.1%. Figure 4.19 shows force estimates when feature option number 5 is selected (Option **5.** in the list). In this case 58 features representing each markers' horizontal and vertical displacements are extracted and the best achieved MSE is 6.3%, scored by the Linear Regressor. Figure 4.20 shows force estimates using 87 features; similarly to the previous option but including the radii's displacements (Option **6.** in the list). Figure 4.21 shows force estimates using the 58 sorted horizontal and vertical displacements (Option **7.** in the list) improving performance of KNN to a MSE of 2.86%.

As shown in Figures 4.18, 4.19, 4.20 and 4.21, the normal force F_z is estimated with a considerable MSE that, depending on the type of extracted feature and used model ranges from 2% to above 100%. Moreover, due to how the sensing device was calibrated, forces are not estimated with the same relia-

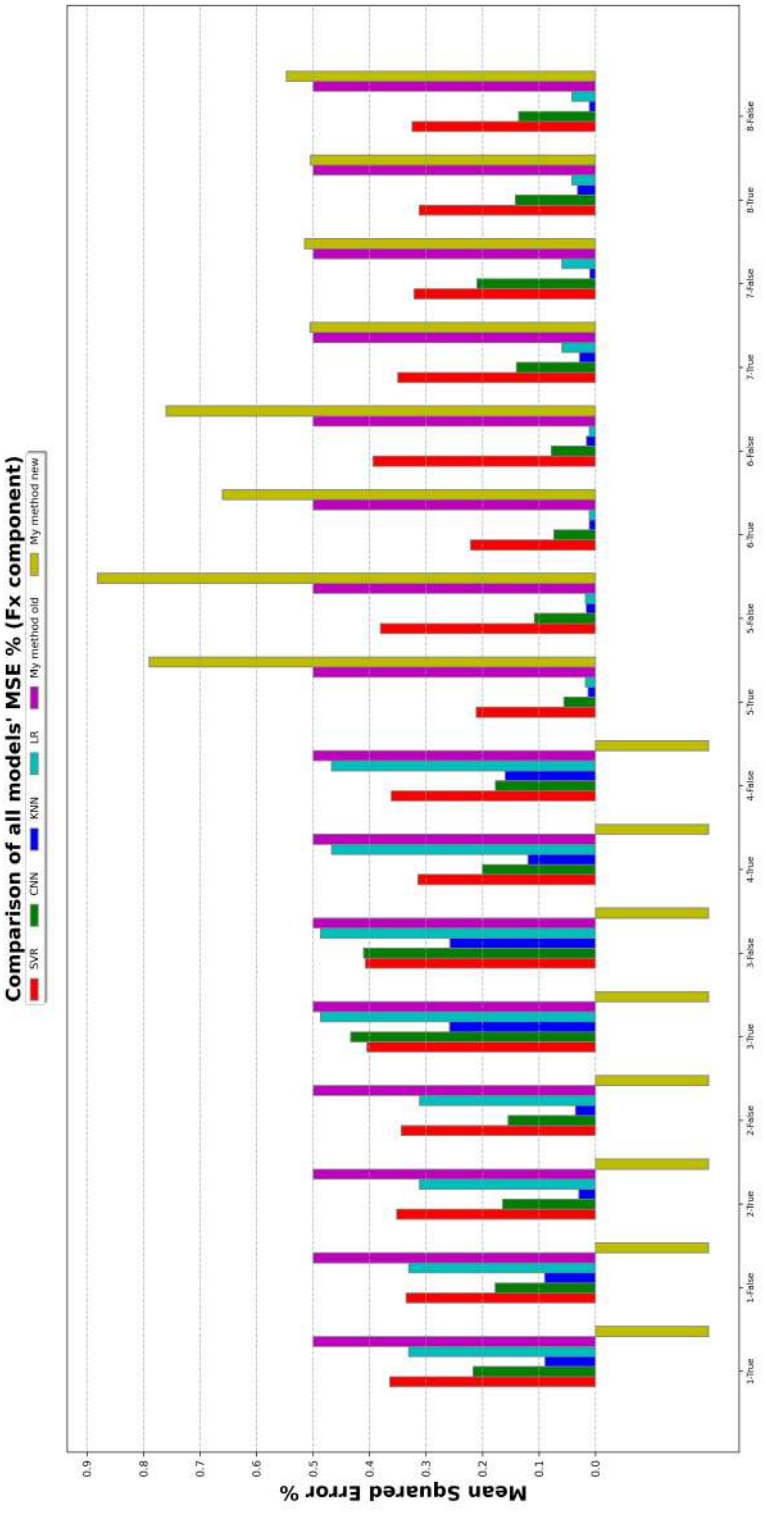

Figure 4.15: *Histogram plot showing the F_x estimation Mean Squared Error depending on the feature type/scaling combination (negative bars correspond to errors above 100%).*

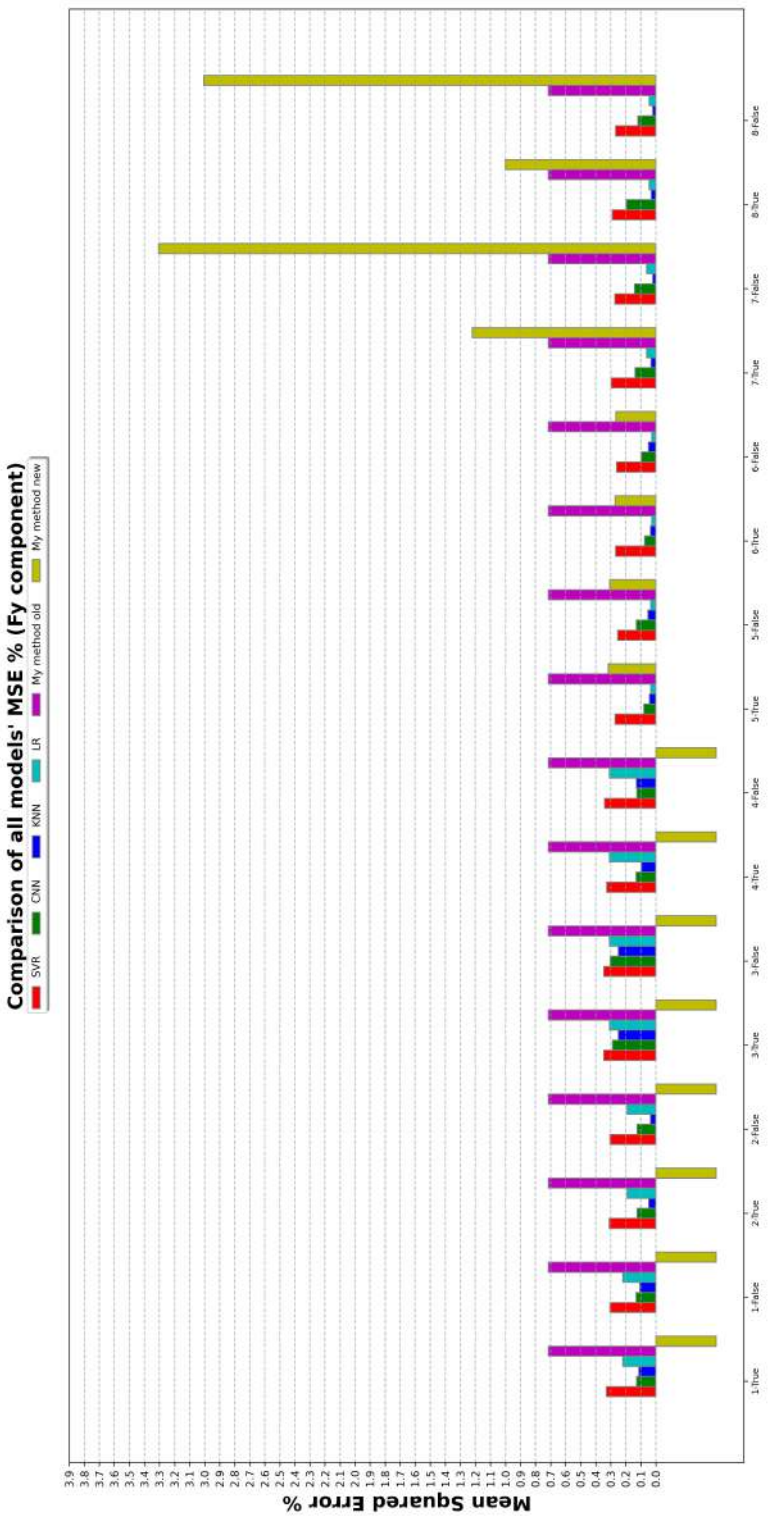

Figure 4.16: *Histogram plot showing the F_y estimation Mean Squared Error depending on the feature type/scaling combination (negative bars correspond to errors above 100%).*

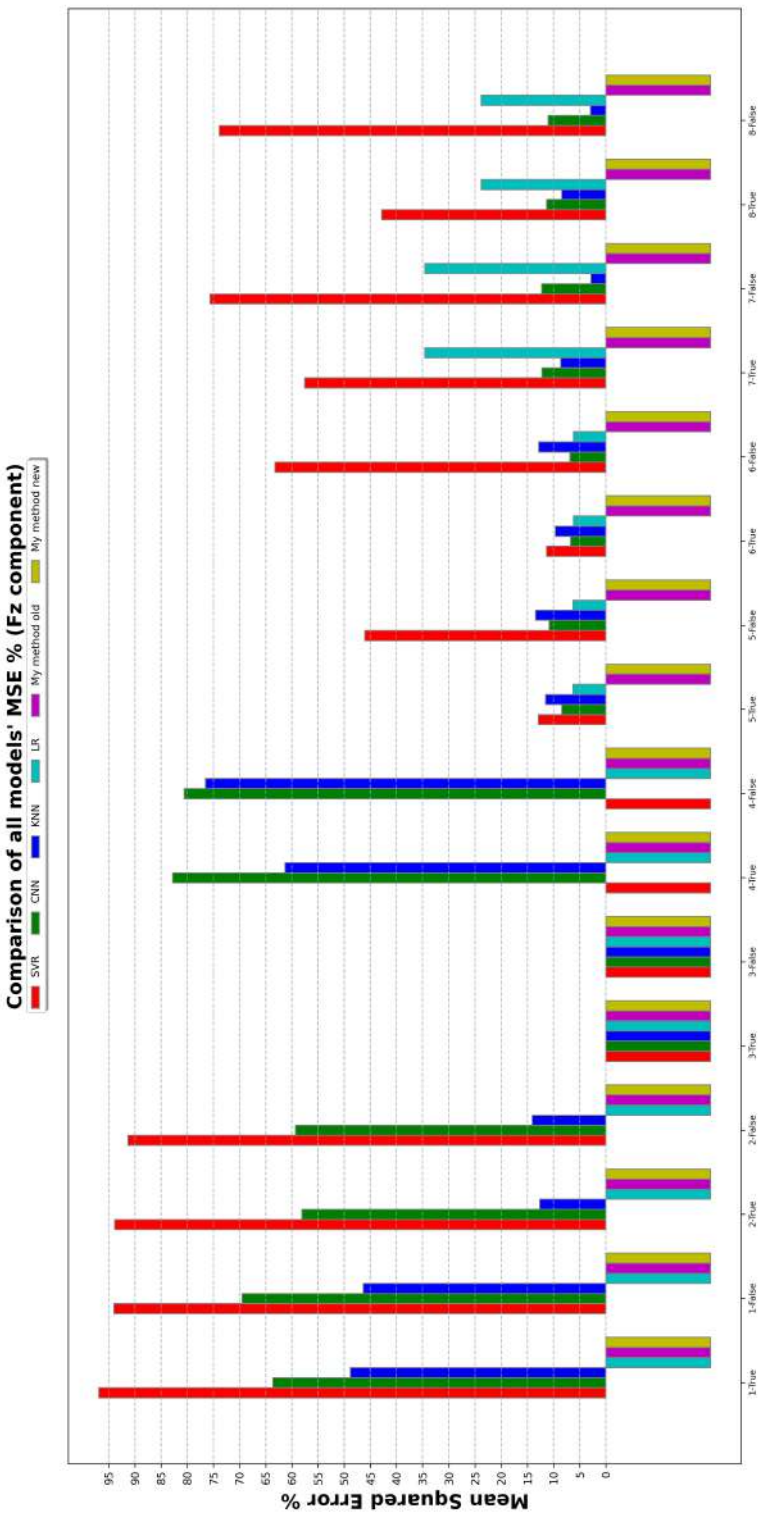

Figure 4.17: *Histogram plot showing the F_z estimation Mean Squared Error depending on the feature type/scaling combination (negative bars correspond to errors above 100%).*

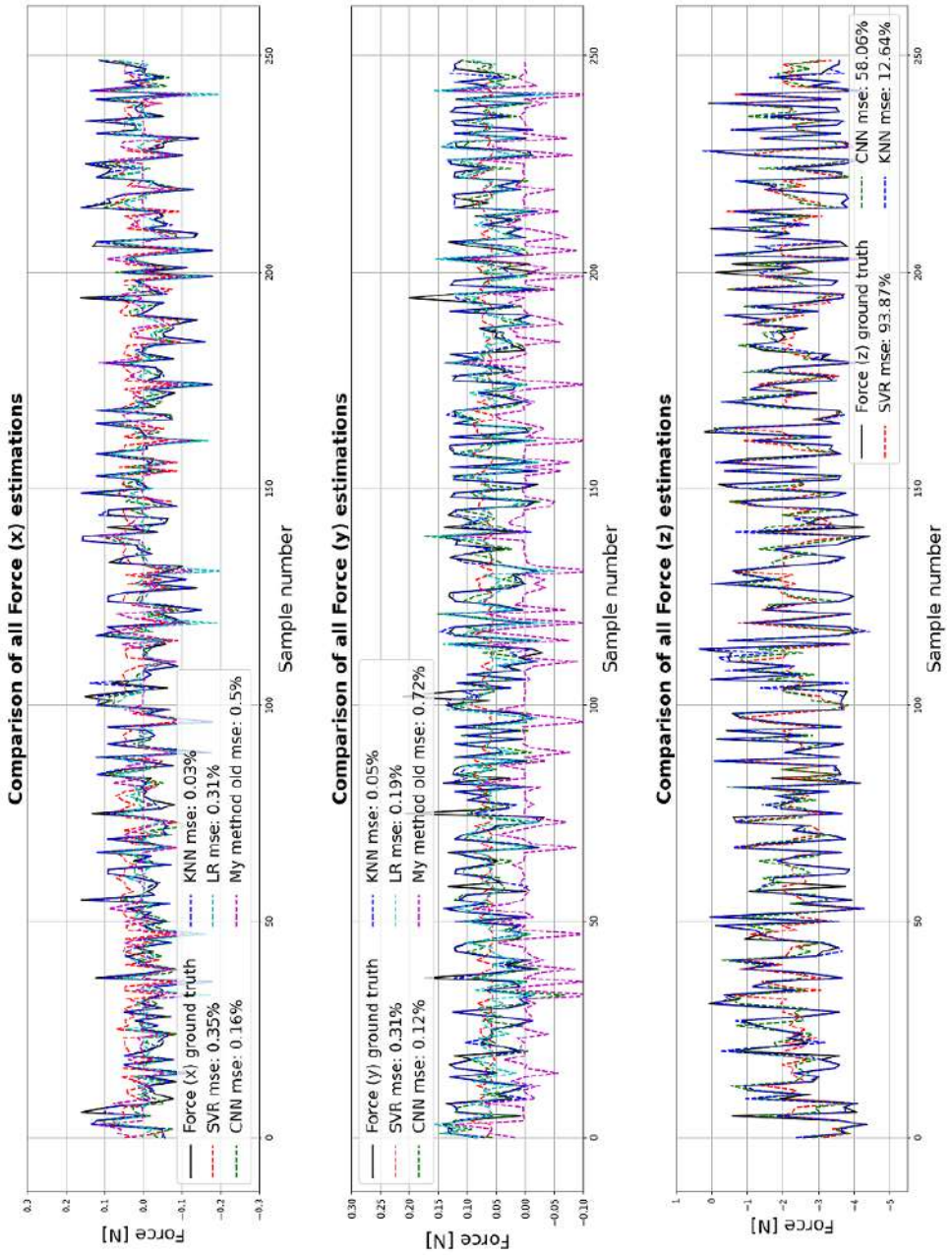

Figure 4.18: *Comparison of the estimated force components, considering feature option number 2 (Option 2. in the list) with feature scaling.*

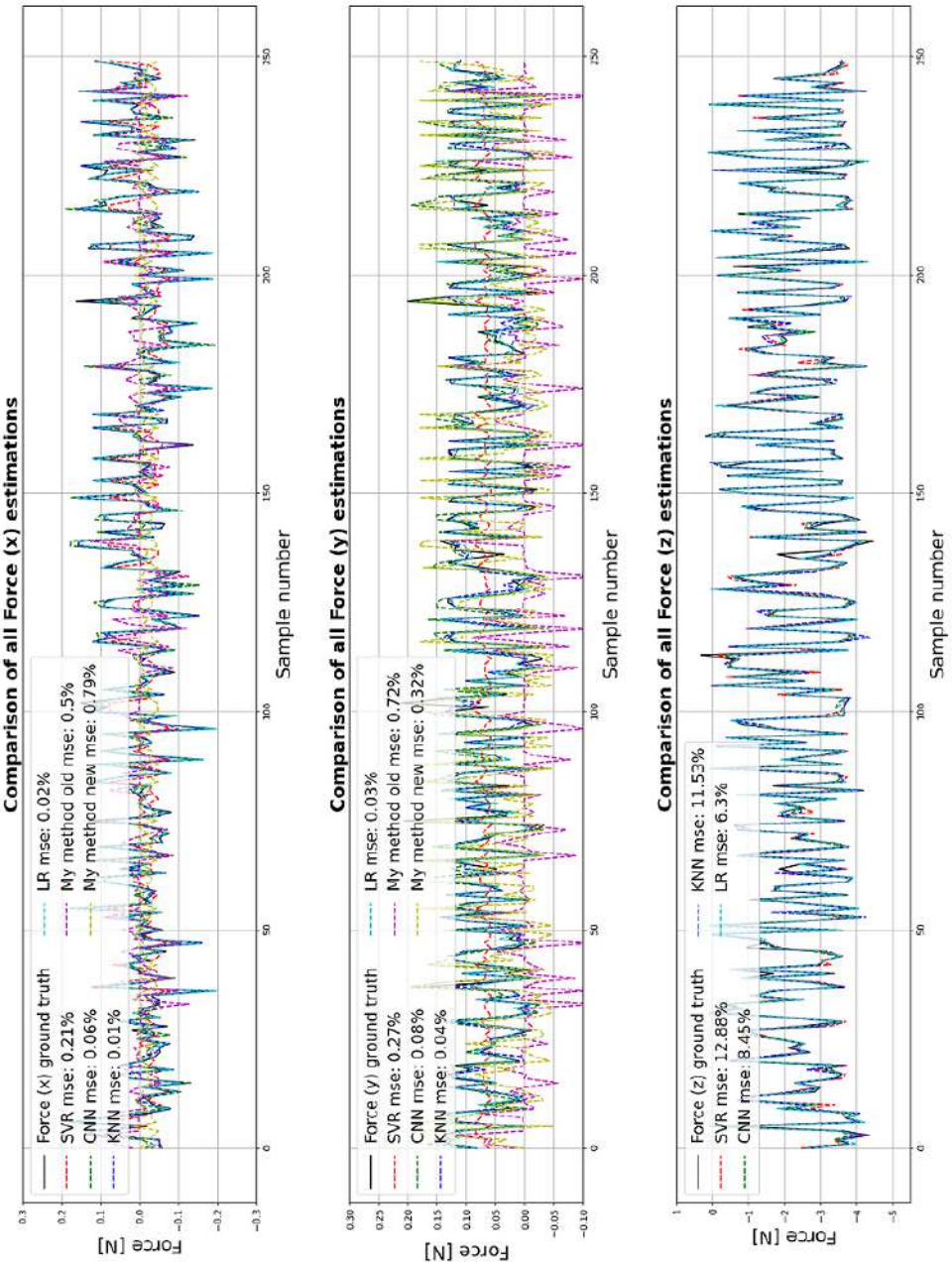

Figure 4.19: *Comparison of the estimated force components, considering feature option number 5 (Option* **5.** *in the list) with feature scaling.*

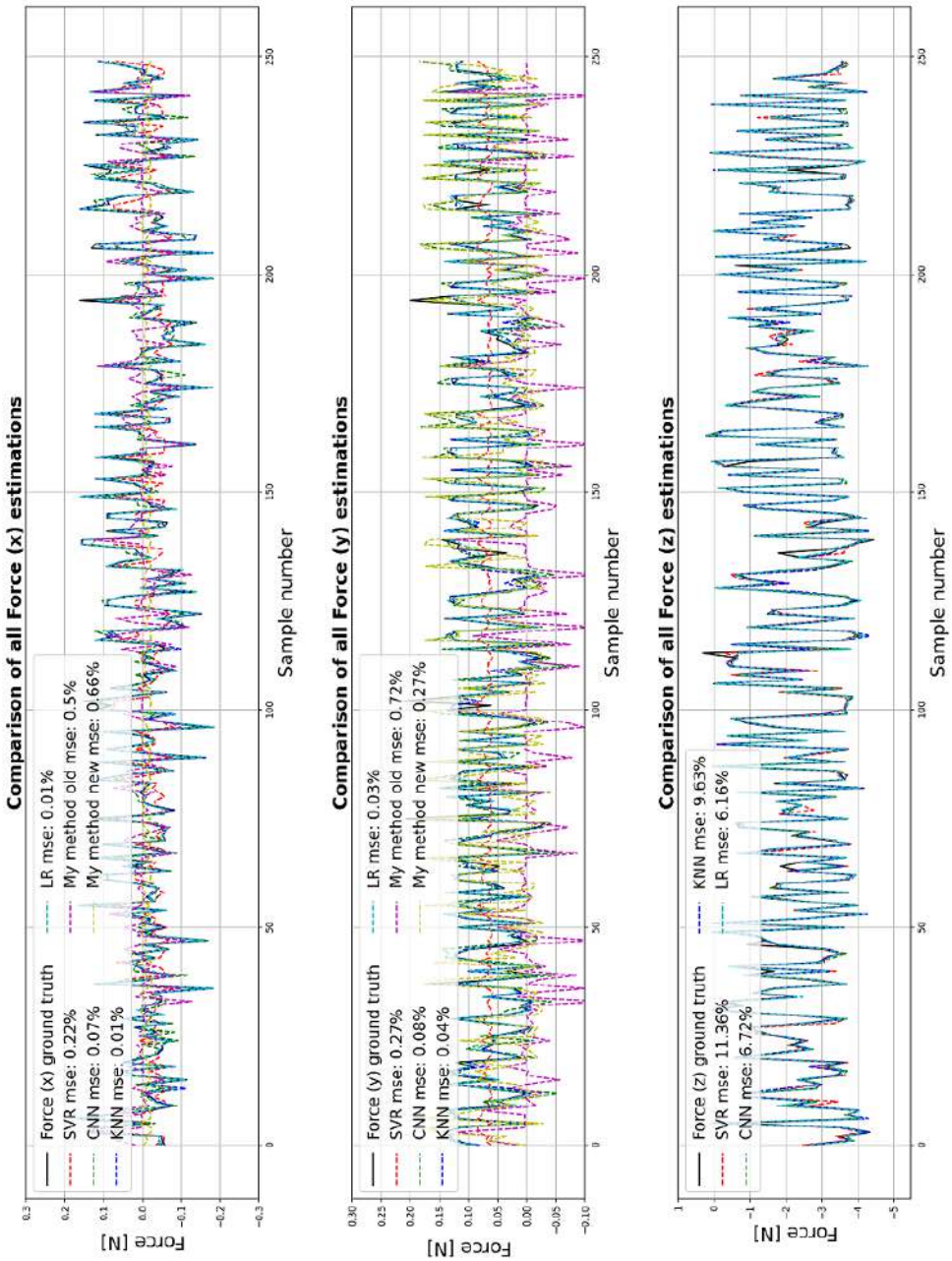

Figure 4.20: *Comparison of the estimated force components, considering feature option number 6 (Option **6.** in the list) with feature scaling.*

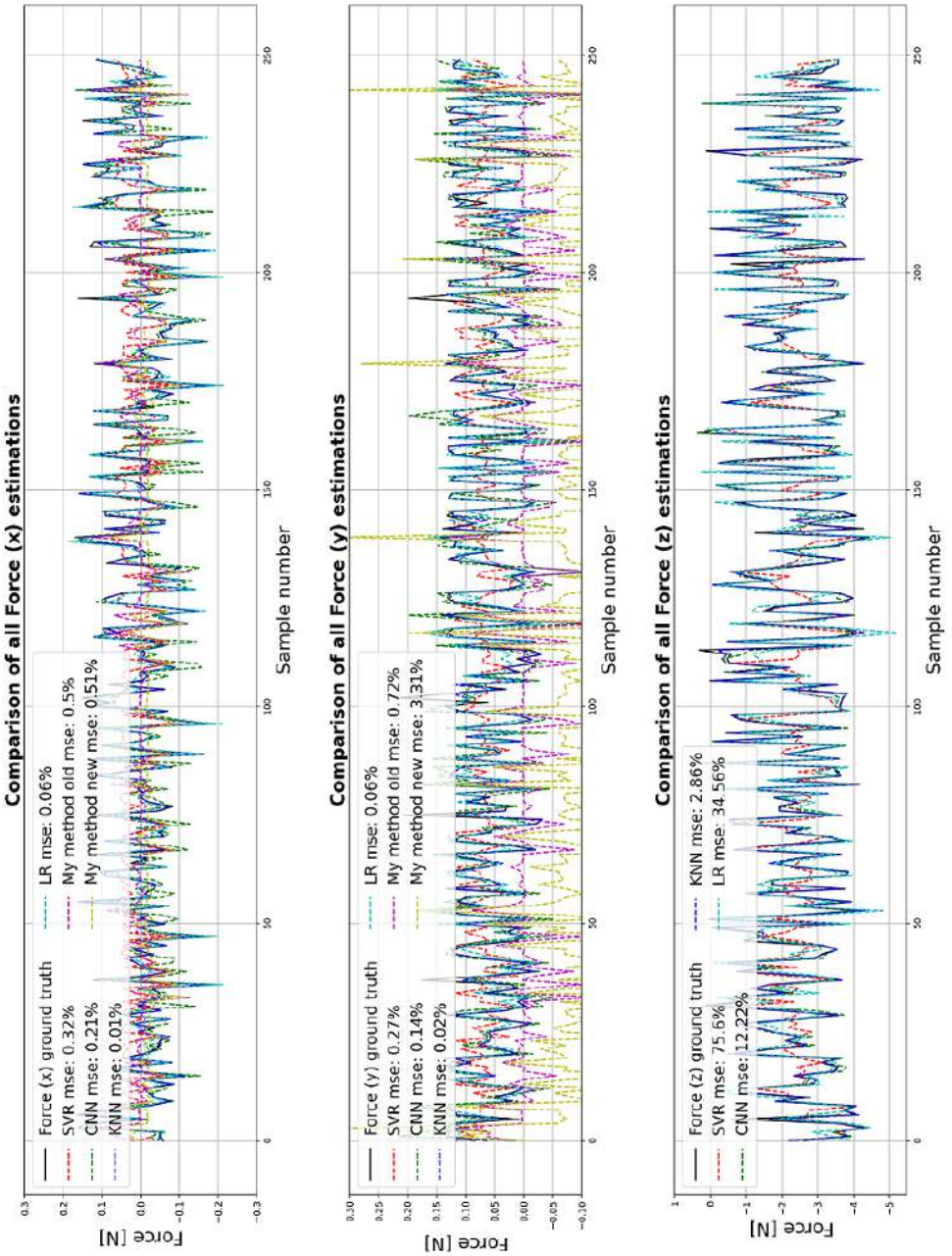

Figure 4.21: *Comparison of the estimated force components, considering feature option number 7 (Option* **7.** *in the list) without feature scaling.*

bility along the 3 directions. Meaning that the dataset is strongly limited along (X, Y) while the (Z) ground truths are more linearly and widely distributed. We think that an unneglectable bias was introduced during data collection and pre-processing, resulting in limited estimation accuracy. Nonetheless, the developed software pipelines can be easily exploited to re-calibrate the sensing device on a wider dataset, including significant shear (tangential) forces. We experimented that the device is sensible enough to perceive normal deformations of 1mm, showing that a wider data collection could lead to significant improvements of the Machine Learning models.

Figure 4.22 shows force estimates without shuffling the testset; this allows to better visualize the forces' trends. As we can see, ground truths are sometimes affected by spikes and noise, even though moving average is performed, that could worsen models' performance. Moreover, the estimates tend to correctly follow the increasing or decreasing trends but slower than the ground truths.

Regarding the Deep Convolutional Neural Network, the achieved results proved to be comparable with the Machine Learning algorithms in terms of performance, with the main difference that the required time to retrieve predictions has to be taken into account. Figures 4.23 and 4.24 show the train and validation losses (MSE) respectively when raw RGB images and binarized images are used as input of the models. Figures 4.25 and 4.26 show the achieved Mean Squared Error when testing on the unseen portion of the dataset. In both cases, after a GridSearch processing, we chose a ResNet50 Deep Convolutional Neural Network model pre-trained on ImageNet [31] with a learning rate of 0.001, a dropout probability of 0.5 and an Adam optimizer, that we trained for 50 epochs.

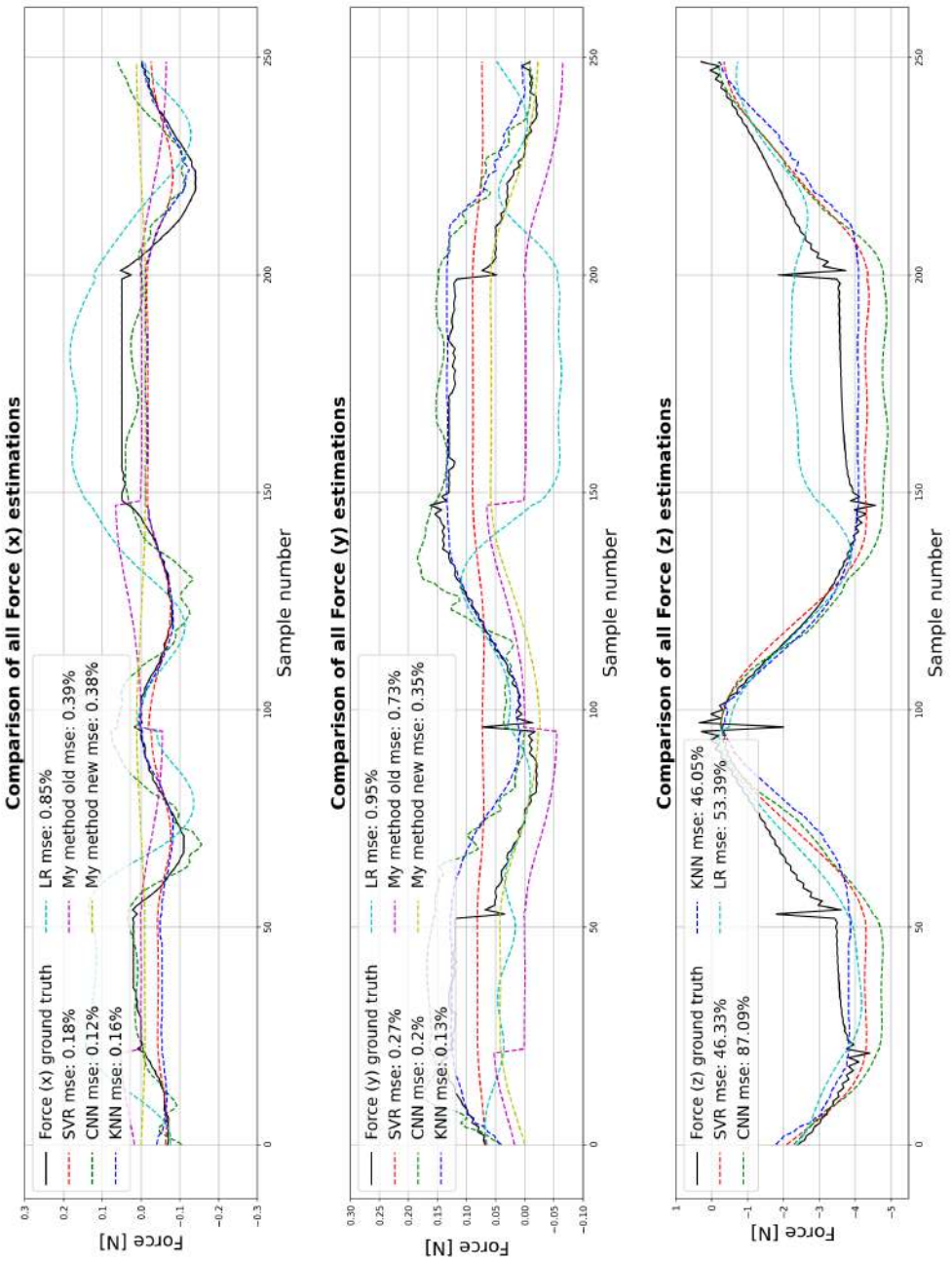

Figure 4.22: *Comparison of the estimated force components, considering feature option number 5 (Option* **5.** *in the list) with feature scaling and without the shuffle option (to see the actual forces' trends).*

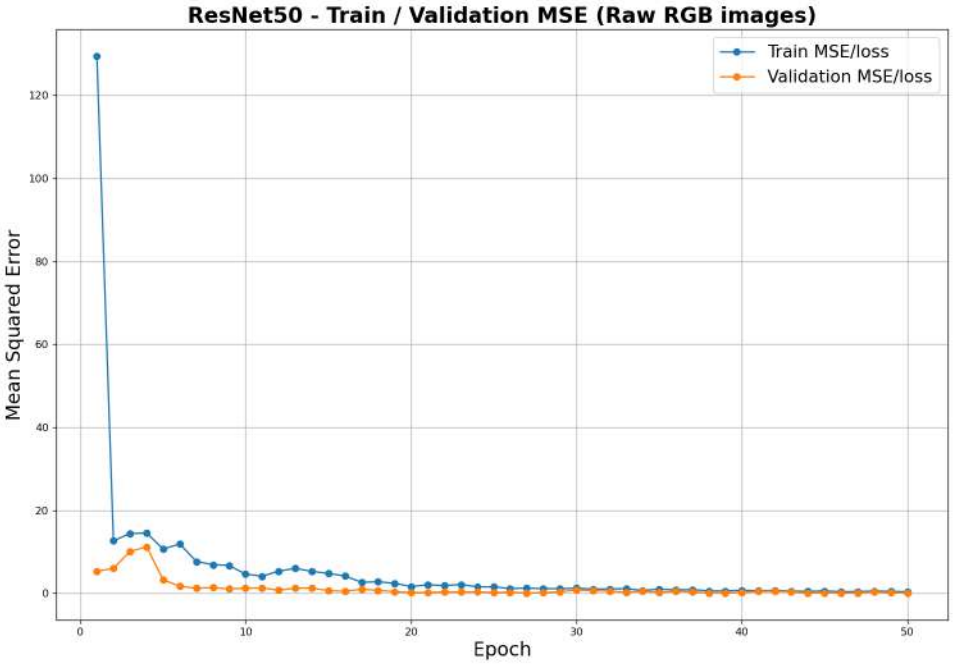

Figure 4.23: *ResNet50 train and validation loss considering raw RGB images as input.*

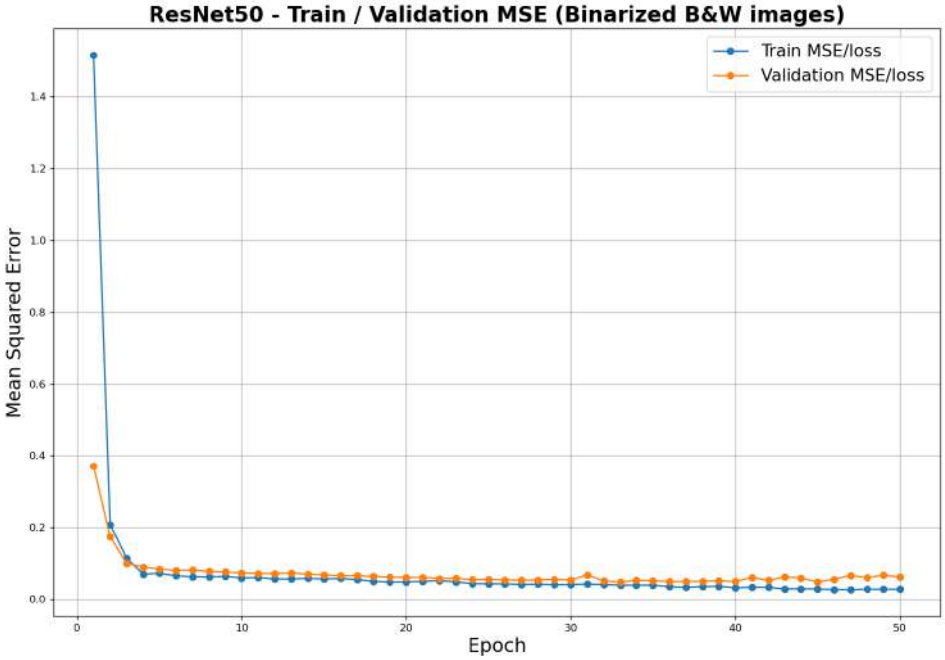

Figure 4.24: *ResNet50 train and validation loss considering Binarized images as input.*

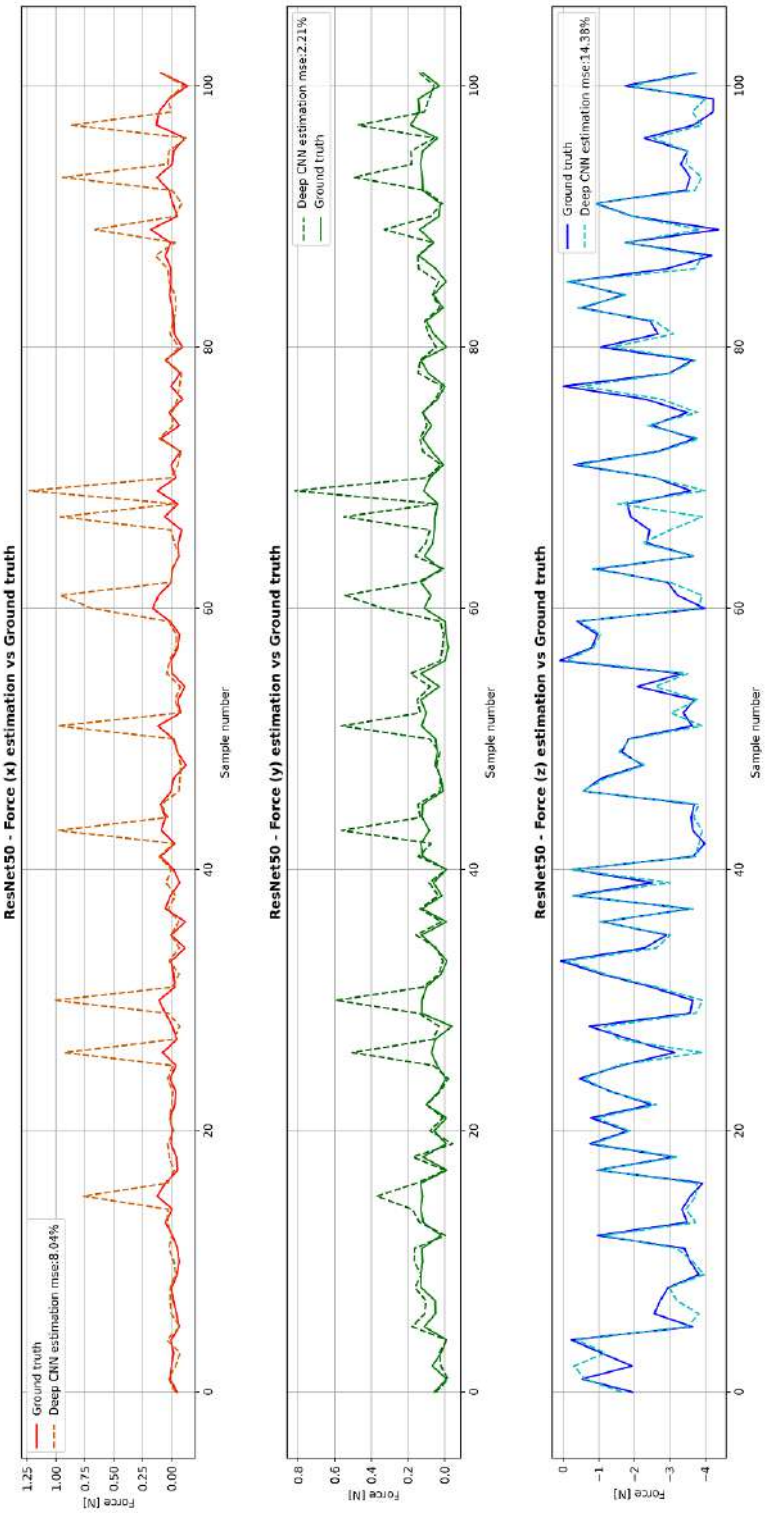

Figure 4.25: *ResNet50 predictions on the testset considering raw RGB images as input.*

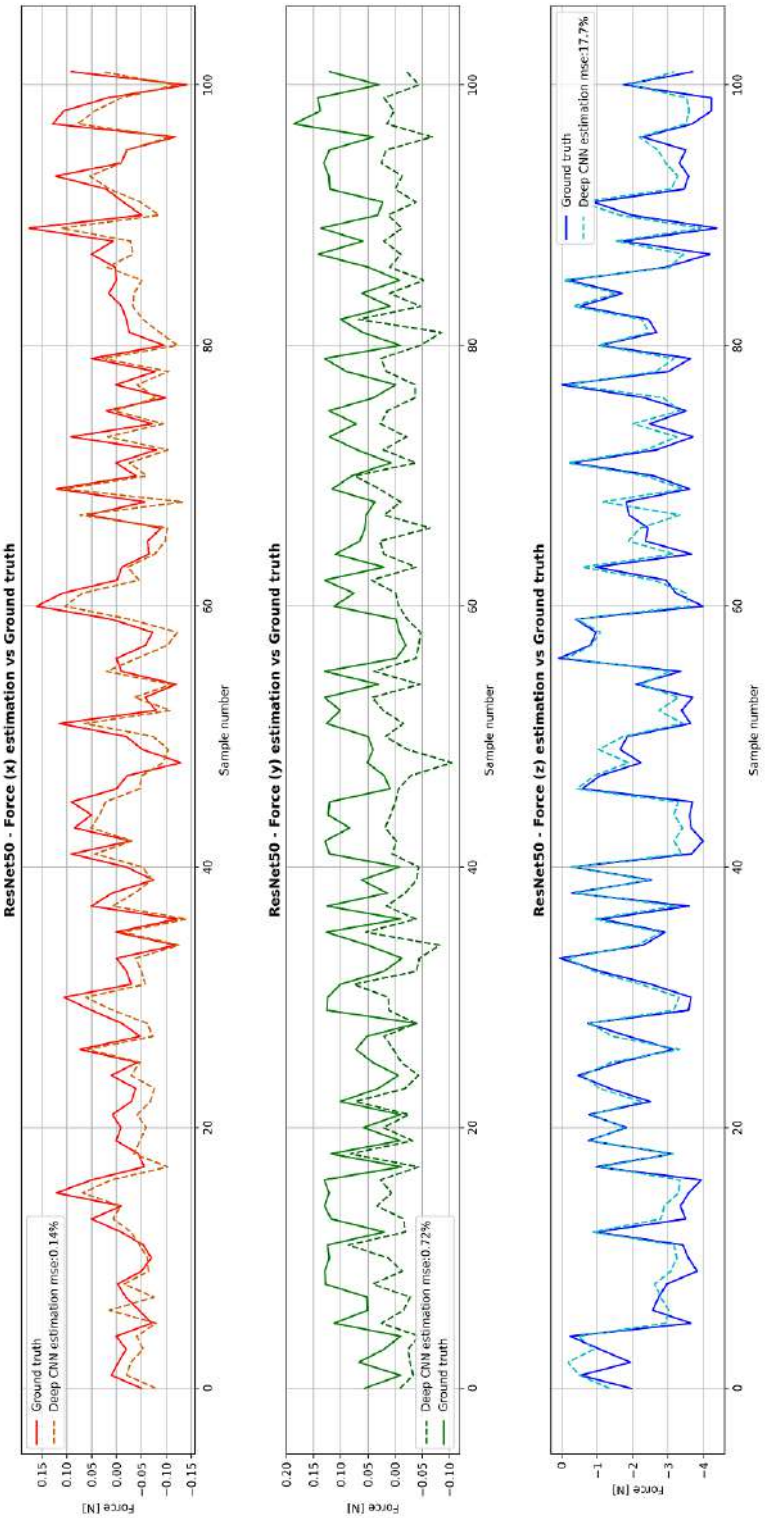

Figure 4.26: *ResNet50 predictions on the testset considering Binarized images as input.*

Table 4.1 shows the best achieved result by every force estimation approach, in terms of Mean Squared Error, when evaluated on the test portion of the dataset. To notice that only the F_z components is considered, being it more representative of the actual models' performance. According to our experiments ***KNN*** obtained the best results during testing, so we chose to use it during the harvesting task.

Model	Feature option	Number of features	Feature scaling	Best MSE
Linear elastic force approximation	-	2	-	$\geq 200\%$
Non-linear compensation	-	58	-	$\geq 200\%$
Linear Regression	**6.**	87	Yes	6.16%
K-Neighbors Regressor	**7.**	58	No	2.86%
Support Vector Regression	**6.**	87	Yes	11.36%
Sequential Neural Network	**6.**	87	Yes	6.72%
Deep Convolutional Neural Network	-	$224 \times 224 \times 3$	-	14.38%

Table 4.1: *Summary of the best performance achieved by every model evaluated on the testset (MSE values refer to the F_z component).*

4.2.2 Evaluation with the robotic gripper

Once the initial dataset was exploited to both train and evaluate the algorithms, the fitted models were further evaluated on the real setup. To do that, the ATI Nano force/torque sensor was fixed into a position and the robotic hand was closed at different speeds and widths (as shown in Figure 4.27). After collecting data from the sensor and the models, predictions were compared against ground truth forces, as shown in Figures 4.28, 4.29 and 4.30. To acquire the necessary data the ROS environment was used and a series of scripts were implemented in order to log images and json streams containing the estimated forces, markers' trajectories and application point. As previously done during the dataset collection, ground truth forces were stored inside a "rosbag"

reading the *netft_data* topic [76].

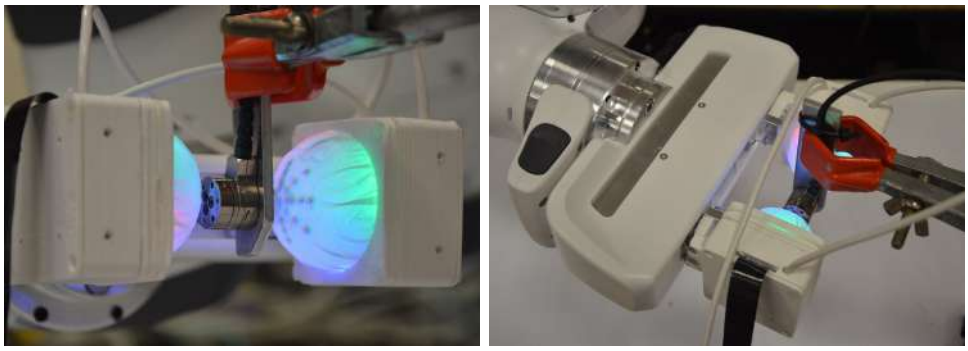

Figure 4.27: *Photographs of the force estimation validation setup.*

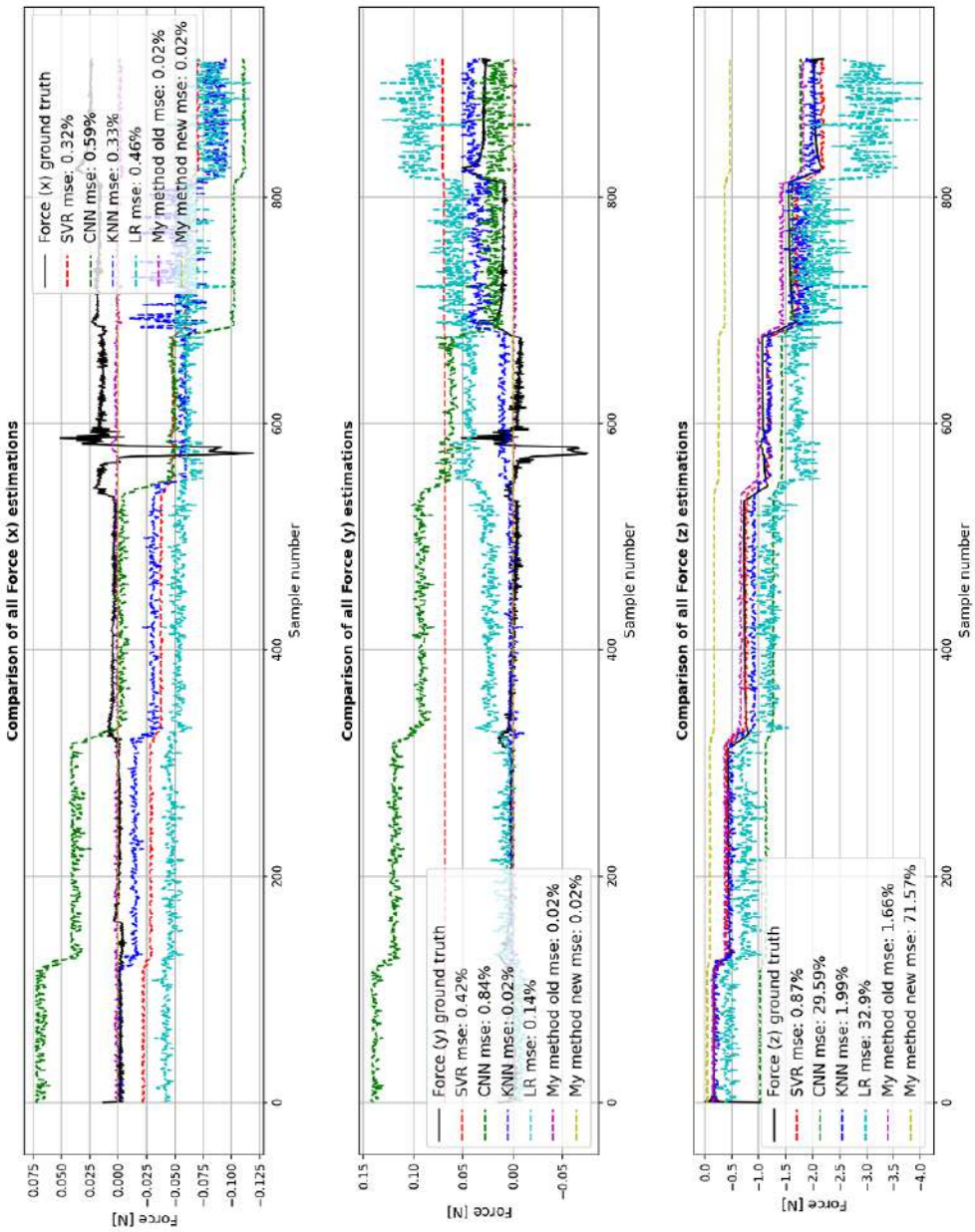

Figure 4.28: *Estimated forces against ground truths during the validation phase using the ATI Nano sensor and pressing with the developed gripper (1).*

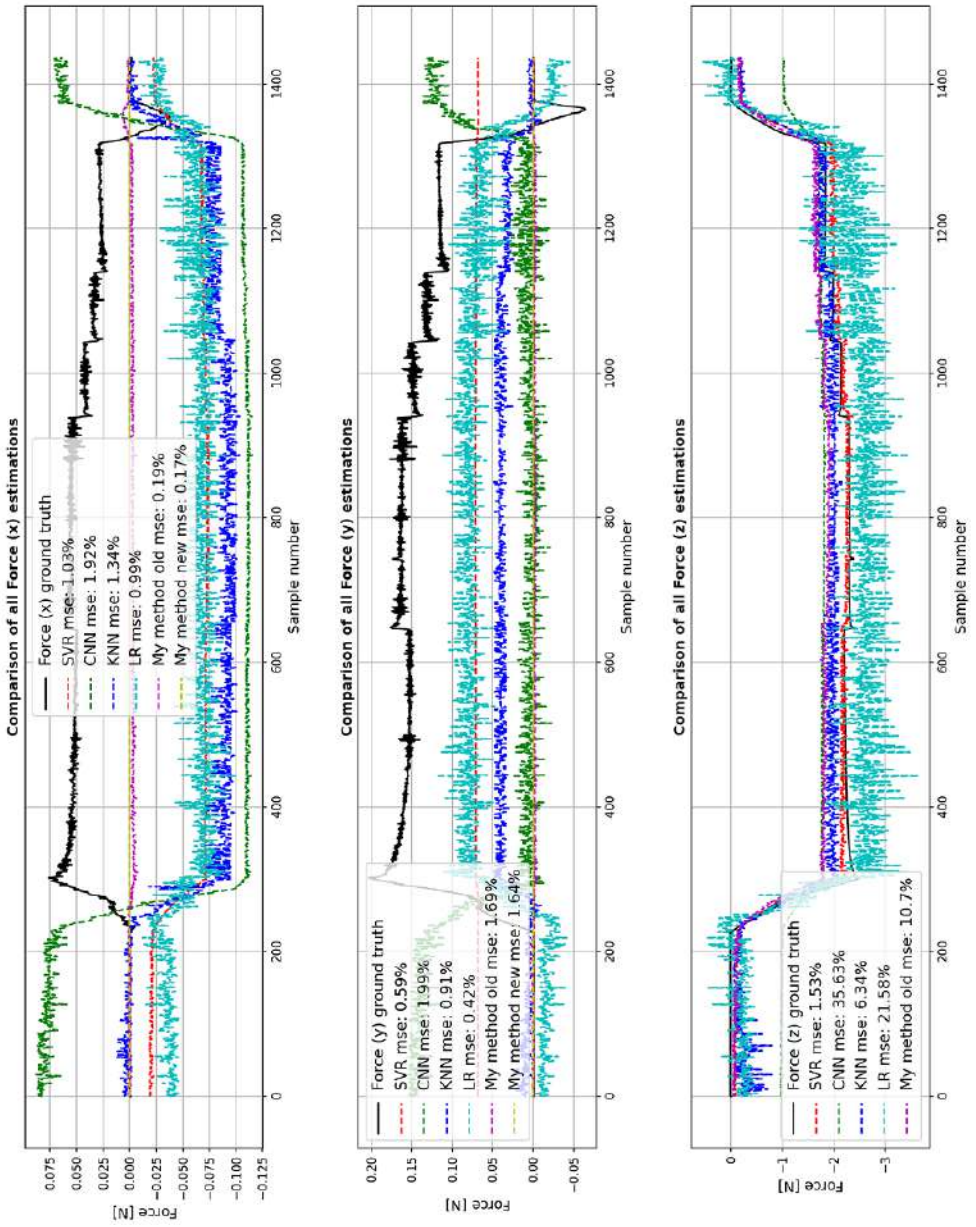

Figure 4.29: *Estimated forces against ground truths during the validation phase using the ATI Nano sensor and pressing with the developed gripper (2).*

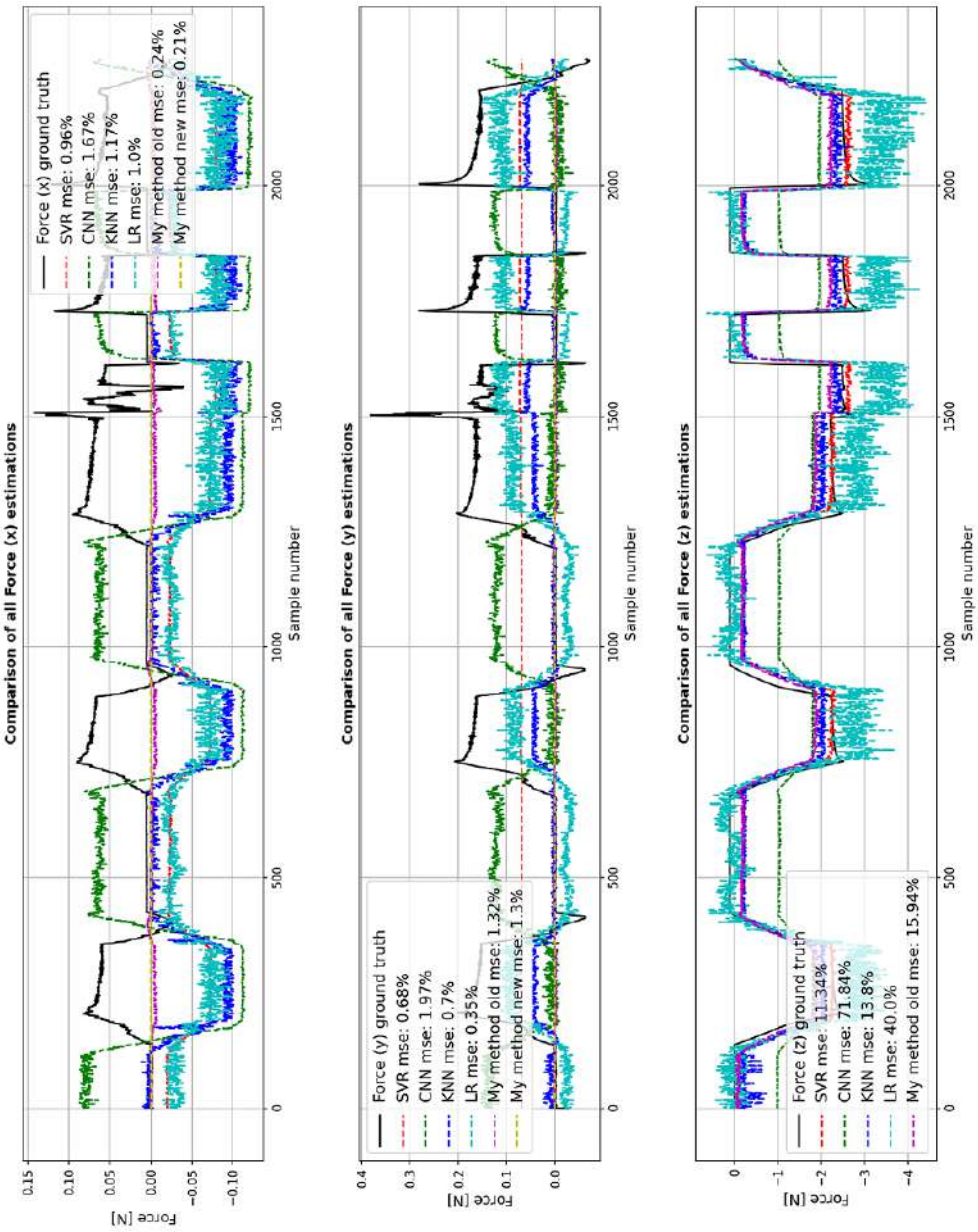

Figure 4.30: *Estimated forces against ground truths during the validation phase using the ATI Nano sensor and pressing with the developed gripper (3).*

4.2.3 Real-time force feedback and strawberry detection

In this Subsection the fruit-picking task is attempted and qualitative and quantitative results are shown. As mentioned at the beginning of this Thesis, the aim of a soft gripper is to gently grasp deformable objects without squeezing or damaging them. To demonstrate that this is possible, even though the gripper design and control architecture can be improved, we setup a series of strawberry plants, emulating a sort of hydroponic culture (shown in Figure 4.31).

Figure 4.31: *Photograph showing how strawberry plants were setup (emulating a hydroponic culture) to perform the picking task.*

Afterwards, we tried to perform a simple harvesting task, starting from the detection of the ripe fruit, all the way to the picking of the strawberry with a suitable motion. This pipeline was implemented in ROS with mainly Python scripts handling the nodes. In fact, the first developed ROS node provides the real-time force feedback using the online estimation pipeline based on the

gripper's deformation (KNN was found to be the most reliable model to use); the second developed node takes care of fruit detection exploiting a pre-trained CNN (YOLOv3) model, that was fine-tuned on a small strawberry dataset for object detection (about 600 images). Figure 4.32 shows a qualitative result of the trained CNN for strawberry detection; it is able to locate the boundaries of the fruit while classifying it as *ripe* or *unripe* with a level of certainty.

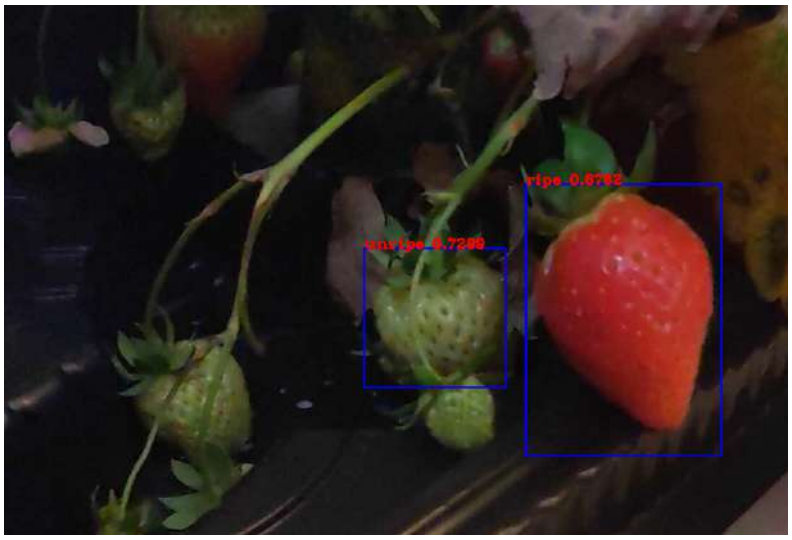

Figure 4.32: *Qualitative result of the fine-tuned CNN tested on one of the plants used for the setup shown in 4.31.*

The developed real-time pipeline requires some supervision due to the temporary setup and a non-perfectly tuned Neural Network and consists in the following steps:

- as shown in Figure 4.33, the robotic harm is placed around 50cm away from the plants and uses the RGBD camera information to both find ripe fruits and compute the 3D location of their central point (center of the estimated 2D bounding box) with respect to the camera;
- the estimated 3D point approximately represents the strawberry bari-

center with respect to the camera. That coordinate is then transformed (exploiting hand-eye calibration results) with respect to the robot's base reference frame;

- as shown in Figure 4.34, the gripper, that is still open, approaches the target point published on a specific ROS topic according to a properly tuned impedence control law;
- once the gripper is correctly positioned, the closing command is published to the *frank_gripper/MoveActionGoal* topic;
- based on the received real-time force feedback, the gripper is stopped to the current position (publishing the last width value), when the normal force reaches the 1.75N threshold (Figure 4.37 shows the real-time force feedback that was estimated during the picking task);
- when the force feedback stabilizes, the gripper performs the suggested picking pattern for strawberries; in particular it slowly rotates and moves back with respect to the plant, harvesting the fruit (this is represented by Figures 4.35 and 4.36).

Figure 4.33: *Phase 1: detection and localization of the ripe fruit.*

Figure 4.34: *Phase 2: approach of the ripe strawberry given the 3D target point.*

Figure 4.35: *Phase 3: application of the picking pattern to harvest the strawberry.*

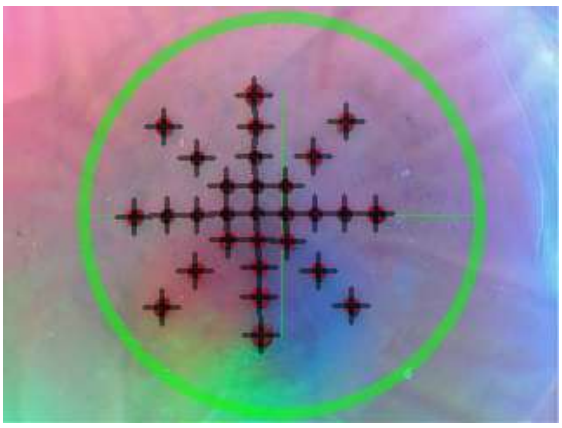

Figure 4.36: *Image sensed by the fish-eye camera when the dome is deformed, with the superimposed detected markers.*

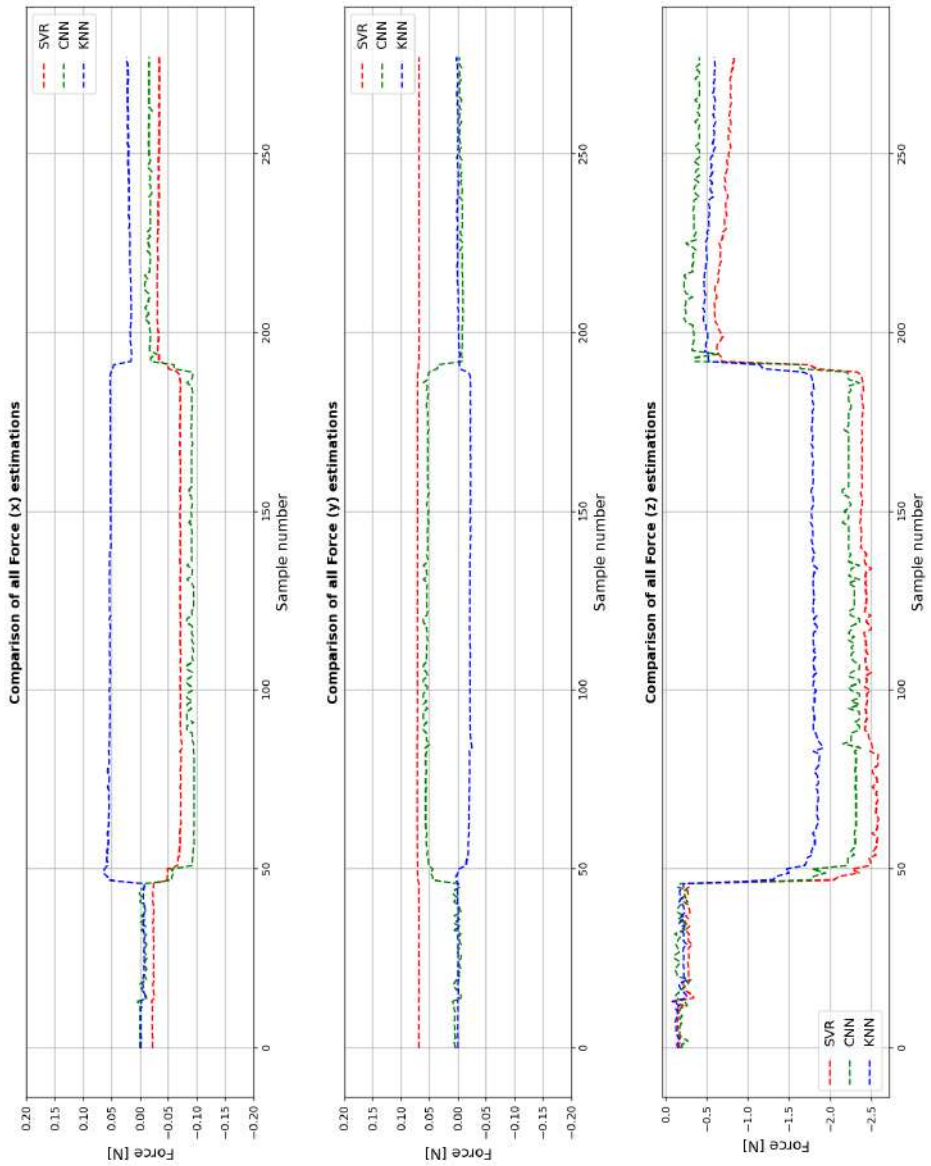

Figure 4.37: *Online force feedbacks while grasping the strawberry (we used the KNN model to estimate normal force and determine if the 1.75 force threshold has been reached).*

Chapter 5

Conclusions

Most of the state-of-the-art work on soft tactile sensors focuses on finger-like or "palmar" soft grippers, that have a quasi-planar geometry, leading to simpler force estimation and overall characterization. We proposed a hemispherical cheap to manufactured 3D printed soft gripper that allows to gently grasp objects. We characterized the sensing device by chosing a suitable placement and density of fiducial markers. By exploiting several Machine Learning and Deep Learning approaches we estimated forces exerted on the dome's surface that were reasonable and sufficient to attempt a picking task. In fact, even though several improvements can be made, the force estimates are stable, allowing to easily set a threshold for stopping the Franka Hand gripper's closure. Moreover, on a real scenario it would be required to gently grasp and handle the fruit or vegetable to preserve its quality, meaning that fast pick and place control isn't probably required nor useful. The proposed force estimation techniques heavily rely on Computer Vision, Machine Learning algorithms and calibration of the device. This means that software updates could surely improve the estimation performance and add new features without necessarily changing the design. Future developments could focus on this subject, providing

Figure 5.1: *Photographs of the developed sensing device holding a strawberry without squeezing it.*

a fully automated calibration process. Also, the fruit detection and picking results should be made more robust against environmental disturbances. To conclude, this was a motivating and inspiring project to work on, also due to the interest that the agricultural industry demonstrates to put into practice the "4.0 transition".

Bibliography

[1] https://www.adafruit.com/product/1643, 2022.

[2] https://www.mccormick.it/as/agriculture-4-0-what-is-it-and-what-are-its-tools-and-benefits, 2021.

[3] https://www.agrobot.com/e-series, 2022.

[4] Alex Alspach, Kunimatsu Hashimoto, Naveen Kuppuswamy, and Russ Tedrake. Soft-bubble: A highly compliant dense geometry tactile sensor for robot manipulation. *IEEE*, 2019.

[5] https://aerial-robotix.com/asctec-falcon-8/, 2022.

[6] https://www.ati-ia.com/products/ft/ft_models.aspx?id=nano17, 2022.

[7] Dominik Bauer, Cornelia Bauer, Arjun Lakshmipathy, and Nancy Pollard. Fully printable low-cost dexterous soft robotic manipulators for agriculture. *AAAI 2022 Workshop*, 2021.

[8] Anelise Borges and Natalie Huet. https://www.euronews.com/my-europe/2020/07/17/invisible-workers-underpaid-exploited-and-put-at-risk-on-europe-s-farms, 2020.

[9] https://burro.ai/, 2022.

[10] https://www.cambridgeconsultants.com/press-releases/hank-dexterous-robot-human-touch, 2022.

[11] Daniel Costa. https://www.epi.org/blog/the-farmworker-wage-gap-continued-in-2020-farmworkers-and-h-2a-workers-earned-very-low-wages-during-the-pandemic-even-compared-with-other-low-wage-workers, 2021.

[12] https://cs231n.github.io/neural-networks-1/, 2023.

[13] https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn, 2023.

[14] Hoang Xuan Diem and Do Thi Thu Thuy. Prospects for agriculture 4.0 in developing countries: Case studies from vietnam. *IDE-JETRO*, 2020.

[15] https://www.dji.com/it/mg-1p, 2022.

[16] Won Kyung Do and Monroe Kennedy. Densetact: Optical tactile sensor for dense shape reconstruction. *ICRA*, 2022.

[17] https://fastled.io/, 2022.

[18] Alfonso J. Fernandez, Huan Weng, Paul B. Umbanhowar, , and Kevin M. Lynch. Visiflex: A low-cost compliant tactile fingertip for force, torque, and contact sensing. *IEEE*, 2021.

[19] https://www.festo.com/, 2022.

[20] https://www.amazon.it/Tangxi-videocamera-Raspberry-Fotocamera-grandangolare/dp/B07WH1D4D4/ref=sr_1_3_sspa?__mk_it_IT=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=1O0NE4N291TA3&keywords=camera+fisheye+raspberry&qid=1675413564&sprefix=

camera+fish+eye+raspberr%2Caps%2C124&sr=8-3-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1&smid=A1UVPWM0U8T1NP, 2023.

[21] https://formlabs.com/it/3d-printers/form-2/, 2022.

[22] https://formlabs.com/materials/, 2022.

[23] https://www.franka.de/, 2022.

[24] https://www.generationrobots.com/media/panda-franka-emika-datasheet.pdf, 2022.

[25] Lucas Gerez, Che-Ming Chang, and Minas Liarokapis. Employing pneumatic, telescopic actuators for the development of soft and hybrid robotic grippers. *frontiers*, 2020.

[26] https://github.com/rpi-ws281x/rpi-ws281x-python, 2022.

[27] Aimee Goncalves, Naveen Kuppuswamy, Andrew Beaulieu, Avinash Uttamchandani, Katherine M. Tsui, and Alex Alspach. Punyo-1: Soft tactile-sensing upper-body robot for large object manipulation and physical human interaction. *IEEE*, 2022.

[28] https://www.harvestcroorobotics.com/, 2022.

[29] Trung Thien Hoang, Jason Jia Sheng Quek, Mai Thanh Thai, Phuoc Thien Phan, Nigel Hamilton Lovell, and Thanh Nho Do. Soft robotic fabric gripper with gecko adhesion and variable stiffness. *Sciencedirect*, 2021.

[30] https://www.plugandplaytechcenter.com/resources/how-automation-transforming-farming-industry, 2022.

[31] https://image-net.org/about.php, 2023.

[32] https://www.intelrealsense.com/depth-camera-d435i/, 2022.

[33] http://www.interactivearchitecture.org/a-wearable-soft-robot-with-variable-material-distribution.html, 2022.

[34] Snehal Jain, Saikrishna Dontu, Joanne Ee Mei Teoh, and Pablo Valdivia Y Alvarado. A multimodal, reconfigurable workspace soft gripper for advanced grasping tasks. *Soft Robotics*, 2020.

[35] https://www.joulin.com/company/vacuum-technology.html, 2022.

[36] https://keras.io/api/applications/, 2022.

[37] https://keras.io/guides/sequential_model/, 2022.

[38] Uikyum Kim, Dawoon Jung, Heeyoen Jeong, Jongwoo Park, Hyun-Mok Jung, Joono Cheong, Hyouk Ryeol Choi, Hyunmin Do, and Chanhun Park. Integrated linkage-driven dexterous anthropomorphic robotic hand. *Nature*, 2021.

[39] Maria Kondoyanni, Dimitrios Loukatos, Chrysanthos Maraveas, Christos Drosos, and Konstantinos G. Arvanitis. Bio-inspired robots and structures toward fostering the modernization of agriculture. *MDPI*, 2022.

[40] Naveen Kuppuswamy, Alex Alspach, Avinash Uttamchandani, Sam Creasey, Takuya Ikeda, and Russ Tedrake. Soft-bubble grippers for robust and perceptive manipulation. *Soft Robotics*, 2020.

[41] https://learnopencv.com/blob-detection-using-opencv-python-c/, 2022.

[42] https://levity.ai/blog/difference-machine-learning-deep-learning, 2023.

[43] Sandra Q. Liu and Edward H. Adelson. Gelsight fin ray: Incorporating tactile sensing into a soft compliant robotic gripper. *IEEE*, 2022.

[44] Mariangela Manti, Taimoor Hassan, Giovanni Passetti, Nicolo‘ D’Elia, Cecilia Laschi, , and Matteo Cianchetti. A bioinspired soft robotic gripper for adaptable and effective grasping. *Soft Robotics*, 2015.

[45] `https://medium.com/@nhan.tran/mean-median-an-mode-in-statistics-3359d3774b0b`, 2023.

[46] `https://www.naio-technologies.com/en/home/`, 2022.

[47] Eduardo Navas, Roemi Fernández, Delia Sepúlveda, Manuel Armada, and Pablo Gonzalez de Santos. Soft grippers for automatic crop harvesting: A review. *MDPI*, 2021.

[48] Luiz F. P. Oliveira, António P. Moreira, and Manuel F. Silva. Advances in agriculture robotics: A state-of-the-art review and challenges ahead. *MDPI*, 2021.

[49] `https://onrobot.com/`, 2022.

[50] `https://www.oreilly.com/library/view/python-machine-learning/9781783555130/ch03s04.html`, 2023.

[51] Serena Giulia Pala. `https://agronotizie.imagelinenetwork.com/agrimeccanica/2022/02/08/incentivi-40-opportunita-per-un-agricoltura-piu-smart/73891`, 2022.

[52] `https://www.public.harvestai.com/`, 2022.

[53] Shengjiang Quan, Xiao Liang, Hairui Zhu, Masahiro Hirano, and Yuji Yamakawa. Hivtac: A high-speed vision-based tactile sensor for precise and real-time force reconstruction with fewer markers. *MDPI*, 2022.

[54] `https://www.raspberrypi.com/products/raspberry-pi-3-model-b/`, 2022.

[55] `https://www.raussendorf.de/en/fruit-robot.html`, 2022.

[56] `https://robotiq.com/it/`, 2023.

[57] Tatsuya Sakuma, Felix von Drigalski, Ming Ding, Jun Takamatsu, and Tsukasa Ogasawara. A universal gripper using optical sensing to acquire tactile information and membrane deformation. *IEEE*, 2018.

[58] Rob B.N. Scharff, Dirk-Jan Boonstra, Laurence Willemet, Xi Lin, and Michael Wiertlewski. Rapid manufacturing of color-based hemispherical soft tactile fingertips. *RoboSoft 2022*, 2022.

[59] `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html`, 2022.

[60] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html`, 2022.

[61] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RANSACRegressor.html`, 2023.

[62] `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html`, 2022.

[63] Jun Shintake, Vito Cacucciolo, Dario Floreano, and Herbert Shea. Soft robotic grippers. *Advanced Materials*, 2018.

[64] https://www.smooth-on.com/page/durometer-shore-hardness-scale/, 2022.

[65] https://www.robotics247.com/article/soft_robotics_raises_10m_to_help_meet_pandemic_induced_demand_for_food_automation, 2021.

[66] https://www.softroboticsinc.com/, 2022.

[67] https://www.solidworks.com/it/product/solidworks-simulation, 2022.

[68] https://store.arduino.cc/products/arduino-uno-rev3/?gclid=Cj0KCQiAic6eBhCoARIsANlox840SmKcsANukbxGxRVvHAKYanTWFUWyRZ-qJ-Cz-vqtDqEffMSQF58aAheIEALw_wcB, 2022.

[69] https://tertill.com/, 2022.

[70] https://www.tevel-tech.com/, 2022.

[71] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, 2023.

[72] https://www.universal-robots.com, 2022.

[73] Santiago Santos Valle and Josef Kienzle. Agricultural robotics and automated equipment for sustainable crop production. *European Commission*, 2020.

[74] https://www.vitirover.fr/, 2022.

[75] Benjamin Ward-Cherrier, Nicholas Pestell, Luke Cramphorn, Maria Elena Giannaccini Benjamin Winstone, Jonathan Rossiter, and Nathan F.

Lepora. The tactip family: Soft optical tactile sensors with 3d-printed biomimetic morphologies. *Soft Robotics*, 2017.

[76] `http://wiki.ros.org/netft_utils`, 2023.

[77] Akihiko Yamaguchi. Fingervision for tactile behaviors, manipulation, and haptic feedback teleoperation. *IEEJ*, 2018.

[78] Akihiko Yamaguchi and Christopher G Atkeson. Implementing tactile behaviors using fingervision. *IEEE*, 2017.

[79] Wenzhen Yuan, Siyuan Dong, and Edward H. Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *MDPI*, 2017.