



UNIVERSITÀ  
DEGLI STUDI  
DI BRESCIA

DOTTORATO DI RICERCA IN INGEGNERIA MECCANICA E INDUSTRIALE  
ING/IND 12 MISURE MECCANICHE E TERMICHE

XXXIII CICLO

TOWARDS TRUE HUMAN-MACHINE COLLABORATION: THE  
CONCEPT OF META-COLLABORATIVE WORKSTATIONS AND  
A FIRST SOFTWARE PROTOTYPE

CRISTINA NUZZI

PROF.SSA GIOVANNA SANSONI

PROF. GIOVANNI LEGNANI

PROF.SSA LAURA ELEONORA DEPERO

# Sommario

Il presente lavoro di tesi ha riguardato lo sviluppo di un sistema interattivo di comando per manipolatori robotici, basato su una comunicazione tramite gesti delle mani riconosciuti da un sistema di visione intelligente. L'idea alla base del progetto è quella di realizzare un primo prototipo di interfaccia uomo-robot che possa rendere la programmazione e l'utilizzo dei manipolatori, sia industriali che collaborativi, semplice e intuitiva anche per utenti non esperti. Questo permette di interfacciarsi al sistema robotico a prescindere da dove esso si trovi (nella cella robotica se industriale o in un ambiente condiviso se collaborativo), utilizzando un sistema comune a prescindere dal brand del robot adottato.

Questo concetto, già parzialmente previsto dal paradigma Industria 4.0, è il primo passo verso una collaborazione reale tra operatori e macchine, dove i due possono lavorare uniti come un team grazie alla definizione di un sistema di comunicazione (i) naturale da usare per gli umani e (ii) semplice da interpretare per i sistemi meccatronici. Rispetto alle soluzioni che la letteratura e il mercato offrono, questo tipo di collaborazione permetterà la creazione di stazioni di lavoro da me battezzate "*Meta-Collaborative*". Questi sistemi sono progettati per essere una via di mezzo tra le attuali stazioni completamente automatizzate che impiegano robot industriali e quelle denominate collaborative che impiegano Cobot.

Il lavoro di ricerca si è concentrato inizialmente sulla definizione del metodo di comunicazione. In base alla letteratura e alle caratteristiche degli ambienti industriali, ho deciso di utilizzare un linguaggio basato sui gesti delle mani realizzato ad hoc. Grazie ai recenti sviluppi nel campo dell'Intelligenza Artificiale, ho individuato una serie di modelli di Deep Learning adatti allo scopo, in grado di riconoscere i gesti presenti nelle immagini con bassi tempi di inferenza e alti valori di accuratezza di riconoscimento. Con questo obiettivo in mente ho progettato il linguaggio a gesti in modo da renderlo semplice da usare e intuitivo, pur mantenendolo sufficientemente robusto da essere riconosciuto con buone performance dal modello di Deep Learning adottato. Tramite tre campagne sperimentali, svolte nell'arco dei tre anni di dottorato, ho definito il dataset completo di gesti statici implementato nel prototipo finale.

---

Esso costituisce il *dizionario dei gesti* dal quale gli utenti possono selezionare gesti singoli (considerati alla stregua di “simboli”) per costruire, tramite una logica binaria che sfrutta la presenza di entrambe le mani contemporaneamente, comandi personalizzabili. Il modello di Deep Learning adottato è un R-FCN Object Detector pre-allenato sul famoso dataset COCO; pertanto, utilizzando il dataset sperimentale, ho eseguito una procedura di fine-tuning per allenarlo sui miei dati, ottenendo una accuratezza di riconoscimento dei gesti superiore al 90%.

La seconda parte del lavoro si focalizza sulla struttura del software di programmazione realizzato, denominato *MEGURU (MEta-collaborative GestUre-based Robot program bUilder)*, e sulla sua valutazione sperimentale in due diversi scenari. Questo ha permesso di ricavare le impressioni degli utenti e i relativi tempi di risposta nell’utilizzo del sistema, confrontati con quelli ottenuti nell’utilizzo del teach pendant sullo stesso task. Dai risultati si evidenzia come, nonostante l’interfaccia di MEGURU lasci spazio a miglioramenti, gli utenti siano in generale soddisfatti delle sue prestazioni, mostrando in diversi casi tempi di risposta complessivamente inferiori rispetto ai corrispettivi ottenuti usando il teach pendant.

Infine, l’ultima parte della tesi presenta il primo modulo di espansione realizzato per il sistema, denominato “*Hands-Free*”. Questo modulo, che può essere utilizzato singolarmente o come parte di MEGURU, permette di teleoperare l’end-effector del manipolatore robotico utilizzando la skeletonizzazione della mano dell’utente ottenuta tramite il software open-source *OpenPose*. Le immagini della mano dell’operatore sono acquisite in tempo reale da una videocamera RGB: in questo modo, a seconda di dove si trovi l’indice all’interno dell’area di lavoro dell’utente definita a priori (*user workspace*), è possibile muovere l’end-effector nel punto corrispondente all’interno di una seconda area di lavoro in cui il robot opera (*robot workspace*). Per valutare le performance di “*Hands-Free*” sono stati condotti tre esperimenti, volti a ricavare il contributo dei diversi errori introdotti dai vari passaggi di calibrazione e conversione del software. Ciò ha evidenziato come la scelta della videocamera e la sua calibrazione siano fondamentali per ottenere una corrispondenza tra metri e pixel il più possibile precisa.

---

## Abstract

This thesis work concerned the development of an interactive command system for robotic manipulators, based on the establishment of a natural communication by means of hand gestures recognized by an intelligent vision system. The project's idea was to realize a first human-machine interface prototype able to ease both the robot's programming and their use, making it simple and intuitive even for non-expert users. This allows the operators to communicate and operate any robotic system, regardless of where the manipulator is located (i. e. in the robotic cell if it is an industrial robot or in a shared environment if it is a collaborative one) and by using the same system regardless of the robot's brand.

This concept, which was partially foreseen by the Industry 4.0 paradigm, is the first step towards a real collaboration between operators and machines, where the two can work together as a team thanks to the definition of a communication system, which is (i) natural to use for humans and (ii) easy to interpret for mechatronic systems. Compared to the solutions offered by the literature and the market, this type of collaboration may allow the creation of workstations, which I baptized with the name "*Meta-Collaborative*". These workstations are designed to be halfway between the current fully automated workstations, which adopt industrial robots, and the so-called Collaborative ones, which adopt Cobots.

The research work initially focused on the definition of the communication method. According to the literature and to the characteristics of industrial working areas, I choose to adopt a tailor-made hand-gesture language. Thanks to the recent advances in the field of Artificial Intelligence, I identified some Deep Learning models suitable to recognize the gestures in the image frames with low inference times and high recognition accuracy. With this objective in mind, I designed the hand-gesture language in order to make it easy to use and intuitive, while also being robust enough to be recognized with good performances by the Deep Learning model of choice. Through three experimental campaigns carried out during my three years of Ph. D. I defined the complete dataset of static gestures implemented in the final prototype. This resulted in the creation of a gestures dictionary from which users can pick single-hand

---

gestures (considered as “symbols”) to build customizable commands leveraging a binary logic that exploits the presence of both hands at the same time. The Deep Learning model adopted is an R-FCN Object Detector pre-trained on the famous COCO dataset; therefore, using the experimental dataset I performed a fine-tuning procedure to train it on my data, obtaining a gesture recognition accuracy over 90%.

The second part of the work focuses on the structure of the developed programming software called *MEGURU (MEta-collaborative GestUre-based Robot program bUilder)*, and on its experimental evaluation in two different campaigns. From the first one, carried out during the *Meet me Tonight* event of 2019, I obtained (i) the users’ feedbacks to evaluate the user experience and (ii) their response times to determine the intuitiveness of the system. From the second one, carried out in collaboration with the STIIMA group of CNR Milan, I compared the performances of MEGURU and of the teach pendant in the same assembling task, to evaluate if the developed system may be a suitable substitute to traditional programming methods, if not better. The results show that, although MEGURU’s interface leaves room for improvement, users are generally satisfied with its performance, showing in several cases response times overall lower than those obtained using the teach pendant.

Finally, the last part of the thesis presents the first expansion module developed for the system, called “*Hands-Free*”. This module, which can be used as a stand-alone package or as an expansion module of MEGURU, allows to teleoperate the robotic manipulator end-effector using the user’s hand skeletonization obtained by the open-source software OpenPose. The hand images are acquired in real time by an RGB camera: in this way, depending on where the index finger is inside a pre-defined user working area (*user workspace*) it is possible to move the end-effector to the corresponding point inside a second working area where the robot operates (*robot workspace*). In order to evaluate the performance of “Hands-Free” three experiments have been carried out, aimed at obtaining the contribution of the different errors introduced by the various software calibration and conversion steps. This highlighted how the choice of the camera and its calibration are fundamental to obtain a correspondence between meters and pixels as precise as possible.

---

# Acknowledgements

These three years have been a journey I will forever hold dear to me. I never imagined to start this career nor to like it as much as I do: it all began with a phone call while I was lost in the mid of Spain's desert area, with only a backpack with me and absolutely no clue of what my future would have been. One step after the other during that fated August of 2017, I weighed the words of Prof. Giovanna Sansoni and finally decided to accept her proposal. That was how I excitedly yet fearfully started my Ph. D. without any previous knowledge about the academic research world I was so keen to dive into.

At first, it was strange. I was excited to start something new but at the same time, I felt like I was still a student, that everything was still the same except for the extra money at the end of the month. However, Prof. Sansoni quickly showered me with responsibilities, prompted me to act and make decisions and I felt urged to grab that independence I felt I was lacking up until that moment. Therefore, my first steps in the academic world started with the right amount of carefulness and boldness, and from the shy version of me who did not know how to approach the other lab mates nor the superiors, I evolved to the person I am now. I met wonderful people along the way, which I want to thank for being close to me for all these years.

I made bonds with Ph. D. students that I am proud to call my friends even today, even if we are from different places of the world and study different things. Thank you, Marco, Lorenzo, Cristiano, Michelangelo, and Michela, for being my friends during that fated summer school of my first year, where we shared laughs and thoughts, drinks and food, happiness and sadness.

Thank you, Fabrizio, for your everyday support. Albeit we are apart and we are facing different struggles in our lives, you still keep in touch with me even when I feel so down I do not want to talk with anyone. I know I may be a prick sometimes, but you still cope with me no matter what.

---

Thank you, Roberto, Stefano, and Federica for being my coffee buddies. I love working with you and being part of a team that we independently decided to create. Going to work knowing that Stefano will be there to chat about user interfaces and curious news from the world while sharing our meal makes me eager to come to the lab. Knowing that Roberto will show up in the afternoon to grab a coffee, share a funny tale and help me with the everyday courses helps me relax. And knowing that sometimes Federica will come too with her usual bashful attitude to talk about our common sport or to ask me for help makes me feel like a big sis.

Most of all I need to thank my lab mates, because without them these three years would have been very different and, probably, difficult. Thank you, Gabriele, for being yourself around me and helping me being myself too, making me feel day after day like one of you and not just an outsider. You have been a friend and an important support for me. Thank you, Simone, for being there every time I needed a hand with my research; for helping me enjoy these three years one coffee break after the other and for the info you shared with me about the academic world and how to be a researcher in the right way. I admired you before and the more I get to know you, I admired you even more: I hope one day to become like you. Thank you, Matteo, Massimiliano, and Luca, for your supporting comments who helped me polish my work and find out new ways to improve, even if sometimes challenging.

Finally, thank you, Giovanna, for being once again my guide during these three years full of up and downs. You gently guided me to the right path, corrected me when needed and challenged me to do better. If I can shine even a little, it is because of you.

A thank goes also to my family and to my close friends that do not work with me but still supported me nonetheless along the way. Your presence has been and still is of utmost importance for me to be myself; to relax; to avoid drowning in the black hole of anxiety and stress that haunts me at night when I think about my future. I love you even if I may not show it as much as I should.

Thank you.

---

## Contents

Sommario .....	ii
Abstract .....	iv
Acknowledgements .....	vi
Chapter 1. Industry 4.0 and Industrial Robotics.....	4
1.1 The Industry 4.0 paradigm.....	4
1.1.1 The main objectives.....	4
1.1.2 Cyber Physical Systems and Cyber Physical Production Systems .....	6
1.1.3 The enabling technologies.....	8
1.1.4 Integrated and Intelligent Manufacturing.....	9
1.2 How the world supported the Industry 4.0 evolution .....	11
1.2.1 Governments plans and actions outside Europe .....	11
1.2.2 European Governments plans and actions .....	12
1.2.3 Industrial plans launched by worldwide companies .....	14
1.3 Industrial thoughts and reactions to Industry 4.0 .....	15
1.3.1 Germany concerns in 2014.....	15
1.3.2 Global trend in 2018.....	17
1.4 The Italian perspective in 2019.....	18
1.4.1 Global Competitiveness of Italy.....	18
1.4.2 Italian Manufacturing .....	24
1.4.3 The level of digitalization in Italy.....	26
1.5 Robots in industry .....	28
1.5.1 Industrial and Collaborative robots.....	29
Chapter 2. Deep Learning for visual data.....	31
2.1 History pills: the evolution of Deep Learning .....	34



2.1.1	Cybernetics Era (1940 – 1980).....	34
2.1.2	Connectionism (1980 – 1990).....	35
2.1.3	Modern Deep Learning (1990 – present).....	37
2.2	Convolutional Neural Networks.....	38
2.2.1	Convolutional layers.....	39
2.2.2	Pooling layers.....	40
2.2.3	Fully Connected layers.....	41
2.3	Object Detectors.....	42
2.3.1	Region-Based CNNs.....	42
2.3.2	You Only Look Once (YOLO).....	44
2.3.3	Region-based Fully Convolutional Neural Network (R-FCN).....	44
2.3.4	Single Shot Detector (SSD).....	45
2.3.5	Performance comparisons.....	46
Chapter 3.	Human-Robot Interaction.....	47
3.1	Communicating by gestures.....	48
3.2	Gesture Recognition.....	50
3.3	The developed gesture language.....	52
3.3.1	Background research.....	52
3.3.2	Evaluation metrics.....	55
3.3.3	First approach: four two-hands gestures.....	58
3.3.4	Second approach: a big single-hand gestures dataset.....	63
3.3.5	Third approach: the gestures dictionary.....	71
Chapter 4.	The MEGURU System.....	80
4.1	The idea.....	80
4.2	System layout in detail.....	85

4.2.1	State Machine node .....	86
4.2.2	Gesture Recognition node .....	90
4.3	Experimental evaluation .....	91
4.3.1	Evaluating the user experience .....	91
4.3.2	Confronting the teach pendant and MEGURU .....	95
Chapter 5.	The “Hands Free” module .....	102
5.1	Set-up calibration and mapping .....	103
5.1.1	User workspace calibration .....	104
5.1.2	Robot workspace calibration .....	105
5.2	Hand-gesture recognition of “Hands-Free” .....	108
5.2.1	Gesture recognition procedure .....	109
5.2.2	Moving the robot end-effector: positioning procedure and filtering .....	111
5.3	Experimental evaluation .....	112
5.3.1	Evaluation of the hand skeleton estimation error .....	114
5.3.2	Evaluation of the camera error .....	116
5.3.3	Evaluation of the robot error .....	118
5.3.4	Discussion .....	121
Conclusions	.....	123
Bibliography	.....	127

## Chapter 1. Industry 4.0 and Industrial Robotics

Industry 4.0 refers to the fourth industrial revolution that the world is experiencing, which started in Germany in 2014 and gradually spread to the USA and Asia.

The *first industrial revolution*, dating back to the second half of the 1700s, introduced automatic machines to replace manual work and, thanks to them, began low-cost mass production, especially in the textile and metallurgical sectors. The *second industrial revolution*, dating back almost a century later around 1870, refers to the introduction of electricity, chemicals, and oil into industrial production. It was with the *third industrial revolution*, dated around 1970, that information technologies such as computers, microprocessors and telecommunications systems began to be introduced into the world of factory production.

What we are experiencing today, however, is a further step forward. Industrial machines and systems are "*smart*", capable of controlling the process to which they are dedicated independently. But what triggered this idea? Differently from the past, the new revolution started thanks to the consumer end world. In fact, the technologies that digitalized the consumers' lives were absolutely alien for the industrial world, which still used technologies adopted during the third industrial revolution. This clear separation between the two worlds could not continue to exist, creating an almost alienating separation between the two. Therefore, in 2014, the **Industry 4.0** paradigm (I4.0) was proposed.

### 1.1 The Industry 4.0 paradigm

#### 1.1.1 The main objectives

The main objectives of the paradigm as it was defined in 2014 are summarized below according to [1]. These are key points that the new industrial revolution aims to achieve or solve using modern technologies and strategies. They are:

- **Short development periods:** the time needed to *innovate* a product and the time needed to *produce* it must both be reduced. This ensures the company to be on the market immediately (short "time to market").
- **Individualization on demand:** today the customers themselves define the purchase conditions of a certain product, imposing strict specifications on the supplier. This implies that each customer will want a product substantially *customized* to his needs. Therefore, the production will no longer be a mass production in huge batches, but a "*batch size one*" production; in other words, tailor-made on the customer.
- **Flexibility:** considering the characteristics of the *batch size one* production, it is necessary to have a flexible production line to be able to manage the different requests of the customers.
- **Decentralization:** organizational hierarchies must be reduced in size to achieve faster decisions on the production line, thus ensuring a short production time. *Decentralization* means the ability of a system to identify itself and connect to a more powerful centralized system, to which it will communicate its position and status. In this way, the computational power can also be in a different place, not necessarily all on board. This is where *Cloud Computing* and *Big Data* management intervene to better manage communication between systems.
- **Resource efficiency:** better management of the raw materials and energy needed for production (electricity, heat, fuel, etc.) not only saves money but also considerably reduces the environmental impact of their use.

To achieve these objectives, the authors of [1] proposed four approaches, which may be adopted individually or jointly. Although these were proposed in 2014, a high number of companies has not yet transformed their production plant to this day, making these four fundamental actions still necessary steps. These are:

- **Increase industrial automation:** more technologically advanced machinery must be introduced in industrial plants. They must be "*smart*", meaning they must autonomously manage their processes, making them more versatile.

- **Digitalization and Networking:** industrial sensors and digital devices in general must be used more intensively (*data digitalization*), interconnecting them within the factory to ensure fast and efficient data management (*networking*). As a result, production processes can be managed quickly and with reliable data. This is also where simulation technologies, data management systems, cybersecurity protocols and advanced techniques such as augmented reality are used as means to digitize processes as much as possible.
- **Miniaturization:** compared to when computers, in the third industrial revolution, were introduced in the industrial world, today's computers occupy a very small space and possess an incredibly greater computational capacity. This makes it possible to create even smaller computers, called *embedded technologies*, which occupy the same space as a microcontroller. The use of these technologies would allow for even better space management and better industrial logistics.
- **Better integration:** this refers to a better integration between sensors and digital technologies within the factory structure, and also to a better integration between various research branches and economic sectors to improve production and plant's management.

### 1.1.2 Cyber Physical Systems and Cyber Physical Production Systems

In particular, the paradigm aims to create **Cyber Physical Systems (CPS)** and **Cyber Physical Production Systems (CPPS)** [2] [3].

A *CPS* is a "smart" physical object that has embedded software that provides it suitable computational power. These devices are specifically designed for a certain task, which they can carry out autonomously thanks to their limited self-managing capabilities.

A *CPPS* is a manufacturing equipment made powerful by the specific hardware and software it has on board that grants it a high computational power. CPPSs know their internal status, their processing capacity, and their internal configurations, allowing them to make their own decisions depending on the task they are completing and the specific situation they are in. These machines are the key to achieving true *batch size one* production, because each of them can be used flexibly to create the desired product, either alone or in combination with other CPPSs.

Using a combination of CPSs and CPPSs will create a *highly interconnected* and *multi-agent* industrial environment. The idea is that manufacturing production will become **decentralized**: a production line where CPPSs are able to carry out their tasks independently by interconnecting with other CPPSs and self-organizing.

Obviously, even if both technologies possess potential and several positive aspects, they also pose several unresolved problems and open challenges. The most troublesome are the following:

- **Time management** issues in IT networks related to when the information will be correctly received with respect to the real time the information has been sent, creating dangerous delays that may harm the production line and the equipment itself.
- The necessary **standardization** of CPSs/CPPSs is complicated because different IT technologies are involved with different standards and requirements.
- **Safety issues** related to the wellbeing of both people and physical equipment and **cybersecurity issues** related to software and data protection.
- CPPSs must be at least **partially autonomous systems**, requiring different methods to provide them with the ability to be *context* and *surrounding aware*, meaning they should be able to perceive and interpret the environment and the physical objects that interact with them.
- CPPSs must be **cooperative**, meaning that they must be able to work with each other, learn collectively and recognize the environment and the different events that may happen in a distributed way.
- They must **manage the scheduling of operations** in a robust way with respect to unforeseen disturbances and/or errors during production.
- Intuitive and smart **Human Machine Interfaces** (HMIs) must be developed to allow human operators to operate the devices in the best possible way.

These are still activities on which research is currently working on, especially in recent years.

### **1.1.3 The enabling technologies**

To reach the paradigm goals, a set of technologies must be adopted. These are called “enabling technologies” which during the years have been researched and optimized by researchers and manufacturers to adopt them in industrial plants safely [4]. The study proposed in [5] describes the enabling technologies considered necessary in 2019, which are the following:

- **Industrial Internet of Things (IIoT):** a computer network between physical objects (sensors, machines, buildings, etc.) that allows interaction and cooperation between them to exchange useful information.
- **Additive Manufacturing (AM):** innovative process of creating products using a 3D printer. The CAD model of the product to be made is realized by joining layers of material one on top of the other. It allows to create truly customized objects with different technological features depending on the material used for printing.
- **Big Data and Advanced Analytics (BD/AA):** the digital devices of an Industry 4.0 company are interconnected and exchange a large amount of data (*BD*), which must therefore be managed and processed in order to be effectively used to improve the production process (*AA*). Using analysis techniques to filter, correlate and evaluate data, users can immediately take decisions on the production. This can be done either on board of the device or within a management and analysis software that accumulates data from all devices and analyzes it.
- **Virtual and Augmented Reality (VR/AR):** *VR* allows to create a simulated environment with which the user, who is completely immersed in it, can interact in three dimensions and where each element that is part of it has a perceptible spatial dimension. The interactions allow the user to control the objects in the virtual world in a realistic way and in real time. On the other hand, *AR* refers to those advanced human-computer interaction techniques that allow the user to insert data and virtual functionalities inside real objects belonging to the physical world. The user, using suitable devices, can see and interact with the virtual data belonging to the physical object and obtain effects on the real world.
- **Cloud Manufacturing (CM):** closely related to the use of *BD* and *AA*, *CM* refers to the ability to keep and process the large amount of data no longer on high-performance and

expensive devices on site but *in the cloud*, simply paying the use of the service to the provider. This also allows the company to connect to its suppliers and eventually to its customers, providing them with a digital support service.

- **Collaborative Robotics (CR):** a robotic system or a machine that complies with technical and safety specifications to work effectively with human beings, without risking the safety of both human beings and the plant's equipment (including themselves). Humans and machines can therefore collaborate and interact physically with each other in a shared environment.

#### *1.1.4 Integrated and Intelligent Manufacturing*

As shown in [6], the modern trend in manufacturing is called **i<sup>2</sup>M**. The concept expands the classic definition of CPSs and CPPSs and better defines the two key points behind it: **integration** and **intelligence**.

Today, Mechanical and Digital devices are considered as one under the name of *Computer-Integrated Manufacturing Systems* (CIMS). Thanks to the development of the Industrial Internet of Things, these systems now also comprehend information, required to truly connect, and integrate the different devices of the plant. CIMS can acquire and process high volumes of data in real time, a processing that is achieved using three types of integration between devices:

1. **Vertical integration:** it addresses the issue of seamless connectivity among all the elements that are included in the product life cycle within an organization. Activities in marketing, design, engineering, production, and sales are all closely integrated. Technologies such as the *manufacturing execution system* (MES) and the *computer-aided process planning* (CAPP) can thus be better utilized to support information and knowledge sharing within the company. In this way, resources within the company, including but not limited to information, data, capital, and human resources, can be used more effectively and efficiently.
2. **Horizontal Integration:** it occurs when a company is closely integrated with its suppliers and partners. Modern industry has already adopted supply-chain management



technology, such that a horizontal value network has been established in many industry sectors. However, challenges still exist in terms of efficiency, intellectual property protection, the establishment of common standards and knowledge sharing. With the implementation of an advanced knowledge base and an *Industrial Internet*, those barriers can potentially be removed. A common knowledge network platform with practical protocols and standards is needed to further enhance the effectiveness and quality of this type of integration.

3. **End-to-End Integration:** this is probably the most active area in the new age of manufacturing. In this case, machine-to-machine integration is provided so that machines are truly an integral part of the production line. The second key aspect is that the integration of customers into the manufacturing system is now possible, thus allowing engineers to obtain feedback from customers easily and quickly. Product-to-service integration is also possible, allowing the manufacturer to monitor the condition of the product in use. This adds value to the customer service of the product and builds a close relationship between customers and suppliers.

Since all the units of a manufacturing system have been integrated into a common system, process decisions become difficult because of the impressive amount of data to be processed. Therefore, the concept of *intelligence* is required to analyze and use the information acquired from the integrated system.

Intelligence is related to sensing, decision-making and action. If sensing can be achieved by using smart sensors, it is not as easy to automatically process the data and obtain as a result a valid and suitable decision about a certain situation or action to be taken. In this sense, Big Data Analytics, Machine Learning & Artificial Intelligence Techniques and Cloud Computing are technologies extremely important and useful to develop a true intelligent system.

Thanks to Big Data Analytics, information can be automatically extracted from the high volume of data acquired by sensors, uncovering hidden clusters and correlations unseen by the humans. This is usually achieved by adopting Machine Learning techniques especially designed to process data.

Artificial Intelligence techniques are necessary to give machines a touch of intelligence, making them able to carry out their tasks autonomously or even learn by their own how to work and/or adapt to the environment.

Finally, Cloud Computing provides all these technologies with the necessary computational power without the need to buy an expensive computer: the idea of “*software as a service*” (SaaS). The Cloud is accessible by every device of the company thanks to IIoT connectivity, allowing them to share knowledge in real-time. This is also true for human knowledge, that can be shared and accessed easily thanks to the Cloud platform. By combining the two types of knowledge sharing it is possible to obtain a marketplace for software and information sharing, an idea called “*platform as a service*” (PaaS).

## 1.2 How the world supported the Industry 4.0 evolution

Although the official name of the new industrial revolution has been created in 2014 in Germany, the Industry 4.0 idea started taking form before that, when the IoT technology (which was mostly aimed at consumer end devices) and CPSs started to be popular. Their popularity gained the interest of different Governments and industries worldwide, which acted to benefit in time from the new digital revolution they were foreseeing [7].

### 1.2.1 Governments plans and actions outside Europe

- **U.S.A. (2011):** the Government began a series of discussions, actions and recommendations entitled “*Advanced Manufacturing Partnership (AMP)*” since 2011. The aim was to prepare the nation to lead the next generation of manufacturing [8].
- **South Korea (2014):** the Government announced a program called “*Innovation in Manufacturing 3.0*” which emphasizes four propulsion strategies and assignments to innovate Korean manufacturing [9].
- **China, 2015:** the Government defines the “*Made in China 2025*” strategy and the “*Internet Plus*” plan to prioritize ten fields of manufacturing to accelerate the digitalization and industrialization of the country [10].

- **Japan (2015):** the Government adopts the “*5th Science and Technology Basic Plan*” focused especially to the manufacturing sector to realize their idea of a “*Super Smart Society*” [11].
- **Singapore (2016):** the Government commits 19 billion dollars to the plan “*Research Innovation and Enterprise 2020*”. Eight key industries have been selected within the advanced manufacturing and engineering domain to lead the transformation [12].

### 1.2.2 European Governments plans and actions

The European commission launched the Digital Transformation Monitor (DTM) website to monitor the national initiatives of European States in detail towards the transformation of the manufacturing sector according to the I4.0 paradigm. The following information has been summarized from the several documents available on the DTM website [13].

- **Germany (2011):** the Government approves the “*High-Tech Strategy 2020*” action plan in 2012, which annually commits billions of euros for the development of cutting-edge technologies. As one of the ten future projects in this plan, the “*Industrie 4.0*” project presented in 2011 represents the German ambitions in the manufacturing sector [14].
- **United Kingdom (2011):** in 2011 the program “*High Value Manufacturing Catapult*” starts. It was a collaboration program between public and private partners to develop advanced technologies for the manufacturing industry. It was only in 2013 that the Government presented a long-term picture for its manufacturing sector until the year of 2050, called the “*Future of Manufacturing*”. It aims to provide a refocused and rebalanced policy for supporting the growth and resilience of UK manufacturing over the coming decades [15].
- **France (2013):** the Government launches a strategic review named “*La Nouvelle France Industrielle*”, in which 34 sector-based initiatives are defined as France’s industrial policy priorities [16]. In 2015, France launches the program “*Industrie du Futur*” to support the digital transformation of industries.
- **Sweden (2013):** the Government launches “*Produktion 2030*”, a program aimed to transform the industry into a sustainable and customizable modern sector, promoting services and products of a higher quality level. To obtain this result, they focused on the enhancement of digital competences and knowledges of workers and on the funding of

several research projects. The program was supported by the local partner VISINNOVA (leader in the technology innovation in Sweden) and several other partner industries.

- **Belgium (2013):** the Government starts the “*Made Different*” initiative to support and push the manufacturing sector towards the “*Factories of the Future*”. The program aims at enhancing the Industry 4.0 technologies awareness of workers and their digital competences, adopting customized coaching programs to support the companies in this evolution. The program also focused on the adoption of efficient and green technologies to reduce waste and production costs.
- **Netherlands (2014):** the Government proposes the “*Smart Industry*” program, focused on the people training and skill development both at school and on the job site. As part of this idea, they built “*Field Labs*” throughout the country: joint labs with industries and Universities to develop prototypes and technological devices suitable for the Industry 4.0 paradigm.
- **Denmark (2014):** the Government issues the “*Manufacturing Academy of Denmark (MADE)*” program, in which 5 important Universities of the country, several manufacturing industries and 3 Registered Training Organizations take part. The aim is to make the manufacturing sector an Academy to boost people’s knowledge and technical skills, creating an ecosystem strongly based on academic research and innovation. MADE project has been funded from 2014 to 2019 with both public and private funds.
- **European Commission (2014):** the EU Commission launched the new contractual *Public-Private Partnership (PPP)* on “*Factories of the Future (FoF)*” under the *Horizon 2020* program that plans to provide nearly 80 billion euros of available funding from 2014 to 2020 [17].
- **Spain (2015):** the Government launches the program “*Industria Conectada 4.0*” to support the evolution of the industry sector towards the fourth revolution. It promotes the development of new technologies and the diffusion of technical skills. Unfortunately, even if in 2019 the Government allocated 97 billion of euros for the program, the local private initiatives worked better during the years, obtaining faster results closely related to the local territory.

- **Czech Republic (2016):** the Government approves the “*Průmysl 4.0*” program to push the local companies to take part in the global supply chains, integrate the Industry 4.0 paradigm in the national industries and enhance the cooperation between research institutes and companies. The goal is to develop software, devices, and patents for the manufacturing sector. The project also aims at transferring the know-how from academies to industries with training programs for the workers.
- **Portugal (2017):** the Government approves the action plan “*Indústria 4.0*”. It is focused on the digitalization of companies and their innovation, as well as on the enhancement of the workers know-how. The strategy aims at the proliferation of technological start-ups to make the country attractive for international investors both legally and financially.
- **Italy (2017):** the Government issues the plan “*Industria 4.0*” aimed at boosting the digitalization and technological development in the country by promoting convenient investments in venture capitals and technological start-ups. One of the main cores of the program revolves around the formation of both the workers and the students on digital competences and technical skills suitable for the future of industry. Because of this, they focused on training centers and Digital Innovation Hubs, as well as funding Technology Clusters and Industrial Ph. D. programs.

### *1.2.3 Industrial plans launched by worldwide companies*

Private funding and programs have been started as well alongside the National ones to further push the transformation and the scientific research on the enabling technologies required by the I4.0 paradigm.

In 2014, AT&T, Cisco, General Electric, IBM, and Intel founded the “*Industrial Internet Consortium (IIC)*” to catalyze and coordinate the priorities and the enabling technologies of the Industrial Internet. Meanwhile, other big companies such as Siemens, Hitachi, Bosch, Panasonic, Honeywell, Mitsubishi Electric, ABB, Schneider Electric and Emerson Electric invested heavily in IoT and CPS related projects [7].

## 1.3 Industrial thoughts and reactions to Industry 4.0

Although the I4.0 paradigm sounds amazing in theory, to make it a reality was not that easy. The industrial world was skeptical at first, and only in late years the adoption of I4.0 technologies has started to become common even for small and medium enterprises (SMEs).

As highlighted in [7] even in 2017 is evident that industrial companies are still skeptical about implementing the I4.0 paradigm, both because they are not sure of the real benefits of it and because the investments to adopt these technologies are very high. Industries are still a long way from scientific research advances: not only they do not even consider it as a mean to implement new industrial technologies, but also they rarely get involved in projects related to Academic research. Moreover, industries are not willing to adopt the few standards and experimental solutions proposed by scientific research.

### 1.3.1 *Germany concerns in 2014*

Several studies have been conducted to understand people fear and skepticism towards the innovation. One of the first ones has been conducted in Germany in 2014 [18] to reveal the reasons for the adaptation and refusal of I4.0 technologies and practices from a managerial point of view. The authors conducted face-to-face interviews with managers from both local companies and consulting business.

One of the most important problems that haunted German managers in 2014 was the dilemma between scale and scope of the production. Back then, this issue was primarily addressed by establishing product families to incorporate **flexibility** into mass production [19], but remained one of the main concerns. In fact, the product design and development represent only 5-10% of the production process but determines more than 80% of the actual cost of the product. This means that the desired flexibility of a product family must be determined at a very early stage but, at the same time, the flexibility benefit is not easy to quantify, hence it is often not included in a classical investment analysis of new machinery [20].

Adding flexibility means to produce customized products as well as standardized ones, hence the problem of **warranties and certification** of these customized products arise. In fact, to properly certificate a product, extensive tests have to be carried out, which is not a practical nor economical procedure, especially if it had to be done on each customized product instead of on the few needed to certify a whole batch. In this regard, manufacturers still lacked knowledge and experience in terms of product safety and component failure, meaning that a reliable customized product was extremely difficult to produce. This led the production to still be heavily characterized by mass production instead of customized.

One of the main reasons that determined the low implementation of I4.0 technologies in 2014 was the **lack of powerful IT systems** and the **absence of a true integration** between each other, as well as the **inadequate employee knowledge** of both production processes and digital skills. For these reasons, most companies did not adhere immediately to the changes introduced by the paradigm.

Another concern was on the topic of **autonomous systems**. The general thought was that they could be extremely effective to better the production but at the same time, they were not reliable enough to be used at their full theoretical potential, still requiring heavy human intervention. Even if the idea of self-organizing machinery had great potential, there were considerable obstacles to overcome to effectively surpass traditional production methods and technologies.

One of the main revolutions introduced by the paradigm was about **information sharing** throughout both the company devices and whole the value chain. In the first case, the technology was not yet ready to successfully adopt interconnected CPSs, while in the second case the concept poses the problem of disclosing sensible information about the company production process to partners. In fact, most of German companies still refused to do it, also because information sharing was a cost which neither the company nor its partners were willing to bear.

The adoption of **advanced simulation and modeling tools** for virtual production strongly depends on the industry and company size. This unfortunately means that SMEs often could not afford expensive workstations for simulation and modeling purposes, let alone the required knowledge to properly use them.

### *1.3.2 Global trend in 2018*

In 2018, the enabling technologies defined by the paradigm were starting to become a reality, thanks to the efforts of both IT industries and academic research. This transformation, however, was not mirrored by most firms worldwide that still refused to adopt advanced I4.0 technologies or were unable to embrace this transformation fully due to the technology costs and the unavailability of the required know-how. As a 2018 study claims [21], the real question at that point was how manufacturers would have made the transition from their present state to their future digitalized state. To answer that question and ease managers' doubts, the idea presented by the authors was to propose a *roadmap* useful to define short, medium, and long-term approaches to evolve the industry.

In fact, it was common knowledge that the benefits introduced by I4.0 would outweigh the costs, but that was particularly true for excellent manufacturers that (i) had the necessary experience and human labor to create and implement the technology trends and (ii) had adequate support from stakeholders to invest in innovation. Moreover, carefully planning the I4.0 transition to balance its risks and possible issues was the best strategy big companies adopted to successfully embrace the digitalization.

Unfortunately, that was not the case for most of the industries, which still are of medium-small size. Not to mention that the manufacturing industry is a network of "*typical*" manufacturers that cannot see the whole picture, investing blindly into new technology without carefully assessing which could really contribute to their company improvement. **SMEs where usually not competent enough to digitize their whole production**, hence limitedly invested in the digitalization to transform only some of their production processes or their warehousing.



However, to fully benefit from the I4.0 transformation, SMEs must better comprehend both their new technologies and their own production chain, which is something that cannot be achieved in short times.

## 1.4 The Italian perspective in 2019

### 1.4.1 *Global Competitiveness of Italy*

The **Global Competitiveness Index 4.0 (GCI 4.0)** [22] provides guidance on what matters for long-term growth and can inform policy choices; help shape holistic economic strategies and monitor progress over time. In this case, “*competitiveness*” stands for the attributes and qualities of an economy that allow for a more efficient use of factors of production, hence the meaning is closely related to productivity. The concept is anchored in growth accounting theory, which measures growth as the sum of growth in the factors of production (labor and capital) and of total factor productivity (TFP), which measures factors that cannot be explained by labor, capital or other inputs. The GCI measures what drives TFP since productivity gains are the most important determinant of long-term economic growth.

The framework places a bonus on factors that will grow in significance as the *Fourth Industrial Revolution* (4IR) gathers pace, namely human capital, agility, resilience, and innovation. It is organized into 12 main drivers of productivity, or “*pillars*”, namely: (i) institutions; (ii) infrastructure; (iii) ICT adoption; (iv) macroeconomic stability; (v) health; (vi) skills; (vii) product market; (viii) labor market; (ix) financial system; (x) market size; (xi) business dynamism and (xii) innovation capability.

For each pillar, a set of indicators are present to address specific areas of the main topic of the pillar. For example, the “Institutions” pillar is subdivided into 8 subcategories which possess a variable number of sub-subcategories as well depending on the topic. For each pillar, the total score achieved by a country is calculated as the mean of the different subcategories scores, which are in turn calculated as the mean of the different sub-subcategories scores. Finally, the overall GCI 4.0 score is calculated as the average of the scores of the 12 pillars.

Table 1 shows the 2019 top-ranking country for each pillar. It is worth noting that for two pillars (*Macroeconomic Stability* and *Health*) multiple countries achieved the best score. The top ten countries ranked according to their 2019 GCI 4.0 score are reported in Figure 1, while the European Union countries ranking is shown in Figure 2.

Italy achieved a total score of 71.5 points, which makes it the 30<sup>th</sup> country of the world and the 12<sup>th</sup> European Union country. A visual representation of the European Union GCI 4.0 scoring is shown in Figure 3.

*Table 1. Pillars best performing countries and top values achieved.*

<b>Pillars</b>	<b>Best performer</b>	<b>Value</b>
Institutions	Finland	81/100
Infrastructure	Singapore	95/100
ICT Adoption	Republic of Korea	93/100
Macroeconomic Stability	Multiple (33 countries)	100/100
Health	Multiple (4 countries)	100/100
Skills	Switzerland	87/100
Product Market	Hong Kong	82/100
Labor Market	Singapore	81/100
Financial System	Hong Kong	91/100
Market Size	China	100/100
Business Dynamism	United States	84/100
Innovation Capability	Germany	100/100

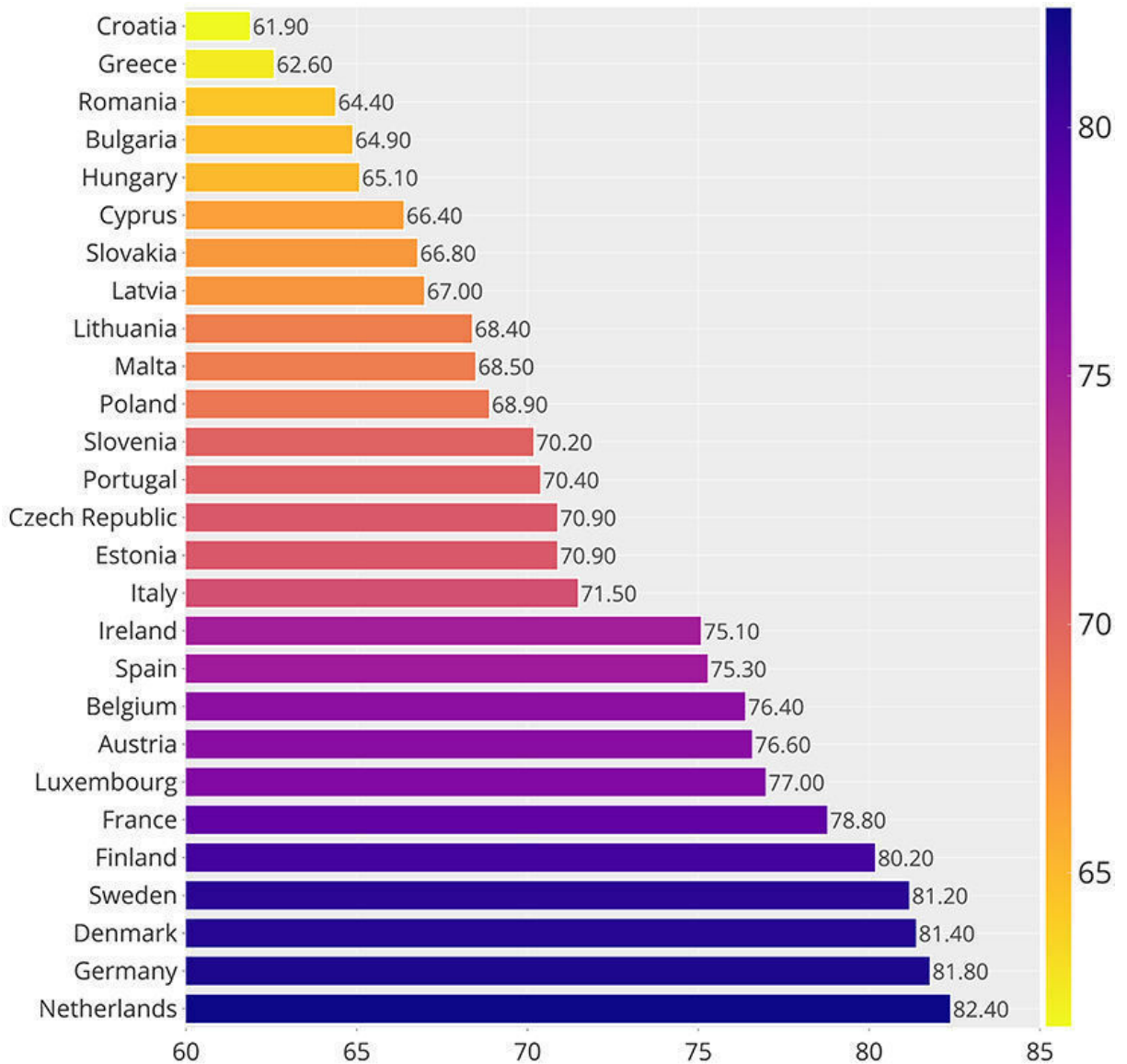


Figure 1. Bar plot of the top ten countries according to their GCI 4.0 score.

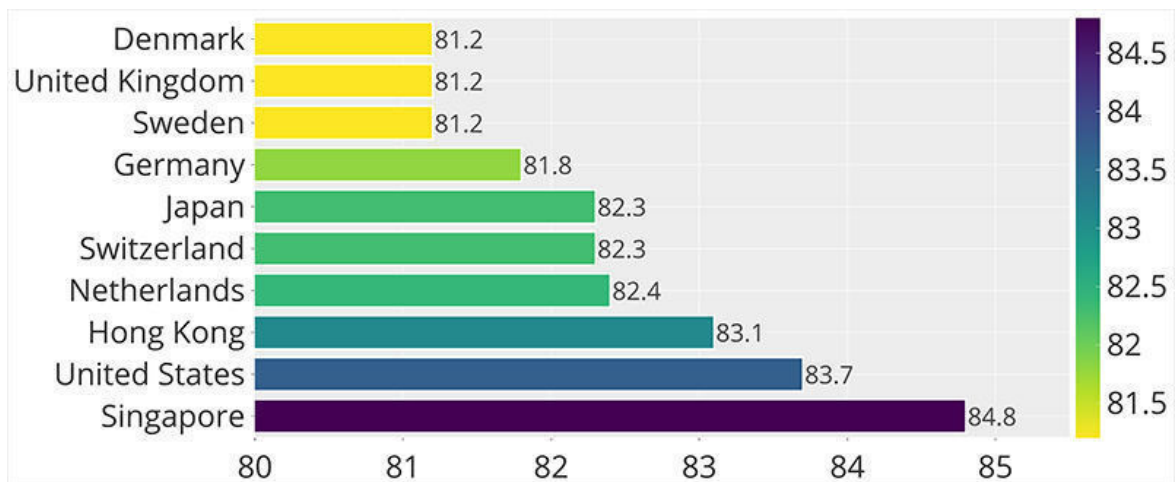


Figure 2. Bar plot of European Union countries ranked according to their GCI 4.0 score.



Figure 3. Map of Europe where European Union countries are colored according to their GCI 4.0 score.

Figure 4 shows the scores of Italy on the different pillars. It is worth noting that values are approximated in excess, which explains why Italy Health rank is shown as 100/100 even if it is not a best performer (actual score: 99.6). Italy achieves the best worldwide rank in a set of sub-subcategories, which are shown in Table 2.

# Performance Overview 2019

Key ◊ Previous edition ◻ High-income group average □ Europe and North America average

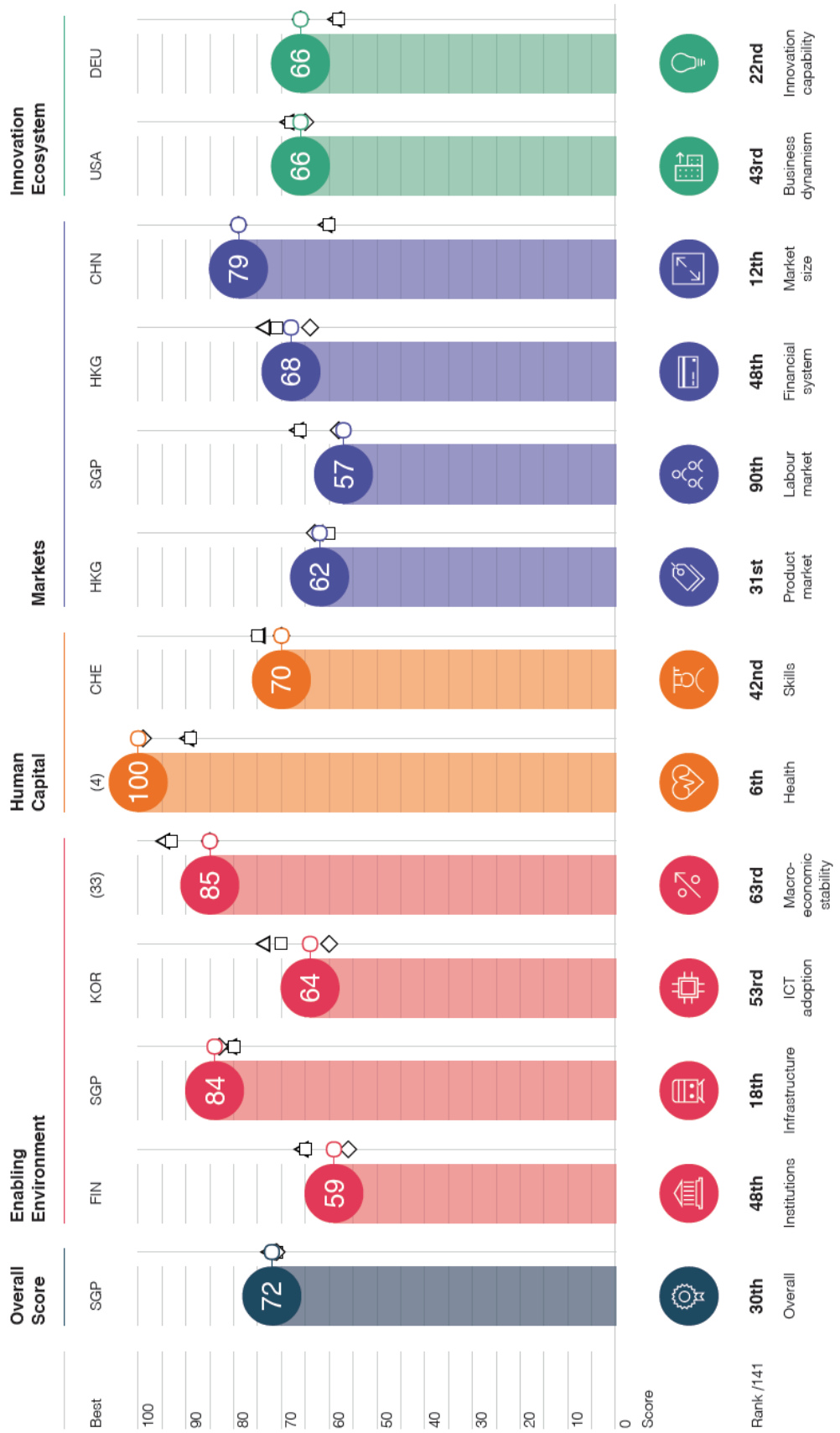


Figure 4. Italy performance overview taken from 2019 GCI 4.0 report.

Table 2. Italy top scores in sub-subcategories, divided by main pillar.

Pillar	Subcategory	Value
Institutions	Energy efficiency regulation	89.2/100
Infrastructure	Railroad density	100/100
Infrastructure	Electricity access	100/100
Infrastructure	Exposure to unsafe drinking water	100/100
ICT Adoption	Mobile telephone subscriptions	100/100
Macroeconomic Stability	Inflation	100/100
Financial System	Insurance premium	100/100
Financial System	Credit gap	100/100
Innovation Capability	State of cluster development	74.9/100
Innovation Capability	Scientific publication score	100/100

Italy's performance has slightly improved compared to 2018, increasing in score by 0.7 and moving up one rank to reach the 30<sup>th</sup> position globally. Italy's performance in 2019 is driven mainly by small advances in the financial system (+3.3 points, 48<sup>th</sup>), where non-performing loans are being gradually absorbed (-2.7% this year), and access to finance to both SMEs and venture capital are slightly improved (+4.5 and +4.8, respectively), though starting from a low base (119<sup>th</sup> and 111<sup>th</sup>, respectively). Similarly, the efficiency of the legal framework has recorded slightly higher scores (+5.1 points, yet again from a low base, 132<sup>nd</sup>), and ICT adoption (+4.2, 53<sup>rd</sup>) and Infrastructure (+1, 18<sup>th</sup>) have gradually improved over the past few years. At the same time, **Italy maintains competitive advantages in terms of Innovation capability** (65.5, 22<sup>nd</sup>) and **Health standards** (99.6, 6<sup>th</sup>).

Yet some bottlenecks are still hindering Italy's competitiveness. Among them, **high public debt** (132% of GDP) represents a looming risk and a burden for economic policy; the **labor market** (56.6, 90<sup>th</sup>) remains to a large extent dual (too rigid in some segments and too precarious in others), despite some recent reforms; **taxes on labor** are high by international comparison (130<sup>th</sup>); and **talent is not sufficiently rewarded** (103<sup>rd</sup>).

**Institutional quality** (58.6, 48<sup>th</sup>) attains a mixed result, combining some positive factors and some areas for improvement. While Italy is a relatively **safe country**, with one of the lowest homicide rates in the World (0.7 cases per 100,000 people, 20<sup>th</sup>) the **Government's capacity to adapt to changes is limited** (28.9, 128<sup>th</sup>) and there is **insufficient administrative efficiency** (45.3, 96<sup>th</sup>).

### *1.4.2 Italian Manufacturing*

In Italy, Manufacturing is the most important sector and accounts for 88% of total production. The biggest segments within Manufacturing are: (i) metallurgy and fabricated metal products (14% of total production); (ii) machinery and equipment (12%); (iii) food, drink and tobacco (10%); (iv) rubber and plastics products and non-metallic mineral products (9%); (v) textile, clothing and leather (8%); (vi) transport equipment (7%); (vii) and other manufacturing, repair and installation of machinery and equipment (7%) [23].

The Italian Flagship Project *Factories of the Future* ("*La Fabbrica del Futuro – Piattaforma manifatturiera nazionale*") is one of the 14 Flagship Projects that started in January 2012 and lasted till December 2018 with a total funding of 10 million euro. The Flagship project defined five strategic macro-objectives for factories of the future to be pursued thanks to the development, enhancement, and application of key enabling technologies. The strategic macro-objectives took inspiration from the research priorities identified at international level, while considering the evolution of the global industrial contexts and, above all, the peculiarities of the Italian manufacturing context [24].

Five strategic macro-objectives were defined for *Factories of the Future*:

1. **Evolutionary and Reconfigurable Factory:** to stay competitive in dynamic production contexts, Italian factories need to react and evolve by exploiting flexibility, reconfigurability, changeability, and scalability. These changes affect both the organizational structure and the production process at the same time.
2. **Sustainable Factory:** green production has a strong impact on the global scenario. Green products are characterized by a limited environmental impact during their life cycle,

including the production phase. This means that factories need to change their production processes to guarantee limited energy consumption and waste, exploiting green energy, innovative natural materials and resources, and energy cogeneration and re-utilization. Sustainable production also means to support the local economy by integrating worker skills and dedicate time and resources to improve them. Furthermore, there is the need to manage the final stages of the product life cycle by implementing product de-manufacturing, re-manufacturing, reuse, recycling, and recovery. This will lead to a new type of factories called “*de-production factories*”.

3. **Factory for the People:** human workers are the core of industrial production. Considering the societal and demographic changes of late years, Italian manufacturing must integrate into their production structure not only young employees but also workers with relevant accumulated knowledge. Moreover, operators must be trained in a multidisciplinary way to flexibly manage the planning and execution of complex production plants. This also means that technologies, machinery, and workplaces must be designed and adapted to cope up with these changes according to the I4.0 paradigm. In fact, robotics and automation technologies will make human-machine interaction even more relevant in the future of manufacturing. New forms of interactions must be investigated to better exploit human-machine cooperation in a shared and safe manufacturing environment, with a focus on safety of both humans and machines and ergonomics.
4. **Factory for Customized and Personalized Products:** according to the I4.0 paradigm, manufacturing production will be less focused on mass production and more focused on *batch size one* production, meaning personalized products and services that are difficult to replicate. The Italian manufacturing industry is traditionally focused on meeting the customer requirements by exploiting process and product know-how together with an attitude for innovation, a concept which is particularly relevant in sectors such as textiles, wearing apparel, footwear, glasses and accessories, luxury goods, and furniture. This tailor-made production (called “*customer-driven*”) requires firms to innovate faster their production processes by using modern technologies and approaches.
5. **Advanced-Performance Factory:** to increase factory performance, it is of utmost importance to minimize the inefficiencies related to external logistics, management of



inter-operational buffers, transformation processes and their parameters, management and maintenance policies, software and hardware tools, quality inspection and control techniques. This innovation requires to adopt smart sensors, innovative mechatronic components, and ICT platforms, as well as the adoption of Advanced Analytics tools to identify beforehand failures, anomalies, and disturbances, moving the industry to an adaptive production model with reduced production delays.

The main enabling technologies considered to be relevant for the macro-objectives of the Flagship are:

- Artificial Intelligence, digital twins, and digital factory technologies for intelligent factories
- Production technologies
- De-Manufacturing and Material Recovery technologies
- Factory Reconfiguration technologies
- Control Technologies of Production Resources and Systems
- Resource Management and Maintenance technologies
- Technologies for Monitoring and Quality Control
- Human-Machine Interaction technologies.

### *1.4.3 The level of digitalization in Italy*

The research presented in [5] in 2019 support the claim described in Section 1.3.2. In the study, Italian SMEs were interviewed to comprehend the level of adoption of a certain technology compared to the level of knowledge of the same technology. The results were disappointing yet not surprising: most of the respondents showed **limited to none knowledge of the enabling technologies** except for the IIoT, which was the first to gain popularity during the years and the most close to the consumer-end world. The study shows that bigger companies intensively adopt I4.0 technologies and methods; hence, the level of knowledge of their employees is higher compared to SMEs, which present the opposite trend.

Surprisingly, IIoT adoption is less than Additive Manufacturing adoption despite the former being more known among employees. As for Big Data, Advanced Analytics, Augmented and Virtual Reality and Collaborative Robotics, **the adoption level is low** and reflects the level of knowledge of workers. The research also shows that SMEs have not activated any new technology-related project, while larger companies have at least activated one. This finding probably relates to the resources availability of SMEs, which is very limited compared to large companies. Finally, if the IT role is strategic and not only operational, a substantial difference can be found in the adoption and on the coverage of IT infrastructure, even for companies that do not adopt a I4.0 technology yet.

Even if I4.0 involves all areas of production, starting from the production line up to managerial and business roles, **for Italy the fourth revolution is still a technological one rather than organizational**. This highlights how Italy is still at the early stage of the transformation, both because the Government actions happened later compared to other countries and because companies have limited resources to access technologies quickly, hence sticking to traditional manufacturing organizations and production processes. It is also worth noting that financial and tax aids from the Government are still lacking, hence slowing the transformation of SMEs.

The **knowledge level of workers** is also one of the barriers that limited Italian digitalization, since technical roles such as Data Scientist, IIoT expert, AR and VR software engineer, and similar figures are extremely rare even today. The absence of a digital expert for the different enabling technologies means that the company, even if it is willing to invest in I4.0, cannot do it or benefit from it in full.

It is also interesting to note that companies that have already adopted at least one technology perceive higher benefits from it compared to companies who have not adopted any. This is not surprising: digitalization completely changes the face of manufacturing production, thus enhancing both the general feeling towards I4.0 and the tendency to adopt more technologies. **The more technologies have been adopted, the more the company is willing to invest in I4.0.**

## 1.5 Robots in industry

The most famous definition of a robot has been given by the *Robot Institute of America* in 1979: “A robot is defined as a reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks.” It is made evident that robots are **reprogrammable**, **autonomous** and **multi-purpose** machines, hence they are very different from traditional industrial machines that are usually designed and programmed to perform only one task and cannot be adapted to suit different ones. In this sense, robots are more similar to humans than to simple machines, combining the best of the two worlds: on one hand, the flexibility and autonomous capabilities of a human being and, on the other hand, the precision and repeatability of a machine [25].

Robots evolved during the years and can be categorized in four “*generations*” [26].

1. **First generation (1950-1967):** robots were stationary non-programmable electromechanical devices with no sensors, unable to control the modality of task execution and to communicate with the external environment. They mostly used pneumatic actuators and logic gates as automatic regulators; hence, they produced a lot of noise when active.
2. **Second generation (1968-1977):** robots were basic programmable machines with limited possibilities of self-adaptive behavior and elementary capabilities to recognize the external world. They used servo-controllers to perform point-to-point movements and continuous paths as well. Their intelligence was made of Programmable Logic Controllers (PLCs) which are widely used even nowadays. They were not versatile; hence, they were usually application-specific devices because a substantial modification of their controller was difficult to be made in a timely manner.
3. **Third generation (1978-1999):** In this generation, the main characteristic was the ability to interact with both operators and environment by using complex interfaces such as vision or voice. They had high computational power intelligence in the form of both microcontrollers and computers to control both the actuation and the several sensors they

had on board. Their programming method was more structured and allowed high-level programming and simulation software.

4. **Fourth generation (2000-present):** starting from the fundamentals laid out by the previous generation, robots evolved again and became smarter. The on-board sensors and intelligence are now high-performing, actuators are mostly electrical and optimized for both electrical and thermal efficiency and their capabilities of interacting with human workers has been further enhanced to mimic humans as much as possible. In this generation, I4.0 robots have been developed under the name of *Collaborative Robots* (Cobots) [27] [28].

### *1.5.1 Industrial and Collaborative robots*

**Industrial robots** of today are heavy-duty robots with high payloads that can tackle dangerous, repetitive, and heavy applications. These robots are commonly used for parts assembly, pick-and-place, and palletizing in many different industries. **Cobots**, on the other hand, are designed to work alongside humans in a shared environment, according to one of the main principles of I4.0. This creates a clear separation between the two types, which mainly revolves around the concept of **safety**. In fact, Industrial robots are kept behind safety cages to protect humans from any harm, because they are usually built to be oblivious of the possible interactions with the world (humans included) and hard programmed. On the other hand, Cobots are designed to safely work around humans; hence, they do not require a protective cage to separate them from the employees. This is made possible by their design: the sensors they possess give them a certain degree of understanding of the surrounding environment, making them capable to detect collisions in advance and prevent them. It is worth noting that, even if Cobots are generally safe to be used around humans, it still depends on the end effector they are using. If a dangerous tool is mounted on their tip, a protective barrier must be adopted to ensure the safety of the human operators [29] [30] [31].

Another difference revolves around their **programming intuitiveness**. Cobots are easier to program because they are designed to be user-friendly, hence their programming time and complexity is highly reduced compared to Industrial robots. In addition, they often have a

manual guidance mode, where users teach the robot the trajectories of the program simply by guiding its end effector manually. It is worth mentioning that, even if Industrial robots are traditionally more complex to program, the market is now changing, opting for more intuitive interfaces also for this category.

However, companies still prefer to buy Industrial robots instead of Cobots. Why? The reason is mostly due to **productivity**. Even if Industrial robots are more expensive, they can produce a very high amount of units in a short period compared to Cobots, not to mention their capability of handling heavy objects, a skill that it is impossible to achieve for Cobots. This productivity issue is reflected on their power consumption, which is higher (therefore more expensive) in the former case compared to the latter [32] [33].

In conclusion, both categories are useful and have different pros and cons, which means that both may be required in a structured I4.0 factory depending on the application. Intensive production requires Industrial robots, while careful assembling or human-assisting operations require a Cobot instead.

## Chapter 2. Deep Learning for visual data

As stated in Chapter 1, advanced technologies and research objectives required to properly adhere to the Industry 4.0 paradigm revolve around the concept of “smart” devices. Historically, humans have always tried to give machines the ability to think, that is, to create **Artificial Intelligence** (or **AI**). In the early days of this research field, scientists tried to solve problems that were intellectually difficult for humans but easy for computers. These problems could be described by a list of mathematical or logical rules; hence, they were straightforward to solve for computers. The true challenge was to solve tasks that were easy for people to solve but hard to describe formally; problems that humans solve intuitively, automatically. For example, it is extremely easy for humans to recognize voices or faces, but it is not as easy for a computer. Why? Because human intelligence is based on **experience**. Humans learn concepts in a hierarchical way from the day of birth to the day of death, building their intelligence block after block, starting from easy concepts up to complex and abstract ones. This allows us to gather knowledge intuitively without formally specify complex concepts [34]. This is the idea behind **Deep Learning**. Building a deep structure composed of concepts of incremental complexity allows us to create a “smarter” computer intelligence. Figure 5 shows a graphical example of how visual data are elaborated by such models: basic concepts such as edges and color spots are learned in the first layer of the structure, allowing to recognize complex patterns such as the ones in the upper layer.

Artificial Intelligence comprehends several research fields. A simple subdivision is shown in Figure 6 that shows how Deep Learning is a part of Machine Learning (ML), which is, in turn, part of AI. Since to obtain a true Artificial Intelligence it is first necessary to develop a way for machines to learn, Machine Learning is the research field that focuses especially on learning from data of any kind in an automatic way. [35] provides a famous definition of what a learning algorithm is: “A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .”

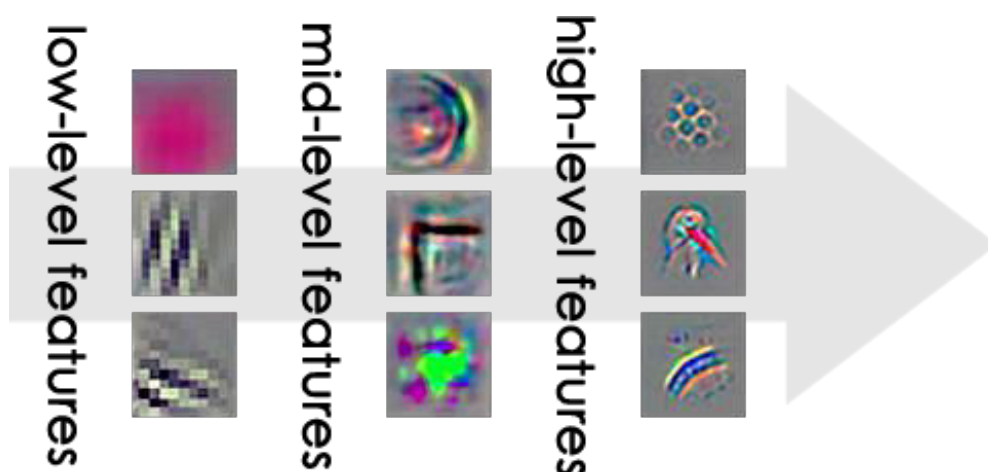


Figure 5. Example of a Deep Learning structure composed of layers of incremental complexity [36].

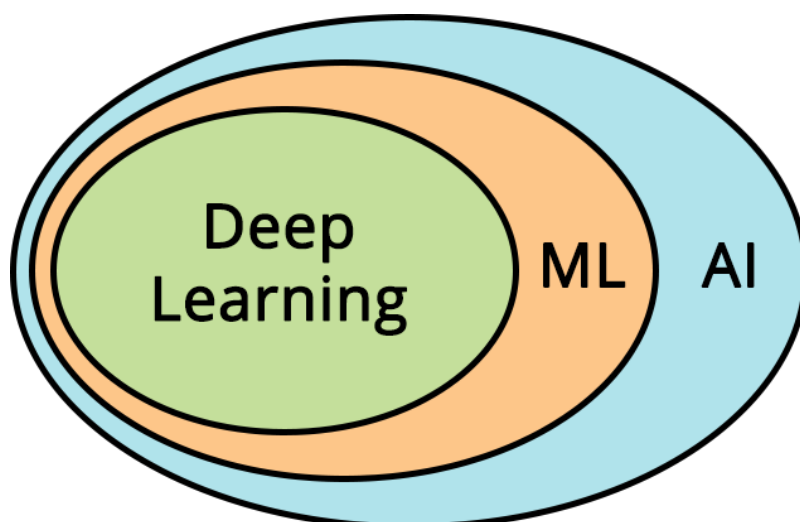


Figure 6. Subdivision of macro research areas of Artificial Intelligence.

A simple example using this definition is the process of learning how to walk. Children learn this by first watching other people walk, and then try themselves in a process of trial and error that slowly builds their muscles and confidence during the walk. In this example, the task  $T$  is “walking”, while the experience  $E$  is obtained one attempt after the other. It is obvious how  $E$  increments efficiently starting from the previous amount of experience gathered, building a solid ability because of the incremental difficulty of the experiments or attempts completed by the child. However, to be able to assess the improvements of the child, we must define a measure  $P$ . In the example,  $P$  may be a set of indicators describing stability, speed, and balance.

Machine Learning models (hence, for extension, Deep Learning models) are based on these concepts. The “task” is the specific problem the algorithm or model is developed to solve, for example the classification of different images in specific categories. The “experience” is provided by a suitable set of data, for example images representing the categories. Finally, the “performance measure” is provided by a set of statistical indicators to measure how well the model performs.

Three approaches are usually adopted in ML algorithms:

1. **Supervised learning:** in this type of learning the focus is on teaching the algorithm the task by showing it examples of inputs matched with the corresponding correct output (called “label”). Classification tasks are usually solved by supervised learning algorithms, where training images are labeled with the corresponding tag.
2. **Unsupervised learning:** the algorithm itself learns a suitable representation from the provided input data and extracts categorizations or information according to patterns learnt during training with no supervision. This kind of algorithms is especially useful to perform a first subdivision of data according to patterns invisible to the human eye (clustering).
3. **Reinforcement learning:** this kind of algorithms strongly differs from the previous ones. The concept of learning is explored by a “software agent” that interact with the “environment” taking certain “actions”. According to the changes that occur in the environment after a certain action has been completed and how this action contributes to the reaching of the agent’s goal, it gets a “reward” or a “penalty”. This system gave the name to this type of algorithms.

It is made evident that to give a device the ability to think, hence making it “smart”, ML and DL models and algorithms are of utmost importance. This Chapter will first briefly show the evolution of this technology through the years and the key factors that sparked it, making DL so important in today’s everyday life. Then two main categories of DL models are presented in more detail: Convolutional Neural Networks and Object Detectors. Both are used to extract information from images in different ways according to the specific model adopted, and are



the key technology to allow a certain machine to “see”. This requirement is of utmost importance for a smart device and it is the first step towards collaboration with humans. Albeit other senses may be exploited to achieve it, vision has been considered the most important one by the research community for years, especially because the huge amount of data available allows these models to be properly trained with low effort. Furthermore, when considering a smart device for industrial plants vision still plays a key role compared to other senses such as hearing. In fact, even if remarkable advances in the field of speech recognition and understanding have been made recently, voice communication between humans and machines (and sometimes even between humans and humans) is not robust enough due to the disturbing noises that may be present on the production line. These considerations motivated me to focus more on vision in the development of this thesis project.

## 2.1 History pills: the evolution of Deep Learning

Although the term “Deep Learning” has gained popularity in recent years, the research field dates back more than fifty years before. Following the subdivision in [34] where the authors divide the evolution in three main “eras” or “trends”, this Section briefly presents the evolution of Deep Learning mainly for visual data and the key concepts that characterize each era.

### 2.1.1 *Cybernetics Era (1940 – 1980)*

In this period, the concept of Intelligence was closely related to how the biological brain of intelligent creatures works. Scientists worked with neuroscientists and psychologists to find out a good mathematical representation of learning, coming up with the idea of “**neuron**” [37].

The first to be presented was the *McCulloch-Pitts Neuron* in 1943 [38]. The idea was simple: given a set of inputs  $x_1, \dots, x_n$  it is possible to associate them to a certain output  $y$  by using weights  $w_1, \dots, w_n$ , hence building a linear model  $f(x, w) = x_1w_1 + \dots + x_nw_n$ .

This idea was further expanded thanks to the *Perceptron* [39], developed in 1958. It was a small mathematical unit able to learn by itself the necessary weights  $w_1, \dots, w_n$ . This learning was possible thanks to learning algorithms of the same type we still use today, called **Stochastic Gradient Descent (SGD)**. Although linear models are still the most used Machine Learning algorithms, they have several limitations that made them not suitable for advance the research quest towards intelligence [40].

In 1965, the first general, working learning algorithm for supervised deep feedforward multilayer perceptrons was published [41] and, in 1971, the same author described what would be later on called a “deep network” with 8 layers [42]. Given a training set of input vectors with corresponding target output vectors, layers are incrementally grown and trained by regression analysis, then pruned with the help of a separate validation set. Regularization is used to weed out superfluous units, thus learning better and better internal representations of the data. The numbers of layers and units per layer can be learned in problem-dependent fashion [43].

### *2.1.2 Connectionism (1980 – 1990)*

The approach used by scientists in this era tackled the problem from a different perspective. In fact, the central idea, which inspired the name of this era, was that a large number of computational units networked together could achieve intelligent behavior.

Compared to the Perceptron, who was a single unit able to learn extremely simple representations, the idea introduced by the *Cognitron* [44] (1975) first and by the *Neocognitron* [45] (1980) second suggested the adoption of such a network; the very first **Artificial Neural Network (ANN)**. This model architecture was inspired by how mammals process image data in their biological visual system and became the basis for modern **Convolutional Neural Networks (CNNs)** developed in 1998 with the name of *LeNet* [46].

Today we still use one of the key concepts introduced by the Cognitron: the **Rectified Linear Unit (ReLU)**. The version we used today has changed through the years thanks to different

viewpoints by both neuroscientists and engineers, which simplified its structure. Thanks to this activation function, the values elaborated by it are mapped according to (1):

$$f(x) = \max(0, x) \quad (1)$$

Which is a simple nonlinearity function that computes the maximum and speed up the convergence of the training process of a network.

Another central achievement in this era is the idea of **distributed representation**: each input to a system should be represented by many features and each feature should be involved in the representation of many possible inputs [47] (1986). To comprehend why this concept is of utmost importance for Deep Learning, let us make a simple example. Considering a model that processes images as inputs, we want the model to learn simple features first and complex ones later. To differentiate between inputs, we may consider the color and the shape; hence, we need our network to learn different colors and different shapes. To do this, we may use as many neurons as many combinations of colors and shapes exists in our dataset. For example, if we consider three colors and three shapes, we will end up with nine possible combinations, hence nine neurons that learn each combination perfectly. The final output would be determined by which of the nine neurons activated the most.

Another way to do this, however, is to use only six neurons: three will learn the different colors and three will learn the different shapes. In this case, the final output will be determined by the combination of the two neurons that mostly activated during the inference phase. This concept allows the network to learn specific features from a wider range of inputs, because to learn “color red” the neuron simply needs to “see” red images, no matter the shape; while the neuron “star shape” only needs to “see” star-shaped images no matter the color [34].

The final and fundamental accomplishment of this era is the definition and successful application of the **Back-Propagation algorithm** [48] in 1986. This algorithm is still used today to learn the weights of the network propagating backwards the gradients of each neuron of the network. This process is called “training” of the network.

### 2.1.3 Modern Deep Learning (1990 – present)

At the end of the Connectionism era, scientists stopped creating algorithms and models inspired by biological brain functions, favoring instead a more mathematical approach to the problem, inspired by different fields. Modern networks, in fact, were able to see the light because scientists stopped trying to create intelligence in the same way human intelligence is built, opting for more realistic achievements that revolved around the machine capabilities of the time. From this separation two fields have been born: Computational Neuroscience, which tries to build more accurate models of how the brain actually works, and Deep Learning, which focuses on building computer systems able to successfully solve tasks that require intelligence [34].

By the early 1990s, experiments had shown that deep feedforward or recurrent networks are hard to train by backpropagation. The reason, scientists found, is due to vanishing or exploding gradients, also called the “*Fundamental Deep Learning problem*” [49]. The very first deep learning model presented in [50] partially overcame it through a deep recurrent neural network (RNN) stack pre-trained in unsupervised fashion to accelerate subsequent supervised learning. Furthermore, in 1997 Long Short-Term Memory (LSTM) RNN became the first purely supervised *very deep learner* [51].

Starting from 2006 with the *Deep Belief networks* published in [52] the field of AI and Deep Learning research attracted more and more attention, making researchers focus on the theoretical importance of depth. However, it was only when the world-famous *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* [53] was launched in 2010 that the field exploded. This competition, which is still organized each year, was established with the aim of propelling Computer Vision research by constructing a big dataset of internet images on which new models developed by contestants should be trained. In fact, thanks to the recent advent of **GPUs**, which allowed rapid parallel computation compared to traditional CPUs, and the **increasing amount of data** available on the Internet, CNNs could be effectively used in this challenge to achieve good results compared to the ones obtained in the previous years.

It was thanks to this competition that, in 2012, the renowned *AlexNet* model [54] was created, achieving impressive results for the time. This success was not only due to the model architecture but also especially due to the clever training method adopted, which utilized two GPUs at the same time to speed up the training process.

In 2014 *VGG* net was developed [55], showing that simple yet deeper models achieve even better results. In fact, the internal structure of the layers was simplified to ease the training process while retaining learning capabilities thanks to the depth structure. That year, though, it was *GoogLeNet* [56] to win the competition, introducing the Inception module and the concept layer that was then used by the *R-CNN* model [57], which was the base model that inspired **Object Detectors** later on.

Starting from the *VGG* model, in 2015 the *ResNet* model was developed and won the competition of that year [58]. This was an incredibly deep model composed of 152 layers with very simple structures, a result achieved thanks to the introduction of the **Residual Block** detailed in [58] [59], necessary to achieve well performing deep architectures.

## 2.2 Convolutional Neural Networks

To teach a computer how to see, a special type of ANN is used called Convolutional Neural Network (CNN). The name derives by the **convolution** operation adopted by the model, which is a mathematical operation that describes how the shape of a certain function  $f$  is modified by another function  $g$ .

Compared to ANN, the architecture of CNNs is very different. In fact, instead of having a set of hidden layers organized in sequence, where a set of neurons are required to elaborate the inputs according to a certain activation function (e. g. the ReLU function), CNNs work with volumes. Since images are stored as matrixes of pixel intensities inside a computer, it is only natural for CNNs to keep the same structure: images are matrixes of data where the  $x$  and  $y$  dimensions represent the width and length of the image and the  $z$  dimension represent its number of channels. For example, RGB images have 3 channels while Depth images only one. Each pixel intensity of the image matrix represents a feature of the input; hence, the structure

has a high number of parameters and requires a lot of time to be trained to learn them. Compared to ANNs, CNNs optimize this aspect reducing the number of parameters needed for the overall structure, learning the features, and keeping under control the layer dimensions thanks to Convolutional and Pooling layers. Finally, to convert the prediction in the desired output (e. g. a class label), Fully Connected layers are adopted. A graphical example of a CNN is shown in Figure 7.

The world-famous architectures of CNNs are AlexNet [54], VGG [55], Inception [56] and ResNet [58]. Starting from them, modern versions have been developed in recent years achieving state-of-the-art results in the task of image classification.

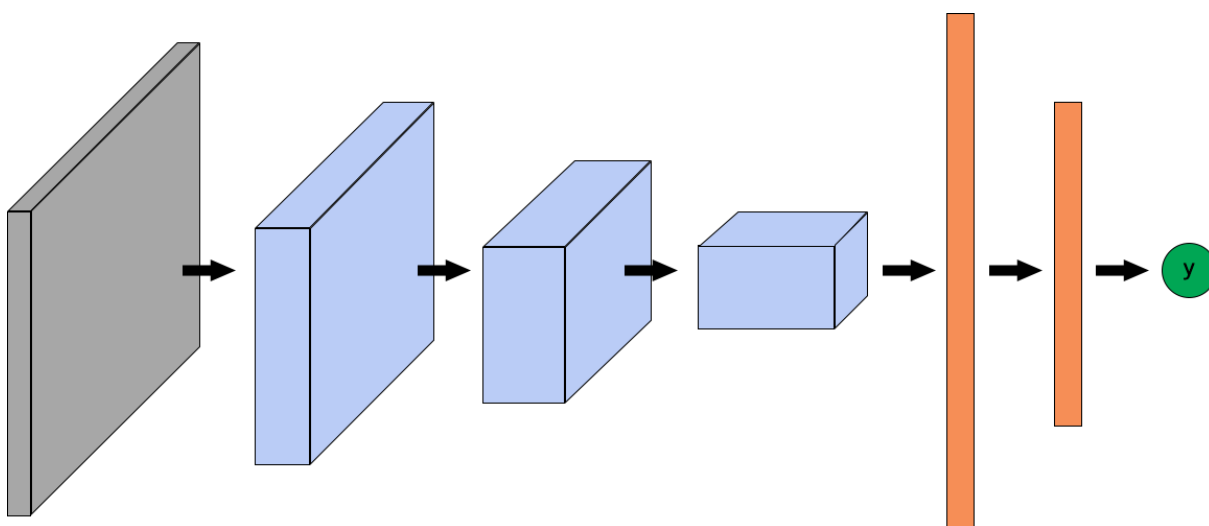


Figure 7. Example of a CNN structure. Starting from the image (grey block) a sequence of Convolutional and Pooling layers learn the features (blue blocks). To obtain the final output, the result is flattened in the final Fully Connected layers (orange vectors), that present the output in a human-readable way (green circle).

### 2.2.1 Convolutional layers

This type of layer allows to learn the features of the input by using **filters** or **kernels**, which are small matrixes that slide across the image and learn features by computing dot products (Figure 8). Each filter used learns a specific characteristic of the image, which will then be aggregated into a **feature map**. The convolution operation is carried out by moving the filter around the image and executing the activation function chosen (e. g. the ReLU function) to transform the data. Feature maps corresponding to each filter adopted are then stacked into

a layer called Convolutional layer. It is important to note that the dimension of the resulting Convolutional layer depends on the original image size and on the number of filters. For example, starting from an image of dimensions  $(500, 500, 3)$  and using 32 filters  $K$  of size  $(3, 3, 3)$  results in 32 feature maps of size  $(500, 500, 1)$ . The step used to slide the filter  $K$  along the image is called *stride*: the higher the number, the more rough and quick will be the learning, therefore reducing the learning ability of the overall network. The dimensions of the Convolutional layer, composed of the stacked feature maps, is  $(500, 500, 32)$ .

The output of a Convolutional layer may be the input of a following one, hence further refining the learning of the features, starting from simple concepts (e. g. edges) to more complex ones (e. g. patterns).

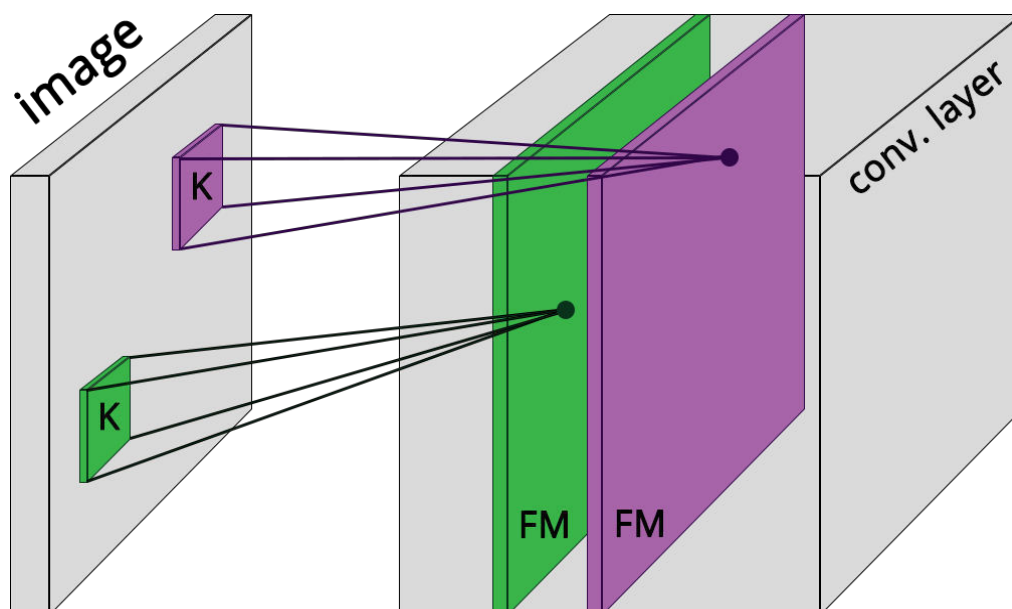


Figure 8. Example showing how a Convolutional layer is built. Starting from the original image, several filters  $K$  slide across the image performing a dot product between the filter values and the image ones, resulting in a series of feature maps according to filter used (green and purple filters result in green and purple feature maps). The feature maps are then elaborated by a chosen activation function and stacked to form the final Convolutional layer.

### 2.2.2 Pooling layers

After a convolution operation, is necessary to reduce the dimensionality of the parameters to learn. Therefore, Pooling layers are usually adopted after Convolutional layers. Reducing the

dimensions helps preventing overfitting, a situation where the model learns extremely well the particular task (for example, to recognize cats in images) but does not generalize well when adopted to elaborate new data (for example, new images of cats unseen during the training phase). Pooling layers down sample each feature map independently by keeping the depth of the map intact but shrinking width and length.

Different types of Pooling layers are available, but one of the most famous ones is the **Max Pooling layer** (Figure 9). Using small filters, these layers extract only the maximum value detected, which represent the most relevant feature of the region. Pooling layers have no parameters: a window slides on the feature map and the value is taken according to the type of Pooling layer adopted.

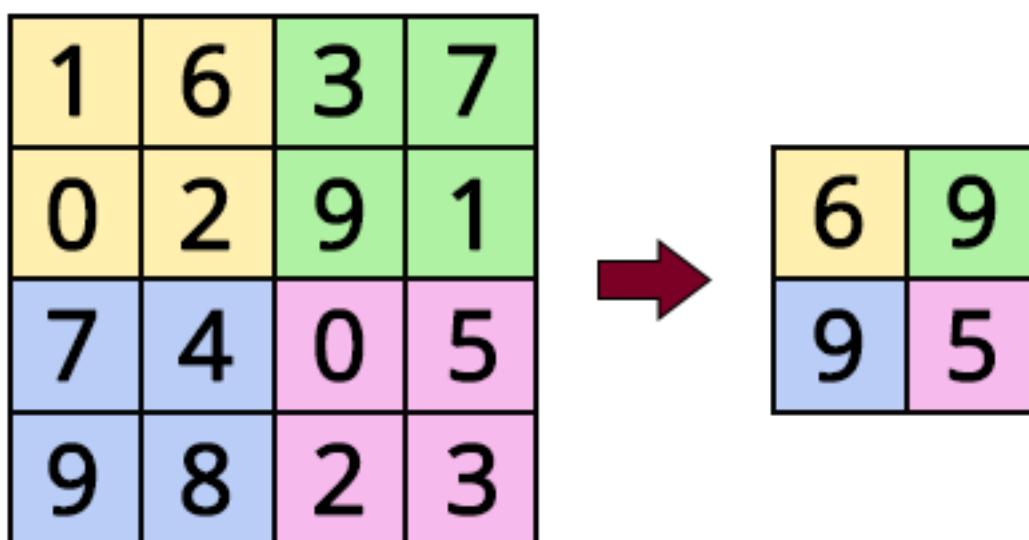


Figure 9. Example of a Max Pooling layer with a window of 2x2 and stride 2. From each window, only the maximum value is passed to the next layer.

### 2.2.3 Fully Connected layers

These layers represent a mapping from the original network multi-dimensional structure into a simple mono-dimensional vector. In fact, the output values of Convolutional and Pooling layers have no meaning for humans; hence, they need to be translated according to the labels defined. This usually means that the mapping performed transforms the output into a vector of probabilities, which are then mapped in text labels according to their positions.



## 2.3 Object Detectors

Traditional CNNs are used to classify images in well-defined classes; hence, the input images need to specifically contain the object to be recognized. Object Detectors try to solve a different task: given a certain image with different objects represented in it, the aim is to detect each object in the image and classify it correctly (Figure 10).

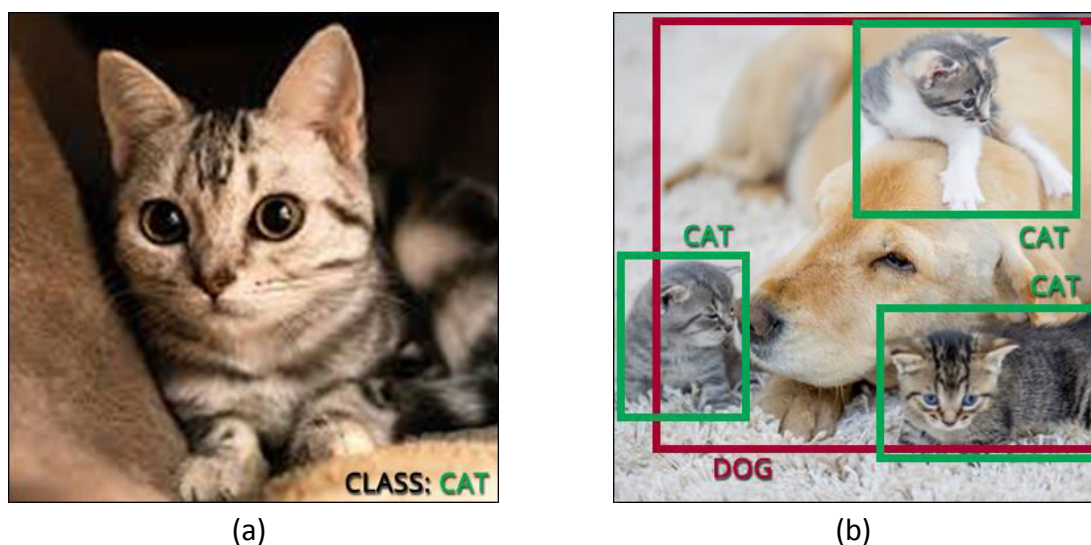


Figure 10. (a) Example of an output image of a CNN. The object is a cat and it is the only element clearly present in the scene, hence increasing the classification capabilities of the network, which outputs the class label. (b) Example of an output image of an Object Detector, where multiple objects are present in the scene. Each of them is precisely located in the scene and the corresponding class label is associated to them. Hence, the output is a set of Regions of Interest and class labels.

### 2.3.1 Region-Based CNNs

Object Detectors date back to when the **Region-Based Convolutional Neural Network (R-CNN)** [57] was first developed. The idea behind this model was to propose a certain number of boxes in the image and check if inside any of the proposed boxes an object could be found. The boxes, called regions, are extracted using a selective algorithm. The procedure carried out by the selective algorithm is the following: (i) the original image is segmented, meaning that different regions of it are recognized as unique blobs; (ii) regions with a certain degree of similarity and closeness are joined together, forming bigger areas (based on color, texture,

size, shape); (iii) these final areas are the region of interest proposed to the algorithm. The image areas proposed by the algorithm are then processed by the CNN model adopted to extract features. These are then passed to a **Support Vector Machine (SVM)** model that classify the object inside the region according to the features extracted by the CNN. A bounding box regression is used to predict the bounding boxes for each positive region (i. e. Rols that contain an object).

Although the R-CNN model was extremely helpful in advancing the task of Object Detection, it had some limitations. In fact, the selective search algorithm required at the start was typically ran around 2 000 times, hence making the necessary steps required to extract, compute, and elaborate a region, very time and resource consuming. Since the model was slow (it took around 40 s to perform a prediction on a single image), it was practically impossible to use for very large dataset, especially if the input images were big.

This led to the developing of **Fast R-CNN** [60]. Instead of running the selective search algorithm 2 000 times, in this model the regions are extracted after just one run of the selective algorithm on the whole image, leveraging the CNN structure. The input is elaborated by a CNN first, that extracts the feature maps that are used to automatically extract the Rols. These regions are pre-elaborated, hence they are resized to a fixed dimension and passed to a fully connected layer that classifies their content.

Alas, even Fast R-CNN had some problems. In fact, the selective search algorithm was still used, even if only once, hence slowing down the whole prediction model. The inference time was around 2 s; an extreme improvement compared to the previous model, but not enough to be used in real-life scenarios.

**Faster R-CNN** [61] solved this issue, finally abandoning the selective search algorithm and adopting a **Region Proposal Network (RPN)** to propose the regions to the CNN. The RPN takes the feature maps of the whole image as input and generates a set of object proposals accordingly. These, in the form of a map, are passed to the classifier as a whole. The inference time obtained with this model is of 0.2 s, another huge improvement compared to the Fast R-CNN model. It is worth noting that, even if Faster R-CNN is a state-of-the-art model, its

structure is a series of network; hence, its final performance heavily depends on how well the previous portions of it performed.

### *2.3.2 You Only Look Once (YOLO)*

The **You Only Look Once (YOLO)** [62] framework deals with the Object Detection task in a different way. Instead of using regions to localize objects in the image, this model takes the entire image as a whole and predicts the bounding box coordinates and classes in one go. This makes the algorithm exceptionally fast (around 45 frames per second), albeit its overall accuracy may be less than Faster R-CNN.

YOLO divides the original image into sections according to a grid. Image classification and localization are applied to each cell of the grid, hence predicting the bounding boxes and their classes. It is important to note that the boxes are calculated according to the grid adopted. This is also true for the input labels: training images are passed to the algorithm with the bounding box coordinates and labels of each object in them, obtained according to where the center of mass of the object is located with respect to grid.

### *2.3.3 Region-based Fully Convolutional Neural Network (R-FCN)*

The idea behind the **Region-based Fully Convolutional Neural Network (R-FCN)** [63] is that even if we have a feature map that detects only a portion of an object, that portion should be enough to detect the whole object. For example, if only handle of a cup is detected with high confidence score, it is reasonable to assume that the whole object is a cup and that the other parts of it are probably around the handle itself.

To achieve this result, the model adopts *position-sensitive score maps*. These maps allow to create a group of feature maps starting from the original one to search for the other cells of the map the other components of the object. Each new feature map is specifically trained to check for a specific feature that is usually located in that position. Then, for each of these regions the model computes a score expressing how much is it likely for the object to be in that cell, obtaining a matrix of votes related to the different cells of the feature map. For example, if we consider a  $3 \times 3$  matrix as our RoI proposal, the model checks 9 cells of the

original feature map according to the proposal position. This process is called *position-sensitive RoI-Pool*. The final score of the whole RoI is computed as the average score of the ones in its matrix.

With this method, the feature maps of the RoI are trained to check specific portion of the object and the final output is a combination of their results. Thanks to this, it is possible to increase the RoI positioning without increasing the inference time. In fact, R-FCN has been demonstrated to be twenty times faster than Faster R-CNN.

#### 2.3.4 Single Shot Detector (SSD)

The **Single Shot Detector (SSD)** [64] architecture has been developed in 2016 with real-time performances in mind. In fact, although Faster R-CNN achieves impressive accuracy results, the whole process still runs at around 7 frames per second. SSD reduces this time by removing the region proposal network and, to cope up with the resulting accuracy drop, it adopts multi-scale features and default boxes.

The model works similarly to YOLO, subdividing the image into a grid. It first extracts features maps using the famous VGG16 structure, then for each cell four object predictions are made. Each prediction is composed of a bounding box and N scores (one for each category plus the “none” category), from which the model picks the one with the highest score to determine the object class. This multi prediction strategy is called *multibox*. Then, the model computes both the bounding box location and class scores using small convolutional filters. To speed up the process, SSD detects objects from multiple layers instead of one, a process called *multi-scale feature maps*. The model adopts lower resolution layers to detect larger scale objects and higher resolution layers to detect smaller objects, where resolution means that the adopted grid of the feature map is thicker the higher the resolution value.

The boundary boxes are determined according to *default boxes*, which are pre-defined boxes of a certain shape and size (e. g. horizontal or vertical rectangles, squares, etc.) defined manually by the user. Starting from the default ones, the model adapts them to fit the object.

SSD is extremely fast (around 59 frames per second) but performs extremely worse compared to Faster R-CNN in terms of accuracy, especially for the detection of small objects. This is mostly due to the fact that high resolution layers are the first ones to be extracted by the feature map extraction network, hence these layers contain only low-level features like edges or color patches, that are less informative for classification. To improve accuracy, the manual definition of default boxes is a key aspect.

### *2.3.5 Performance comparisons*

Authors of [65] performed several tests to compare famous Object Detectors developed in TensorFlow, a popular framework to develop and test Deep Learning algorithms. In the tests, they used the **MS COCO** dataset [66] for training because its composition establishes a controlled environment and makes tradeoff comparisons easy. MS COCO is a large-scale dataset for object detection and several other tasks related with the detection of objects in cluttered scenes. It has around 330 000 images of which more than 200 000 are labeled, 80 object categories and 91 stuff categories.

It is important to note that there is not a “best” detector. In fact, everything depends on the end-user needs: does the task require speed or accuracy? The study shows that Faster R-CNN using Inception Resnet with 300 proposals gives the highest accuracy at the cost of a very slow inference time (1 frame per second) and, in general, this is the model with the highest accuracy among the tested ones even when changing the parameters, although it requires at least *100 ms* per image. The fastest models are SSD and R-FCN models at the cost of a reduced accuracy; in particular, R-FCN is a good trade-off between a fast model and a high-accuracy one, scoring exactly in-between SSD and Faster R-CNN depending on the parameters used. For more details about the tests and the results, see [65].

## Chapter 3. Human-Robot Interaction

Robots have been considered merely as necessary tools to achieve a goal, but what if they could be able to communicate? When humans work together in organized groups everyone has a specific task to carry out and, to properly collaborate as a team, the individuals need to communicate. For example, by watching a colleague working it is possible to understand if it requires assistance (body language); or by asking a colleague for information it is possible to better adjust our own work accordingly (voice communication) [67].

If humans work as partners, why robots and humans should not? In fact, human workers have unique problem-solving skills, sensory and motor capabilities but are restricted in force and precision. On the other hand, robots provide higher speed, repeatability and can manipulate heavy objects with ease, but are restricted in flexibility [68]. By establishing an effective communication method between the two, it is possible to obtain a human-robot collaborative team where human workers do not have to carry out heavy or dangerous tasks while still being able to exploit their unique intelligence and knowledge to guide the robot, this time as a partner.

Hence, communication is the key to achieve effective collaboration between two individuals. Different types of communication may be established between humans and robots according to the sensor technology and approach. Generally, interactions may be:

- **Physical:** by touching the robot in a certain way it is possible to issue a command. This type of communication is not always possible nor feasible if the robot moves at very high speed or if it is carrying heavy objects, but it may be the preferred type of interaction for companion robots or service robots;
- **Verbal:** by using voice it is possible to give information to the robot or issue a command, for example to require assistance or to order it to carry out a certain action. This type of interaction is based on speech recognition and it is heavily language-dependent, not to mention that background noises, usually found in manufacturing plants, could terribly

lower the chances of the system to recognize and interpret correctly the user voice command;

- **Non-Verbal:** body language can be as expressive as voice if properly designed, as the sign language communication suggests. Hence, by adopting suitable technologies to acquire and interpret body gestures, it is possible to easily communicate with the robot. This approach allows communicating regardless of the user language by using a pre-defined gesture dictionary.

### 3.1 Communicating by gestures

As discussed in [69], gestures represent a complex type of communication. They can be used as part of an utterance to represent a meaning in a complex, holistic way, concurrently with speech or as an alternative, serving as a component of a sentence that might otherwise have been spoken.

For example, considering the phrase: “...*he was very stressed and started smoking*” where the speaker mimics the act of smoking with its hands. In this case, the gesture adds a behavior example to the message that enriches the original meaning of the spoken phrase. This also happens when gestures are used as substitutes for words or complex meanings. In the phrase: “...*her parents are extremely—*” the speech interrupts and the speaker stretches out both hands fully open with the fingers slightly bended as if to simulate the act of grasping an object, moving the hands up and down with a stressed-out facial expression. This complex gesture conveys the general idea of over-controlling parents, but at the same time, a broad set of meanings linked to this concept that are difficult to explain with words but are immediately understood thanks to this simple gesture. Therefore, if in the first case the gesture simply added meaning to an already well-built and understandable sentence, in the second case, the gesture does not substitute a single word and its presence is necessary to understand the original sentence in full.

A different scenario is represented by gestures used as the only communication channel between two or more individuals. This is the case of sign language or coded languages in general, where the people involved all know how to translate the symbol to a meaning

according to a pre-defined symbolism that standardizes the language. These symbols are much more like words than anything are, in fact, they are used as building blocks to create a complex language structure similar to how spoken language is built. It is interesting to note that, considering deaf elementary school children who did not have an already established sign language between them, they naturally elaborated a pantomimic gesture form to represent something and mutually agreed on that something [70]. For example, the gesture to represent their teacher facial appearance became a method to refer specifically to her, by pointing the index finger to the cheek. This gesture changed from “iconic” to “arbitrary” and even transformed to the gesture that refers to the School Principal. The original gesture is freed from the requirement of being a picture of something (the teacher), so it becomes free to take a general meaning (a general teacher or the School Principal) and being recombined with other forms or participate in compound signs or sentences. These forms emerge during their use during interactions; hence, this transformation only happen when a community of users is involved and actively uses them. If a gesture is established between two individuals it may be “private” at first, conveying only a certain meaning known to them but not to others, like a secret code. Nevertheless, when the gestures are used to communicate with a broad audience, their meaning cannot be private and must be general and understandable: this is the process of *lexicalization of a gesture*.

The author of [71] highlights that gestures are not just movements but are *symbols* that exhibit meanings in their own way. In their experiments, different people described the same event with both words and hand gestures and, albeit with some differences, the main concepts were expressed by similar gestures. The individually different gestures each subject used had a common core of meaning, not because of a code or a gesture language but because each subject separately created its own manual symbol of the event. Gestures are linked to the words used; in fact, 54% of the verbs with the “downward” meaning used by the subjects were accompanied by a downward movement of the gesture, while 0% of them were accompanied with an upward movement of the gesture. Similarly, 73% of the verbs with the “horizontal” meaning co-occurred with gestures that included left or right movements, while only 8% occurred with a downward movement and 17% with an upward movement.



An interesting analysis concerns two-handed gestures, which can be of two kinds: gestures where the hands move in the same pattern but mirrored and gestures where the hands perform different movements. The latter are interesting because usually these gestures are used to express a fixed reference with one hand and an active motion with the other; hence, these gestures have a stronger meaning compared to single-hand gestures.

## 3.2 Gesture Recognition

The term *Gesture Recognition* refers to a process where the gestures made by a user are recognized by a receiver. Gestures are expressive body motions that involve physical movements of fingers, hands, arms, head, face, or full body, used to convey information or to interact with the environment. Since gestures depend on the language and on the culture of the user, (i) the same information may be expressed using a different gesture or (ii) a certain gesture could express very different information according to the user, even if a proper gestures dictionary has been defined in advance [72].

Gestures may be:

- **Static:** the user assumes a certain pose or configuration with its body;
- **Dynamic:** the gesture changes over time, hence there are different phases called pre-stroke, stroke, and post-stroke. This implies that the gesture has start and end points both in space and time, therefore automatic recognition may be difficult or prone to errors according to the gestures in the dictionary.

It is worth noting that different gestures may be (i) hand and arm gestures (e. g. sign language, hand poses), (ii) head and face gestures (e. g. direction of eye gaze, emotion recognition) and (iii) body gestures (e. g. full body motion for tracking, rehabilitation or behavior monitoring). According to [72], the meaning of a gesture typically depends on where it occurs (spatial information), what trajectory or path it takes (path information), what sign or symbol it makes (symbolic information), and, if it has an emotional quality like for facial gestures, what is it (affective information).

The authors of [68] propose a gesture recognition model based on five steps:

1. **Sensor data collection:** the raw data of the gesture is acquired by sensors;
2. **Gesture identification:** in each frame, a gesture is located from the raw data;

3. **Gesture tracking:** the located gesture is tracked during the gesture movement. This step is unnecessary if the gesture is static;
4. **Gesture classification:** tracked gesture movement is classified according to pre-defined gesture types;
5. **Gesture mapping:** the gesture recognition result is translated into robot commands and sent back to the workers.

Although gesture recognition is usually carried out by means of a vision system to “see” the gestures, different types of sensors can also be adopted. For example, gloves provide high precision in the recognition but may cause problems in the working environment due to their calibration needs and required cable connections. Band sensors are another type of wearable device that adopt electromyogram technology to sense the user movements albeit with less precision compared to gloves. Non-wearable devices include radio frequency-based sensors that track gestures according to the changes in the signals, even from afar, but the precision of the recognition is very low.

As pointed out in [73], there is not a standardized set of gestures suitable for human-robot communication. Albeit many gesture recognition works and applications are available in literature, these are mostly designed to perform laboratory experiments and test the system performance, not to perform tasks in the real industrial environment. Gestures must be easy to make, as close as possible to the common use of finger, hand, and arm gestures (i. e. natural gestures), clearly distinguishable one from another and easy to remember. They must be different from gesticulation and socially acceptable, minimizing the cognitive workload. Moreover, even if a sign language like the American Sign Language is rich in syntactic and semantic features to the point of not needing the contemporary use of spoken language, it is, on the other hand, a quite complicated way to communicate and require a great learning workload, making sign languages not suitable to express simple commands for human-robot communication [69].

Another way to categorize gestures is proposed in [71], following this subdivision:

- **Iconic gestures:** they represent images of concrete or abstract entities and/or actions, resembling the event or the objects as if they have a semantic connection to them;
- **Metaphoric gestures:** related more to the representation of abstract entities or concepts rather than concrete objects;
- **Deictic gestures:** refer to entities, actions or objects present in the surrounding environment where the person currently is but may also be abstract. This type of gestures is the “pointing index” kind that can be performed with the different body parts;
- **Beats gestures:** this category refers to gestures where the movement of the limb or the hand is rhythmic like it is a beating time. Typically, this is what happens with Gesticulation, with no semantic correspondence to the speech.

Hence, for human-robot communication iconic and/or deictic gestures are the most suited ones to express understandable commands to the robot.

### 3.3 The developed gesture language

#### 3.3.1 Background research

To effectively communicate with a machine in a natural and intuitive way the communication language adopted should be easily learned by the operator and robustly interpreted by the robot. The need of a flexible and intuitive programming language for industrial robots has been thoroughly investigated in [74], which highlights that not only modern companies need a flexible set-up to adjust the production based on market needs, but also that complex robot tasks can be decomposed into small and simple actions. The experiments carried out by the authors show that skill-based programming performed by different means (such as gesture recognition and kinesthetic teaching) greatly speed up the programming time required by the operators, regardless of their experience.

As pointed out in [75] and [67], a natural and robust interaction is based on body language, in particular on hand gestures. Hand gestures recognition is a topic that has been vastly explored

through the years and a plethora of sensors may be used according to the type of communication method, for example by using wearable sensors such as haptic gloves or EMG or IMU devices [75]. In fact, authors of [76] propose an HRI system based on hand gesture recognition, which leverages wearable sensors such as IMUs. They adopted a combination of 8 static gestures and 4 dynamic gestures. The IMUs' data is acquired and elaborated by an unsupervised segmentation algorithm based on motion and classified afterwards by an Artificial Neural Network (ANN) purposely trained for the task. In late years, though, the adoption of vision systems to perform hand-gesture recognition has increased also because these sensors are contact-less, reliable, and cost-effective, considering that a single camera can be adopted in combination with a suitable Computer Vision or Deep Learning algorithm to achieve satisfactory results [77]. Furthermore, vision sensors may also be used to monitor the scene effectively, allowing the eventual introduction of safety strategies based on human-robot relative positions [78] [31] in parallel with the human-robot communication system. Vision-based methods traditionally involve Computer Vision techniques such as the hand segmentation using the skin color or the depth information to accurately find the hand in the image frame. A successful example is detailed in [79], where the authors developed an open-source API to recognize both static hand gestures and dynamic hand gestures combining the RGB and depth information provided by a Kinect sensor. They were able to implement in the API the full ASL finger-spelling alphabet and a rich set of 16 uni-stroke gestures, using a Support Vector Machine (SVM) to classify static gestures based on features extracted from a Gabor filtering, and a Hidden Markov Model (HMM) to recognize dynamic gestures based on the computed trajectories of the segmented hands. In [80] a flexible system for gesture-based HRI is presented, where the authors propose a system based on the recognition of upper arms gestures, utilizing a Machine Learning algorithm called Adaptive Naïve Bayes Classifier that classifies the gestures by analyzing the pose skeletonization extracted from the depth image of the scene. Authors of [81] present a similar system, based on ROS and on the recognition of full body poses acquired by a Kinect camera and a Leap Motion sensor. Here, the hierarchical representation of tasks allows users to build programs in a flexible and intuitive way. Similarly, in [82] a system based on a Gestures Dictionary is used to dynamically build a robot program where the adopted gestures are acquired by a Kinect sensor that extracts the

human skeleton (static gestures) and by the Leap Motion sensor that tracks the hands movements (dynamic gestures). The gestures are designed to move the robot in a modality similar to the traditional Jog mode, where the robot end effector position is adjusted by a certain pre-defined step according to the direction of the movement (up, down, left, right). The experimental results show that even for non-expert users the adoption of body gestures to move the robot was more convenient in the case that the accuracy of the positioning was not required, and overall this approach was preferable compared to traditional methods.

Compared to those methods, nowadays hand gestures can be easily recognized by using convolutional neural networks (or CNNs) [77]: the hands are treated like symbols or objects and thus can be easily extracted from single frames with an inference time of *200 ms* and classified correctly in most cases. An example of this approach is detailed in [83], where the authors present a small set of very different single-hand gestures acquired by an RGB camera. To ensure a high recognition confidence, they use a solid color background and extract the hands in different regions by using a skin model, filtering out non-hand pixels. They also perform a calibration of the system, to ensure its capability of recognizing the gestures even when they are performed in slightly different poses and orientations. A different solution is to adopt an Object Detection model. In this case, the network is composed of two parts, which solve two different aspects of the problem: the object localizer, which finds the objects in the scene regardless of the background noise, and a CNN-based object classifier, which identifies the objects obtained from the previous step. After training, the network localizes the objects in a cluttered scene and classifies them correctly. In this way, it is possible to perform the recognition of multiple objects in the same scene, reducing the computational complexity of the system, speeding it up [84], and avoiding the use of a solid color background, which is not a feasible option in industrial environments. Object Detectors are intensely used especially in the autonomous driving research, to localize pedestrians, other vehicles, street signs and so on [85]. This suggests the idea of applying Object Detectors to recognize human hands, a task successfully achieved in [86], where it is demonstrated that Object Detectors can be effectively used to localize human hands and to classify the single-hand gestures performed in cluttered environment.

### 3.3.2 Evaluation metrics

To evaluate the performances of a gesture recognition model trained on a certain dataset it is common practice to use a set of statistical tools. Let us consider a classification example where a CNN model is trained to recognize pictures of cats as “Cat” and, if no cat is present in the image, the model should output “None” (Table 3). **True Positive samples (TP)** are obtained when the original image represented a cat and the model correctly outputs the correct label “Cat”. **True Negative samples (TN)** are obtained when the original image contained no cat and the model correctly outputs the “None” label. **False Positive samples (FP)** are obtained when the original image did not represent a cat, but the model outputs the class label “Cat”. **False Negative samples (FN)** are obtained when the original image represented a cat but is classified as “None”.

Table 3. Example of a truth table considering the case of a two-class classification problem. In this case, the problem is designed to check for the presence/absence of a single object, hence it is a binary problem.

		Ground Truth		
		Cat	None	
Predicted	Cat	TP	FP	PPV
	None	FN	TN	NPV
		TPR	TNR	

Hence, the indexes considered to evaluate the model performances are the following:

- **Positive Predictive Value (PPV, also called Recall):** it represents the ratio of correctly predicted positive observations out of all predicted class observations. It answers the question: “*what proportion of predicted class positive observations is truly positive?*”

$$PPV = \frac{TP}{TP + FP} * 100$$

- **True Positive Rate (TPR, also called Precision):** it represents the ratio of correctly predicted positive observations out of the number of actual class observation. It answers the question: *“what proportion of actual class positive observations is correctly classified?”*

$$TPR = \frac{TP}{TP + FN} * 100$$

- **Negative Predictive Value (NPV):** it represents the ratio of correctly predicted negative observations out of all predicted class observations. It answers the question: *“what proportion of predicted class negative observations is truly negative?”*

$$NPV = \frac{TN}{TN + FN} * 100$$

- **True Negative Rate (TNR):** it represents the ratio of correctly predicted negative observations out of all actual class observations. It answers the question: *“what proportion of actual class negative observations is correctly classified?”*

$$TNR = \frac{TN}{TN + FP} * 100$$

- **Recognition Accuracy (RA):** it represents the number of total correct classifications over the total number of test examples, answering the question: *“what proportion of occurrences both positive and negative were correctly classified?”*

$$RA = \frac{TP + TN}{TP + TN + FP + FN} * 100$$

- **F1-Score:** it is the harmonic average of TPR and PPV. It is used as a tool to compare different classifiers, giving a larger weight to lower values. It is worth noting that this index is useful for a quick comparison, but the context of the classification problem must be considered. For example, classifying a defective gear as optimal has a different cost from classifying an optimal gear as defective, and this should be reflected in the way weights and costs are used to select the best classifier for the problem.

$$F1 - Score = 2 * \frac{TPR * PPV}{TPR + PPV}$$

- **Confusion Matrix:** it is a table reporting the number of all the predicted observations of each class with respect to their ground truth. It gives a quantitative measure of the ability of the model to classify the different classes. When normalized, each value of the table is averaged over the total occurrences of the specific class.

The multi-class case is a little bit trickier. Although the definition of each index is still valid, the adopted formula is not the same. Considering the example in Table 4, we have three animal classes “Cat”, “Dog” and “Mouse”, plus a non-class “None” representing the case where the image is showing a different animal or something unrelated to the three animals in the dataset. In this case, to correctly determine the index values it is best to look at the matrix and keep in mind the question each index is trying to answer. Hence, the PPV of each class is calculated as the number of TP of the class over the sum of predictions of said class (horizontal sum), while the TPR is calculated as the number of TP of the class over the sum of actual class observations (vertical sum). For the negative values, the same consideration must be considered but only for the “None” class, representing the negative observations.

Table 4. Example of a truth table considering the case of a three-class classification problem.

		<b>Ground Truth</b>				
		<b>Cat</b>	<b>Dog</b>	<b>Mouse</b>	<b>None</b>	
<b>Predictions</b>	<b>Cat</b>	<b>TP</b>	<b>FP</b>	<b>FP</b>	<b>FP</b>	<b>PPV</b>
	<b>Dog</b>	<b>FP</b>	<b>TP</b>	<b>FP</b>	<b>FP</b>	<b>PPV</b>
	<b>Mouse</b>	<b>FP</b>	<b>FP</b>	<b>TP</b>	<b>FP</b>	<b>PPV</b>
	<b>None</b>	<b>FN</b>	<b>FN</b>	<b>FN</b>	<b>TN</b>	<b>NPV</b>
		<b>TPR</b>	<b>TPR</b>	<b>TPR</b>	<b>TNR</b>	



### **3.3.3 First approach: four two-hands gestures**

As a first approach to the problem, a very small gesture dataset has been developed and tested with only 4 gestures [87] [88]. The idea behind this first test was to choose gestures that were easy to remember and perform and that, at the same time, could express suitable commands. The focus of this first approach was to experiment with gestures following similar examples shown in literature. Simple and easy single-hand gestures were usually adopted, such as the ones in [89], and detected using visual techniques like hand segmentation using color thresholding and background subtraction. The static gestures adopted in works that used the whole body posture were also very simple albeit not natural, a necessary trade-off to improve robustness of the recognition. These methods were often combined with the skeleton information or by segmenting the human from the colored image thanks to the depth frames in order to detect the whole pose correctly [82] [90] [91]. Some works also fuse the visual information taken from RGB-D cameras with other sensors such as haptic gloves [92].

#### **3.3.3.1 Dataset definition**

Since even the best performing DL algorithms may recognize a gesture wrong, to ensure that the hand gesture command is properly recognized and to expand the dictionary of possible commands, I choose to build a gesture language with both hands at the same time. A similar idea was also explored in [93], where two hands were used to form complex meanings such as the action of putting an object over something else. I opted for simpler gestures; hence, in this dataset the second hand acts as an “anchor” with no meaning for the command. The “anchor” role is to force the DL model to recognize the complete gesture only in its presence, improving the robustness of the recognition. I choose as “anchor” the left hand, which must be held close showing the fingers to the camera.

Figure 11 shows the four gestures adopted, briefly described as follows:

- **START:** left hand closed (anchor), right hand closed (Figure 11a).
- **STOP:** left hand closed (anchor), right hand opened (Figure 11b).

- **RIGHT:** left hand closed (anchor), right hand pointing right (palm facing the camera, Figure 11c).
- **LEFT:** left hand closed (anchor), right hand pointing left (back facing the camera, Figure 11d).

These were mostly inspired by existing literature regarding gestures in industrial plants, which adopt only a few of easy and intuitive commands [82] [94].

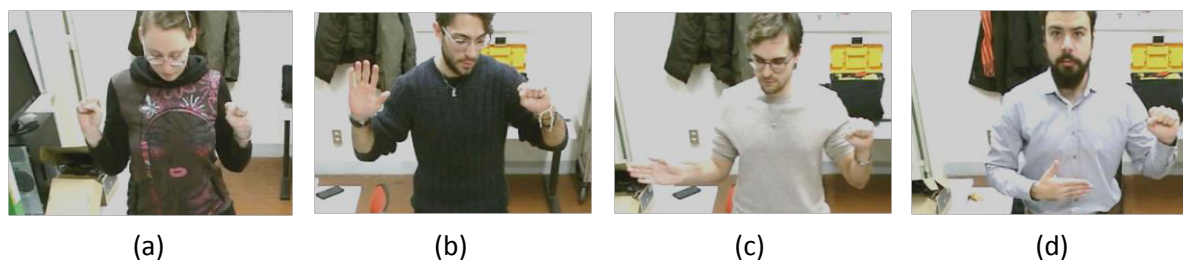


Figure 11. Examples of the four gestures performed by different actors. (a) *START* gesture, performed with both hands closed, (b) *STOP* gesture, performed with the right hand open, (c) *RIGHT* gesture, performed with the right hand pointing right, (d) *LEFT* gesture, performed with the right hand pointing left.

The *Complete* dataset is composed of a combination of four small datasets containing RGB images acquired by a Kinect v2 [95] camera in different set ups. The actors moved around the test area while performing the gestures, resulting in gestures performed while front facing the camera and while being laterally oriented. To ease the training process, each image in the datasets has a standard size of 224x224 px. The four datasets are:

- **Base Dataset:** it is composed of 609 images containing gestures taken from 15 different operators;
- **Light Colors Dataset:** it is composed of 383 images containing gestures taken from 5 different operators. These are taken with operators wearing clothes similar to their skin color or similar to the background color, to test the performance of the model in problematic conditions;
- **Gloves Dataset:** it is composed of 400 images of gestures taken from 5 different operators. These are taken with the operators wearing light blue gloves, to create some contrast;
- **Zoom Dataset:** it is composed of 710 images of gestures taken from 7 different operators. These are basic gestures taken with the camera positioned very close to the operator to reduce the disturbances from the background.

The combination of these four datasets creates a *Complete* dataset. It is worth noting that for each dataset 80% of the images have been used for training and 20% for testing. To build the *Complete* dataset, the 80% of the images have been selected from each smaller dataset, ensuring an adequate contribution from each of them, for a total of 1681 images for training and 421 for testing. I also create an extra dataset named *Errors* dataset composed of 140 images taken from the *Base* dataset, 17 images from the *Light Colors* dataset and 9 images taken from the *Zoom* dataset. These images contain mostly purposely “None” gestures, which represent not allowed gestures and situations where multiple operators not performing any gesture are in the scene, and only a few of correct gestures.

### *3.3.3.2 Experimental system description*

The model chosen for this first test was a Faster R-CNN model trained in MATLAB without pre-trained weights. This choice was motivated by the fact that albeit famous DL frameworks such as Caffe or TensorFlow were available at the time, they still required the user to fully comprehend their structure before being ready for use. MATLAB DL module, on the other hand, was a faster alternative to test this first dataset idea and get some results, while still following famous model structures such as the one adopted. I chose the Faster R-CNN model because it was the best performing one available in MATLAB as an off-the-shelf algorithm, allowing me to focus more on the dataset definition and hyperparameters tuning than on the whole model structure.

The four single-hand gestures are individually recognized by the DL model and, after the prediction set has been obtained, the constraints are checked to verify if the recognized gestures comply with them. In this way, it is ideally possible to build different commands by combining single-hand gestures into a two-hand one afterward. This checking procedure is carried out by a custom-made filtering function called *Custom Prediction Function* (CPF) I purposely designed for the task. The CPF takes the model output as input, which is a set of predictions comprehending the predicted class label, bounding box coordinates and confidence score. For each image, only the predictions with a confidence score greater than 90% are retained; among them, the presence of at least one closed-hand gesture is verified,

otherwise the predictions are discarded, and the gesture is classified as “None”. Residual predictions are good candidates to represent gestures: the challenge is to combine predictions in *pairs* of single-hand gestures, where one is the left-hand anchor and the other is the right hand (closed, open or pointing left or right, depending on the gesture). To this aim, the CPF block checks the Euclidean distance of the centroids of each bounding box detected in the image. If this distance is inside a threshold (not too close, not too far), the pair is retained. If some predictions overlap, only the one with the highest confidence score is retained, no matter the class label. Finally, only the pair with the highest total confidence score is retained among the residual pairs. The complete procedure is described in detail in [88].

### 3.3.3.3 Performances evaluation

I trained each dataset in Table 5 individually, obtaining five different models. The results relative to each dataset are obtained testing each model on its respective test dataset, while the results relative to the *Errors* dataset are obtained considering the model trained on the *Complete* dataset and tested on the *Errors* dataset. It is worth noting that the performances shown in the Table have been rounded. Considering the results obtained, it is not surprising that the *Light Colors* dataset performs worse than the others do because the skin color of the hands is often too light or too similar to the shirt worn by the operator, hence reducing the model recognition capabilities. The limited number of images in this dataset is also one of the reasons why its performances are very low compared to the other ones.

Table 5. Test results of the model trained on each dataset. Results have been rounded.

<b>Dataset</b>	<b>PPV</b>	<b>TPR</b>	<b>NPV</b>	<b>TNR</b>	<b>F1-Score</b>	<b>RA</b>
<b>Base</b>	91%	83%	20%	100%	87%	84%
<b>Light Colors</b>	79%	66%	20%	67%	68%	66%
<b>Gloves</b>	89%	90%	14%	8%	87%	78%
<b>Zoom</b>	96%	98%	90%	79%	97%	95%
<b>Complete</b>	92%	93%	81%	72%	93%	91%
<b>Errors</b>	1%	6%	100%	73%	2%	70%

The idea of wearing gloves of a bright color shows interesting results, in fact the *Gloves* dataset, despite having few images, achieves a total recognition accuracy close to 90%. This finding suggests that, in industrial scenarios where gloves may be a required gear to wear for operators, designing a gesture dataset based on hand gestures with gloves is a feasible solution. As expected, the *Zoom* dataset performs best among the datasets, not only because of the high number of images in it but also because the hands are perfectly recognizable and occupy a big portion of the image, hence the cluttered background noise is reduced. While the *Base* dataset achieves overall good performances, the real improvement can be seen in the *Complete* dataset, where even the NPV and TNR values are high. The reduction of the overall performance values compared to the *Zoom* dataset is due to the fact that, in this dataset, even the difficult images from the *Base* and *Light Colors* datasets are considered. Figure 12 shows the normalized confusion matrix obtained for the *Complete* dataset test.

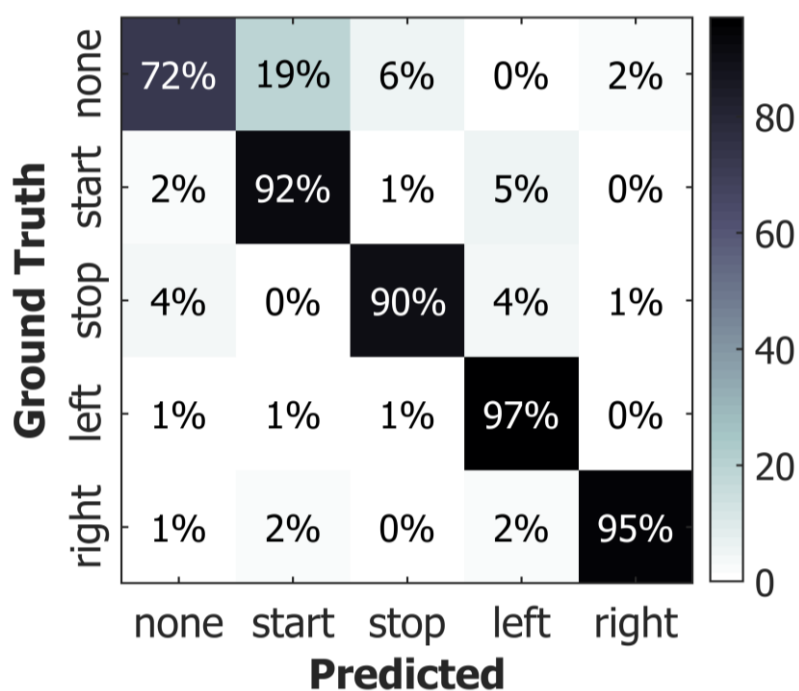


Figure 12. Normalized confusion matrix of the Complete dataset test.

The performances of the *Errors* dataset are interesting because, in this case, only the NPV and TNR values are informative. The high values achieved show that, when a purposely wrong gesture is presented to the model, the chances of recognizing it as “None” are extremely high. From this experiment, it is made evident that four gestures are not enough to properly

command a robot end-effector, but the idea of a two-hand language to boost the recognition accuracy of the complete gesture shown promising results.

### *3.3.4 Second approach: a big single-hand gestures dataset*

In this second approach I expanded the single-hand gestures dictionary and tested the capabilities of the same model (Faster R-CNN model designed and trained in MATLAB) to recognize single-hand gestures only in cluttered environments. Some ideas regarding this second dataset structure came from the literature. For example, the work detailed in [96] shows a successful application of a custom-made Object Detector trained to detect single-hands gestures. Its robustness is improved by the presence of the human skeleton, used to filter out incorrect predictions also according to the performing stance defined by the authors, which is with both arms at shoulder length front facing the camera. This combination proved to be successful at the cost of a reduced inference time; in fact, the whole human-robot interaction may be too slow to be smooth. Furthermore, in a second release of the previous work presented in [97], the authors adopted multiple gestures and augmented their dataset using artificial backgrounds.

#### *3.3.4.1 Dataset definition*

The new dataset has been manually acquired using different smartphones: a person moved around the actor with the device, acquiring the gestures with different angles. A total of 15 actors have been registered performing the gestures, 5 females and 10 males (Figure 13). Different backgrounds were chosen for each actor, to increase the variability of the dataset and prevent the network to learn how to recognize the background instead of the hands. It is worth noting that 5 actors presented a solid bright green color background; two of them have the hands segmented out from the rest of the body by a green frame (samples (k) and (l) in Figure 13).



Figure 13. Examples taken from the dataset for every actor. Actors from (a) to (l) have been used for training, while actors from (m) to (o) have been used for testing.

The dataset has been augmented using a random combination of 3 augmenting techniques:

- **Random spatial distortion:** applies a virtual zoom to the original HD image and a dimensional distortion, thus losing the original aspect ratio. The distorted images have been then cropped to the standard size of 224x224 px (samples (a) and (b) in Figure 14);
- **Random color distortion:** multiplies the channels of the image for a random numerical value. To increase variability, the program selects randomly the color space on which perform this distortion (RGB, HSV). Speckle noise is added after-wards with intensity randomly chosen (samples (a) and (b) in Figure 14);
- **Artificial background:** 5 actors have been purposely acquired with a solid bright green color background. This color is easily selected by the program and substituted with a random background downloaded from the internet, followed by a low-pass filtering step to obtain a more realistic fusion between the actor and the new background (sample (c) in Figure 14).



The total number of images is 39584 for the training dataset and 5739 for the test dataset, both randomly shuffled. The gestures chosen for the experiment are shown in Figure 15.

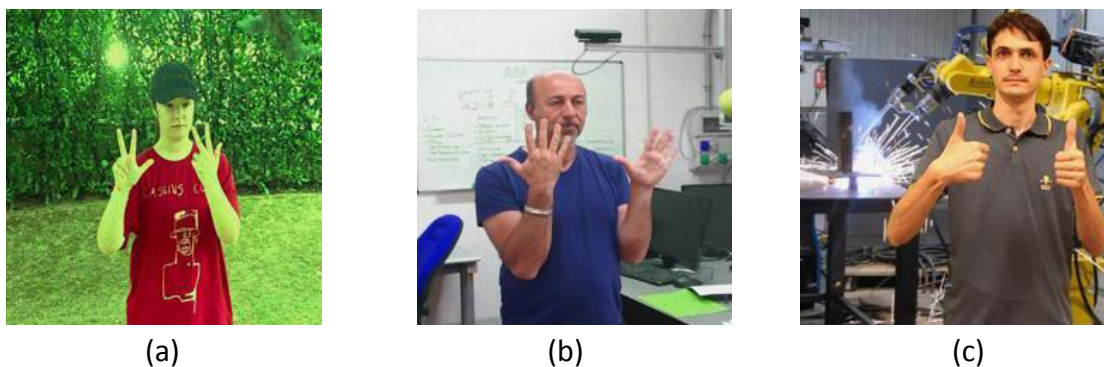


Figure 14. Examples of the augmenting techniques. (a) Spatial distortion along the y-axis, RGB distortion and speckle noise added on top, (b) spatial distortion along the y-axis, HSV distortion and speckle noise added on top, (c) artificial background added, random crop and zoom, speckle noise added on top.

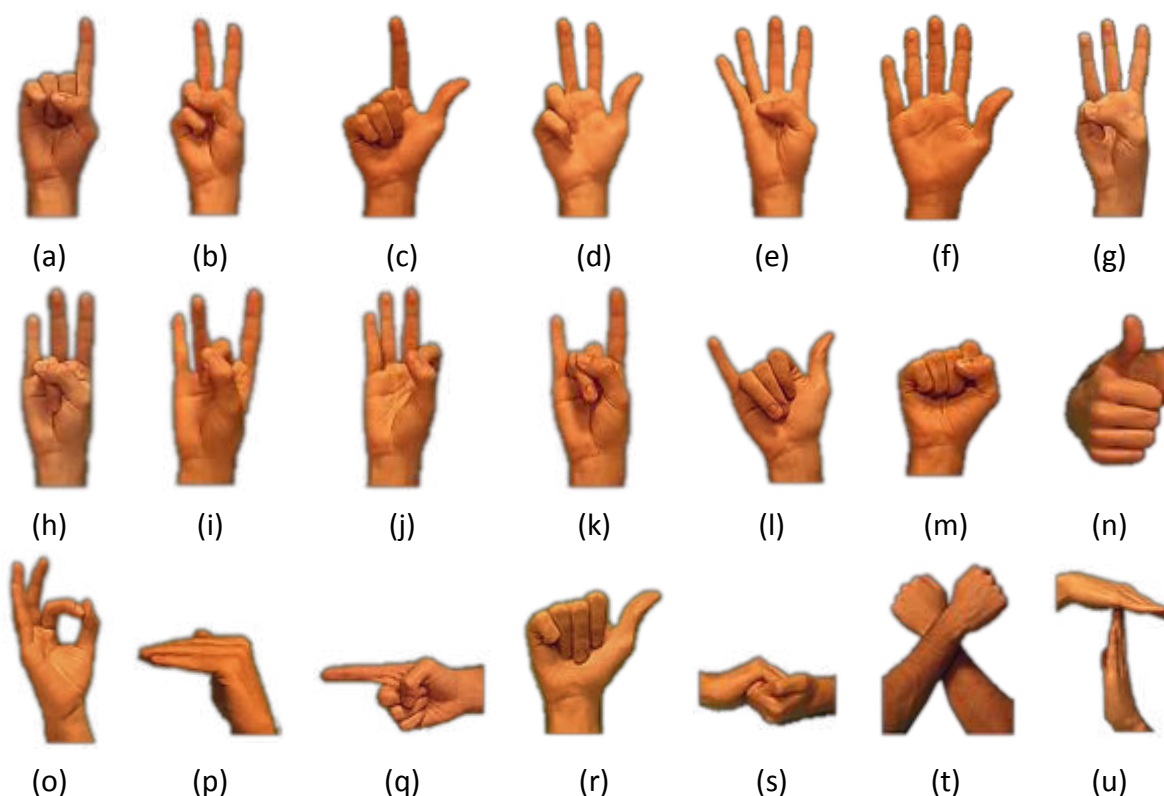


Figure 15. Experimental gestures proposed. (a) One\_FR, (b) Victory\_FR, (c) Two\_FR, (d) Three\_FR, (e) Four\_FR, (f) Five\_FR, (g) Six\_FR, (h) Seven\_FR, (i) Eight\_FR, (j) Nine\_FR, (k) Rock\_FR, (l) Span\_FR, (m) Punch\_FR, (n) Thumb\_R, (o) Ok\_R, (p) Vel\_R, (q) OneH\_FR, (r) ThumbH\_FR, (s) Collab, (t) XSign, (u) TimeOut.



As a first approach, I wanted to understand if the model was able to properly recognize left-hand and right-hand gestures and, at the same time, back-hand and front-hand gestures. Because of this, gestures from (a) to (l) have 4 variants. Gestures from Figure 15(m) to (p) have only left and right variants, while gestures from (q) to (r) have four variants since these both are intended as “directional gestures”. Finally, gestures from (s) to (u) are two-hand gestures, thus having only one variant.

### 3.3.4.2 Performances evaluation

The first row of shows the results obtained from a training made by using only back-hand and front-hand gestures of the same hand (left or right only). This first experiment highlights that back-hand and front-hand gestures are often misclassified with each other by the model, and that gestures from “Six” to “Nine” are extremely hard to be recognized, given their similarities. The results for the other variants are supposed to be similar to the abovementioned ones, hence were not computed.

The low performances achieved are not surprising, especially considering the high number of gestures in the dataset compared to the four gestures dataset (Section **Errore. L'origine riferimento non è stata trovata.**) and their similarities, not to mention the difficulties introduced by the augmenting techniques. Hence, given the low performances obtained, I (i) removed almost every back-hand gesture and (ii) used only a limited number of gestures, some of which have been merged in a single class even if they were intended as different classes or with different meaning with respect to the original one.

Table 6. Model performances on the two different tests.

<b>Dataset</b>	<b>PPV</b>	<b>TPR</b>	<b>NPV</b>	<b>TNR</b>	<b>F1-Score</b>	<b>RA</b>
<b>Test 1</b>	73%	55%	0%	0%	58%	53%
<b>Test 2</b>	83%	82%	0%	0%	81%	77%

The modified gestures are: (i) the “Thumb” gesture, used to represent number one because the latter is frequently classified as the “Two” or the “Victory” gestures due to the fingers positions, or as the “Punch” gesture because of the presence of the knuckles; (ii) the “Span”

gesture, used to represent number four because the “Five” and “Four” gestures are frequently misinterpreted due to a wrong placing of the bounding box, which cuts off the thumb of the hand. This issue may be fixed by adopting a different model (i. e. the R-FCN model, which adopts a voting method inside the bounding box to correctly place it around the object of interest). Finally, (iii) the “Nine” gesture has been merged with the “Ok” gesture because the two can be considered the same gesture according to how much the “o” shape of the gesture is shown to the camera.

The selected gestures chosen after this evaluation are:

- The “**Punch**” gesture, in front-hand left and right variants, both under the same label name (gesture (m) in Figure 15);
- The “**Thumb**” gesture, in front-hand left and right variants, both under the same label name (gesture (n) in Figure 15);
- The “**Two**” gesture, in front-hand left and right variants, with two different label names to differentiate between them (gesture (c) in Figure 15);
- The “**Three**” gesture, in front-hand left and right variants, with two different label names to differentiate between them (gesture (d) in Figure 15);
- The “**Span**” gesture, necessary since the original “Four” gesture was often misclassified. Both front and back-hand, left and right variants are used under the same label name, to increase robustness (gesture (l) in Figure 15);
- The “**Five**” gesture, in front-hand left and right variants, with two different label names to differentiate between them (gesture (f) in Figure 15);
- The “**Ok**” gesture, merged with the Nine gesture to form a single gesture. Both left and right front-hand variants are used under the same label name (gestures (j) and (o) in Figure 15);
- The “**Collab**” gesture (gesture (s) in Figure 15);
- The “**XSign**” gesture (gesture (t) in Figure 15);
- The “**TimeOut**” gesture (gesture (u) in Figure 15).

The model has been trained and tested again on these gestures; the results are reported in the second row of . For this experiment, both right-hand and left-hand gestures have been

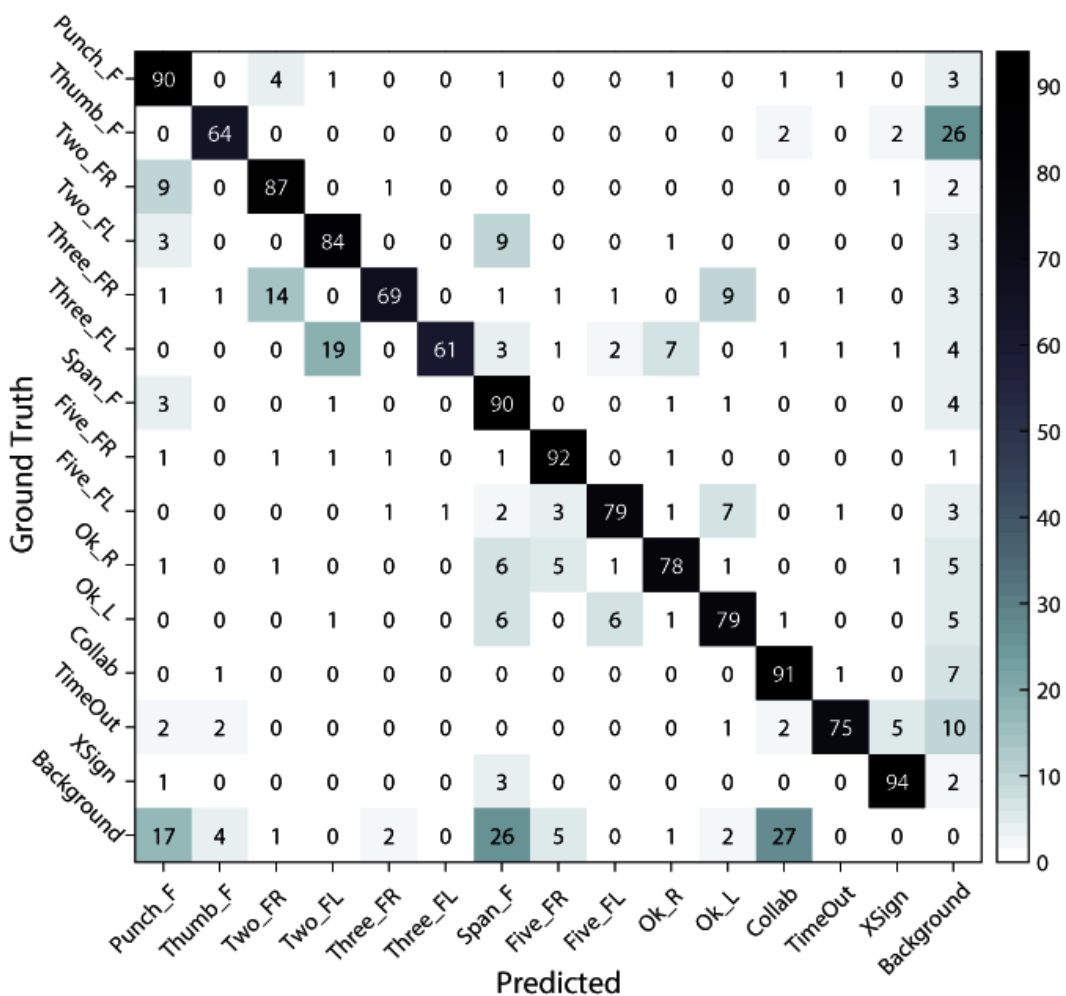


Figure 16. Normalized confusion matrix of the results of Test 2.

used for the selected gestures, for a total of 29542 images for training and 4066 for testing. It is worth noting that the number of True Negative samples was zero for both tests, resulting in the null TNR value. Figure 16 shows the normalized confusion matrix of this test. As expected, the overall performances increased compared to the previous test, while still being low with respect to state-of-the-art gesture recognition models. This was probably due to the fact that the model was not pre-trained on large datasets and that the Faster R-CNN model implemented in MATLAB was not able to properly place the bounding box around the hands, as shown in Figure 16 where the “Three” gestures are misclassified with the “Two” gestures. There is also a relatively high amount of misclassifications of the “Background” class: the “Punch”, “Span” and “Collab” gestures are mistakenly predicted when the ground truth class label was “Background” (last row of the confusion matrix in Figure 16) and the “Background”

gesture was mistakenly predicted as “Thumb” (last column of the confusion matrix in Figure 16).

### 3.3.4.3 Experimental system description

Compared to the approach seen in Section 583.3.3, this second version of the gesture language was used to move a real industrial robot in real time thanks to a command software based on a State Machine structure. Hence, the gestures are not only gestures but also operative commands when elaborated by the developed software detailed in [98]. Similar to the first approach, a filtering procedure has been implemented to improve the detection robustness and combine single-hand gestures in two-hand gestures accordingly. After the detector makes a prediction, the centroid of the predicted box undergoes a checking procedure according to several conditions detailed in [98] such as: (i) the *confidence score*, which has to be equal or greater than 80%, (ii) the *working zone* of the centroids, (iii) the *operative state* of the State Machine, which accepts only a pre-defined subset of gesture commands, (iv) the presence of the requested second hand gesture if necessary and (v) the number of predictions of the same class around the same centroid position, which must be at least 2 on a total count of 4 consequent predictions.

The *working zones* are defined by the software according to the centroid position of the human operator, found after the initialization step (**Errore. L'origine riferimento non è stata trovata.**). These are:

- **Left zone**, where only left-hand gestures can be detected;
- **Right zone**, where only right-hand gestures can be detected;
- **Center zone**, where only two-hand gestures can be detected.

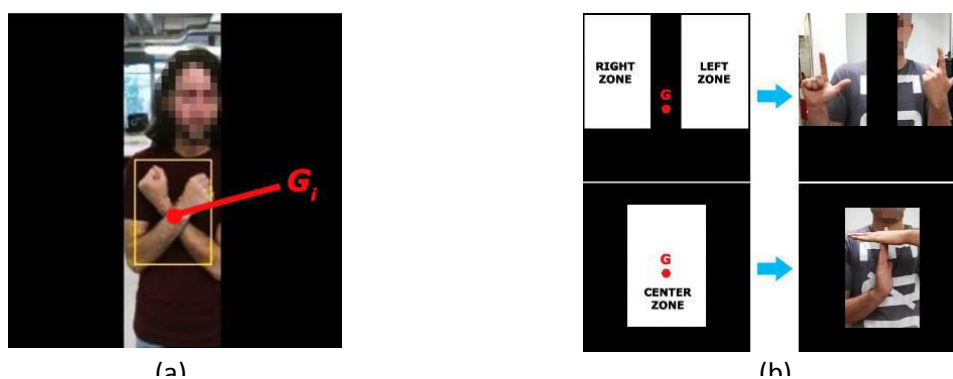


Figure 17. Initialization procedure. (a) Centroid detection, (b) working zones definition.

Furthermore, I expanded the number of possible commands by using both single-hand gestures and two-hands gestures, obtained as a combination of single-hand gestures. Commands are then subdivided by their meaning and usage as follows:

- **Numerical commands:** gestures from “Punch” to “Five”, indicating numbers from 0 to 5. These are performed with a single hand;
- **Interface commands:** these are used to confirm, cancel or undo the given commands, thus avoiding giving the robot the wrong command by mistake (Figure 18). Each command is performed with both hands to create a more robust instruction. To confirm an instruction the “Punch” gesture must be performed by both hands at the same time. To delete an instruction and re-enter it in the system, the “Five” gesture must be performed with one hand and the “Punch” gesture with the other. To cancel an instruction, the “Five” gesture must be performed by both hands at the same time;
- **System commands:** these commands are used to send precise instructions to the robot or to the Command System. The “XSign” gesture is used both for starting the communication with the system and for closing the communication. To start a robot execution the “Punch” gesture must be performed with both hands, while to stop it, it is enough to show to the camera a single hand “Five” gesture. The emergency stop is obtained by using both hands to perform the “Five” gesture. The “TimeOut” gesture is used to pause a robot execution (i. e. a loop task), that can be re-started by the start command. The “Collab” gesture is used to display the instructions available in that state of the state machine. The “Ok” gesture is used to change the robot speed: it is expressed as a percentage, obtained as the slope of the segment having the extremes in the centroids of the boxes of the two hands both performing the “Ok” gesture.

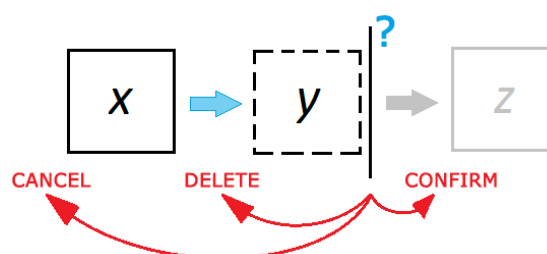


Figure 18. Example of the Interface commands. When the user is prompted with an Interface command request, it is possible to confirm the instruction and proceed with the operation, or to delete (thus returning to a previous state and re-enter the command) or cancel the operation.

Working zones are necessary to filter out incorrect predictions that may be detected on wrong body parts (i. e. the head) or on the background. This obviously pose the problem of the naturalness of the interaction: albeit no experimental evaluation of the user experience has been conducted to determine its impact, it is very likely that it reduces the overall satisfaction of the user when using the system. Using working zones proved to be a necessity for this system because of both the dataset composition and the model adopted. In fact, using such a vast number of gestures and allowing the operator to use only one hand at a time instead of two like in the previous approach detailed in Section 3.3.3 required the system to use strict constraint. A better model or a better training procedure (e. g. fine-tuning) could improve the recognition ability of the network, thus reducing the number of necessary constraints. On the other hand, even if the adoption of Interface Commands may seem like another limit to the naturalness of the interaction, they are very useful to filter out logic mistakes made by the user. For example, when a person accidentally sends a wrong waypoint to the robot it would be best to stop it before execution: this can only be made if a confirmation procedure intervenes before it, thus improving the overall safety of the system with respect to this kind of events.

### *3.3.5 Third approach: the gestures dictionary*

Thanks to the findings of the previous approaches, I finally designed the definitive version of the proposed gesture language. Single-hand gestures are not enough to create a flexible and robust gesture language because, by using only single-hand gestures, the obtained language would be unary, thus highly inefficient. Not to mention that Object Detectors are noisy in their detections, resulting in wrong predictions because of light, scene, and color conditions as seen in the previous tests. Adding the second hand means that the language obtained can express more information and is more robust because both right and left-hand gestures must be correctly recognized at the same time. This is why in this final approach I defined two-hands gestures as *Commands*, divided in two groups: **Static Commands**, which represent always the same operation or a set of operations with very similar meaning, and **Parametric Commands**, that can be dynamically composed during execution according to the specific value set by the user. This is the case of quantities, which can be single-digit or two-digit, and movements,

which are represented by a number indicating the robot joint to move and a gesture representing the type of movement required. Commands define the **gestures dictionary**, which users must use to communicate with the robot.

### 3.3.5.1 Dataset definition

The HANDS dataset is an improved version of the dataset described in Section 3.3.4. It is composed of 15 static hand-gestures represented in Figure 19, which are:

- Digits from 1 to 9 from (a) to (i), the 0 digit is represented by the gesture in **Errore. L'origine riferimento non è stata trovata.**(j);
- A Span gesture, shown in (k);
- A directional gesture pointing left or right accordingly, shown in (l);
- Two-hand gestures that represent intuitive commands, from (m) to (o).

Gestures from (a) to (k) have two variants: one performed with the right hand and one performed with the left hand. Gesture (l) has four variants, according to the direction the hand is pointing to and the hand used: right hand pointing left (back facing the camera) and right (palm facing the camera), left hand pointing left (palm facing the camera) and right (back facing the camera). Gestures performed with both hands have only one variant (from (m) to (o)). Thus, considering all the variants, the HANDS dataset comprehends 29 different classes.

To correctly assign the labels to each variant, each class is named after the gesture name (e. g. “Nine”) and has a suffix to identify the variant of the gesture, following this structure:

- **V** stands for vertical gesture, while **H** stands for horizontal gesture;
- **F** identifies the front facing camera version of the gesture, while **B** identifies the back facing the camera version;
- **R** stands for right hand gesture, while **L** stands for left hand gesture.

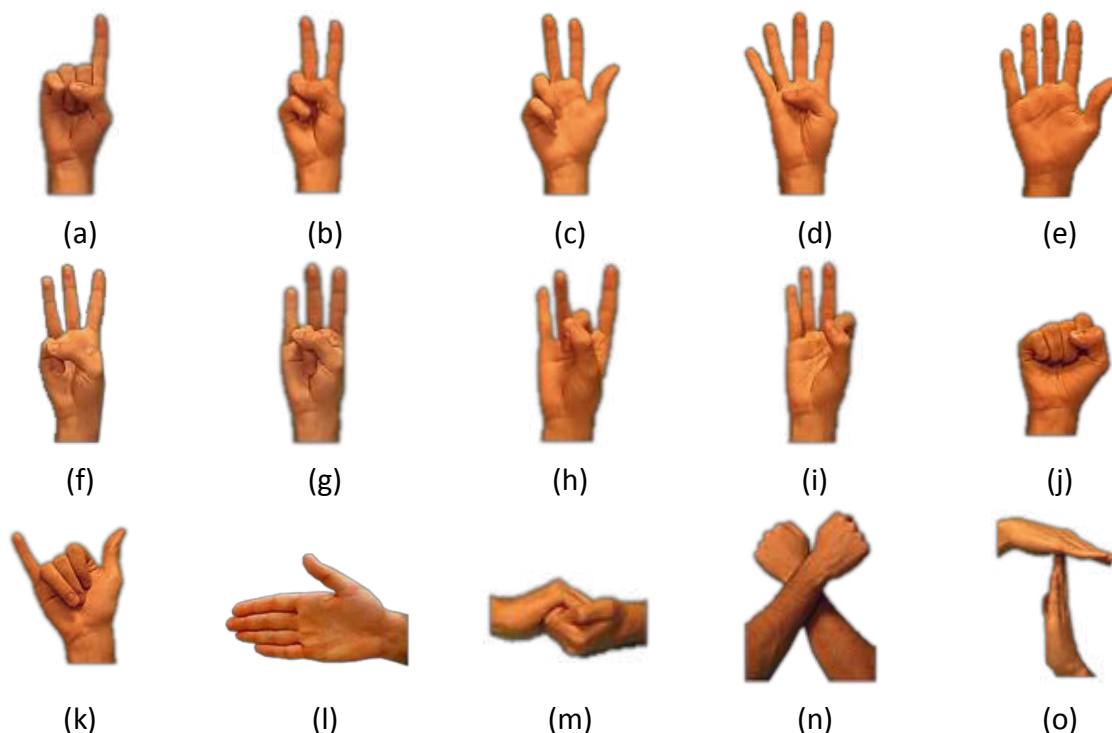


Figure 19. Front right-hand gestures variants. (a) One\_VFR; (b) Two\_VFR; (c) Three\_VFR; (d) Four\_VFR; (e) Five\_VFR; (f) Six\_VFR; (g) Seven\_VFR; (h) Eight\_VFR; (i) Nine\_VFR; (j) Punch\_VFR; (k) Span\_VFR; (l) Horiz\_HFR. (m) Collab; (n) XSign; (o) TimeOut.

Table 7. Static Commands.


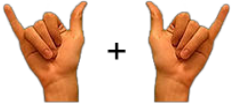



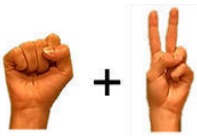
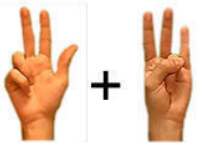
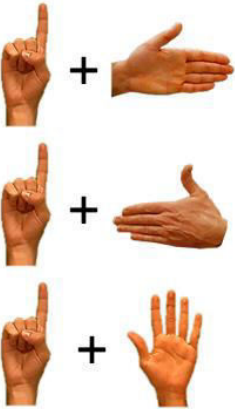
Gestures	Description
	<b>CONFIRM:</b> Punch_VFR and Punch_VFL as in Figure 19 (j)
	<b>DELETE:</b> Span_VFR and Span_VFL as in Figure 19 (k)
	<b>EXIT:</b> XSign gesture as in Figure 19 (n)
	<b>PAUSE:</b> TimeOut gesture as in Figure 19 (o)
	<b>OPEN FILE:</b> Collab gesture as in Figure 19 (m)

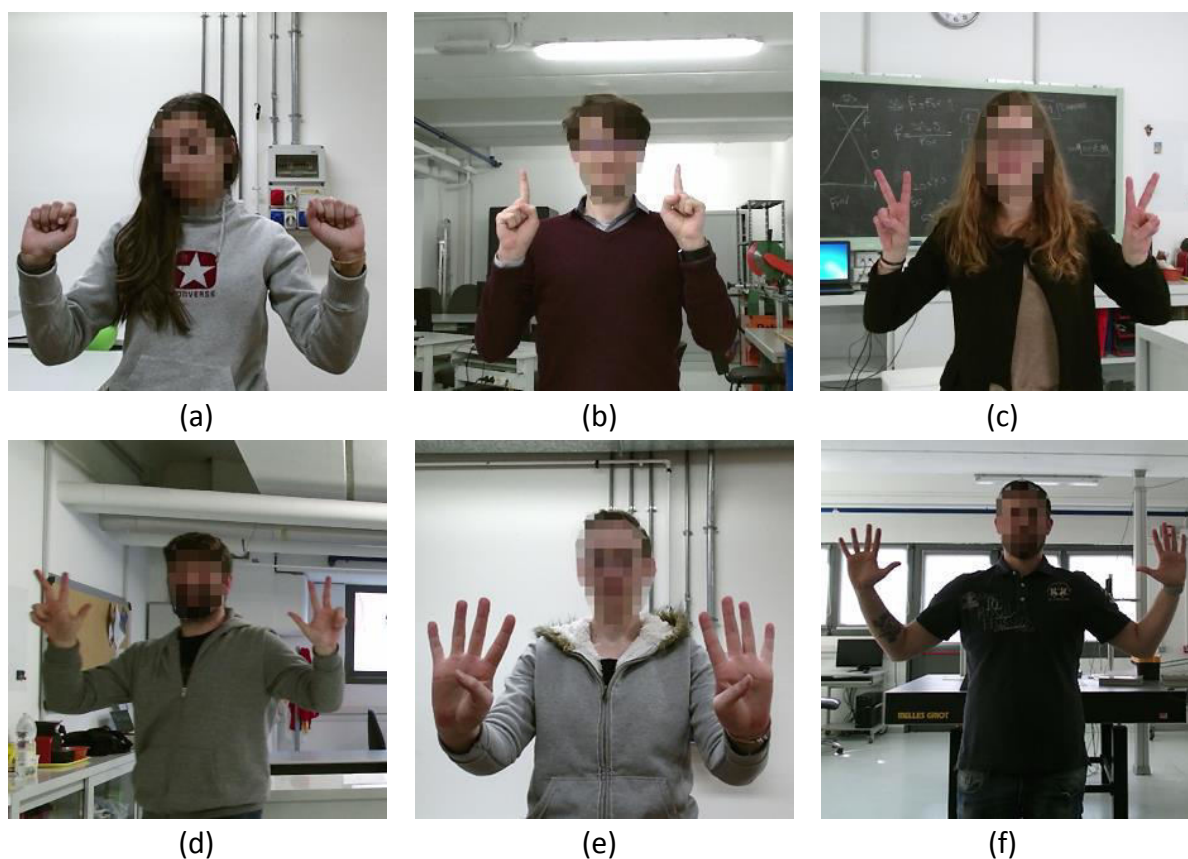


Table 8. Parametric Commands.

Gestures	Description
	<p><b>SINGLE-DIGIT NUMBER:</b> <i>Punch_VFR</i> as in Figure 19 (j) and <i>Number_VFL</i> (gestures from Figure 19 (a) to Figure 19 (i) performed with the left hand). The value is calculated as <math>0 + N = N</math>. For example, the figure shows number 2.</p>
	<p><b>TWO-DIGIT NUMBER:</b> <i>Number_VFR</i> + <i>Number_VFL</i> (gestures from Figure 19 (a) to Figure 19 (i), where the right-hand gesture represents tens). For example, the figure shows number 37 composed by using number 3 (Figure 19 (c), right-hand) + number 7 (Figure 19 (g), left-hand).</p>
	<p><b>JOINT MOVEMENTS:</b> Number (from Figure 19 (a) to Figure 19 (i)) + Value (Figure 19 (l) or Figure 19 (e)). In this case the order of the hands is not important, so the number to represent the robot Joint can be performed either using the right or the left hand, and the Value gesture with the other hand. To increase the position of the robot Joint the user must use the Hand gesture pointing to the right (<i>Hand_HBL</i> and <i>Hand_HFR</i>); to decrease the position of the robot Joint, the ones pointing to the left (<i>Hand_HFL</i> and <i>Hand_HBR</i>). To stop the robot in a specific position the Value gesture is the Five gesture (Figure 19 (e), VFR or VFL accordingly).</p>

A total of 6 actors have been acquired performing the gestures, equally divided between males and females. For each actor, a different background has been chosen to increase the variability of the dataset (Figure 20). To speed up both the acquisition and the labelling process, the actors were told to perform the same gesture with both hands. The performing style of the actors was twofold: actors 1, 2, 5 and 6 performed the gestures standing in the

same position front-facing the camera, the hands moving (i) towards the camera, in order to acquire different depth values for the hands, and (ii) laterally at shoulder height, carefully rotating the hands in order to still be able to recognize the correct gesture. On the other hand, actors 3 and 4 performed the gestures while moving randomly in the field of view of the camera, also moving the hands towards the camera and laterally. Light conditions in the scenes were also different: in the scenes used for actors (a) and (e) only artificial light was used, while for the others a combination of artificial and natural light has been used. After carefully selecting the most representative images for every gesture and actor, discarding the blurred and occluded ones, we selected a total of 150 RGB frames and their corresponding 150 depth frames per gesture for every actor. This assures that every actor contributes to the training in the same way.



*Figure 20. Examples taken from each actor dataset. It is worth noting that even if gray color is always present in the background, around the hands the background can vary in color and light conditions.*

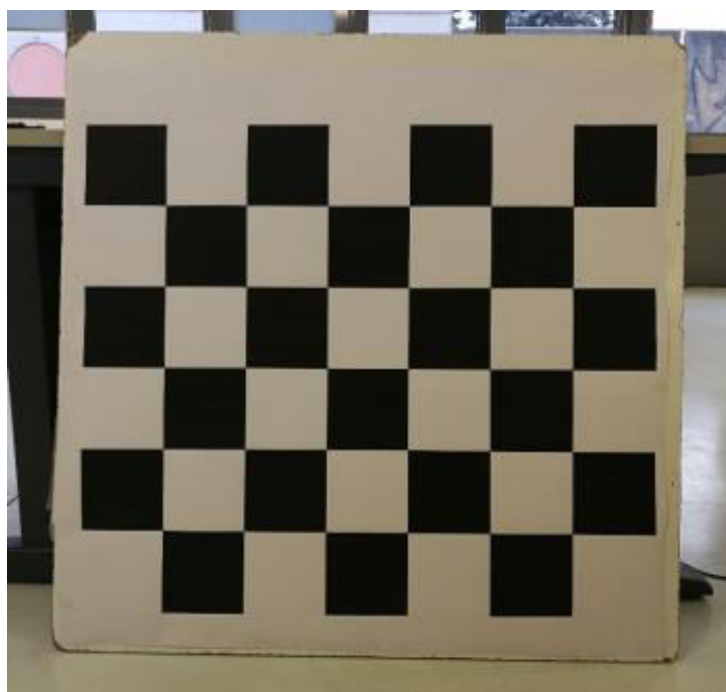
The dataset is composed as follows:

- **Actor 1 (F):** 2400 RGB images and their corresponding 2400 Depth images, 150+150 frames per gesture. Grey uniform background with artificial light only, actor standing still;
- **Actor 2 (M):** 2400 RGB images and their corresponding 2400 Depth images, 150+150 frames per gesture. Cluttered background with artificial light only, actor standing still;
- **Actor 3 (F):** 2400 RGB images and their corresponding 2400 Depth images, 150+150 frames per gesture. Cluttered background with a combination of natural and artificial light, actor moving;
- **Actor 4 (M):** 2400 RGB images and their corresponding 2400 Depth images, 150+150 frames per gesture. Cluttered background with a combination of natural and artificial light, actor moving;
- **Actor 5 (F):** 2400 RGB images and their corresponding 2400 Depth images, 150+150 frames per gesture. Grey uniform background with artificial light only, actor standing still;
- **Actor 6 (M):** 2400 RGB images and their corresponding 2400 Depth images, 150+150 frames per gesture. Cluttered background with intense natural light from behind, actor standing still.

The dataset has been acquired using a Kinect v2 sensor [95] intrinsically calibrated to *spatially align* the Depth and RGB frames. Since the sensor acquires the frames at different times, a *temporal alignment* procedure of the frames has been carried out afterwards. The images have been acquired frame by frame at 30 fps using ROS [99], `libfreenect2` [100] and `iai_kinect2` [101] ROS packages. I saved each acquisition into a *rosbag* and processed them afterwards using MATLAB. To achieve the **spatial alignment**, it is required that the sensor used is intrinsically calibrated to properly align the colour information on top of the depth information. This procedure has been carried out in advance using the calibration tool of the `iai_kinect2` package: a chessboard with a grid of 7x6 squares of 120 mm has been used to calibrate the sensor (Figure 21). It is also worth noting that the intrinsic calibration allows the `iai_kinect2` package to correct the camera distortions while acquiring the frames: because of this, I acquired the corrected frames with resolution of 960x540 px.

To achieve the **temporal alignment**, each acquisition has been elaborated using MATLAB:

- To perform the calibration, a reference stream must be identified to which refer the others. I selected the reference frame from the list of messages according to which one of the two streams contained less messages;
- Then, I have found the corresponding frame in the other stream that has the minimum temporal distance from the selected reference frame;
- If the temporal distance is less than *66 ms*, the couple is valid and properly saved.



*Figure 21. The chessboard used to calibrate the Kinect v2 sensor. It was printed and glued on a wooden support to move it around easily during the calibration process.*

### *3.3.5.2 Experimental system description*

Thanks to the findings observed in the previous approaches, I decided to change the model type and framework as well as the training procedure. In this case, I fine-tuned an R-FCN Object Detector [63] pre-trained on the COCO dataset using the TensorFlow Object Detector API [65]. Albeit HANDS is composed of both RGB and depth images spatially and temporally aligned, I choose to only train the model with the RGB frames. The training lasted 400000 epochs with a batch size equal to 1 on a machine equipped with a single GPU Nvidia GeForce GTX 1060 6GB and 16 GB RAM. To further increase the generalization capability of the model

and to decrease the influence of the light conditions of the scene, I chose to apply a set of augmenting techniques during the training of the dataset. These were chosen from the list of available augmenting operations of the API, which are: (i) random scaling of pixels, (ii) random scaling of the image, (iii) random conversion of the image to a grayscale image, (iv) random adjustments of brightness, hue, contrast and saturation, (v) random color distortion and (vi) random application of black patches on the image.

### 3.3.5.3 Performances evaluation

The resulting model described in the previous Section achieves the performances reported in Table 9 on the test dataset. The null values of NPV and TNR are due to the lack of purposely wrong gestures in the test dataset, resulting in the lack of True Negatives. The performances obtained show that the choice of the model as well as the new dataset definition are a remarkable improvement compared to the previous approaches, achieving competitive results with state-of-the-art models. This is due two main reasons: the fine-tuning procedure and the model characteristics, which allow for a precise bounding box positioning around the hands, thus allowing to differentiate gestures such as “Four” and “Five” which only differ for the presence of the thumb. Nonetheless, the model shows some recognition faults in the case of “Seven” and “Eight” gestures, that are misclassified in a high number of cases as the “Six” gesture or as “None”, as it can be seen from the normalized confusion matrix in Figure 22.

It is worth noting that, even though the HANDS dataset comprehends depth images aligned both temporally and spatially with their corresponding RGB images, the former images have not been used at the present time but may be used in a fusion network to further improve the gestures recognition.

Table 9. Test results of the R-FCN model tested on the HANDS test dataset.

<b>Dataset</b>	<b>PPV</b>	<b>TPR</b>	<b>NPV</b>	<b>TNR</b>	<b>F1-Score</b>	<b>RA</b>
<b>HANDS</b>	94%	90%	0%	0%	91%	90%

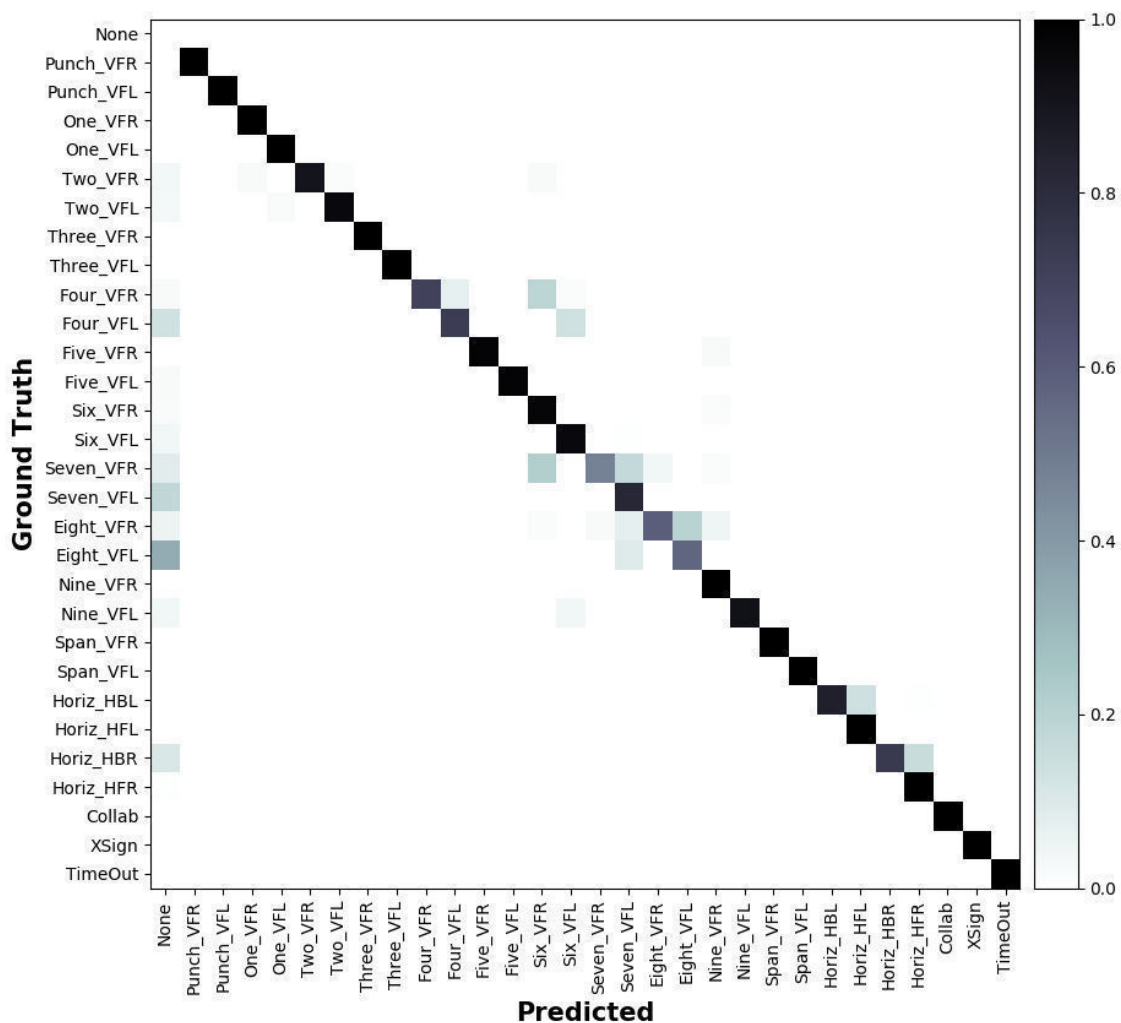


Figure 22. Normalized confusion matrix obtained testing the fine-tuned model on the test dataset. The “None” label refers to frames where no gestures were performed. It is worth noting that each frame in the dataset contains at least one gesture, resulting in a row of zeros in correspondence of the “None” Ground Truth label.

## Chapter 4. The MEGURU System

As discussed in Chapter 1, the Industry 4.0 paradigm introduced the concept of “collaboration” between humans and machines, which resulted in the development of Co-bots [28] [27]. Therefore, two types of robotic units populate the production lines of manufacturing industries: **Industrial robots**, which operate at high speed and force and are kept inside safety cages to protect humans from any harm, and **Collaborative robots**, which are designed to operate near humans at lower speed and force and possess a safety system to prevent dangerous contact and reduce the eventual harmful impacts [32] [33]. These two types of robotic units are adopted in two different workstations: **fully automated workstations**, characterized by a high production rate, and **collaborative workstations**, where the robot assists the human and speed up a traditionally slower procedure.

This thesis work proposes a new type of workstation called “**Meta-Collaborative workstation**” (MCW), where the robot can operate behind a safety cage, either physical or virtual, and the operator can interact with the robot, either Industrial or Collaborative, in an intuitive and natural way. To this aim, I developed **MEGURU** (MEta-collaborative GestUre-based Robot program bUilder), an open-source software to easily build robot programs purposely designed for MCWs that may be a valid alternative to traditional robot programming methods (e. g. the teach pendant). As a further contribution, I made the HANDS dataset (Section 3.3.5) and the project code publicly available [102].

### 4.1 The idea

To program a robot, either Industrial or Collaborative, different programming methods can be adopted as stated in [103] [104] [105]. Conventional approaches are subdivided in **online** and **offline** programming methods: in the former case, the robot cell is actively used during the programming process, while in the latter the robot cell is not involved in the task definition, thus non-productive robot time is kept at minimum. Online teaching methods involve *walk-through programming*, where the operator manually moves the robot along the trajectory to

be reproduced, and *lead-through programming*, where the operator defines the robot program using only the teach pendant, a method that does not require any contact with the robot. Both methods require the operator to have proper knowledge of the robot to be used and a suitable programming experience. Offline programming methods require a higher knowledge of both programming languages and robotics fundamentals and are usually adopted in a simulated environment to build complex robotic cell programs. These methods are extremely time consuming and often stress the operator. Collaborative Robots often adopt a hybrid programming method combining both typologies: the logic of the program can be developed offline in a simulated environment, while the motion instructions and the positioning can be programmed online.

As discussed in Section 1.3, key technologies such as virtual/augmented reality and Co-bots are not properly known and used by the majority of SMEs, especially in Italy [5]. This may be due to the fact that companies do not properly understand the technology or the application and that the required hardware and know-how to adopt it is too high or unavailable. Hence, a simple, cost-effective and intuitive communication system that can be used by operators with limited programming experience may be a strategic choice for small-medium companies to move towards the Industry 4.0 revolution. To this aim, intuitive programming methods have been proposed by the scientific community to make the robot programming phase more intuitive for users, allowing ordinary people to program robots with minimal skills and training. Wearable sensors may be an interesting solution: for example, authors of [76] propose an HRI system based on hand gesture recognition which leverages wearable sensors such as IMUs. They adopted a combination of 8 static gestures and 4 dynamic gestures and the IMUs' data is acquired and elaborated by an unsupervised segmentation algorithm based on motion and classified afterward by an Artificial Neural Network (ANN) purposely trained for the task.

In late years, vision has become the preferred method to perform a plethora of different tasks, including hand-gesture recognition, robot teleoperation, scene monitoring, and scene reconstruction. This is due to the fact that these sensors are contact-less, reliable, and cost-effective, considering that a single camera can be adopted in combination with a suitable Computer Vision or Deep Learning algorithm to achieve satisfactory results depending on the



task. Adopting vision sensors to monitor the surroundings of the robot allows the definition of safety strategies based on human-robot relative positions in parallel with the human-robot communication system [78] [31], also adopting mixed reality techniques as shown in [106]. A working example of this idea is presented in [107], where the authors developed a framework to allow the safe coexistence of humans and robots in the same production cell based on two Kinect v2. According to the position of the operator obtained from the depth sensors, the robot behavior is modified in real-time to avoid unsafe interactions and any possible collisions. A very limited set of hand gestures is used to guarantee a few collaborative interactions, recognized by extracting the body skeleton of the user leveraging the sensor built-in skeletonization algorithm.

Modern techniques such as (i) multimodal approaches, (ii) virtual reality and (iii) augmented reality approaches, which largely use image and speech recognition algorithms, are the most promising methods to further simplify robot programming. Among the proposed methods, multimodal interfaces for human-robot interaction (HRI) based on speech and/or gesture recognition systems have been intensely studied in the last years. In fact, as highlighted in [108] HRI is based on cognitive capabilities such as perception, which can be guaranteed by adopting and integrating sensors and actuators. Although speech is considered the most intuitive method of communication for humans, it is unpractical to adopt in industrial environments, traditionally characterized by strong noises. Furthermore, a communication system based on speech recognition depends on the user language and accent, forcing the adoption of different language models. Hence, gesture and pose recognition methods are a more viable solution. In [80] a flexible system for gesture-based HRI is presented, where the authors propose a system based on the recognition of upper arms gestures, utilizing a Machine Learning algorithm called Adaptive Naïve Bayes Classifier that classifies the gestures by analyzing the pose skeletonization extracted from the depth image of the scene. Authors of [81] present a similar system, based on ROS and the recognition of full-body poses acquired by a Kinect camera and a Leap Motion sensor. Here, the hierarchical representation of tasks allows users to build programs flexibly and intuitively. Similarly, in [82] a system based on a Gestures Dictionary is used to dynamically build a robot program where the adopted gestures

are acquired by a Kinect sensor that extracts the human skeleton (static gestures) and by the Leap Motion sensor that tracks the hands' movements (dynamic gestures). The gestures are designed to move the robot in a modality similar to the traditional Jog mode, where the robot end-effector position is adjusted by a certain pre-defined step according to the direction of the movement (up, down, left, right). The experimental results show that even for non-expert users the adoption of body gestures to move the robot was more convenient in the case that the accuracy of the positioning was not required, and overall this approach was preferable compared to traditional methods. Authors of [97] present an application of a human-robot interaction framework based on the recognition of 10 static hand gestures and a finite state machine, an approach also adopted in this work. The procedure developed by the authors first extracts the human skeleton from the RGB-D frames acquired by a Kinect v2 camera using the OpenPose network [109] to find the position of the user's hands. Then, their algorithm places a bounding box around the obtained location and the object inside it is classified by using a version of the Inception V3 CNN network fine-tuned to recognize the proposed gestures. The main issue of this approach is that the procedure is time-consuming and computationally expensive; hence, the recognition system requires at least a dedicated GPU to achieve a satisfactory frame rate.

Ubiquitous Computing and Manufacturing (UC and UM) are concepts in which computing may be performed everywhere, hence promoting a decentralized computing structure and a "design anywhere, make anywhere, sell anywhere, and at any time" paradigm [110]. Hence, since MCWs are designed to be flexible MEGURU has three fundamental characteristics:

1. **Robot Independent:** it is based on ROS, hence it can be used regardless of the robot manufacturer brand and of the robot type (industrial robots or Co-Bots);
2. **Easy to program:** MEGURU adopts an innovative programming method based on hand-gestures, designed to allow users to create robot programs in an intuitive and easy way. As a result, even operators without programming experience can efficiently use the software to program a robot operation, reducing the training time required to learn how to use the specific robot;

3. **Flexible production:** the programming structure of MEGURU allows users to define tasks (e. g. a pick and place task) that can be used as building blocks to build more structured operations. This feature allows an easy and fast reconfiguration of the MCW to better adapt to a mixed production.

MEGURU's programming structure is based on three types of building blocks of increasing complexity:

- **Points:** these are the base blocks that compose MEGURU programming structure. They represent the positions of the robot end effector in the current reference system, expressed by (i) cartesian coordinates or by (ii) Joints position. Points are collected by the user either using ROS or the robot proprietary software, and are stored in an ordered list, thereafter called "Points file", as shown in Figure 23 (a);
- **Actions:** these are parametric functions that represent a simple action of the robot (e. g. opening and closing a gripper or moving the robot to a certain point). A dedicated Python library based on ROS communication functionalities has been developed to (i) guarantee the independency of the Actions from the robot manufacturer platform and (ii) allow the user to easily define different Actions according to the application needs, as shown by the pink blocks in Figure 23 (b);
- **Operations:** by using MEGURU, users can build two types of Operations: **Simple Operations (SOPs)**, obtained by combining different Actions, and **Combined Operations (COPs)**, obtained by joining multiple SOPs. As a result, MEGURU allows users to reconfigure the robot tasks in reduced times and to adapt the robot to a mixed production minimizing production downtime. An example of a SOP is shown in Figure 23 (b), with respect to a Pick&Place task, while examples of the resulting COPs are shown in Figure 23 (c) and Figure 23 (d).

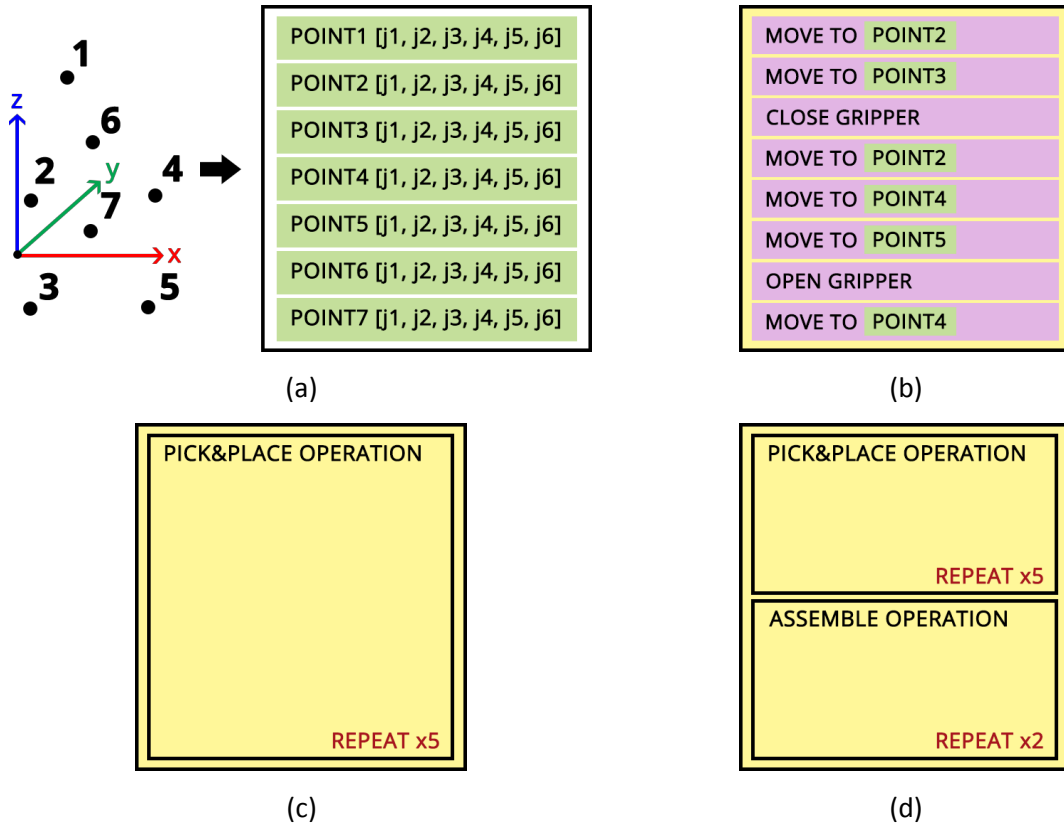


Figure 23. (a) Example of a Points file. Points are represented as green blocks in the file. (b) Example of a SOP representing a Pick&Place task. Actions are represented as pink blocks in the SOP file, represented as a yellow block. (c) Example of a COP obtained by repeating the Pick&Place SOP for 5 iterations. (d) Example of a COP obtained by joining the Pick&Place SOP (repeated for 5 iterations) and the Assemble SOP (repeated for 2 iterations).

## 4.2 System layout in detail

The layout of the MEGURU system is represented in Figure 24. White blocks are ROS nodes and gray blocks are ROS topics, used to provide the communication between nodes. It is worth noting that to effectively communicate with a robot, it is necessary to translate ROS commands into operative robot-specific controller instructions. In the current version of the system, this is provided in two ways: one is based on the use of ROS-Industrial (ROS-I) [111] “Robot Drivers”, the other one is based on the use of Move-It [112]. In both cases, ROS reads the joint positions from the `/joint_path_command` topic and writes the robot joint states in the `/joint_states` topic by using the chosen robot-specific driver. A list of robot drivers is available in [113]. Custom made solutions can be easily adopted and used as “Robot Drivers”, provided that the communication between them and the State Machine node adopts the abovementioned two topics.

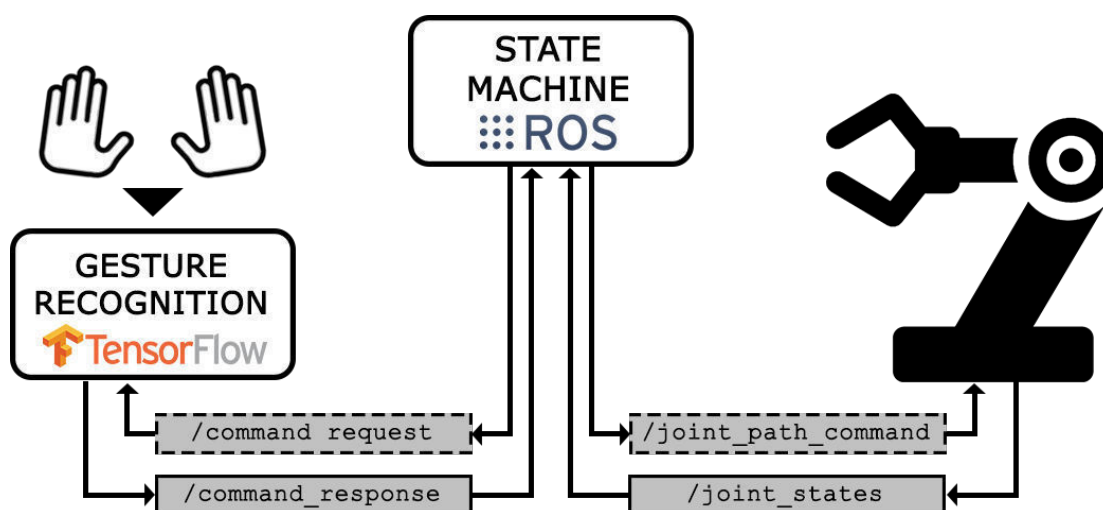


Figure 24. Scheme of ROS nodes and their respective topics. Gray blocks with dashed edges are the request topics, used by the State Machine node to request a specific Command (*command\_request*) or a robot movement (*joint\_path\_command*). Gray blocks with a solid edge are the response topics, where the recognized Command (*command\_response*) or the robot position feedback (*joint\_states*) are contained.

#### 4.2.1 State Machine node

The layout of the State Machine is presented in Figure 25. When launched, the State Machine is in the initial state, called **Ready State**. In this state, the State Machine waits for the user Commands and can move to (i) the EXIT State (where the program quits) or (ii) to the **Home State**. From this state, the user can access to one out of the following four states (see Table 7 and Table 8 for the list of Commands):

1. **SOPs Building State (SB)**: in this state, the user selects Actions from the Action library and, if requested by the Action, selects Points from the Point File. Each single Action results in the corresponding robot task and is immediately executed by the robot. Complex tasks are implemented as SOPs built by means of a user-machine collaboration that combines different Actions and saves them in the corresponding Operation file.
2. **COPs Building State (CB)**: the State Machine enters in a loop where the user can select, for each single SOP, the corresponding Operation file and the number of iterations that SOP must be repeated for; then, the State Machine moves to the **COPs Launch State (CL)** where each single Action of the COP is sent to the robot until the whole sequence is performed. The execution of each Action can be paused by the user using the “PAUSE”

gesture and resumed by performing the “CONFIRM” gesture. It is also possible to stop the execution completely by performing the “EXIT” gesture at any time: in this case, the State Machine moves back to the Ready State.

3. **Jog State (J)**: in this state, users may perform the robot jog mode, which allows to manually move the robot joints. It is composed of two states: the **Jog Mode State (JM)**, where operators may increment or decrement the position of the gesture-selected robot joint, and the **Jog Step State (JS)**, where operators may increase or decrease the default Jog step size by changing the parameter in the corresponding file. Users can save a certain position by performing the “CONFIRM” command without exiting or pausing the Jog State.
4. **Robot Speed State (RS)**: in this state, the overall movement speed of the robot can be modified. The default speed is set to 100%, but users can decrease it by performing the gesture which correspond to a lower percentage of the total speed of the robot (e. g. to obtain the 80% of the total speed, the user must perform the instruction which corresponds to number 8). It is important to stress that this parameter affects the whole

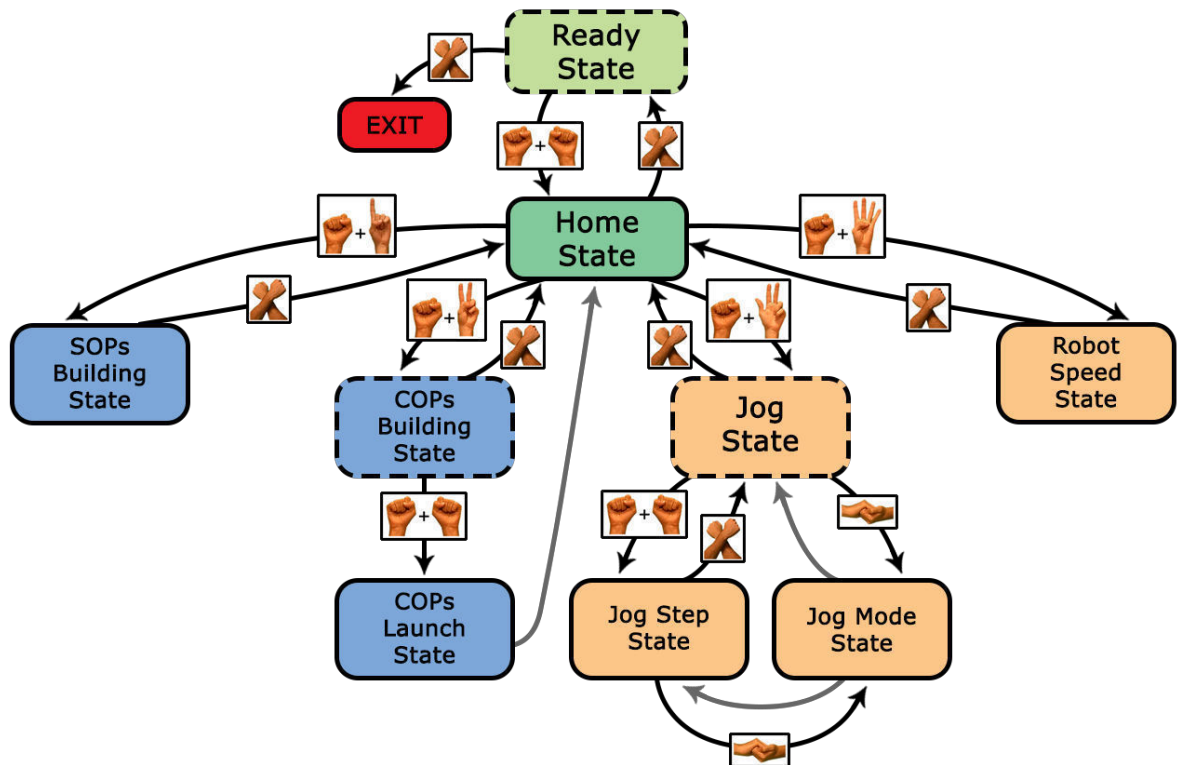


Figure 25. Layout of MEGURU State Machine. Dashed edges identify states containing sub-states. The “EXIT” status quits the application. Blue blocks represent Operative States and orange blocks represent Service States. Gray arrows represent automatic transitions between States.

Actions equally, meaning that the robot speed does not change dynamically; instead, it is the same for the whole SOP until it is changed by the user. This allows the users to modify the robot speed parameter according to their needs without the aid of the Teach Pendant. For a more precise control of the robot speed or to set it dynamically (e. g. each Action of a SOP must be executed at a certain velocity different from the others) this state is not enough.

The State Machine has been implemented using SMACH [114], a ROS package for designing State Machines. The State Machine node manages the communication using a request-response system which leverages ROS functionalities. An example is shown in Figure 26: the State Machine node sends a Command request in the `/command_request` topic, to enable the Gesture Recognition node detection function. The Gesture Recognition node outputs a valid Command in the `/command_response` topic and, upon receiving the response, the State Machine node reads the Command and acts accordingly. To correctly match requests and responses, each request has an incremental ID value called `request_number`, which must be the same of the corresponding response. To correctly decode the gestures detected in the correct Command value, the request asks for a specific `request_type`. This field is useful as a debug tool for users, as well as the `active_state` field, which memorizes which state was the active one when the request was sent. The time stamp of each message is saved in the `header` portion of the messages, defined as the standard Header message of ROS.

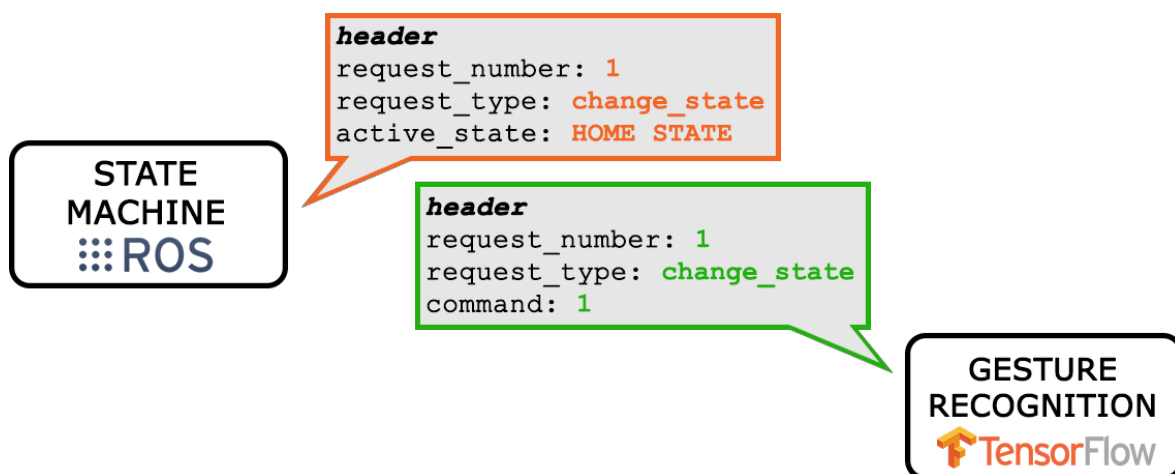


Figure 26. Example of a request-response exchange between the State Machine node and the Gesture Recognition node.

A similar communication procedure is carried out between the State Machine node and the robot. In this case, the State Machine node sends a `JointTrajectory` message in the `/joint_path_command` topic. This is a standard message of ROS, and it is used to communicate the cartesian position to the robot. The State Machine node waits until the position is reached, or a “STOP” or “PAUSE” Command is received. Thanks to ROS, it is possible to read the robot encoder to obtain the current pose of the robot, which is written as a `JointState` message in the `/joint_states` topic. Hence, to know if the desired position has been reached the State Machine checks if the position values in the last `JointState` message received are equal to the desired position values  $\pm 0.01$  radians.

To guarantee that the correct Command has been received, the State Machine prompts the operator to confirm, cancel or delete the last Command by means of a “**check interface**”. This is a safety strategy to further enforce the filtering of wrong behaviors of the system, an idea that was also adopted in [76]. The check interface behavior is shown in Figure 27: the operator moves from the Home State to another state (green block of Figure 27) and selects an operative Command from the list of available operative Commands of the State. The check interface intervenes, asking the operator if the Command received is the correct one or not. If the operator confirms the Command (using the “CONFIRM” gesture), the portion of the program linked to that operative Command is executed. If the operator deletes the last Command (using the “DELETE” gesture), the check interface deletes it and asks the user to make a new selection from the list of operative Commands of the State. If the operator cancels the Command (by performing the “EXIT” gesture), the check interface forces the State Machine to move from the current State to the previous State. If the current state is the Home State, this closes the communication and the State Machine moves to the Ready State.

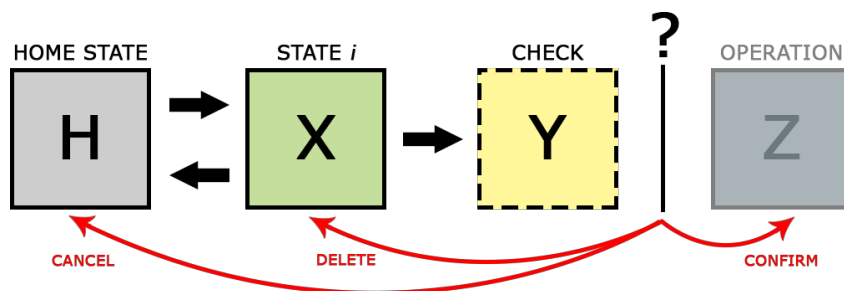


Figure 27. Structure of the check interface. By performing the corresponding Command, the user can confirm, delete, or cancel the last Command given.



### 4.2.2 Gesture Recognition node

This node implements the R-FCN Object Detector model detailed in Section 3.3.5 used to interpret the gestures according to the proposed gestures dictionary. The detector takes as input RGB frames of 960x540 px resolution acquired from the Kinect v2 camera, and outputs a set of predictions which comprehend, for each predicted gesture, (i) the class name, (ii) the confidence score and (iii) the bounding box coordinates. To improve robustness, the following filtering procedure is carried out on the predictions:

- Single-hand gestures are checked; only valid pairs of gestures are retained, according to the request *request\_type*, and are classified as Commands. It is worth noting that left and right variants are predicted incorrectly very rarely, and even when this happens, the procedure does not accept a Command composed of gestures of the same hand. Hence, the label name used to discriminate left and right gestures is used to compose the Command: this is an improvement compared to [88], where discriminating the left from the right hand is accomplished by calculating the barycenter positions of the boxes, at the expense of an increased processing time;
- Commands must be recognized for a certain number of consecutive frames (*chain value*) to prevent the system from sending erroneous interpreted Commands due to sudden environmental disturbances and/or small or incorrect movements of the operator. It has been experimentally found that setting the chain value to 7 is a good tradeoff between robustness and speed; however, this value can be reduced up to 2 or 3 in case of controlled set ups and expert users.

After this filtering procedure, the valid Command is written in the *command* field of the response, which is then published in the `/command_response` topic. To provide the user with a visual feedback, the camera feed is shown on screen. The detected bounding boxes are plotted on each frame with their corresponding label name, as well as suitable debug information such as the inference time recorded frame per frame, the latest *request\_number* received, and the latest Command value published (Figure 28).

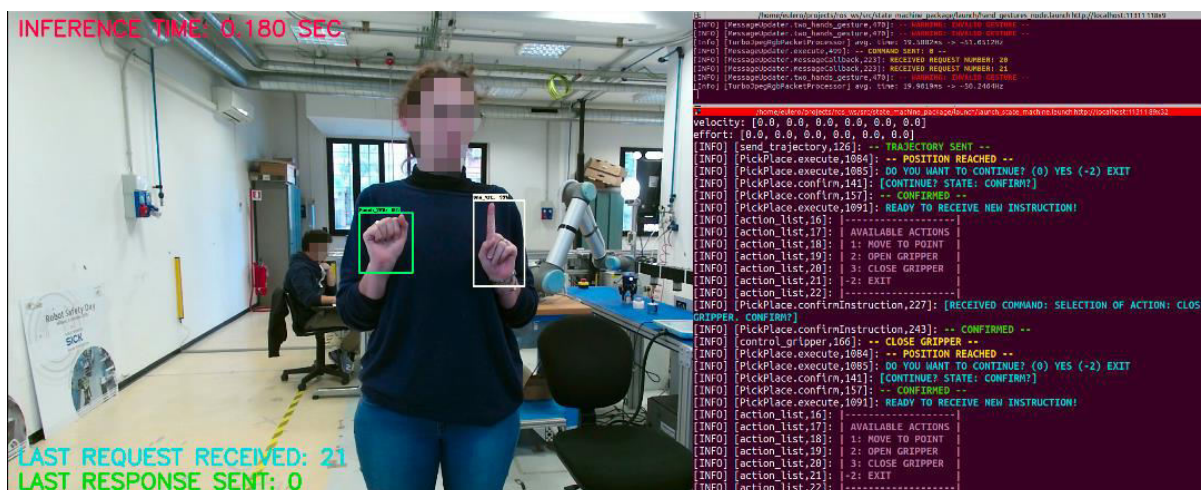


Figure 28. Screenshot of the MEGURU system in use. On the left, the camera feed with the detected boxes drawn on the hands, the inference time, the last request number received, and the last Command sent visualized. On the right, the state machine logs with different colors, representing different system states and actions.

### 4.3 Experimental evaluation

To evaluate the proposed system, two different experiments have been carried out. The first one was aimed at evaluating the user experience of people of different age, sex, and professional background, while the second one was aimed at comparing the teach pendant programming method and MEGURU.

#### 4.3.1 Evaluating the user experience

This experiment was conducted during the Meet me Tonight 2019 event held in Brescia. The robot used was a **Rethink Robotics Sawyer** one arm Collaborative Robot without gripper to avoid unsafe interactions. A simplified version of MEGURU's State Machine called "demo State Machine" was adopted, resembling the layout of the original State Machine presented in Section 4.2.1 (Figure 29). The demo functionalities were:

1. **Hello Trajectory:** upon entering the state, a pre-defined trajectory is sent to the robot to make it wave at the user three times (purple block in Figure 29);
2. **Move Sawyer:** users can physically move the robot around exploiting its manual guidance feature by pressing the corresponding buttons positioned on the end-effector and build a custom trajectory. To save a Point, users must perform the "CONFIRM" gesture to the camera. Each Point is stored in the corresponding Points file and, when the user performs

- the “OPEN FILE” gesture, the robot moves to each Point, automatically interpolating the resulting trajectory. An Operation file corresponding to the obtained trajectory is saved before exiting the state (blue block in Figure 29);
3. **Launch Trajectory:** upon entering the state, the file paths of all the Operation files existing in the corresponding directory of the package are loaded into memory and are presented to the user as an ordered list of file names. The user can choose which Operation file to load by performing the corresponding numerical gesture to the camera. When the Operation file is loaded, the robot moves to each Point of the saved trajectory (orange block in Figure 29);
  4. **Handshake Trajectory:** upon entering this state, a pre-defined trajectory is sent to the robot to make it perform a handshake-like movement (purple block in Figure 29).

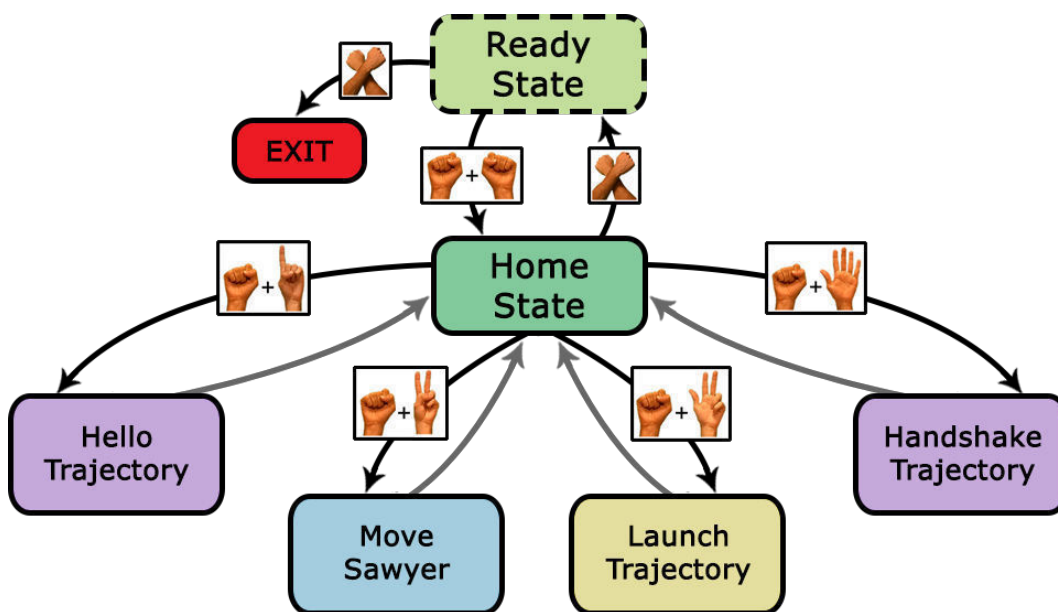


Figure 29. Layout of the demo State Machine. To move to a specific state from the Home State the corresponding numerical gesture is used. After the state execution is complete, the State Machine automatically moves back to the Home State (gray arrows).

At the end of each demo, users were requested to fill in a survey. They had to rate the effort required to learn gesture language on a scale from *Extremely Easy* to *Extremely Hard*; in addition, they had to rate the communication language intuitiveness on a scale from *1 star* to *5 stars*. A total of 28 Subjects have been interviewed, equally divided between males and females, and subdivided in two age groups of equal numerosity (age group 14-30 and 30-70,

14 Subjects per group). In Table 10 and Table 11 the contingency maps are reported, showing the number of voters for each value. Figure 30 and Figure 31 present the results graphically: the bars show the number of votes given by the participants expressed as a percentage versus the proposed rates (left scale), while the dot graphs represent the mean age of the participants who expressed the same vote (right scale).

Table 10. Contingency table about the effort required by the Subjects to learn the gesture language.

Age group	Extr. Easy	Easy	Neutral	Hard	Extr. Hard
14 - 30	4	9	1	0	0
30 - 70	3	7	4	0	0

Table 11. Contingency table about the gesture intuitiveness rating.

Age group	1 star	2 stars	3 stars	4 stars	5 stars
14 - 30	1	0	4	6	3
30 - 70	1	1	5	3	4

As highlighted by the surveys, people from age group 14-30 were generally satisfied with the demo and approached it with enthusiasm, also demonstrating a low effort to properly learn the gesture language. In contrast, people belonging to age group 30-70 experienced more difficulties using the system, as shown by their response mean time.

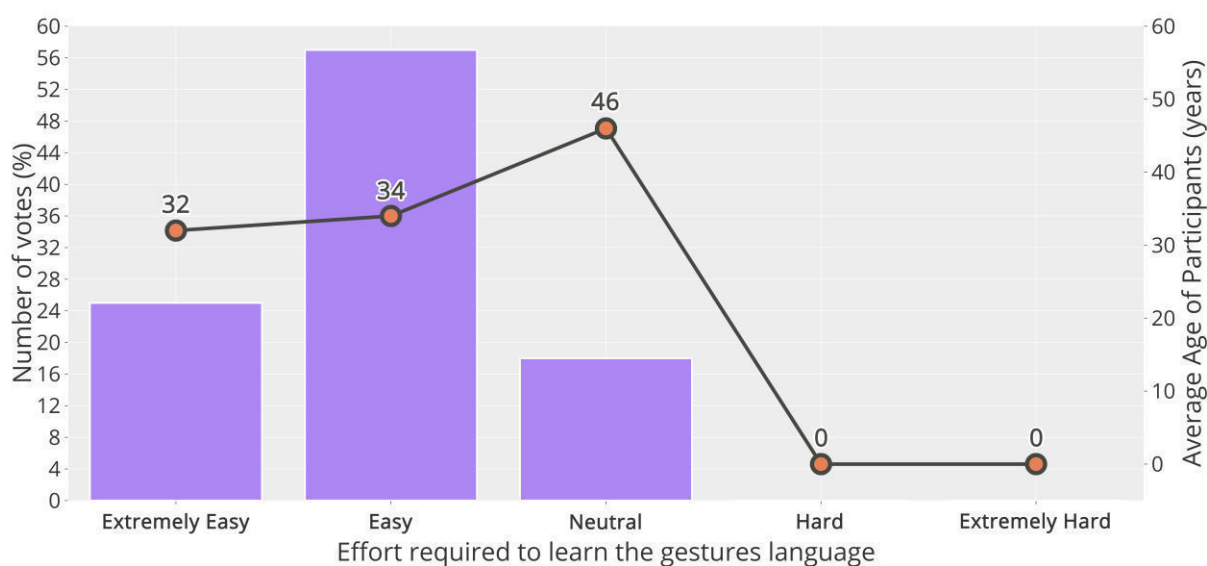


Figure 30. Graph that represents the effort required for each participant to learn the gestures language.

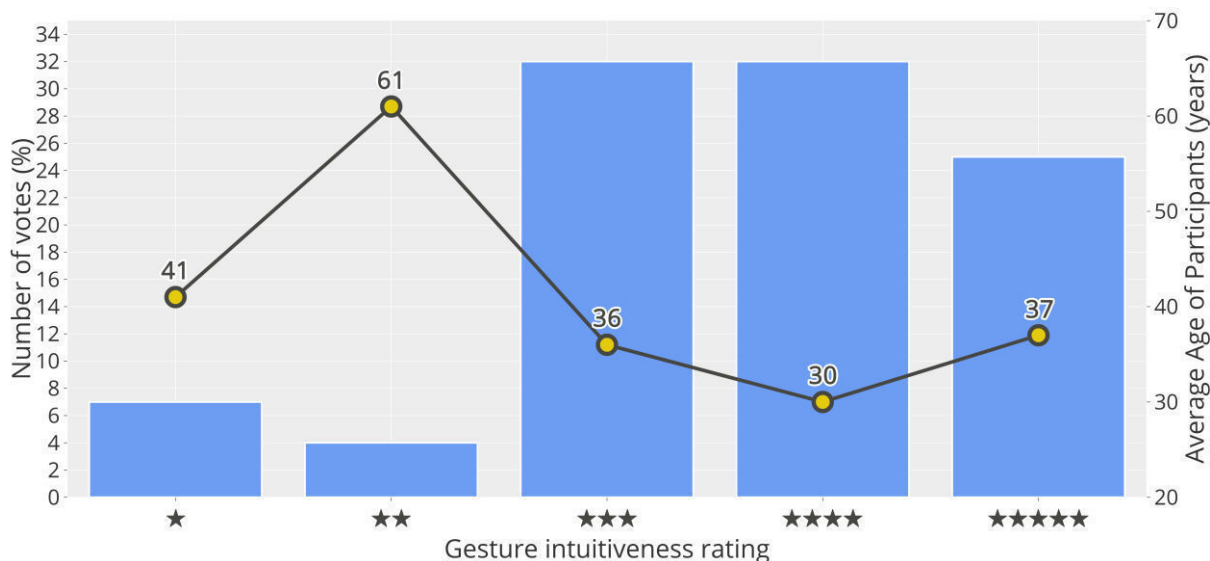


Figure 31. Graph that represents how much the participants felt the gesture language to be intuitive.

The response time of a Subject is calculated considering (i) the time elapsing between a Command request sent by the State Machine Node to the Gesture Recognition Node and (ii) the corresponding Command response sent by the Gesture Recognition Node to the State Machine Node (Section 4.2.1). For each participant, the response times have been calculated, excluding the ones corresponding to the transition from the Ready State to the Home State and the transitions from the Home State to the EXIT. Then, the mean is computed. Figure 32 represents the response mean time of participants who used the “Hello Trajectory” or the “Handshake Trajectory” functionality (the two purple blocks in Figure 29), as they were similar and were the most used by the participants. If a participant used both functionalities, only the lowest response mean time achieved is considered. The pink dashed line in the figure represents the reference time achieved by an expert user, corresponding to 3.40 s.

Wrong gesture recognitions are considered in the calculation of the response time in two ways: when a wrong prediction occurs, the system (i) does not send a response, because the recognized gesture is not an acceptable gesture for the specific state or (ii) sends a response, but the user corrects it exploiting the check interface (Section 4.2.1). As a result, the response mean time is a parameter useful to measure (i) the user ability to use the system (a more skilled user performs the gestures faster and with less faults) and (ii) the system difficulty to recognize the user hands properly.

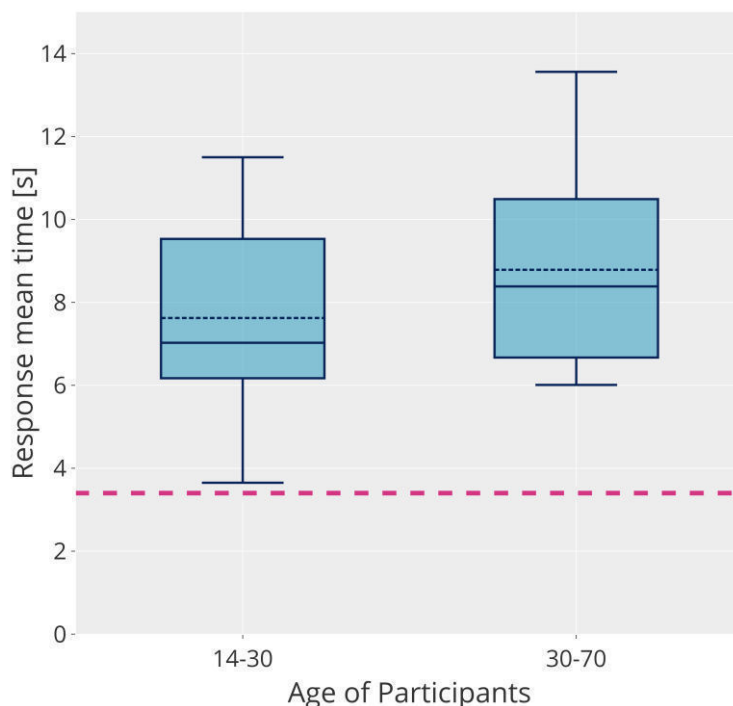


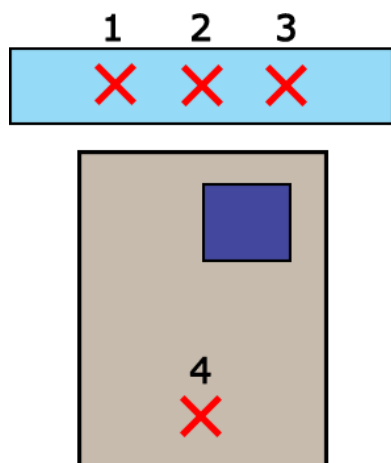
Figure 32. Response mean time obtained by each participant. The pink dashed line represents the response mean time achieved by an expert user, equal to 3.4 s. The solid blue line represents the median of the boxplot, while the dashed blue line represents the mean.

Considering these results, it is made evident that young people performed better, adapting fast to the system interface and language. Subjects of age greater than 40 demonstrated to have less patience, in fact, when performing the gestures, they usually did not stand still waiting for the system response, violating the chain value filtering parameter (Section 4.2.1). This behavior resulted in further increasing their response time. These users expressed the opinion that the reaction times of MEGURU should have been decreased.

#### 4.3.2 Confronting the teach pendant and MEGURU

The second experiment was carried out in collaboration with the **STIIMA** group of the National Research Center of Milan (CNR). The experiment simulated a manufacturing set-up aimed at assembling a moka coffee maker, using a robotic manipulator and a human operator in a MCW. The robot manipulator used for the tests was a **Universal Robot UR10** equipped with a **ROBOTIQ 2F-85 Adaptive Gripper**. The task was to assemble the moka coffee maker (i) by programming the robot using the teach pendant and (ii) by programming the robot using MEGURU. The set-up of the experiments is shown in Figure 33: the heating vessel, the filtering

funnel and the coffee collector of the moka coffee maker are positioned in positions 1-3 respectively, while position 4 corresponds to the area where the parts must be moved and assembled.



(a)



(b)

Figure 33. (a) Set-up of the experiments in top-view. Positions from 1 to 3 are for the moka coffee maker parts; position 4 denotes where the robotic manipulator (represented as a blue block) must move the objects and perform the collaborative assembling with the operator. (b) Image of the experimental set-up. The moka coffee maker parts are shown here: in position 1 is the heating vessel, in which the filtering funnel (position 2) must be placed. The coffee collector in position 3 must be screwed on top by hand.

The experiment has been conducted using the following schedule:

1. **Teach Pendant test (TP):**

- a. *Coarse Programming:* build the robot program using the teach pendant and use the manual guidance mode or the teach pendant to move the robot;
- b. *Fine Programming:* check if the robot works as intended and adjust the program for optimization.

2. **MEGURU Collaborative test (MC):**

- a. *Points Acquisition:* move the robot in the proper cartesian positions either by using the manual guidance of the robot or the ROS interface to move the joints. Save the joint coordinates of each Cartesian position in the Points file;
- b. *Operation Building:* perform the assembling task using the SOPs Building State and the Points file as input.



Participants were required to carry out the assembling task avoiding input constraints such as the number of points to collect or the sequence to follow to assemble the components, to let them free of approaching the tests as much naturally as possible. The set of points chosen by each participant to complete the robot trajectory in the TP test was the same one used to obtain the points elements required by the MEGURU interface, following the process described in Section 4.2.1. Subjects 1, 2, 3, 4 and 6 performed the TP test first and Subjects 5 and 7 performed the MC test first. The number of points chosen by each Subject is shown in Table 12.

Table 12. Results of the second experiment. The lower time for each Subject is highlighted in bold.

Subjects	Expert user of which modality?		# points	TP total time	MC total time
	TP	MC			
Subject 1	X	-	12	15 min	<b>11 min</b>
Subject 2	-	X	11	<b>12 min</b>	16 min
Subject 3	-	-	12	12 min	<b>9 min</b>
Subject 4	-	-	19	21 min	<b>19 min</b>
Subject 5	X	-	8	<b>12 min</b>	18 min
Subject 6	-	X	10	14 min	<b>12 min</b>
Subject 7	-	-	8	16 min	16 min

Figure 34 shows the time required to complete both the TP and the MC tests: pink and orange bars represent the time required to complete the Coarse and the Fine programming phases of the TP test respectively; green and blue bars refer to the time required to carry out the Point Acquisition and the Operation Building steps of the MC Test respectively; the results are reported in pairs for each Subject. From this figure it can be observed that four participants (1, 3, 4 and 6) took less time to carry out the MC test than the TP test. Among these, three Subjects were non-expert users of the MC modality (1, 3, 4): this suggests that the MEGURU interface is easy enough to be used properly even by non-expert users. Subject 4, which was a non-expert user of both methods, is the one with the highest values: this can be related to the high number of used points (19 points, as seen in Table 12). It is worth noting that the time required to complete the MC test depends on the wrong predictions of the gestures.



Specifically, the high values of the MC test achieved by Subject 2, 5 and 7 are related to the recognition difficulties of the hand-gesture recognition model.

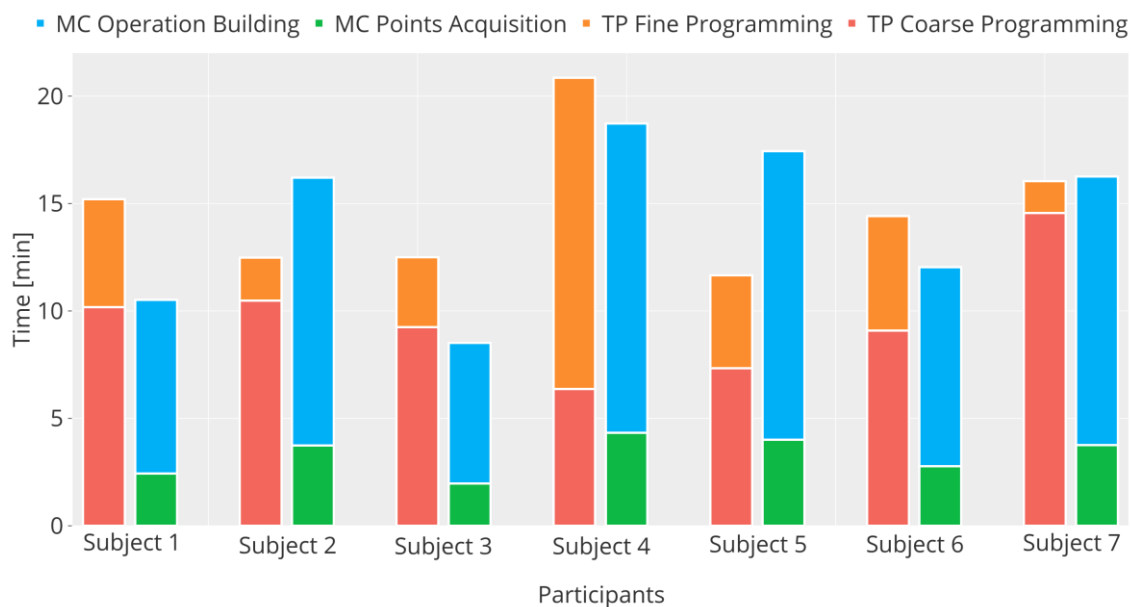


Figure 34. Time required by each participant to complete the assembling task. The pink bar represents the TP Coarse Programming time, the orange bar represents the TP Fine Programming time, the green bar represents the MC Points Acquisition time and the blue bar represents the MC Operation Building time.

The map of erroneous predictions observed in the experiment is shown in Figure 35: rows refer to the Subjects and columns correspond to the gestures in which errors have been observed. Prediction errors are related to three types of situations:

- **Case 1:** Gestures correctly recognized by the system but corresponding to wrong Actions (i. e. user command logic error): in this case the operator had to perform the “DELETE” gesture and reset the correct Action (Subjects 1 and 3).
- **Case 2:** Gestures erroneously recognized by the system, as a consequence of the operator ability to perform the gestures. In this case, the level of familiarity with the use of the gesture-based communication system determined the final time: Subjects 1, 2, 3 and 6 responded promptly to the instructions of the visual feedback provided by MEGURU, and were fast enough to adapt the pose and the orientation of their hands to ensure that the right Commands were sent to the State Machine node. Subjects 4, 5 and 7, instead, were slower in the interaction, thus having to perform the “DELETE” gesture and to prompt their

gesture again. In both cases, a time delay was introduced in the communication between the ROS nodes.

- Case 3:** Gestures erroneously recognized due to the weaknesses of the recognition chain, specifically in the presence of single-hand gesture commands, where the hand dimension and the skin color with respect to the background color and to the light conditions determined the visibility of the features in the frames. Very slender and pale hands were almost invisible for operators wearing light color jerseys, while large, darker hands were robustly detected thanks to the increased contrast between the hand color and the background. In fact, Subject 2 (a female), had small, pale hands and, although very experienced in making gestures and in using MEGURU, reported a high number of erroneous predictions, especially whenever the single-hand gesture in Figure 15 (h) (*Eight\_VFR/L*) was prompted. On the other hand, Subject 3 hands were large enough to be easily detected by the system, and, although he was not an expert user, he showed just three wrong predictions.

Referring to Figure 35, the highest values of incorrect predictions (5 and 10) occurred in correspondence of single-hand gestures *Four\_VFR/L*, *Eight\_VFR/L* and *Span\_VFR/L* (Figure 15 (d), (h) and (k) respectively). This is related to the fact that (i) gesture *Four\_VFR/L* is easily confused with gestures *Five\_VFR/L* and *Six\_VFR/L* and (ii) gestures *Eight\_VFR/L* and *Span\_VFR/L* require a fine finger mobility that some Subjects lacked, thus leading to incorrect predictions.

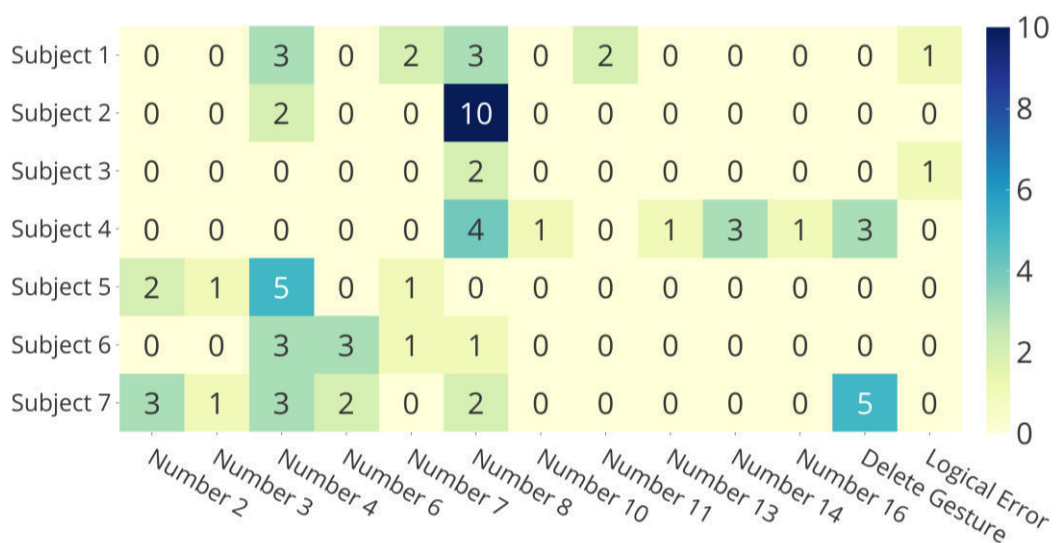


Figure 35. Errors made by each Subject. The highest the number, the highest the time lost to correct them.

After each test, the participants were required to fill in a survey of four questions and to add their comments. The results are shown in Figure 36, where for each question a pie plot reports the percentage of votes for each rating (possible values: *Extremely Easy*, *Easy*, *Neutral*, *Hard*, *Extremely Hard*).

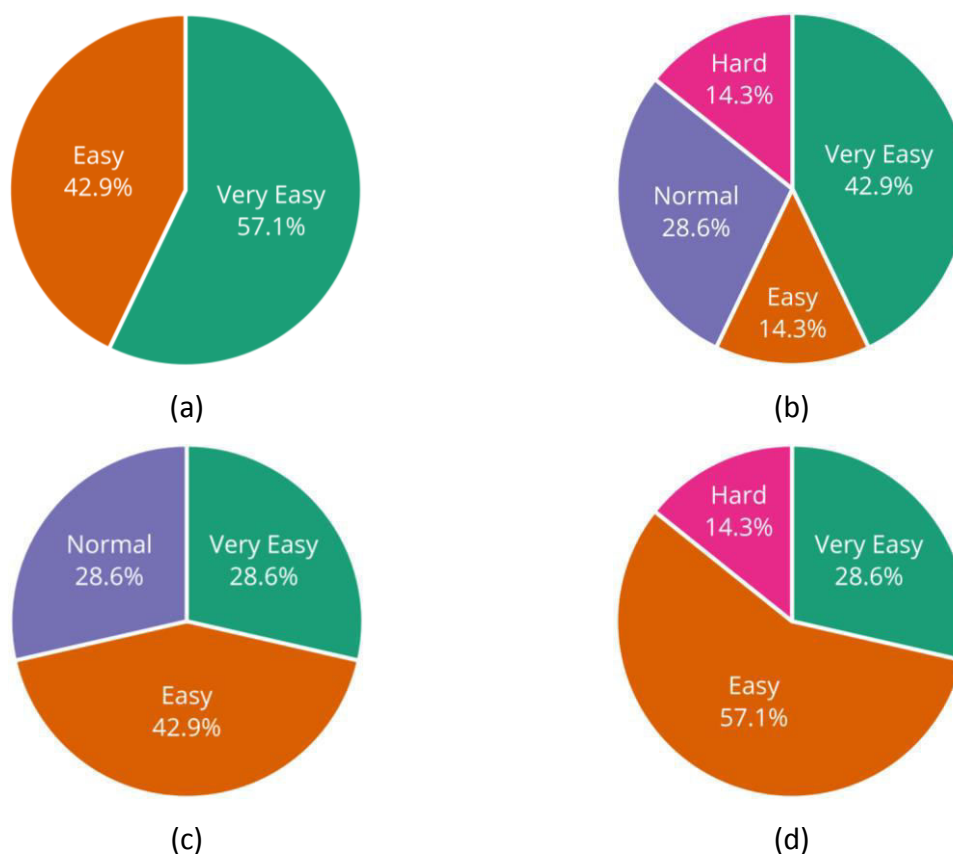


Figure 36. Results of the questionnaire. (a) How easy do you think it is to learn the TP programming modality? (b) How easy do you think it is to operate using the TP programming method? (c) How easy do you think it is to learn the MC programming modality? (d) How easy do you think it is to operate using the MC programming modality?

The answers and the comments collected for each participant highlight that for the TP modality the programming part was straightforward to be learnt even by inexperienced users, but that the real difficulty was that the teach pendant itself was quite heavy and uncomfortable to be carried around, often forcing the user to place it on a table to use it properly. To use the manual guidance modality, which is the easiest way to obtain a robot position, the operator had to keep pushed a button on the back of the teach pendant, thus he/she had to move the robot using one hand. This resulted in a very uncomfortable experience for some users, especially those who were not strong enough to easily move the robot around (e. g. Subject

2). A more comfortable and precise way to take the robot positions was to move the joints using the teach pendant interface, but this required a deeper knowledge of the robot reference system in order to move it without faults and took more time to reach the desired position. In contrast, learning the MEGURU programming method was more difficult for inexperienced users, since they had to learn the gesture dictionary and how to use the system, but they quickly got used to that. In fact, most users reported that using the MEGURU interface was very simple and straightforward after only few Commands were practiced, in accordance with the results obtained from the first experiment (Section 4.3.1). The only exception was Subject 7, which reported a strong difficulty in the gesture execution due to his finger mobility issues. In addition, all participants, except for Subject 2, reported pain at both their shoulders and arms, since the body posture required to perform the gestures was quite uncomfortable and they were not used to it.

## Chapter 5. The “Hands Free” module

Despite advances in robotic perception are increasing autonomous capabilities, human intelligence is still considered a necessity in unstructured or not predictable environments. Typical scenarios concern the detection of random shape objects, manipulation, or custom robot motion; hence, human and robots must achieve mutual **Human-Robot Interaction** (HRI) [115]. HRI can be physical (pHRI) or not, depending on the assigned task. For example, when the robot works in a dangerous environment or must handle hazardous materials, pHRI is not recommended. In such cases, it may be necessary to **teleoperate** the robot, allowing the user to control the robot remotely [116]. A plenty of human-machine interfaces for teleoperation have been developed considering a mechanical interface, including exoskeletons [117] or gloves [118]. Such systems are particularly helpful to achieve bilateral teleoperation [119], where they can transmit or reflect back to the user reaction forces from the task being performed. In this case, a high perception with complete haptic feedback [120] [121] is achieved. Other types of controllers may be a mouse and keyboard interface, switchboxes, touchscreen displays and joysticks. The latter type in particular is usually a better type of control device than others are, because the operators identify with the task in a more immersive way [122]. On the other side, if bilateral interaction is not required, a vision-based interface is preferable because it does not require physical contact with external devices such as cables, connectors, and objects outside of the user working area. This grant a more natural and intuitive interaction, which is reflected on the task performance: as shown in [123], the accuracy of object gripping tasks is improved by mean of a contactless vision-based robot teleoperation method.

Since MEGURU’s structure presented in Section 4.2 is modular and can be easily expanded adding standalone functionalities as blocks, the “Hands-Free” module has been developed to allow users to teleoperate the robot easily in 2D using their hands and a simple RGB camera, hence granting greater operating distance at a reduced price compared to the Leap Motion controller [124]. The system is based on the idea that the user moves its hands in a certain space, called *user workspace*, and that its movements are replicated in a second space where

only the robot operates, called *robot workspace*. According to how the two workspaces are defined, they may be (i) of different in size, (ii) very distant from each other or (iii) they may even be the same workspace, in total accordance with the idea of Meta-Collaborative workstations presented in Section 4.1, where the human and the robot workspaces may be overlapping or not while still achieving human-robot collaboration. To detect the human hands, I used the open-source OpenPose software [125] [109] based on the Deep Learning framework Caffe [126]. This software elaborates RGB frames and extracts the human body's skeletonization; in this case, only the hands module is used since no other body parts are present in the frames. This choice is motivated by the fact that the precise joint positioning of the fingers in the image is the fundamental requirement to calculate the position of the corresponding robot waypoint. Using the same approach described in Chapter 4 would not be enough because from the bounding box surrounding the hand only the centroid of it could be extracted, not the finger joints.

## 5.1 Set-up calibration and mapping

In Figure 37 the proposed set-up adopted for the experimental evaluation of the “Hands-Free” module is represented. Albeit different set-ups may be adopted (i. e. with overlapping workspaces or with different orientations of the planes), in this case the user workspace is the horizontal green plane defined by reference system  $H$  and the robot workspace is the vertical purple plane defined by reference system  $W$ .

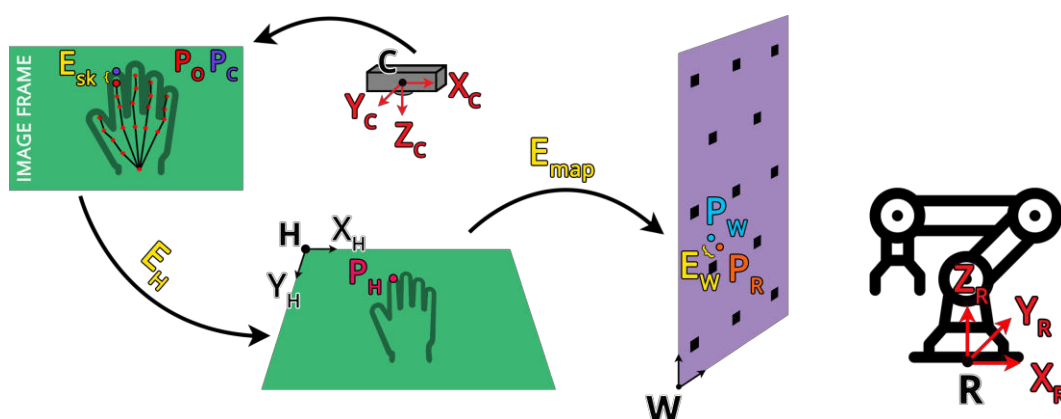


Figure 37. Scheme of the calibration steps. The user workspace is represented by the green plane while the robot workspace is represented by the purple plane. In this set-up, the two have different orientations: one is horizontal while the other is vertical. According to the set-up definition, a suitable calibration procedure of both workspaces must be performed to correctly use the “Hands-Free” module.

Cartesian points in the user workspace that (i) may be reached by the user hand and (ii) that are correctly viewed by the camera, correspond to precise robot end-effector Cartesian points in the robot workspace. Hence, to obtain the mapping between the hand positions and the robot end-effector positions, it is necessary to perform a set of **calibration** procedures.

### 5.1.1 User workspace calibration

In the user workspace an RGB camera is used to recognize the hand skeleton in real-time. Therefore, it is necessary to properly calibrate the camera relative to the user-defined reference system. This procedure is called **camera calibration**, and it may be easily realized following standard procedures such as the one detailed in [127].

The projection mapping for a generic point  $\mathbf{P}_C = (u, v)$  in the camera image plane with reference frame  $C$  to its corresponding real-world coordinate point  $\mathbf{P}_H = (x, y, z)$  in reference frame  $H$ , is defined by the following formula where homogeneous coordinates are used:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} [\mathbf{R}|\mathbf{t}] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2)$$

However, since we are looking for the position of point  $\mathbf{P}_H$  in frame  $H$  by back-projecting a 2D point to 3D, it is necessary to invert equation (2):

$$\mathbf{P}_H = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{R}^{-1} \left( \mathbf{K}^{-1} s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} - \mathbf{t} \right) \quad (3)$$

The parameters in the equations above are:

- $s \in \mathbb{R}$  is the scalar representing the scale factor of the image;
- $\mathbf{K} \in \mathbb{R}^{3 \times 3}$  is the camera matrix containing the intrinsic parameters of the camera, such as focal length and optical center obtained through the calibration procedure;
- $[\mathbf{R}|\mathbf{t}] \in \mathbb{R}^{3 \times 4}$  is the rigid transformation matrix containing the extrinsic parameters for rotation ( $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ ) and translation ( $\mathbf{t} \in \mathbb{R}^3$ ) of the camera reference frame  $C$  relative to the calibration master reference frame  $H$ .

To obtain matrix  $\mathbf{K}$  it is necessary to perform a calibration procedure. A well-performed calibration procedure allows to obtain a satisfactory estimation of the camera parameters. To correctly map image points to the corresponding real-world coordinates, the rigid transformation matrix must be estimated with respect to the user-defined reference system of the calibration master. Thus, if reference frame  $H$  changes or the camera frame  $C$  moves, it is necessary to estimate the correct rigid transformation matrix again. This procedure has been automatized by “Hands-Free” software: a calibration script calculates matrix  $\mathbf{K}$  using a set of calibration images acquired by the user, then it takes a new frame from the actual set-up to estimate the position of reference system  $H$ .

By using the abovementioned formulas, after each frame is processed by the OpenPose network to extract the hand keypoints, it is possible to obtain the real-world coordinates corresponding to them during the execution of the software (image frame in Figure 37).

### 5.1.2 Robot workspace calibration

The robot workspace refers to the space in which the robot moves (reference system  $W$  of Figure 37) with respect to the user workspace (reference system  $H$  of Figure 37). In this case, the user hand real-world position in reference system  $H$  is mapped to the new reference system  $W$ . The mapping between reference system  $H$  and reference system  $W$  is obtained easily if the two workspaces have the same dimension (matrix  $[\mathbf{R}|\mathbf{t}]$  is the identity matrix) or if one workspace is a scaled version of the other one (matrix  $[\mathbf{R}|\mathbf{t}]$  is the identity matrix multiplied by the scale factor).

To correctly move the robot in a cartesian position of reference system  $W$ , it is necessary to perform a calibration between reference system  $W$  and the robot reference system  $R$ . This procedure has been carried out experimentally by moving the robot (using its manual guidance mode) in different Cartesian positions of reference system  $W$ . The robot correct positioning on top of each calibration position has been assured by using a 3D-printed centering tool purposely developed for the **Rethink Robotics Sawyer** robot adopted (Figure 38). The tool must be centered manually on each calibration marker and secured in place, and



then the robot end-effector can be moved on it and carefully positioned inside the purposely made circular cavity of the tool. When the positioning is complete, the robot coordinates (both in the Cartesian space and in the Joints space) corresponding to that particular marker (of which the positioning is known with respect to reference system  $W$ ) can be extracted using ROS or the robot proprietary software.

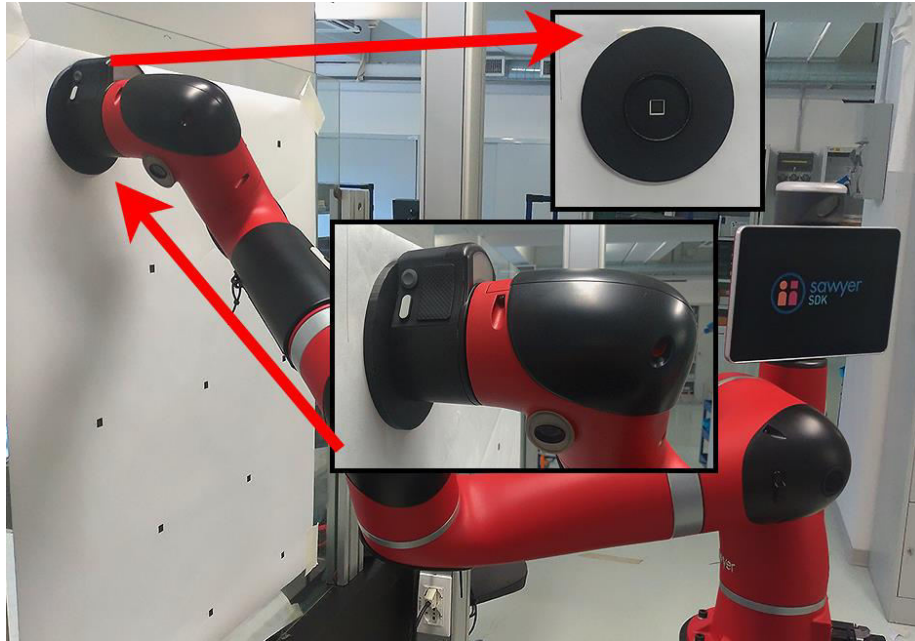


Figure 38. Example of the robot calibration procedure. The calibration pose tool is placed in correspondence of a calibration marker, and then the robot end-effector is carefully placed inside the tool. The calibration pose tool cavity has been purposely made to fit the robot end-effector.

When a satisfactory number of calibration positions has been acquired, it is possible to estimate the rigid transformation matrix between workspaces  $W$  and  $R$ . It is worth noting that the proposed system currently involves only a planar motion; thus, the mapping procedure does not consider the  $z$ -axis.

Referring to the calibration example in Figure 39, the plane position of a point  $\mathbf{P}_1 \in \mathbb{R}^2$  is calculated with respect to both frame  $W$  ( $\mathbf{P}_{1,W}$ ) and frame  $R$  ( $\mathbf{P}_{1,R}$ ). The distance between the two reference frames is  $\mathbf{T}_R \in \mathbb{R}^2$ , hence:

$$\mathbf{P}_{1,R} = \mathbf{P}_{0,W} + \mathbf{T}_R \quad (4)$$

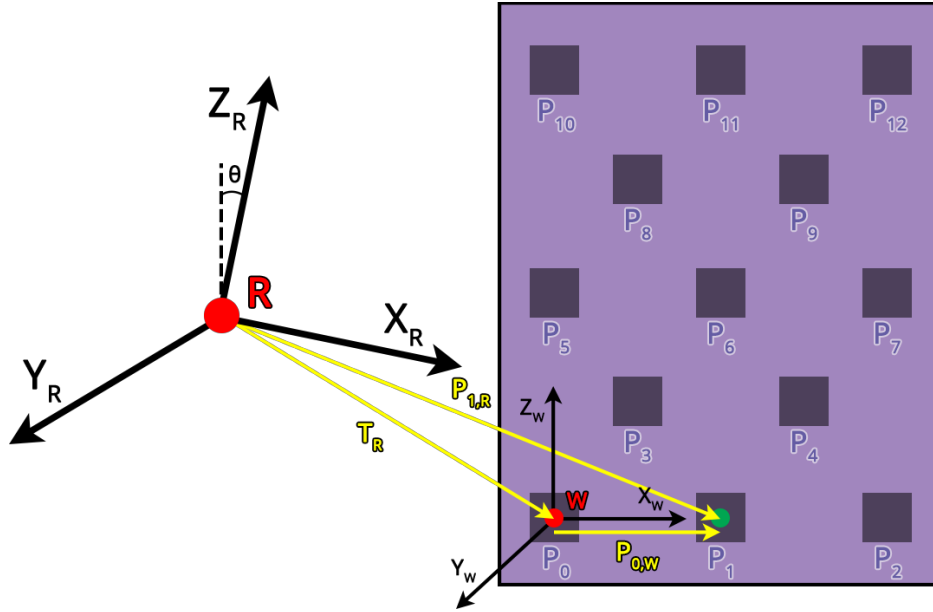


Figure 39. Calibration master used to calibrate the second user-defined reference system  $W$  with the robot reference frame  $R$ . The figure illustrates the position of point  $P_1$  in both reference frames. To properly calibrate the system, the position of each point is required, both for frame  $W$  and  $R$ . In the procedure, 13 calibration points have been acquired.

By using homogeneous coordinates, it is possible to rewrite the previous equation as matrix products:

$$\mathbf{P}_{1,R} = \mathbf{M}_R^W \mathbf{P}_{0,W} \quad (5)$$

Where  $\mathbf{M}_R^W \in \mathbb{R}^{3 \times 3}$  is the rigid transformation matrix between the two reference systems defined as:

$$\mathbf{M}_R^W = \begin{bmatrix} \cos \theta & \sin \theta & x_{T_R} \\ -\sin \theta & \cos \theta & y_{T_R} \\ 0 & 0 & 1 \end{bmatrix} \quad (6)$$

Hence, by outlining equations from (5), we obtain:

$$\begin{cases} x_{P_{0,W}} = x_{P_{1,R}} \cos \theta + y_{P_{1,R}} \sin \theta + x_{T_R} \\ y_{P_{0,W}} = -x_{P_{1,R}} \sin \theta + y_{P_{1,R}} \cos \theta + y_{T_R} \end{cases} \quad (7)$$

The aim of the calibration procedure is to calculate  $\mathbf{M}_R^W$  in order to find the correct position and orientation of the reference frame  $W$  with respect to frame  $R$ . However, considering only one calibration position point  $P_1$ , the system in (7) results underdetermined, hence, a minimum of  $n > 2$  calibration points is required to solve the system. To minimize the

calibration error, I considered  $n = 13$  points for the calibration; thus, the system in (7) becomes an overdetermined system  $\mathbf{Ax} = \mathbf{b}$  that has been solved using the least square method:

$$\mathbf{A} = \begin{bmatrix} x_{P_{0,R}} & y_{P_{0,R}} & 1 & 0 \\ -y_{P_{0,R}} & x_{P_{0,R}} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{P_{n-1,R}} & y_{P_{n-1,R}} & 1 & 0 \\ -y_{P_{n-1,R}} & x_{P_{n-1,R}} & 0 & 1 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} \cos \theta \\ \sin \theta \\ x_{T_R} \\ y_{T_R} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} x_{P_{0,W}} \\ y_{P_{0,W}} \\ \vdots \\ x_{P_{n-1,W}} \\ y_{P_{n-1,W}} \end{bmatrix} \quad (8)$$

The rigid transformation matrix  $\mathbf{M}_R^W$  used to identify the reference frame  $W$  from  $R$  is defined by the components of  $\mathbf{x}$ . Considering the overall scheme in Figure 37, the generic point  $\mathbf{P} \in \mathbb{R}$  in the robot reference frame  $R$  with respect to the camera frame  $C$  is calculated as follows:

$$\mathbf{P}_R = \mathbf{M}_R^W \mathbf{P}_W \quad (9)$$

The same point in the robot workspace  $W$  is:

$$\mathbf{P}_W = K_s \mathbf{P}_H \quad (10)$$

Where  $K_s \in \mathbb{R}$  is the scaling factor between the robot and the user workspaces and  $\mathbf{P}_H$  is defined in (3). Finally, considering equations (9), (10) and equation (3), the resulting point  $\mathbf{P}_R$  in the robot reference frame using the camera coordinates is calculated as:

$$\mathbf{P}_R = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = K_s \mathbf{M}_R^W \mathbf{R}^{-1} \left( \mathbf{K}^{-1} s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} - \mathbf{t} \right) \quad (11)$$

The space coordinates  $(u, v)$  are the output of the hand-gesture recognition algorithm, while Cartesian coordinates  $(x, y, z)$  are the calculated position of the robot end-effector.

## 5.2 Hand-gesture recognition of “Hands-Free”

The proposed teleoperation method is based on the recognition of the user hands' skeleton. Each frame acquired by the RGB camera (in this case a Kinect v2 camera) is processed by the software, which leverages the OpenPose hand skeleton recognition network to predict the hand skeleton, following the details of [125]. The gesture recognition procedure is based on the position of the reference keypoint (red keypoint 0 in Figure 40) and on the position of the four knuckles keypoints (blue keypoints 5, 9, 13, 17 in Figure 40). I defined two gestures used

to carry out basic teleoperation tasks: the **open hand gesture** (Figure 40, top-right) and the **index gesture** (Figure 40, bottom-right).

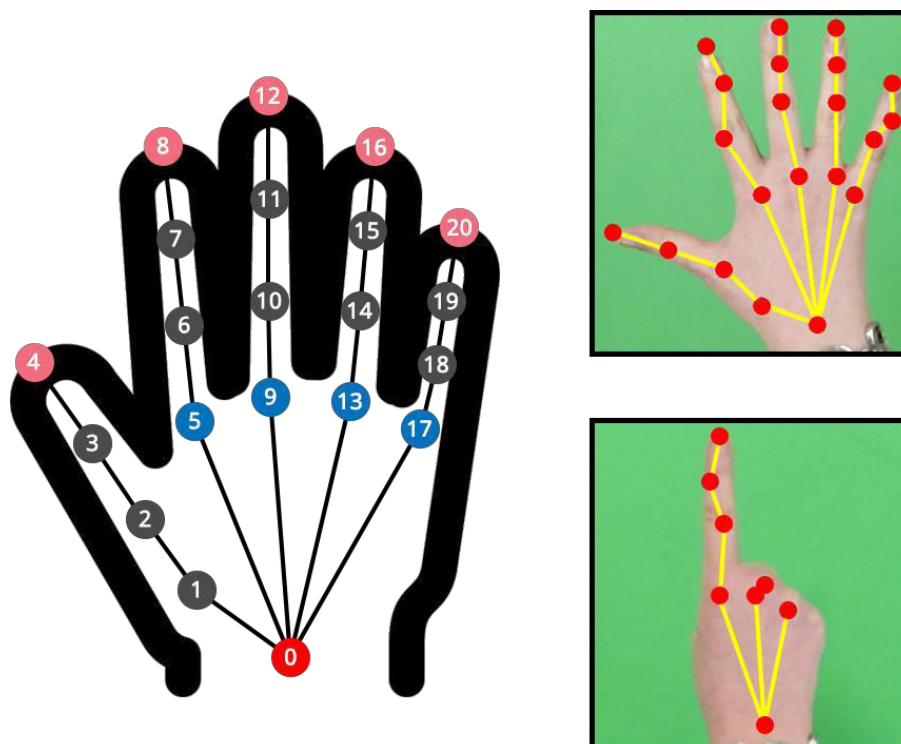


Figure 40. Scheme of the hand skeleton predicted by OpenPose neural network. The red keypoint is the reference keypoint, the blue keypoints are the knuckles keypoints and the pink keypoints are the fingertips keypoints. Examples of correctly recognized gestures: open hand gesture (top-right) and index gesture (bottom-right).

### 5.2.1 Gesture recognition procedure

The gesture recognition procedure is based on the recognition of the fingers, which may be opened or closed. The prediction score threshold adopted to determine if a certain keypoint  $k_j$  has been correctly predicted by the OpenPose network or not has been set to 40%.

To robustly recognize if a finger is opened or closed, the following procedure is performed to output a **handmap**, representing which finger  $f_i$  is considered open (value 1) and which finger is considered closed (value 0). For example, if the thumb ( $f_0$ ) and the pinkie ( $f_4$ ) are opened, the corresponding handmap is  $\mathbf{h} = [1 \ 0 \ 0 \ 0 \ 1]$ .

The procedure is iteratively repeated for each finger  $f_i$  as follows:

1. **Finger keypoints saving:** in this step, the procedure checks if all the keypoints of the considered finger  $f_i$  have been correctly predicted by the network and saves their position (in pixels) in vector  $\mathbf{F}_{f_i}$ . If a certain keypoint is absent, the position of  $k_0$  is saved in its place. For example, considering the index finger ( $f_1$ ), the procedure outputs:

$$\mathbf{F}_{f_1} = [k_5 \quad k_6 \quad k_7 \quad k_8]$$

where  $k_5$  represents the knuckle keypoint and  $k_8$  represents the fingertip keypoint.

2. **Relative distances calculation:** for each value of  $\mathbf{F}_{f_i}$  the relative distance from reference keypoint  $k_0$  ( $D_{0,j}$ ) is calculated. In particular, the procedure checks the relative distance of the fingertip: if it is equal to zero, the finger is considered closed as a safety measure. The fingertip relative distance of each finger is saved in vector  $\mathbf{T}$ . In the abovementioned example, the procedure checks the relative distance between  $k_0$  and  $k_8$  ( $D_{0,8}$ ).
3. **Overall finger length calculation:** to check if the finger keypoints are collapsed around the knuckle keypoint, the overall length of the finger  $\mathbf{L}_{f_i}$  is calculated considering the relative distance from  $k_0$  of the fingertip keypoint ( $D_{0,T}$ ) and of the knuckle keypoint ( $D_{0,K}$ ), expressed as a percentage of  $D_{0,T}$ :

$$\mathbf{L}_{f_i} = \frac{D_{0,T} - D_{0,K}}{D_{0,T}}$$

If this distance is less than 10%, the finger keypoints are collapsed around the knuckle keypoint, therefore the finger is considered *closed*.

4. **Index finger superimposing:** sometimes, the absence of the thumb finger generates wrong skeletonizations of the hand; hence, the index gesture may be incorrectly detected. To compensate this effect, the procedure checks which fingertip relative distance is the maximum in vector  $\mathbf{T}$ : if the maximum distance corresponds to the index finger, then the second value of the handmap is changed to 2. For example, a resulting handmap corresponding to the index gesture is  $\mathbf{h} = [1 \quad 2 \quad 0 \quad 0 \quad 1]$ . It is worth noting that with this method it is sufficient to have value 2 as the second element of the handmap to recognize the index gesture, reducing its recognition error.

### 5.2.2 Moving the robot end-effector: positioning procedure and filtering

Considering the calibration procedure detailed in Section 5.1, a certain position of the index finger  $\mathbf{P}_O$  in the image frame  $C$  as calculated by OpenPose corresponds to a real-world index finger position  $\mathbf{P}_H$  in the user workspace  $H$  which, in turn, corresponds to a certain robot end-effector position  $\mathbf{P}_W$  in the robot workspace  $W$ . Hence, to move the robot end-effector in position  $\mathbf{P}_W$  using “Hands-Free”, the users must:

1. place their hand in position  $\mathbf{P}_H$  (corresponding to position  $\mathbf{P}_W$ ), using the real-time visualization of the software as guidance;
2. perform the open hand gesture to allow the coordinate extraction;
3. perform the index gesture by carefully pointing the index finger to position  $\mathbf{P}_H$ .

It is worth noting that, since the hand skeleton is obtained by a neural network that estimates the joints coordinates frame per frame, their position in consecutive frames may vary. Therefore, “Hands-Free” adopts a **filtering procedure**, which, referring to Figure 37, extracts  $N$  different pixel coordinates of  $\mathbf{P}_O = (x_o, y_o)$  from  $N$  consecutive index gestures recognized in consecutive frames. The average coordinates are extracted to reduce positioning errors introduced by the hand skeleton recognition network; hence, the final point  $\overline{\mathbf{P}}_O$  is obtained as:

$$\overline{\mathbf{P}}_O = \left( \frac{1}{N} \sum_{n=1}^N x_o, \frac{1}{N} \sum_{n=1}^N y_o \right) \quad (12)$$

The higher the value of  $N$ , the higher the error reduction, at the cost of a higher delay before  $\overline{\mathbf{P}}_O$  coordinates are extracted. In particular, setting  $N = 7$  has proven to be a good trade-off between robustness and speed of the recognition. After a position  $\overline{\mathbf{P}}_O$  is obtained, the corresponding real-world position  $\mathbf{P}_H$  is calculated using Equation (3). Then, the corresponding robot position  $\mathbf{P}_R$  is calculated using Equation (11) and the robot is moved on it using the ROS interface. To move the robot again, the positioning procedure must be repeated from the start.

### 5.3 Experimental evaluation

A reliable teleoperation system is obtained if the robot moves to the desired position with a low positioning error. In the proposed set-up, the positioning error is obtained as a sum of different errors, as shown in Figure 37. First, when the user points the index finger to a Cartesian point in workspace  $H$ , OpenPose neural network estimates the index position in the image as point  $\mathbf{P}_O = (x_o, y_o)$  in pixel coordinates (corresponding to keypoint 8 in Figure 40) which, according to the index finger filtering procedure detailed in Section 5.2.2 corresponds to  $\overline{\mathbf{P}}_O$  as detailed in Equation (12). In matrix notation:

$$\overline{\mathbf{P}}_O = \begin{bmatrix} \frac{1}{N} \sum_{n=1}^N x_o \\ \frac{1}{N} \sum_{n=1}^N y_o \end{bmatrix} \quad \forall 0 \leq n \leq N \quad (13)$$

Since the hand skeleton is an estimate of the position of the hand in the image, there may be a certain error  $\mathbf{E}_{sk}$  between the estimated skeleton position  $\overline{\mathbf{P}}_O$  and the real hand position  $\mathbf{P}_C$ , as shown in Figure 37. The estimation error  $\mathbf{E}_{sk}$  is obtained as the pixel distance between the real index position ( $\mathbf{P}_C$ ) and the estimated index keypoint position ( $\overline{\mathbf{P}}_O$ ) as:

$$\mathbf{P}_C = \overline{\mathbf{P}}_O + \mathbf{E}_{sk} = \begin{bmatrix} \frac{1}{N} \sum_{n=1}^N x_o \\ \frac{1}{N} \sum_{n=1}^N y_o \end{bmatrix} + \mathbf{E}_{sk} \quad (14)$$

Thanks to the camera calibration procedure (Section 5.1.1), a point in the acquired image frame  $\mathbf{P}_C$  [px] corresponds to a Cartesian point  $\dot{\mathbf{P}}_C$  [m] rototranslated in workspace  $H$ .  $\dot{\mathbf{P}}_C$  corresponds to the real position  $\mathbf{P}_H$  of the original  $\mathbf{P}_C$  with an estimation error  $\mathbf{E}_H$  which depends on the accuracy of the calibration and on the camera resolution. Thus,  $\mathbf{P}_H$  is defined as follows, where the dot represents the conversion from pixels to meters performed applying Equation (3) (Section 5.1.1):

$$\mathbf{P}_H = \mathbf{P}_C + \mathbf{E}_H = \overline{\mathbf{P}_O} + \mathbf{E}_{sk} + \mathbf{E}_H = \begin{bmatrix} \frac{1}{N} \sum_{n=1}^N x_O \\ \frac{1}{N} \sum_{n=1}^N y_O \end{bmatrix} + \mathbf{E}_{sk} + \mathbf{E}_H \quad (15)$$

The robot operates in workspace  $W$ , therefore we must obtain the position of  $\mathbf{P}_W$ , which corresponds to  $\mathbf{P}_H$  according to the specific mapping between the two workspaces. This mapping introduces error  $\mathbf{E}_{map}$ , hence:

$$\mathbf{P}_W = \mathbf{P}_H + \mathbf{E}_{map} = \begin{bmatrix} \frac{1}{N} \sum_{n=1}^N x_O \\ \frac{1}{N} \sum_{n=1}^N y_O \end{bmatrix} + \mathbf{E}_{sk} + \mathbf{E}_H + \mathbf{E}_{map} \quad (16)$$

Finally, to correctly move the robot end-effector to  $\mathbf{P}_W$ , we must obtain the corresponding  $\mathbf{P}_R$  in the robot reference system  $R$  which adds error  $\mathbf{E}_W$ . It is worth noting that  $\mathbf{E}_W$  comprehends both the robot workspace calibration error and the robot intrinsic positioning error that may be due internal characteristics such as motor vibrations and encoder resolution. The correspondence between  $\mathbf{P}_W$  and  $\mathbf{P}_R$  is obtained from the robot calibration procedure detailed in Section 5.1.2, resulting in:

$$\mathbf{P}_R = \mathbf{P}_W + \mathbf{E}_W = \begin{bmatrix} \frac{1}{N} \sum_{n=1}^N x_O \\ \frac{1}{N} \sum_{n=1}^N y_O \end{bmatrix} + \mathbf{E}_{sk} + \mathbf{E}_H + \mathbf{E}_{map} + \mathbf{E}_W \quad (17)$$

It is made evident that moving from one reference frame to another introduces errors.  $\mathbf{E}_{map}$  can be assumed equal to zero if workspace  $H$  and workspace  $W$  dimensions are kept equal; hence, considering Equation (17) three experiments have been designed to determine the contribution on the overall positioning error of (i) the skeletonization error  $\mathbf{E}_{sk}$ , (ii) the camera calibration error  $\mathbf{E}_H$  and (iii) the robot error  $\mathbf{E}_W$ .



### 5.3.1 Evaluation of the hand skeleton estimation error

For each considered point  $p$  the positioning error  $\mathbf{E}_{sk}$  due to the estimation of the hand skeleton made by OpenPose neural network has been evaluated considering the difference between (i) theoretical positions  $\mathbf{T}_p$  [px] corresponding to the index positions in the image frame and (ii) actual positions  $\mathbf{A}_p$  [px] corresponding to the index joint position in the image frame calculated by OpenPose (Figure 41).

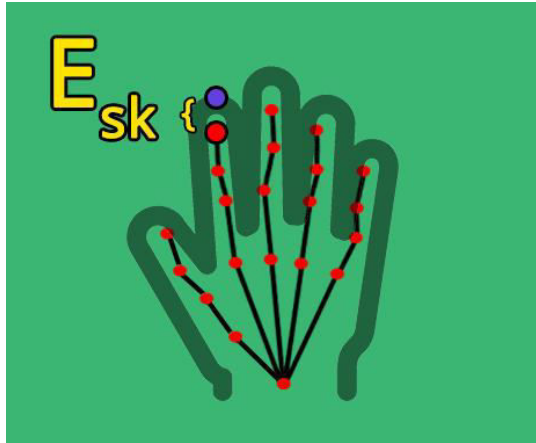


Figure 41. Graphical representation of the hand skeleton estimation error. The purple dot is the theoretical position of the index finger, while the red dot is the actual position of the index keypoint estimated by OpenPose.

When users point their finger to a position, they must keep the index gesture firmly in place until  $N = 7$  consecutive index gestures have been successfully detected by the software; hence, both theoretical and actual positions are obtained as the mean value over  $N$  consecutive frames. Considering that the values are in pixels, the resulting mean is floored. Error  $\mathbf{E}_{sk}$  is therefore calculated as follows for each point  $p$ :

$$\mathbf{E}_{sk}^p = (\mathbf{T}_p - \mathbf{A}_p) = \begin{bmatrix} \mathbf{T}_{x,p} \\ \mathbf{T}_{y,p} \end{bmatrix} - \begin{bmatrix} \mathbf{A}_{x,p} \\ \mathbf{A}_{y,p} \end{bmatrix} \quad (18)$$

The user participating in this evaluation experiment moved its hand to  $p = 14$  randomly chosen locations of workspace  $H$ , corresponding to  $p * N = 14 * 7 = 98$  couples of image frames and index joint estimations. Theoretical positions  $\mathbf{T}_p$  have been manually selected from each acquired frame considering the tip of the index finger, while the actual positioning  $\mathbf{A}_p$  of each frame corresponds to the predicted index keypoint ( $k_8$ ) obtained from

OpenPose neural network. The user hand in the acquired frames is both vertically oriented and left or right oriented and an equal number of left-hand and right-hand frames have been selected for the evaluation. It is worth noting that when users point their finger to a position, they must keep the index gesture firmly in place until  $N = 7$  consecutive index gestures have been successfully detected by the software. Hence, for each point  $p$  the resulting theoretical and actual positions are obtained as the floored mean value over  $N$  consecutive frames, namely  $\overline{\mathbf{T}}_p$  and  $\overline{\mathbf{A}}_p$ . Consequently, at each position  $p$  correspond  $N$  errors  $\mathbf{E}_{sk}^p$ ; therefore it is possible to calculate a mean error  $\overline{\mathbf{E}}_{sk}^p$  and a standard deviation  $\sigma\overline{\mathbf{E}}_{sk}^p$  as follows:

$$\overline{\mathbf{E}}_{sk}^p = \begin{bmatrix} \overline{\mathbf{E}}_{sk,x}^p \\ \overline{\mathbf{E}}_{sk,y}^p \end{bmatrix} = \begin{bmatrix} \frac{1}{N} \sum_{n=1}^N (\mathbf{T}_{x,n}^p - \mathbf{A}_{x,n}^p) \\ \frac{1}{N} \sum_{n=1}^N (\mathbf{T}_{y,n}^p - \mathbf{A}_{y,n}^p) \end{bmatrix} \quad (19)$$

$$\sigma\overline{\mathbf{E}}_{sk}^p = \begin{bmatrix} \sigma\overline{\mathbf{E}}_{sk,x}^p \\ \sigma\overline{\mathbf{E}}_{sk,y}^p \end{bmatrix}$$

Table 13 shows the values of  $\overline{\mathbf{T}}_p$ ,  $\overline{\mathbf{A}}_p$ ,  $\overline{\mathbf{E}}_{sk}^p$  and  $\sigma\overline{\mathbf{E}}_{sk}^p$  for each position  $p$ . The mean values of  $\overline{\mathbf{E}}_{sk}^p$  and  $\sigma\overline{\mathbf{E}}_{sk}^p$  over  $p = 14$  for both components are (2.06, -2.64) [px] and (3.61, 3.49) [px] respectively. The negative sign represents the case when the actual positioning  $\mathbf{A}_p$  is overestimated with respect to the corresponding theoretical positioning  $\mathbf{T}_p$ .

It is worth noting that the standard deviation  $\sigma\overline{\mathbf{E}}_{sk}^p$  gives the measure of how much the user kept the hand firmly in place for each location. This result is useful to understand in which position  $p$  the user moved its hand too much, thus reducing the accuracy of the estimation of  $\overline{\mathbf{A}}_p$ , an effect that could lead to an incorrect placing of the robot end-effector.

Table 13. Averaged valued of theoretical and actual positions  $\overline{T}_p$  and  $\overline{A}_p$ , of the hand skeleton estimation error  $\overline{E}_{sk}^p$  and of its standard deviation  $\sigma\overline{E}_{sk}^p$  for each point  $p$ .

Points	$\overline{T}_p$ [px]		$\overline{A}_p$ [px]		$\overline{E}_{sk}^p$ [px]		$\sigma\overline{E}_{sk}^p$ [px]	
	x	y	x	y	x	y	x	y
<b>P<sub>1</sub></b>	332	346	333	349	-1.29	-3.29	3.92	4.06
<b>P<sub>2</sub></b>	485	276	483	278	1.71	-2.14	3.06	3.00
<b>P<sub>3</sub></b>	437	264	437	269	-0.29	-4.86	4.49	6.49
<b>P<sub>4</sub></b>	509	278	507	282	1.43	-4.14	2.77	5.19
<b>P<sub>5</sub></b>	554	269	552	269	2.29	0.14	4.43	1.96
<b>P<sub>6</sub></b>	552	275	552	280	0.43	-5.43	2.56	1.18
<b>P<sub>7</sub></b>	614	199	610	205	4.14	-6.14	4.09	2.53
<b>P<sub>8</sub></b>	411	247	411	247	0.29	0.00	3.73	4.07
<b>P<sub>9</sub></b>	587	273	584	276	3.71	-3.00	3.84	3.82
<b>P<sub>10</sub></b>	401	255	400	256	1.29	-0.43	3.92	1.84
<b>P<sub>11</sub></b>	380	269	376	271	4.14	-1.43	4.09	4.03
<b>P<sub>12</sub></b>	446	174	442	176	3.29	-1.71	4.46	2.55
<b>P<sub>13</sub></b>	455	94	450	97	4.71	-3.00	2.37	4.24
<b>P<sub>14</sub></b>	496	243	493	245	3.00	-1.57	2.93	3.96

### 5.3.2 Evaluation of the camera error

Thanks to the calibration procedure detailed in Section 5.1.1, a point in the image frame can be converted to its corresponding point in workspace  $H$  by using Equation (3). The conversion from pixel coordinates to real-world coordinates introduces error  $\mathbf{E}_H$ , which in this experiment has been estimated considering the difference between (i) theoretical positions  $\mathbf{T}_p$  [mm] corresponding to the real-world coordinates of the centroid of markers  $\mathbf{P}_p$  in workspace  $H$ , calculated with respect to reference point  $\mathbf{P}_0$  (Figure 42) and (ii) the converted actual positions  $\mathbf{A}_p$  [mm] corresponding to real-world coordinates of the same markers, obtained converting their pixel coordinates (taken from the image frame) in real-world coordinates using Equation (3). It is worth noting that workspace  $H$  has been placed at 1 m from the camera and that its markers are squares of 1x1 cm positioned according to theoretical positions in Table 14, as shown in Figure 42.

Hence, for each marker position the total error  $\mathbf{E}_H$  is obtained as follows:

$$\mathbf{E}_H^p = (\mathbf{T}_p - \mathbf{A}_p) = \begin{bmatrix} \mathbf{T}_{x,p} \\ \mathbf{T}_{y,p} \end{bmatrix} - \begin{bmatrix} \mathbf{A}_{x,p} \\ \mathbf{A}_{y,p} \end{bmatrix} \quad (20)$$

Table 14 reports the values obtained for each point  $\mathbf{P}_p$ , where the negative sign represents the case when the actual position  $\mathbf{A}_p$  is overestimated with respect to the corresponding theoretical position  $\mathbf{T}_p$ . The resulting mean of the values of  $\mathbf{E}_H^p$  in Table 14 is  $(-10.33, -8.26)$  [mm], while the corresponding standard deviation is  $(7.40, 6.36)$  [mm].

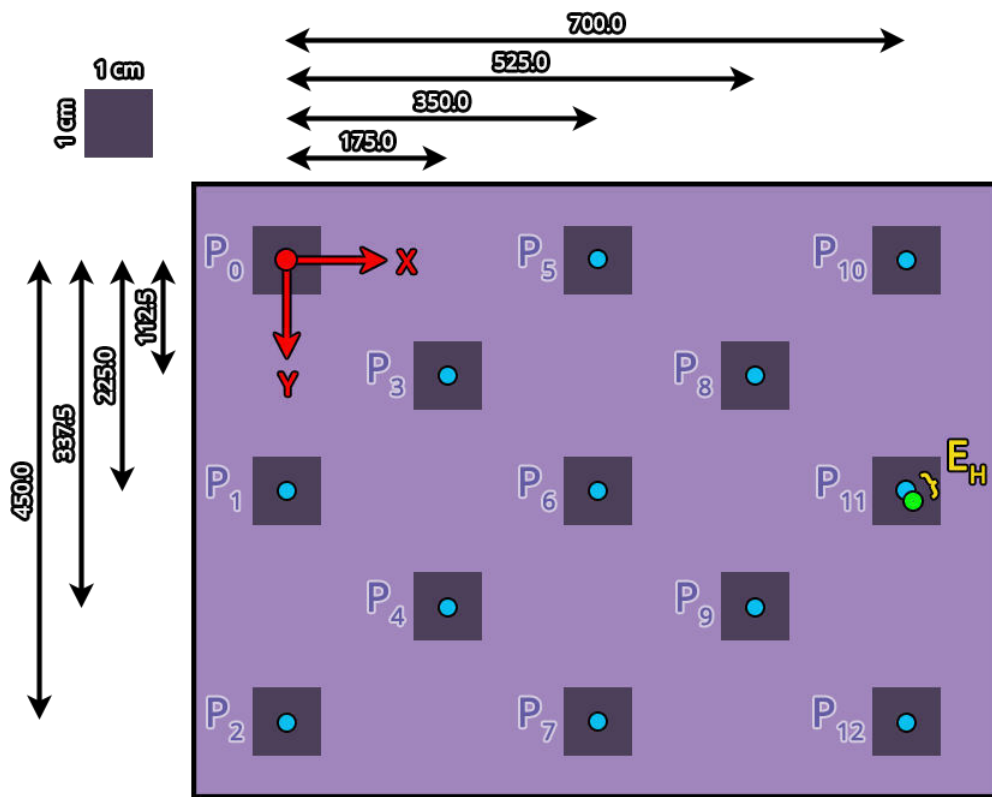


Figure 42. Scheme of workspace  $H$  representing all the markers  $\mathbf{P}_p$  positions with respect to reference point  $\mathbf{P}_0$ . Theoretical positions are represented by the centroid of the markers drawn in blue, while actual positions are calculated by the software using Equation (3) (an example is drawn in green). The red dot represents the centroid of reference point  $\mathbf{P}_0$ .

Table 14. Values of theoretical positions  $T_p$ , actual positions  $A_p$  and of the calculated camera error  $E_H^p$  for each marker  $P_p$  of workspace  $H$ .

Points	$T_p$ [mm]		$A_p$ [mm]		$E_H^p$ [mm]	
	x	y	x	y	x	y
$P_0$	0.00	0.00	-0.68	-0.66	0.68	0.66
$P_1$	0.00	225.00	0.99	234.17	-0.99	-9.17
$P_2$	0.00	450.00	4.52	462.42	-4.52	-12.42
$P_3$	175.00	112.50	180.19	118.14	-5.19	-5.64
$P_4$	175.00	337.50	180.91	349.22	-5.91	-11.72
$P_5$	350.00	0.00	360.32	-0.71	-10.32	-0.71
$P_6$	350.00	225.00	359.17	234.13	-9.17	-9.13
$P_7$	350.00	450.00	357.07	464.28	-7.07	-14.28
$P_8$	525.00	112.50	541.19	118.19	-16.19	-5.60
$P_9$	525.00	337.50	538.16	351.06	-13.16	-13.56
$P_{10}$	700.00	0.00	725.08	-2.64	-25.08	2.64
$P_{11}$	700.00	225.00	721.12	235.02	-21.12	-10.02
$P_{12}$	700.00	450.00	716.21	469.87	-16.21	-19.87

### 5.3.3 Evaluation of the robot error

The robot error  $E_W$  has two components: (i) error  $E_{W,a}$  which is due to the robot calibration procedure and (ii) error  $E_{W,b}$  which is due to physical robot characteristics, such as motor vibrations and encoder resolution:

$$E_W = E_{W,a} + E_{W,b} \quad (21)$$

Hence, since only  $E_{W,b}$  may be estimated with an experimental evaluation, it is possible to obtain error  $E_{W,a}$  by subtraction. Considering Equation (17) we obtain:

$$E_{W,a} = P_R - P_W = P_R - \begin{bmatrix} \frac{1}{N} \sum_{n=1}^N x_O \\ \frac{1}{N} \sum_{n=1}^N y_O \end{bmatrix} - E_{sk} - E_H - E_{map} - E_{W,b} \quad (22)$$

In fact, by moving the robot end-effector using “Hands-Free”,  $\mathbf{P}_W$  and  $\mathbf{P}_R$  are obtained automatically by the software using the conversion formulas in (3) and (11) respectively.

In this experiment the aim is to determine the value of  $\mathbf{E}_{W,b}$ . To avoid adding other contributions to its estimation, the robot is moved to the predefined theoretical positions of workspace  $W$  ( $\mathbf{T}_p = \mathbf{P}_W$ ) by sending to the robot encoder the precise coordinates using ROS. Hence, the actual position ( $\mathbf{A}_p = \mathbf{P}_R$ ) where the robot moves to is only affected by  $\mathbf{E}_{W,b}$ .

The robot has been moved to 7 theoretical positions 3 times each, corresponding to the centroid of markers  $\mathbf{P}_3, \mathbf{P}_4, \mathbf{P}_5, \mathbf{P}_6, \mathbf{P}_7, \mathbf{P}_8$  and  $\mathbf{P}_9$  of workspace  $W$  as in Figure 39. Therefore, for each point  $\mathbf{P}_p$ , we obtain:

$$\overline{\mathbf{E}_{W,b}^p} = \frac{1}{3} \sum_{n=1}^3 (\mathbf{T}_n^p - \mathbf{A}_n^p) = \frac{1}{3} \sum_{n=1}^3 \begin{bmatrix} \mathbf{T}_{x,n}^p \\ \mathbf{T}_{y,n}^p \end{bmatrix} - \begin{bmatrix} \mathbf{A}_{x,n}^p \\ \mathbf{A}_{y,n}^p \end{bmatrix} \quad (23)$$

To robustly determine the actual positions  $\mathbf{A}_p$  during the experiment, a 3D printed carrier holding a bright red laser (*Lasiris laser 635nm, 10mW*) has been mounted on the end-effector as shown in the bottom-right corner of Figure 43. To correctly visualize the laser dot, an RGB camera (*IDS Imaging UI-1460C*) has been mounted behind the glass pane. A measuring software has been developed using LabVIEW to measure the distance ( $\mathbf{E}_{W,b}^p$ ) between the theoretical position  $\mathbf{T}_p$  (automatically calculated as the centroid of the marker of workspace  $W$ , represented as the green dot in Figure 43, top-left corner) and the actual positioning  $\mathbf{A}_p$  (automatically detected as the centroid of the laser blob, represented as the red dot in Figure 43, top-left corner).

Table 15 reports the values of  $\overline{\mathbf{E}_{W,b}^p}$  and the corresponding standard deviation among the 3 repetitions for each point  $\sigma \overline{\mathbf{E}_{W,b}^p}$ . By calculating theoretical positions  $\mathbf{T}_p$  using the rigid transformation matrix that converts  $\mathbf{P}_W$  to  $\mathbf{P}_R$  as shown in Equation (9) instead of sending them precisely to the encoder, it is possible to estimate  $\mathbf{E}_{W,a}^p$  by performing again the subtraction between  $\mathbf{T}_p$  and  $\mathbf{A}_p$  for each  $\mathbf{P}_p$  and subtracting the average error for that point  $\overline{\mathbf{E}_{W,b}^p}$ . It is

worth noting that in Table 15 the negative sign represents the case when the actual position  $\mathbf{A}_p$  is overestimated with respect to the corresponding theoretical position  $\mathbf{T}_p$ . The resulting mean of the values of  $\overline{\mathbf{E}_{W,b}^p}$  is (0.66, 1.66) [mm], while the corresponding standard deviation is (0.13, 0.09) [mm]. On the other hand, the mean of the values of  $\mathbf{E}_{W,a}^p$  is (2.04, 0.49) [mm].

It is worth noting that according to the task repeatability reported in the robot datasheet, I assumed that it could reach the desired position with a minimal error. This assumption has been proved true, in fact the resulting values of  $\overline{\mathbf{E}_{W,b}^p}$  suggest that a joint-level calibration of the robot was not needed.

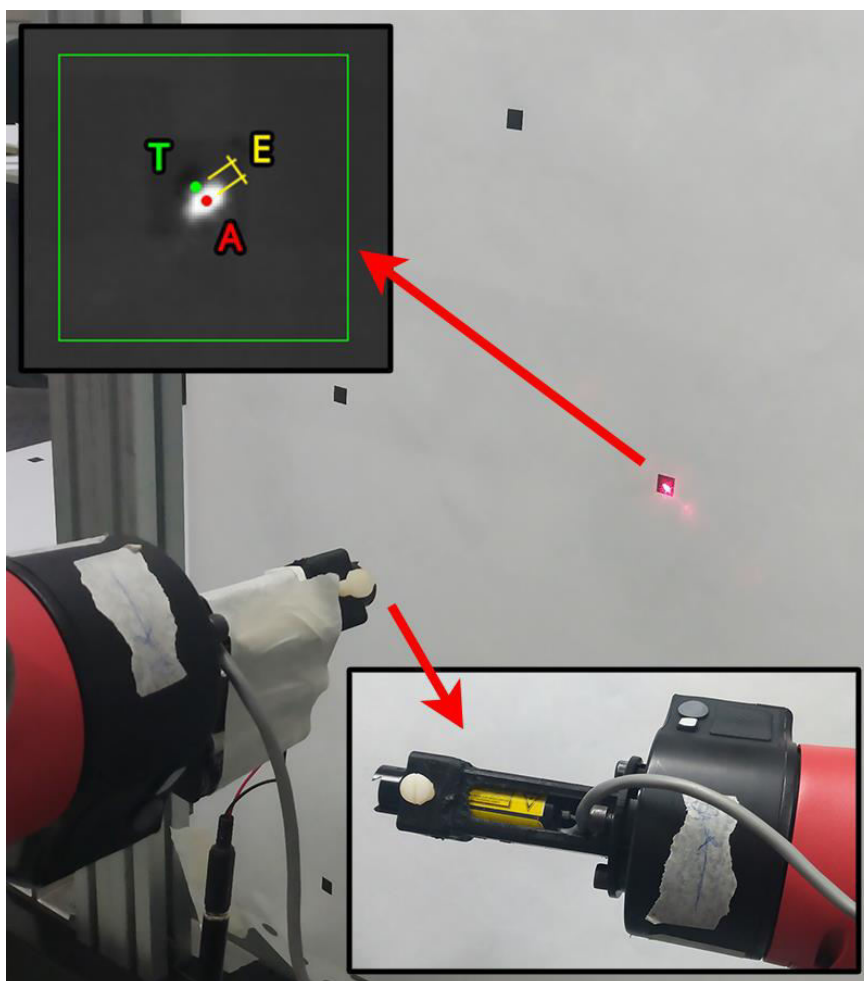


Figure 43. The image shows a close-up of the 3D printed laser carrier used (bottom-right corner) and of the image seen by the measuring software (top-left corner). Note that the image is captured by the RGB camera mounted behind the glass pane in order to see the markers and the laser blob.

Table 15. Values of the average intrinsic robot error  $\overline{E_{W,b}^p}$  and of its standard deviation  $\sigma E_{W,b}^p$ , as well as the calculated robot calibration error  $E_{W,a}^p$  for each marker  $P_p$  of workspace  $W$  used in the experiment.

Points	$\overline{E_{W,b}^p}$ [mm]		$\sigma E_{W,b}^p$ [mm]		$E_{W,a}^p$ [mm]	
	x	y	x	y	x	y
$P_3$	0.26	1.86	0.01	0.05	0.30	1.10
$P_4$	0.19	1.59	0.11	0.27	1.70	0.00
$P_5$	1.63	0.58	0.14	0.05	6.70	0.70
$P_6$	0.77	1.44	0.04	0.05	0.60	0.40
$P_7$	0.70	1.15	0.02	0.12	2.60	0.20
$P_8$	0.20	1.50	0.17	0.06	0.40	0.50
$P_9$	0.91	1.40	0.40	0.04	2.00	0.50

### 5.3.4 Discussion

Table 16 reports the mean values of each error obtained in the experiments. In particular, the values of  $E_{sk}$  have been converted from pixel to millimeters to allow the comparison with the other results.

Table 16. Average values of errors  $E_{sk}$ ,  $E_H$ ,  $E_{W,b}$  and  $E_{W,a}$  obtained in the experiments.

$E_{sk}$ [mm]		$E_H$ [mm]		$E_{W,b}$ [mm]		$E_{W,a}$ [mm]	
x	y	x	y	x	y	x	y
0.55	-0.70	-8.26	-10.33	0.66	1.66	2.04	0.49

I can conclude that the final position of the robot end-effector  $P_R$  is mostly affected by the camera error  $E_H$ . This result is not surprising, considering that the Kinect v2 sensor is not suited for applications that require high resolution images; hence, by adopting an industrial RGB camera with better performances the values of  $E_H$  may significantly improve.

Albeit the values of error  $E_{W,b}$  are aligned with the information found in the robot's datasheet, by improving the calibration between reference system  $W$  and reference system  $R$  it is possible to further reduce the contribution of error  $E_{W,a}$ , hence achieving a lower overall



robot error  $\mathbf{E}_W$ . It is worth noting that adopting a different robot with better performances may be necessary for precise positioning, as required by some applications (e. g. medical applications).

Finally, even if error  $\mathbf{E}_{sk}$  is the lowest of the three, by changing the camera and improving the set-up illumination conditions it may be possible to further improve the results. The operator ability of keeping its hand still on top of the desired position is another key factor to reduce the error values, albeit the filtering procedure helps minimizing this effect as much as possible.

At this stage of development, “Hands-Free” is an easy to use open-source software, which I made available on GitHub [128] as a further contribution. Future developments include adding the z-axis spatial control, for example by adopting wearable devices or traditional controllers to guarantee precise control over the robot end-effector. Improving the graphical interface is also a necessity to make the overall system intuitive and easy to use even for non-expert users. By adding other functionalities such as the hand-tracking to reproduce a certain trajectory in both 2D and 3D space, “Hands-Free” may be a valid alternative to teleoperate robotic manipulators intuitively and with reduced costs. Furthermore, the software may be used as a standalone package or in combination with MEGURU; in this case, it is considered as a plug-in easily accessible from MEGURU’s State Machine.

## Conclusions

Starting from the Industry 4.0 (I4.0) paradigm objectives, a set of enabling technologies has been defined through the years to push the fourth revolution forward. One of the ideas is to design *Cyber Physical Systems*, which are smart systems partially autonomous and cooperative, able to work in teams of machines or with humans. To this aim, the field of Collaborative Robotics plays a central role as one of the key technologies to create such intelligent systems. Albeit the paradigm was launched in 2014, up until today a limited number of firms all around the world adhere to its principles. Specifically, Italy started focusing on the digital transformation with funding plans only since 2017, as reflected on the adoption level of I4.0 technologies, which is extremely low. It has been hypothesized that this behavior may be due to the reduced level of knowledge of the firms' staff about the paradigm and the enabling technologies, as well as the limited funding possibilities of small-medium companies to buy these technologies and educate their staff. The topic of human-robot collaboration in industrial production lines has been thoroughly researched during the years, but even if the scientific community has proposed successful applications these are not currently adopted by firms. In fact, they adopt only three types of workstations: (i) *completely manual*, where the operator is in charge of the production process from start to finish; (ii) *completely automated*, where a set of industrial robots and manipulators carry out the production process automatically and (iii) *collaborative workstations*, where Cobots perform a portion of the process while being near the human operator, which performs other tasks. Alas, this type of collaboration is still inadequate, since Cobots are mechanically limited to fulfill the safety regulations needed to work alongside humans. Moreover, operators still communicate with them using traditional interfaces such as teach pendants and complex proprietary software, losing a lot of time to program the robots' tasks, which makes the collaborative workstation not flexible enough to meet the increasing need of production flexibility, as pointed out by the I4.0 paradigm.

This thesis work proposes a new workstation type, named "**Meta-Collaborative**": the idea is to create an intuitive and easy to use interface that allows users to communicate with any

kind of robot, being it Industrial or Collaborative, regardless of their workspace, which may be overlapping or not. To achieve a successful communication between the two, Gesture Recognition plays a central role. In fact, it has been studied that communicating by body and hand gestures is straightforward and natural for humans as much as using their voice. Compared to vocal communication, though, by using their body any person may successfully operate the robot regardless of its language and of the background noises that may be present in industrial plants. Nowadays, Gesture Recognition includes different technologies according to the sensor adopted, but the scientific community that adopt visual systems and machine learning models has proposed the most interesting results. In this case, intelligent models recognize the gesture from images, often acquired in real-time from the camera, and output the recognized gesture class with low inference times.

By following this trend, in this thesis work I experimentally studied gestures and Deep Learning models to build an intuitive and effective *gesture dictionary* to be used as the communication language between human operators and robotic manipulators. With this language, which is based on the presence of both hands at the same time, users may effectively command the robot by using my open-source software named MEGURU (*MEta-collaborative GestUre-based Robot program bUilder*). MEGURU is aimed at substituting the traditional robot interface, for example the teach pendant, hence its functionalities mirror the classic ones. It has been designed as a State Machine where users navigate using the developed gesture dictionary, and it is currently capable of building robot programs by using simple actions as building blocks (e. g. moving to a point, opening the gripper, etc). MEGURU has been tested in two experiments to evaluate (i) the user experience and (ii) its effectiveness when compared to the teach pendant when building a robot program. The experimental results show that the perfect target users for MEGURU are people of age 20-40, which showed low response times and a better comprehension of the overall structure, allowing them to conclude their tasks in less time and with low recognition errors. Furthermore, when compared to the teach pendant programming method, MEGURU showed competing performances which strongly depend on the number of recognition errors. These are often due to the background and illumination characteristics of the scene, parameters that may be controlled in industrial plants to further

increase the system's robustness. It is worth noting that in future validation campaigns rigid procedures to determine the Subjects participating in the tests will be used according to the literature. Designing a robust and reliable questionnaire to evaluate the user's experience with a certain device or software is not an easy task either; hence, in future studies existing state-of-the-art techniques to build the surveys will be adopted as well. To provide a better user experience, a graphical interface should be developed in the future to guide the operator through MEGURU's States and internal functionalities, also providing them feedback to adjust their position accordingly and improve the recognition of the gestures. Augmented Reality is also a technology that may work well with MEGURU; hence, future research directions may be aimed at incorporating these techniques in the software. For example, the graphical interface may be developed in an AR fashion and compared with a more traditional one in a set of experiments aimed at evaluating the user experience with both.

Finally, since MEGURU has been designed as a modular core, I designed an expansion module named "Hands-Free". Its aim is to allow the robot's end-effector teleoperation by using only the index finger viewed by an RGB camera. To achieve this, "Hands-Free" extracts the skeletonization of the user's hand by using OpenPose, an open-source Deep Learning-based software that estimates the skeleton of the human body from RGB images. Teleoperation is achieved thanks to precise camera and robot calibration procedures and a transformation chain to correctly map the user workspace (where the index finger moves) and the robot workspace (where the end-effector moves). The final end-effector position is affected by some errors, which have been estimated through three experiments: (i) the *camera error*, which is due to the camera calibration procedure necessary to transform a point from pixel coordinates to real-world coordinates and vice-versa; (ii) the *skeletonization error*, which is due to the skeleton estimation produced by OpenPose and (iii) the *robot error*, which is both due to the robot internal mechanics and its calibration procedure. Our experiments show that the highest one is the camera error, which may be improved by adopting a better performing camera with higher resolution. Improving the robot calibration procedure and the illumination conditions of the set-up are also feasible options to obtain lower values of the respective errors. Overall, "Hands-Free" demonstrated to be an effective teleoperation method if no

precise positioning is required by the application. Furthermore, this version only includes a 2D teleoperation method: a new version is currently under study to add the third axis control by means of different controllers.

## Bibliography

- [1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, M. Hoffman, «Industry 4.0,» *Business & Information Systems Engineering*, vol. 6, n. 4, pp. 239-242, 2014.
- [2] L. Monostori, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, K. Ueda, «Cyber-Physical Systems in Manufacturing,» *CIRP Annals - Manufacturing Technologies*, vol. 65, n. 2, pp. 621-641, 2016.
- [3] F. Almada-Lobo, «The Industry 4.0 revolution and the future of manufacturing execution systems (MES),» *Journal of Innovation Management*, vol. 3, n. 4, pp. 16-21, 2015.
- [4] L. D. Xu, E. L. Xu, L. Li, «Industry 4.0: state of the art and future trends,» *International Journal of Production Research*, vol. 56, n. 8, pp. 2941-2962, 2018.
- [5] T. Zheng, M. Ardolino, A. Bacchetti, M. Perona, M. Zanardini, «The impacts of Industry 4.0: a descriptive survey in the Italian manufacturing sector,» *Journal of Manufacturing Technology Management*, 2019.
- [6] C. Yubao, «Integrated and Intelligent Manufacturing: Perspectives and Enablers,» *Engineering*, vol. 3, n. 5, pp. 588-595, 2017.
- [7] Y. Liao, F. Deschamps, F. de Freitas Rocha Loures, L. Pierin Ramos, «Past, present and future of Industry 4.0 - a systematic literature review and research agenda proposal,» *International Journal of Production Research*, vol. 55, n. 12, pp. 3609-3629, 2017.
- [8] R. Reif, A. J. Shirley, A. Liveris, «Report to the President Accelerating U.S. Advanced Manufacturing,» in *The President's Council of Advisors on Science and Technology*, Washington, DC, 2014.
- [9] H. S. Kang, Y. L. Ju, S. C. Sang, K. Hyun, H. P. Jun, Y. S. Ji, H. K. Bo, D. N. Sang, «Smart Manufacturing: Past Research, Present Findings, and Future Directions,» *International Journal of Precision Engineering*, vol. 3, n. 1, pp. 111-128, 2016.
- [10] K. Li, «Made in China 2025,» Beijing: State Council of China, 2015.

- [11] Cabinet Office, «Report on The 5th Science and Technology Basic Plan,» Tokyo: Cabinet Office of Japan, 2015.
- [12] National Research Foundation, «Research, Innovation and Enterprise (RIE) 2015 Plan,» Prime Minister's Office of Singapore, 2016.
- [13] European Commission, «Digital Transformation Monitor: National Initiatives,» [Online]. Available: <https://ec.europa.eu/growth/tools-databases/dem/monitor/category/national-initiatives>. [Consulted in November 2020].
- [14] K. Henning, W. Wahlster, J. Helbig, «Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0,» Berlin: Industrie 4.0 Working Group of Acatech, 2013.
- [15] Foresight, «The Future of Manufacturing: A New Era of Opportunity and Challenge for the UK,» London: UK Government Office for Science, 2013.
- [16] Conseil National de l'Industrie, «The New Face of Industry in France,» Paris: French National Industry Council, 2013.
- [17] European Commission, «Factories of the Future PPP: Towards Competitive EU Manufacturing,» Bruxelles: European Commission, 2016.
- [18] M. Brettel, N. Friederichsen, M. A. Keller, M. Rosenberg, «How Virtualization, Decentralization and Network Building Change the Manufacturing Landscape: An Industry 4.0 Perspective,» *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, vol. 8, n. 1, pp. 37-44, 2014.
- [19] G. Schuh, J. Arnoscht, S. Rudolf, «Integrated development of modular product platforms,» in *Proceedings of PICMET 2010 Technology Management for Global Economic Growth*, 2010.
- [20] P. P. Rogers, D. Ojha, R. E. White, «Conceptualising complementarities in manufacturing flexibility: a comprehensive view,» *International Journal of Production Research*, vol. 49, n. 12, pp. 3767-3793, 2011.
- [21] M. Ghobakhloo, «The future of manufacturing industry: a strategic roadmap toward Industry 4.0,» *Journal of Manufacturing Technology Management*, vol. 29, n. 6, pp. 910-936, 2018.

- [22] K. Schwab, «The Global Competitiveness Report 2019,» Switzerland: World Economic Forum, 2019.
- [23] «Trading Economics,» [Online]. Available: <https://tradingeconomics.com/>. [Consulted in May 2020].
- [24] W. Terkaj, T. Tolio, «The Italian Flagship Project: Factories of the Future,» in *Factories of the Future: The Italian Flagship Initiative*, Springer International Publishing, 2019, pp. 3-35.
- [25] G. Legnani, I. Fassi, «Robotica Industriale,» 2019.
- [26] A. Gasparetto, L. Scalera, «A Brief History of Industrial Robotics in the 20th Century,» *Advances in Historical Studies*, vol. 8, n. 1, pp. 24-35, 2019.
- [27] M. Akella, M. Peshkin, J. E. Colgate, W. Wannasuphoprasit, «Cobots: a novel material handling technology,» in *Proceedings of the ASME Winter Annual Meeting*, 1998.
- [28] J. E. Colgate, W. Wannasuphoprasit, M. A. Peshkin, «Cobots: Robots for collaboration with human operator,» in *Proceedings of the ASME Dynamic Systems and Control Division*, 1996.
- [29] V. Villani, F. Pini, F. Leali, C. Secchi, «Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications,» *Mechatronics*, vol. 55, pp. 248-266, 2018.
- [30] M. P. Polverini, A. M. Zanchettin, P. Rocco, «A computationally efficient safety assessment for collaborative robotics applications,» *Robotics and Computer-Integrated Manufacturing*, pp. 25 - 37, 2017.
- [31] S. Pasinetti, C. Nuzzi, M. Lancini, A. Fornaser, F. Docchio, G. Sansoni, «Development and characterization of a safety system for robotic cells based on multiple Time of Flight (TOF) cameras and point cloud analysis,» in *Proceedings of the 2018 IEEE International Workshop on Metrology for Industry 4.0 and IoT*, Brescia, Italy, 2018.
- [32] J. Krüger, T. K. Lien, A. Verl, «Cooperation of human and machines in assembly lines,» *CIRP Annals*, vol. 58, n. 2, pp. 628 - 646, 2009.



- [33] M. P. Pacaux-Lemoine, D. Trentesaux, G. Z. Rey, «Human-machine cooperation to design Intelligent Manufacturing Systems,» in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Florence, 2016.
- [34] I. Goodfellow, Y. Bengio, A. Courville, «Deep Learning,» MIT Press, 2017.
- [35] T. M. Mitchell, «Machine Learning,» New York: McGraw-Hill, 1997.
- [36] M. D. Zeiler, R. Fergus, «Visualizing and Understanding Convolutional Networks,» in *European conference on computer vision (ECCV)*, 2014.
- [37] D. H. Hubel, T. N. Wiesel, «Receptive Fields, Binocular Interaction And Functional Architecture In The Cat's Visual Cortex,» *Journal of Physiology*, vol. 160, n. 1, pp. 106-154, 1962.
- [38] W. S. McCulloch, W. Pitts, «A logical calculus of the ideas immanent in nervous activity,» *The bulletin of mathematical biophysics*, vol. 5, n. 4, pp. 115-133, 1943.
- [39] F. F. Rosenblatt, «The perceptron: a probabilistic model for information storage and organization in the brain,» *Psychological review*, vol. 65, n. 6, pp. 386-408, 1958.
- [40] M. L. Minsky, S. A. Papert, *Perceptrons*, Cambridge: MIT Press, 1969.
- [41] A. G. Ivakhnenko, V. G. Lapa, «Cybernetic Predicting Devices,» 1965.
- [42] A. G. Ivakhnenko, «Polynomial theory of complex systems,» *IEEE Transactions on Systems, Man and Cybernetics*, vol. 4, pp. 364-378, 1971.
- [43] J. Schmidhuber, «My First Deep Learning System of 1991 + Deep Learning Timeline 1960-2013,» 2013. [Online]. Available: <http://people.idsia.ch/~juergen/firstdeeplearner.html>. [Consulted in November 2020].
- [44] K. Fukushima, «Cognitron: A self-organizing multilayered neural network,» *Biological Cybernetics*, vol. 20, pp. 121-136, 1975.
- [45] K. Fukushima, «Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,» *Biological Cybernetics*, vol. 36, pp. 193-202, 1980.
- [46] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, «Gradient-based learning applied to document recognition,» *Proceedings of the IEEE*, vol. 86, n. 11, pp. 2278-2324, 1998.

- [47] G. E. Hinton, J. McClelland, D. Rumelhart, «Distributed representations,» in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Cambridge, MIT Press, 1986, pp. 77-109.
- [48] D. Rumelhart, G. Hinton, R. Williams, «Learning representations by back-propagating errors,» *Nature*, vol. 323, pp. 533-536, 1986.
- [49] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, «Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,» in *A Field Guide to Dynamical Recurrent Neural Networks*, IEEE press, 2001.
- [50] J. Schmidhuber, «Learning complex, extended sequences using the principle of history compression,» *Neural Computation*, vol. 4, n. 2, pp. 234-242, 1992.
- [51] S. Hochreiter, J. Schmidhuber, «Long Short-Term Memory,» *Neural Computation*, vol. 9, n. 8, pp. 1735-1780, 1997.
- [52] G. E. Hinton, S. Osindero, Y. Teh, «A fast learning algorithm for deep belief nets,» *Neural Computation*, vol. 18, pp. 1527-1554, 2006.
- [53] O. Russakovsky , J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, «ImageNet Large Scale Visual Recognition Challenge,» *International Journal of Computer Vision*, vol. 115, n. 3, pp. 211-252, 2015.
- [54] A. Krizhevsky, I. Sutskever, G. E. Hinton, «Imagenet classification with deep convolutional neural networks,» *Advances in neural information processing systems*, pp. 1097-1105, 2012.
- [55] K. Simonyan, A. Zisserman, «Very Deep Convolutional Networks for Large-Scale Image Recognition,» in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, 2014.
- [56] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, «Going deeper with convolutions,» in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- [57] R. Girshick, J. Donahue, T. Darrell, J. Malik, «Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,» in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 2014, pp. 580 - 587.

- [58] K. He, X. Zhang, S. Ren, J. Sun, «Deep residual learning for image recognition,» *arXiv preprint arXiv:1512.03385*, 2015.
- [59] H. Wang, B. Raj, «On the origin of deep learning,» *arXiv preprint arXiv:1702.07800*, 2017.
- [60] R. Girshick, «Fast R-CNN,» in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, IEEE Computer Society, 2015, pp. 1440 -1448.
- [61] R. Shaoqing, H. Kaiming, R. Girshick, S. Jian, «Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks,» in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, MIT Press, 2015, pp. 91 - 99.
- [62] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [63] J. Dai, Y. Li, K. He, J. Sun, «R-FCN: Object Detection via Region-based Fully Convolutional Networks,» *arXiv preprint arXiv:1605.06409*, 2016.
- [64] W. Liu, Dragomir A., D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, A. C. Berg, «SSD: Single Shot MultiBox Detector,» in *ECCV*, 2016.
- [65] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, K. Murphy, «Speed/accuracy trade-offs for modern convolutional object detectors,» in *CVPR*, 2017.
- [66] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, L. C. Zitnick, P. Dollár, «Microsoft COCO: Common Objects in Context,» in *arXiv preprint arXiv:1405.0312*, 2014.
- [67] T. Fong, C. Thorpe, C. Baur, «Collaboration, Dialogue, Human-Robot Interaction,» in *Robotics Research*, 2003, pp. 255 - 266.
- [68] L. Hongyi, W. Lihui, «Gesture Recognition for Human-Robot Collaboration: a Review,» *International Journal of Industrial Ergonomics*, 2017.

- [69] A. Kendon, «How gestures can become like words,» in *Crosscultural perspectives in nonverbal communication*, Toronto, Canada: Hogrefe, Potyatos, F., 1988, pp. 131-141.
- [70] B. T. Tervoort, «Esoteric symbolism in the communication behavior of young deaf children,» *American Annals of the Deaf*, vol. 106, pp. 436-480, 1961.
- [71] D. McNeill, «Hand and Mind: what gestures reveal about thought,» Chicago, USA: University of Chicago press, 1992.
- [72] S. Mitra, T. Acharya, «Gesture Recognition: A Survey,» *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 37, n. 3, pp. 311-324, 2007.
- [73] P. Barattini, C. Morand, N. M. Robertson, «A proposed gesture set for the control of industrial collaborative robots,» in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, 2012, pp. 132 - 137.
- [74] M. R. Pedersen, L. Nalpantidis, R. S. Andersen, C. Schou, S. Bøgh, V. Krüger, O. Madsen, «Robot skills for manufacturing: From concept to industrial deployment,» *Robotics and Computer-Integrated Manufacturing*, vol. 37, pp. 282-291, 2016.
- [75] H. Liu, L. Wang, «Gesture recognition for human-robot collaboration: A review,» *International Journal of Industrial Ergonomics*, vol. 68, pp. 355-367, 2017.
- [76] P. Neto, M. Simão, N. Mendes, M. Safeea, «Gesture-based human-robot interaction for human assistance in manufacturing,» *The International Journal of Advanced Manufacturing Technology*, vol. 101, n. 1-4, pp. 119-135, 2019.
- [77] Y. Lecun, Y. Bengio, G. Hinton, «Deep Learning,» *Nature*, vol. 521, n. 7553, pp. 436 - 444, 05 2015.
- [78] N. Nikolaos, M. Vasilis, M. Sotiris, «A cyber physical system (CPS) approach for safe human-robot collaboration in a shared workplace,» *Robotics and Computer Integrated Manufacturing*, vol. 56, pp. 233-243, 2018.
- [79] F. Pedersoli, S. Benini, N. Adami, R. Leonardi, «XKin: an open source framework for hand pose and gesture recognition using kinect,» *The Visual Computer*, vol. 30, n. 10, pp. 1107-1122, 2014.

- [80] A. Tellaeché, J. Kildal, I. Maurtua, «A flexible system for gesture based human-robot interaction,» *Procedia CIRP*, vol. 72, pp. 57-62, 2018.
- [81] S. Makris, P. Tsarouchi, A. S. Matthaiakis, A. Athanasatos, X. Chatzigeorgiou, M. Stefos, K. Giavridis, S. Aivaliotis, «Dual arm robot in cooperation with humans for flexible assembly,» *CIRP Annals*, vol. 66, n. 1, pp. 13-16, 2017.
- [82] P. Tsarouchi, A. Athanasatos, S. Makris, X. Chatzigeorgiou, G. Chryssolouris, «High level robot programming using body and hand gestures,» *Procedia CIRP*, vol. 55, pp. 1-5, 2016.
- [83] H. I. Lin, M. H. Hsu, W. K. Chen, «Human hand gesture recognition using a convolution neural network,» in *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, 2014, pp. 1038 - 1043.
- [84] G. Ross, D. Jeff, D. Trevor, M. Jitendra, «Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,» in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 2014, pp. 580 - 587.
- [85] P. Dollar, C. Wojek, B. Schiele, P. Perona, «Pedestrian Detection: An Evaluation of the State of the Art,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 743 - 761, 2012.
- [86] J. O. P. Arenas, R. J. Moreno, P. C. U. Murillo, «Hand gesture recognition by means of region-based convolutional neural networks,» *Contemporary Engineering Sciences*, vol. 10, n. 27, pp. 1329 - 1342, 2017.
- [87] C. Nuzzi, S. Pasinetti, M. Lancini, F. Docchio, G. Sansoni, «Deep Learning based Machine Vision: first steps towards a hand gesture recognition set up for Collaborative Robots,» in *Proceedings of the 2018 IEEE International Workshop on Metrology for Industry 4.0 and IoT*, Brescia, Italy, 2018.
- [88] C. Nuzzi, S. Pasinetti, M. Lancini, F. Docchio, G. Sansoni, «Deep Learning-based hand gesture recognition for Collaborative Robots,» *IEEE Instrumentation & Measurement Magazine*, vol. 22, n. 2, pp. 44-51, April 2019.

- [89] J. L. Raheja, R. Shyam, U. Kumar, P. B. Prasad, «Real-Time Robotic Hand Control Using Hand Gestures,» in *2010 Second International Conference on Machine Learning and Computing*, 2010.
- [90] P. Tsarouchi, A.-S. Matthaïakis, S. Makris, G. Chryssolouris, «On a human-robot collaboration in an assembly cell,» *International Journal of Computer Integrated Manufacturing*, vol. 30, n. 6, pp. 580-589, 2017.
- [91] K. Konda, H. Schulz, A. Königs, D. Schulz, «Real time interaction with mobile robots using hand gestures,» in *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2012.
- [92] Y. Xiao, Z. Zhang, A. Beck, J. Yuan, D. Thalmann, «Human–Robot Interaction by Understanding Upper Body Gestures,» *Presence: Teleoperators and Virtual Environments*, vol. 23, n. 2, pp. 133-154, 2014.
- [93] C. Hu, M. Q. Meng, P. X. Liu, X. Wang, «Visual gesture recognition for human-machine interface of robot teleoperation,» in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.
- [94] S. Waldherr, R. Romero, S. Thrun, «A Gesture Based Interface for Human-Robot Interaction,» *Autonomous Robots*, vol. 9, p. 151–173, 2000.
- [95] J. Sell, P. O'Connor, «The Xbox One System on a Chip and Kinect Sensor,» *IEEE Micro*, vol. 34, n. 2, pp. 44-53, 2014.
- [96] O. Mazhar, S. Ramdani, B. Navarro, R. Passama, A. Cherubini, «Towards Real-Time Physical Human-Robot Interaction Using Skeleton Information and Hand Gestures,» in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.
- [97] O. Mazhar, B. Navarro, S. Ramdani, R. Passama, A. Cherubini, «A real-time human-robot interaction framework with robust background invariant hand gesture detection,» *Robotics and Computer-Integrated Manufacturing*, vol. 60, pp. 34-48, 2019.
- [98] C. Nuzzi, S. Pasinetti, R. Pagani, D. Franco, G. Sansoni, «Hand gesture recognition for collaborative workstations: a smart command system prototype,» in *Image Analysis and Processing -- ICIAP 2019*, Trento, 2019.

- [99] M. Quigley, B. P. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Ng, «ROS: an open-source Robot Operating System,» in *ICRA Workshop on Open Source Software*, 2009.
- [100] L. Xiang, F. Ehtler, C. Kerl, T. Wiedemeyer, Lars, Hanyazou, R. Gordon, F. Facioni, laborer2008, R. Wareham, M. Goldhoorn, Alberth, bagorpapp, S. Fuchs, jmtatsch, J. Blake, Federico, H. Jungkurth, Y. Mingze, vinouz, D. Coleman, B. Burns, R. Rawat, S. Mokhov, P. Reynolds, P. E. Viau, M. Fraissinet-Tachet, Ludique, J. Billingham, Alistair, *libfreenect2: Release 0.2*, 2016.
- [101] T. Wiedemeyer, «IAI Kinect2,» 2015. [Online]. Available: [https://github.com/code-iai/iai\\_kinect2](https://github.com/code-iai/iai_kinect2). [Consulted in January 2018].
- [102] C. Nuzzi, S. Pasinetti, R. Pagani, S. Ghidini, M. Beschi, G. Coffetti, G. Sansoni, «MEGURU: a gesture-based robot program builder for Meta-Collaborative workstations,» *Robotics and Computer-Integrated Manufacturing*, vol. 68, p. 102085, 2021.
- [103] K. Krot, V. Kutia, «Intuitive methods of industrial robot programming in advanced manufacturing systems,» in *International Conference on Intelligent Systems in Production Engineering and Maintenance*, 2018, pp. 205-214.
- [104] Z. Pan, J. Polden, N. Larkin, S. Van Duin, J. Norrish, «Recent progress on programming methods for industrial robots,» *Robotics and Computer-Integrated Manufacturing*, vol. 28, pp. 87-94, 2011.
- [105] V. Villani, F. Pini, F. Leali, C. Secchi, «Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications,» *Mechatronics*, vol. 55, pp. 248-266, 2018.
- [106] M. Khatib, K. Al Khudir, A. De Luca, «Human-robot contactless collaboration with mixed reality interface,» *Robotics and Computer-Integrated Manufacturing*, vol. 67, p. 102030, 2021.
- [107] E. Magrini, F. Ferraguti, A. J. Ronga, F. Pini, A. De Luca, F. Leali, «Human-robot coexistence and interaction in open industrial cells,» *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101846, 2020.

- [108] P. Tsarouchi, S. Makris, G. Chryssolouris, «Human–robot interaction review and challenges on task planning and programming,» *International Journal of Computer Integrated Manufacturing*, vol. 29, n. 8, pp. 916-931, 2016.
- [109] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, Y. Sheikh, «OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields,» *arXiv preprint arXiv:1812.08008*, 2018.
- [110] T. Chen, H.-R. Tsai, «Ubiquitous manufacturing: Current practices, challenges, and opportunities,» *Robotics and Computer-Integrated Manufacturing*, vol. 45, p. 126–132, 2017.
- [111] S. M. Edwards, C. L. Lewis, «ROS-Industrial - Applying the Robot Operating System (ROS) to Industrial Applications,» in *Presented at the International Conference on Robotics and Automation/Robot Operating System Developer Conference (ICRA/ROSCon)*, St. Paul, Minnesota, 2012.
- [112] «Move-It Motion Planning Framework,» [Online]. Available: <https://moveit.ros.org/>. [Consulted in June 2019].
- [113] «ROS-I Supported Hardware,» [Online]. Available: [http://wiki.ros.org/Industrial/supported\\_hardware](http://wiki.ros.org/Industrial/supported_hardware). [Consulted in June 2019].
- [114] J. Bohren, S. Cousins, «The SMACH High-Level Executive [ROS News],» *IEEE Robotics & Automation Magazine*, vol. 17, n. 4, pp. 18-20, 2010.
- [115] H. A. Yanco, J. L. Druri, «A Taxonomy for Human-Robot Interaction,» in *Proceedings of the AAAI Fall Symposium on Human-Robot Interaction*, 2002.
- [116] J. Vertut, P. C. Coeffet, *Robot Technology*; vol. 3A Teleoperation and Robotics Evolution and Development, Prentice Hall, 1986.
- [117] J. Rebelo, T. Sednaoui, E. B. Den Exter, T. Krueger, A. Schiele, «Bilateral robot teleoperation: A wearable arm exoskeleton featuring an intuitive user interface,» *IEEE Robotics & Automation Magazine*, vol. 21, n. 4, pp. 62-69, 2014.
- [118] X. Lv, M. Zhang, F. Cui, X. Zhang, «Teleoperation of robot based on virtual reality,» in *16th International Conference on Artificial Reality and Telexistence--Workshops (ICAT'06)*, 2006.



- [119] P. F. Hokayem, M. W. Spong, «Bilateral teleoperation: An historical survey,» *Automatica*, vol. 42, n. 12, pp. 2035-2057, 2006.
- [120] C. Glover, B. Russell, A. White, M. Miller, A. Stoytchev, «An effective and intuitive control interface for remote robot teleoperation with complete haptic feedback,» in *Proceedings of the Emerging Technologies Conference-ETC*, 2009.
- [121] M. Tavakoli, A. Aziminejad, R. V. Patel, M. Moallem, «High-Fidelity Bilateral Teleoperation Systems and the Effect of Multimodal Haptics,» *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, n. 6, pp. 1512-1528, 2007.
- [122] R. G. Boboc, H. Moga, D. Talaba, «A Review of Current Applications in Teleoperation of Mobile Robots,» *Bulletin of the Transilvania University of Brasov. Engineering Sciences. Series I*, vol. 5, n. 2, pp. 9-16, 2012.
- [123] J. Kofman, X. Wu, T. J. Luu, S. Verma, «Teleoperation of a robot manipulator using a vision-based human-robot interface,» *IEEE Transactions on Industrial Electronics*, vol. 52, n. 5, pp. 1206-1219, 2005.
- [124] F. Weichert, D. Bachmann, B. Rudak, D. Fisseler, «Analysis of the Accuracy and Robustness of the Leap Motion Controller,» *Sensors*, vol. 13, pp. 6380-6393, 2013.
- [125] T. Simon, H. Joo, I. Matthews, Y. Sheikh, «Hand Keypoint Detection in Single Images using Multiview Bootstrapping,» in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017.
- [126] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, «Caffe: Convolutional Architecture for Fast Feature Embedding,» in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014.
- [127] Z. Zhang, «A flexible new technique for camera calibration,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, n. 11, pp. 1330-1334, 2000.
- [128] C. Nuzzi, S. Ghidini, R. Pagani, S. Pasinetti, G. Coffetti, G. Sansoni, «Hands-Free: a robot augmented reality teleoperation system,» in *2020 17th International Conference on Ubiquitous Robots (UR)*, 2020.