POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

AUTONOMOUS MISSION CONFIGURATION ON SPOT FROM BOSTON DYNAMICS

Academic Supervisor:
Dr. Primatesta Stefano

Internship Supervisor:
Dr. Pistamiglio Oscar

Candidate:
Vanniyakulasingam Vinogiga

April 2023

Table of contents

Acknowledgements

This thesis work marks the end of two years of hard work and dedication and closes a meaningful chapter of my life. Despite the onset of the covid-19 pandemic, it has been a time of deep learning and personal growth. I am grateful for all the people who have supported me during this journey, starting from my academic supervisor, Dr. Primatesta Stefano, who has always been available for clarifications and suggestions. Special thanks go also to my internship supervisor Pistamiglio Oscar for giving me the opportunity to work on this formative experience and for always being there to help me. Further, I would like to thank my internship team leader Sedini Andrea for all the precious teachings he gave me and my thesis colleagues Mazza Paolo, Compagnone Pierluigi and De Santis Francesco for their incredible support and collaboration. I am truly grateful to have worked with people like you.
I wish to extend my thanks to my friends, especially Viglietti Claudia and Zanetta Alessandra, who have always been by my side, despite the difficulties.
Last but not least, I sincerely thank my family for their support, even financially, and for the trust they placed in me: I would not be here without them.
For you all I dedicate my thesis work!

Vinogiga Vanniyakulasingam

Abstract

Nowadays, robotic technologies are becoming popular and almost essential in many industrial sectors. They are used for different purposes as they provide huge benefits to companies and workers; for example, in industrial environments, inspection tasks have been increasingly automated. Plant inspection and maintenance are generally jobs susceptible to human errors as they require tedious and repetitive tasks; consequently, this may entail a decrease in the attention and focus of the employees, leading to errors that may have serious repercussions. In this context, those tasks can be accomplished by robots more safely and efficiently in terms of time and cost. This allows to deploy the human workforce in non-automatable activities from which the companies can benefit and to reduce the overall cost. Autonomous inspections have an important role in this thesis work which is part of the project carried out by Sprint Reply, the society of Reply group specialized in hyperautomation with focus on Robotic Process Automation, Computer vision & ICR, AI & Machine learning and Process mining. The project aims to develop a robotic-aided solution for a world leader company in Oil & Gas industry. The main objective is to detect and monitor the conditions of working machines by reading both analog and digital manometers located in one of the company's plants. To this purpose the quadruped canine-inspired robot from Boston Dynamics was employed. The latter is known as Spot and it is an agile mobile robot equipped with cameras and sensors located around its body; also, payloads both from the robot's mother company and third-party companies can be mounted on top of it to improve its performances. The robot has great stability and ability to adapt autonomously to height differences in the terrain. These features allow Spot to automate routine inspection tasks of any area of interest and to monitor different scenarios, detecting and recognizing specific objects and capturing data safely, accurately, and frequently. Two phases were involved in the project. The first phase included data collection, gathering and labelling, model training exploiting machine learning techniques, computer vision algorithms development and their packaging into Docker containers. Specifically, the data collection focused on taking pictures of target manometers located in three different areas of the Oil & Gas customer's plant. Some examples of the mentioned manometers are analog gauges, mechanical and digital water meters, digital indicators, expansion vessels, status indicator valves and liquid level gauges. The photos were gathered both using phone cameras and Spot PTZ, that is a camera of the Spot CAM+ payload provided with 30x optical zoom. Consequently, there were differences in camera quality and resolution that led to the building of a heterogeneous and varied dataset consisting of almost a thousand of photos. Subsequently, a labelling phase followed. This is an essential step to take before training the model because it allows artificial intelligence and machine learning algorithms to build an accurate understanding of real-world environments and conditions. For the purposes of this project, the labelling activity has been done

manually by five people exploiting the LabelImg tool. The latter is a tool for visual image annotations which saves annotations as XML files in PascalVOC format. This implies that for each image, the target meter has been enclosed in a bounding box and labelled with the proper class generating an XML file containing the details. Once the whole dataset has been labelled, the model training phase began. This is a fundamental step as the goal is to make the robot able to recognize autonomously the target objects located in the customer plant. To this aim, the use of the YOLO tool has been adopted. It is a real-time object detection algorithm and its main feature is the use of a single neural network to carry out both classification and prediction of the bounding boxes of the detected objects. Finally, the dataset has been split into training and validation sets and the model has been trained. Almost in parallel, computer vision algorithms development started. Computer vision in robotics is defined as the process of extraction, characterization and interpretation of information coming from images of a tridimensional world. Thus, in this project, the development of those algorithms played an important role. This activity is the one that required the most time and effort from me and my three thesis colleagues. In particular, given that for each of the classes obtained from the labelling phase there may be more than an occurrence, that is different models of meters that detect the same measure, it has been decided to develop a different algorithm for each occurrence; in addition, a json configuration file has been implemented for each occurrence to take small differences between similar components into account. This made it possible to reduce the number of different algorithms developed still considering all the cases. For the development, Python programming language has been adopted and the OpenCV and PyTorch libraries have been used; also, the pipeline was based on pre-processing, segmentation, description, recognition, and interpretation processes. The algorithms were equally divided between me and my colleagues: everyone developed two or three different codes. Specifically, the codes which I contributed to this project with are the 'round digital water meters' and the 'mechanical water meters'. Lastly, the algorithms have been packed into Docker containers and deployed in a custom webapp through which it was possible to control both the missions and the robot.
The second phase of the project was held on-site and involved mission configuration on Spot, testing, validation, and analysis of the obtained results. In addition, on an experimental level, Spot Arm has been joined to the robot already in use. Before configuring the missions, three patrol tours have been identified: MISS01, MISS02 and MISS03. For each of them, it has been decided the number of checkpoints that Spot had to take i.e., the number of stops for acquiring photos. Then, mission configuration on the robot started. Generally, two steps are involved while configuring an autonomous mission on Spot: mission recording and mission replay. Mission recording is the process by which the robot is instructed on the path to take and any eventual actions to perform. This can be done exploiting the Autowalk function in the Spot tablet. In particular, a qualified operator must first scan a QR-like code with Spot: this defines the robot's starting point; then, he moves the robot along the desired path and lastly, generally the Dock action is added to the mission so the robot can dock itself in its docking station to autonomously recharge the battery: this defines the

robot's ending point. In the case of this project, it has been decided to define loop paths, so the starting and the ending points coincide. One mission was recorded for each of the three defined patrol tours. Once all the missions have been recorded, they could be replayed. To this aim, it was decided to replay at least eight missions per day to get a large amount of data so that the results obtained could be more reliable. This marked the beginning of the testing phase in which the computer vision algorithms were revised and adjusted to reach the final objective. It was a month-long trial-and-error process in which the goal was to have AI-powered reading achieve ever higher success rates among the photos Spot acquired during the missions replay. During this period, we had to deal with a certain number of faults due to the robot, its firmware, the payloads, and the tablet. Specifically, during the mission replay, the robot had to walk near a wall and, even if the registration was rather smooth, the robot showed a strange behavior that can be described as zigzag. This was solved by replacing the robot with a newer one with an updated firmware. Also, the tablet often showed connection faults which led to a sudden stop of the mission when replayed in supervised mode. This was solved by replacing the tablet with a newer version. Lastly, the PTZ camera showed some random faults probably due to degraded connectivity to the robot. This was not solved during the on-site pilot, but it will be solved in the future by replacing the camera with a newer model. Experiments with Spot Arm began from the third week out of five of the on-site work. Specifically, it was used to open doors to the other robot while the latter was replaying the configured missions. Unfortunately, at the current stage, it is not possible to configure the door opening automatically by Spot except for few standardized door handles. With the latter feature, an operator needs to indicate, using the tablet, the location of the handle and of the hinge of the door and whether the robot must push or pull it. After this, Spot opens the door and enters inside. For the purposes of the project, it was needed that Spot Arm opens the door while the other robot enters inside. Therefore, an operator was essential to make the robot open and close the doors with its arm by controlling its motions via the tablet. This is how I contributed to the project on field. As concerns the validation of the results, a comparison between the human and artificial intelligence-based readings have been made. Specifically, to be validated, the result coming out from the AI-based reading needed to be exactly equal or to differ by less than a certain tolerance to the number read by the human eye on the meters of interest. This comparison has been done exploiting Excel's capabilities. Moreover, charts have been carried out to get easy to visualize data. From the results it emerges that the target meters reached a success rate of about 70%. This value was obtained first collecting the number of successes for each patrol tour and then dividing the sum of the number of successes of all the missions by the number of total AI-based readings. In conclusion, the success rate obtained is acceptable for the limited time available: the computer vision algorithms took only a few months to be developed and fine-tuned. At the end of the on-field work, the results were illustrated in a steering committee meeting with the customer's representatives who seemed satisfied with the work, although there is room for improvement.

1. Introduction

1.1 Problem

In engineering activities inspection involves the measurements, tests, and gauges applied to certain characteristics regarding an object or activity. The results are usually compared to specified requirements and standards for determining whether the item or activity is in line with these targets. Inspections are usually non-destructive and nowadays they may be:

- a visual inspection or involve sensing technologies.
- accomplished with a direct physical presence or remotely.
- manually or automatically operated.

Over the past decades, inspection was an entirely human based activity. This involved an increase in the number of personnel and in the time taken to perform this task. With the advent of robotics, inspection became more and more an automated operation. From companies' side, automating the process exploiting robotic workforce allow to dramatically decrease the number of people employed in inspection tasks and, consequently, the overall cost. Another advantage is given by the quality of the results the robots can provide: often it is the same or better than the results carried out by the human workforce. For this reason, the latter can mostly be deployed to perform more complex and profitable tasks for the company. It is important to note that, at this stage, robots do not totally substitute the human workforce, instead they work together to deliver better results. This human-robot collaboration is a must since controlling the robots is challenging due to the variation and unpredictability of the environment.

1.2 State of art

The use of robotics to deal with autonomous inspections has become a topic of increasing interest both for academics and companies. In the oil and gas industry, recent major incidents such as the tragic industrial disaster happened in Gulf of Mexico in 2010 [1], have accelerated the importance of safety and reliability of structures and machinery. For this reason, this industrial sector operates under the most stringent standards requiring consistent and accurate inspections.
So far, most inspection processes are performed manually by qualified operators. The process is subjective and the operators need to face very uncomfortable and even dangerous conditions such as dust environments, absence of light or toxic substance exposition. As concerns the oil and gas industry, the main inspection methods used are visual inspections, factory and machinery inspections, final random inspections

(FRI), pre-shipment inspections and acoustic emission testing [2]. Visual inspections are a cost-effective test method used to inspect equipment for flaws and are often the first inspection in the oil and gas industry. An expert technician visits the site and proceeds to detect any defects that can be detected by the naked eye or with a borescope. These include structural failure, welding flaws, corrosion, or cracks. Using basic condition monitoring and corrective measures, hazardous situations can be prevented before they occur. A main disadvantage of this type of inspection is operator's distraction and human error in general. Another inspection method is factory and machinery inspection that involves rigorous testing of equipment within factories, ensuring all equipment is free of faults. As far as FRI type of inspection is concerned, it is essentially used to witness and evaluate all physical and chemical tests. The inspections are done to ensure the conformity of product packaging, color codes, and the packaging process. All of these are reviewed and tested to meet client's requirements. Once the results of the tests are evaluated, a report is issued. Moreover, a certificate is issued if the report concludes the inspection was a success. A similar inspection method is pre-shipment inspection which aims to ensure all goods are safe and within regulations before shipment occurs. Lastly, acoustic emission testing inspection method is mainly used to detect the rarefaction waves associated with leaks in a pipeline.
Over the last years, with the increasing demand for energy, the oil and gas industry market raised the demand of inspection robots, as shown by the forecast in the figure below (Fig. 1).
Since most of oil and gas assets are located in extreme environments, adopting robotic technology provides an alternative way to perform quality inspection tasks collecting different types of data while ensuring efficiency, productivity, and safety at a reduced cost. This gives rise to autonomous or semi-autonomous inspections.

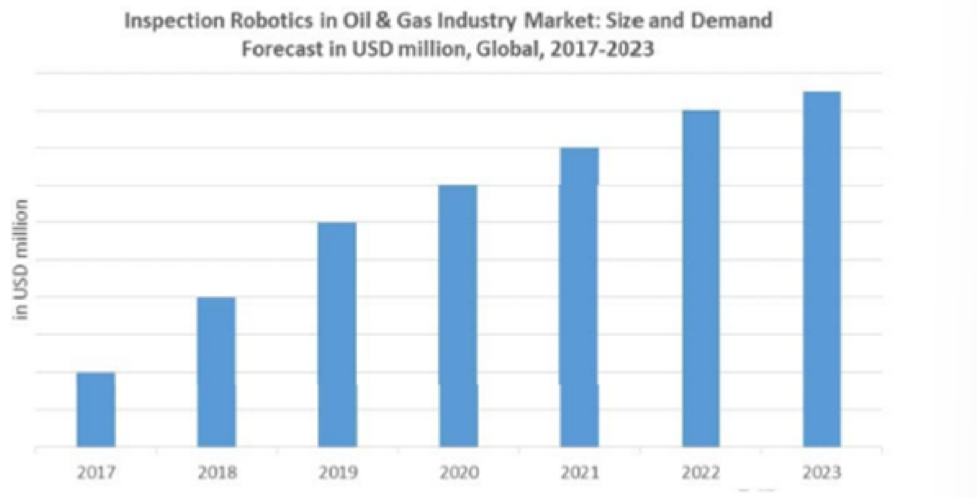


Fig. 1: size and demand of inspection robots in Oil & Gas industry.

According to vehicle type, mechanisms or structures, the market of inspection robots in the oil and gas industry can be divided into the following categories: remotely operated vehicles (ROVs), autonomous underwater vehicles (AUVs), unmanned ground vehicles (UGVs), and unmanned aerial vehicles (UAVs) [3]. Some of them are meant to examine oil storage tanks while others are designed to inspect pipelines. In any case, most of these inspection robots require qualified operators to manipulate them throughout the inspection process. Thus, the aforementioned non-autonomous inspection methods can be replaced by semi-autonomous robotic inspection. In this phase human-robot collaboration is necessary as, at the current stage, controlling robots is challenging due to the variation and unpredictability of the environment. Autonomous localization and planning, autonomous decision-making, including safety and even ethical decisions, and verification and validation testing to ensure robot's safety and reliability are some of the topics still under research [4]. In the near future, hopefully, it could be possible to make use of fully autonomous inspection robots.

1.3 Hard and soft robotics

1.3.1 Hard robotics

Generally, robots can be classified as hard or soft on the basis of the compliance of their underlying materials [5]. As noted in the report [6], traditional robots can be made of a number of rigid links connected to each other with single degree of freedom joints, as hyper-redundant manipulators or invertebrate-like robotic topologies (Fig. 2), where each joint is controlled independently by the actuators to realize a task or purpose. This is a classical approach that has been used in many robotic designs and is commonly known as hard robotics. This approach requires intricate algorithms to control the position of each link and of the whole robot and/or to control the contact force during the physical interaction and the interface of the robot with its environment. Although all the difficulties in developing hard robots, they are still commonly employed since they ensure high accuracy, high speed, and force application [5]; in addition, thanks to the rigid materials of which they are made, hard robots are more damage resistant. Some examples include conventional manipulators, humanoid robots, legged robots, and wheeled robots. Conventional manipulators consist mainly of rigid links and are employed for object manipulation that can occur with the use of specialized end-effectors. Usually, they come with a fixed base, therefore the robot's working area is limited to the environment surrounding this base. In contrast, mobile robots are able to move around the environment so they can obviously operate in a larger working area. A rough classification of the latter type of robots includes legged robots, tracked robots, wheeled robots and aerial robots (Fig. 3); a detailed analysis on this topic will be presented later.

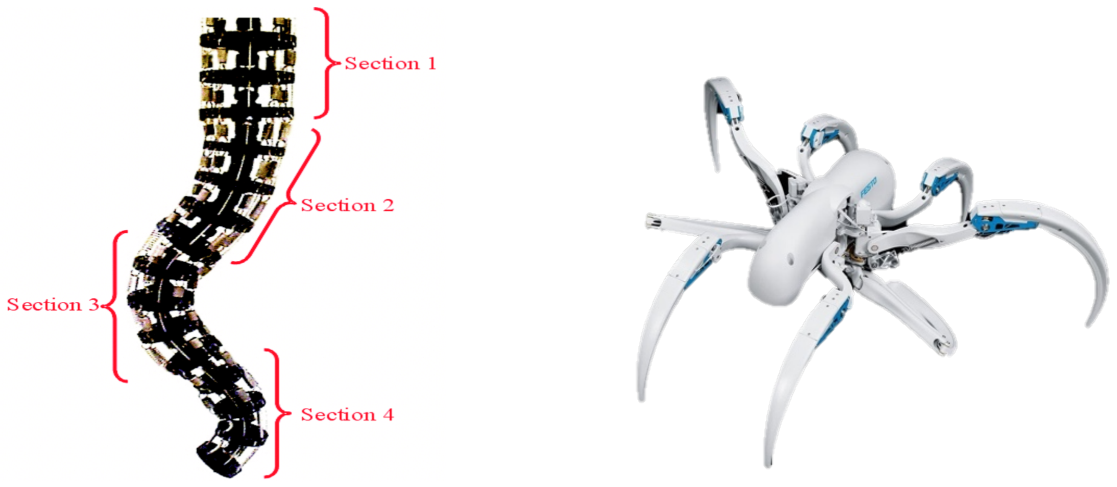


Fig. 2: example of a hyper-redundant manipulator (left) and of an invertebrate-like robot (right).

Fig. 3: examples of mobile robots.

1.3.2 Soft robotics

Recently biology has inspired researchers to design and build what are known in literature as soft robots, giving birth to another branch of robotics: soft robotics. The name comes from the soft structure of which these robots are made. This characteristic together with redundant degrees of freedom feature allow robots to offer unprecedented solutions for applications involving smooth touches, safe interaction with humans, adaptability to wearable devices, and simple gripping system [7]. Furthermore, different form of soft robots is available depending on the application; some examples are robot manipulators, grippers, medical robots, agricultural robots, rehabilitation robots and similar (Fig. 4). In line with recent progress in soft smart materials or electro-materials and additive manufacturing techniques, these robots consist of a monolithic body containing actuation and sensing elements, mechanical structure, energy storage units with a minimum footprint. Such robots are expected to change their effective stiffness in order to provide a desired force or compliance when interacting with the environment including physical interaction with humans.

As regard the actuation systems used in soft robotics, the three major types are: variable length tendon, fluidic actuation, and electro-active polymer (EAP). Besides, a remark must be made concerning the deformable property of these robots as it restricts the use of many conventional rigid sensors such as encoders, strain gauges, or inertial measurement units. Thus, contactless approaches for sensing and/or sensors with low modulus are preferable for soft robots [7]. In conclusions, the two mentioned branches of robotics mainly differ from materials, control, and applications; both have interesting features and some limitations, so a detailed analysis is necessary in order to decide which type of robot suits best for a specific application.

Fig. 4: examples of soft robots.

1.4 Mobile robots

Nowadays, mobile robotics is one of the fastest expanding fields of scientific research. A primary distinction of mobile robots is based on their ability to move. In fact, they can move semi-autonomously, that is with assistance from a qualified operator, or autonomously where there is no need of external operators. This ability together with the intelligence to react and make decisions based on the perception system received from the environment is what distinguishes autonomous mobile robots with respect to other robots [8]. Furthermore, in order to respond to a changing world, these robots necessitate of some source of input data, some way of input decoding, and a way of taking actions including their own motion. In addition, a powerful cognition unit or a control system is required to coordinate all the subsystems that comprise the robot and to make it adapt to an unknown environment.
The basics of mobile robotics consist of the fields of locomotion, perception, cognition, and navigation [9]. Locomotion problems are solved by understanding the mechanism and kinematics, dynamics, and control theory. Perception involves the areas of signal analysis and specialized fields such as computer vision and sensor technologies. Cognition is responsible for analyzing the input data from sensors and taking the corresponding actions to achieve the objectives of the mobile robot. Navigation requires knowledge of planning algorithms, information theory, and artificial intelligence. For the correct operation of the mobile robot, the locomotion system and kinematics, the perception system (sensors), the localization system, and the

navigation system have to be integrated by a control unit so that the work or mission of the mobile robot is carried out in a coherent way.

1.4.1 Locomotion

Although the motion of mobile robots usually takes place in known, controlled environments, on other occasions they have to move in dangerous, inhospitable, and extreme environments. An example is the Sojourner robot used in the Mars Pathfinder mission to explore Mars in 1997 [10]. The robot's locomotion system is an important aspect to consider for a correct design of the robot itself. It is therefore necessary to make a preliminary analysis of the medium in which the robot moves (in the air, under the water, on the Earth's surface, etc.) but also to consider technical criteria such as maneuverability, controllability, ground conditions, efficiency, and stability. Depending on this, robots may be able to walk, jump, run, slide, swim, and fly.

According to their locomotion system, mobile robots can be classified into the following major categories:

- stationary such as arms and manipulators.
- land-based such as wheeled robots and legged robots.
- air-based that is aerial robots (UAV).
- water-based.
- other.

In the following a more-in-dept description for each category will be presented.

1.4.1.1 Arms and manipulators

A manipulator is a device used to manipulate materials without direct physical contact by the operator. The applications were originally for dealing with radioactive or biohazardous materials or they were used in inaccessible places. In more recent developments they have been used in diverse range of applications including welding automation, robotic surgery, and in space. It is an arm-like mechanism that consists of a series of segments, usually sliding or jointed called cross-slides, which grasp and move objects with a number of degrees of freedom. These robots usually come with a fixed base which allow the manipulator to move in a controlled and known environment; for this reason, they are categorized as stationary robots. There exist also manipulators placed at the top of mobile robots such as wheeled and legged robots (Fig. 5). In this case, the environment is not always known and controlled, therefore more attention must be paid in controlling and moving the robot.

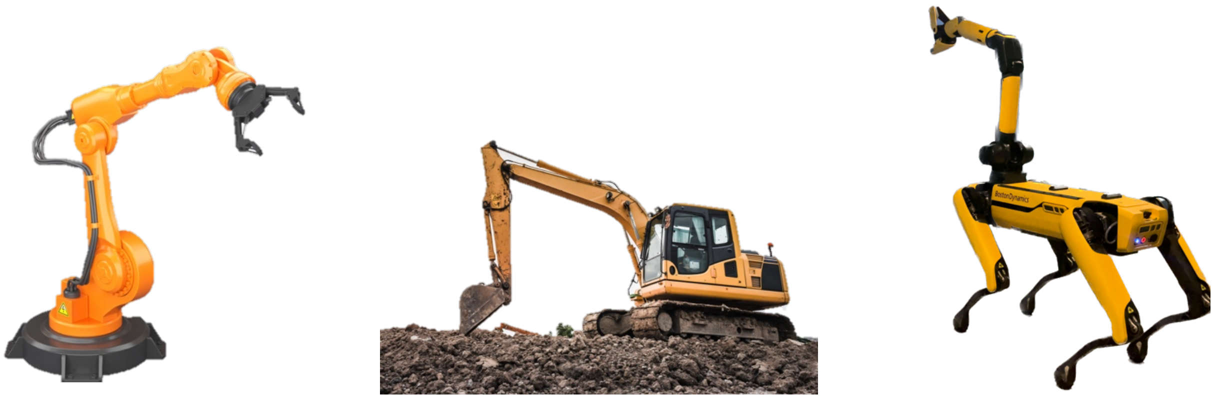


Fig. 5: examples of manipulators with fixed and mobile base.

1.4.1.2 Legged robots

Legged robots are a type of mobile robots which use articulated limbs, such as leg mechanism, to provide locomotion. These robots, compared to wheeled robots, are more versatile and can traverse many different terrains but these advantages require increased complexity and power consumption. Legged robots often look like legged animals; some examples are Festo’s kangaroo robot and Cyberdog, Xiaomi’s robotic dog (Fig. 6). These robots are categorized by the number of limbs they use, which determines the different gaits available. Obviously, many-legged robots tend to be more stable, while fewer legs lend itself to greater maneuverability. There exist also hybrid robots which use a combination of legs and wheels; an example is Handle from Boston Dynamics (Fig. 7), a bipedal robot with wheels on both legs.

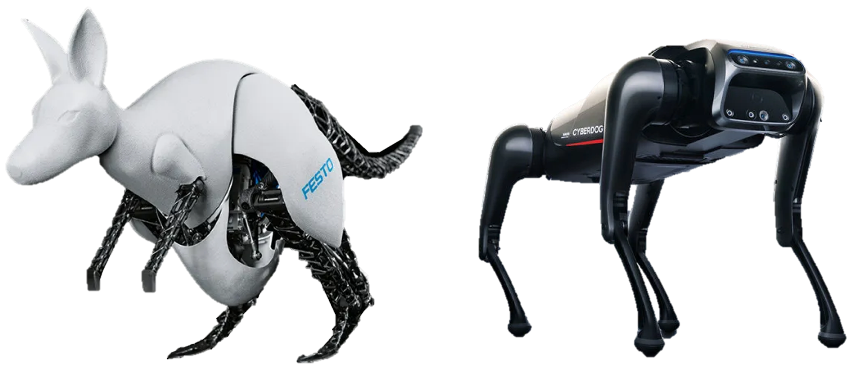


Fig. 6: examples of legged robots.

Fig. 7: Handle from Boston Dynamics.

1.4.1.3 Wheeled robots

Wheeled robots are robots that navigate around the ground using motorized wheels to propel themselves. Compared to robots with treads or legs, the use of wheels makes this type of robots easier to design, build, and program for movement in flat terrain. They are also more well controlled than other types of robots. Disadvantages of wheeled robots are that they are not able to navigate well over obstacles such as rocky terrain, sharp declines, or areas with low friction. Wheeled robots can be classified based on the type and the number of wheels. Depending on the application, a variety of wheels are already available: simple non steering wheels, simple steering wheels, castor wheels, omniwheels, spherical omniwheels, cylindrical wheel (Fig. 8). In particular, the differential steering of wheeled robots provides low cost and simplicity and, for this reason, they are the most popular among the consumer market. Robots can have any number of wheels, but three wheels are sufficient for static and dynamic balance. Additional wheels can be added to balance; however, additional mechanisms will be required to keep all the wheels in the ground when the terrain is not flat.

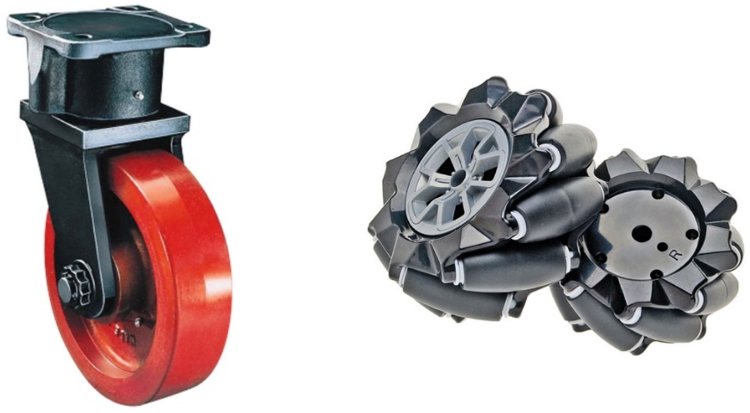


Fig. 8: example of a castor wheel (left) and of omniwheels (right).

1.4.1.4 Tracked robots

The distinctive feature of tracked robots is the use of treads or caterpillar tracks instead of wheels (Fig. 9). From a broader perspective, tracks refer to sets of linkages which are connected to drive gears, wheels, rollers, or sprockets, allowing them to run a certain length along the robot's chassis in a similar manner to a conveyor belt. Tracks can vary widely in design: they are usually made of metal or rubber and they feature grooves, treads or screws built into them to provide traction along the floor.
An important distinction concerns tracks and simple drive chains. In particular, the latter merely provide drive power by connecting the motor output shaft to the robots' wheels or gears. Numerous advantages come from the use of tracked robots. For example, the use of tracks gives much more traction on the floor than wheels, offering better acceleration. Generally, tracked robots have larger ground contact patches and this fact plays a major role to improve their maneuverability on loose surface in comparison to conventional wheeled robots. Another advantage comes from balance: tracks offer better balance than wheels as a result of their length and larger mass. As regards the disadvantages of this typology of robots, because of the large ground contact patch of which these robots are provided, it is meaningful to mention the difficulty to predict the exact robot's center of rotation and to determine accurately the change in position and direction which is subject to variation on account of the ground friction. Consequently, dead reckoning in tracked robots is very imprecise.

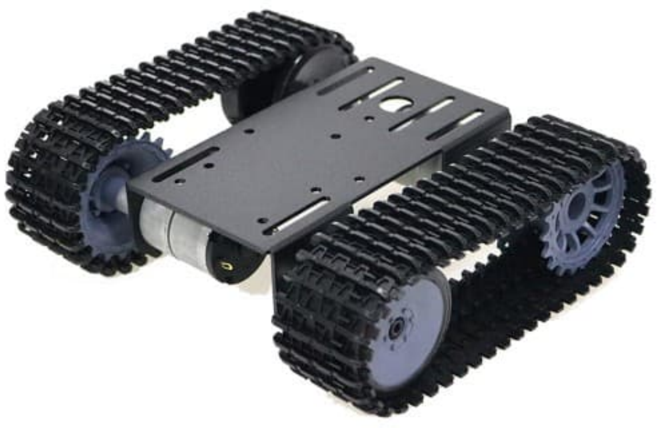


Fig. 9: example of a tracked robot.

1.4.1.5 Aerial robots

Robots belonging to this category are the ones that exploit air as their medium. A well-known flying robot is the modern passenger airliner which requires two humans to manage it. Other flying robots are uninhabited and are known as unmanned aerial vehicles (UAVs), commonly known as drones. They can be smaller and lighter without a human pilot on board and they can fly into dangerous territory for example for military surveillance missions. The flight of UAVs may operate under remote control by a human operator or with various degrees of autonomy up to fully autonomous aircraft that have no provision for human intervention. UAVs were originally developed through the twentieth century for military missions too dull, dirty, or dangerous for humans and then they had become essential assets to most militaries. As control technologies improved and costs fell, their use expanded to many non-military applications. These include forest fire monitoring, aerial photography, product deliveries, agriculture, policing and surveillance, infrastructure inspections, entertainment, science, and drone racing. Some examples are reported in the figure below (Fig. 10).

Fig. 10: examples of aerial robots.

A complete and detailed discussion on this topic can be find in the report [9].

1.4.2 Perception

Besides locomotion, another important aspect while designing autonomous mobile robots is perception since it helps the robot to acquire knowledge about its work environment and itself. This is achieved by means of sensors from which measurements of the relevant information are subsequently extracted.
The use of sensors in robotic applications make it possible to perform different tasks like robot positioning and localization [11], mapping and representation [12], object recognition [13] as well as the simplest obstacle avoidance during robot navigation [14]. From a broader perspective, sensors can also be seen as the robot control system's window to the world since no mobile robot can work without these devices and their software. Fields of recent expansion such as drones and automated driverless cars could not exist without sensors. Moreover, the latest advances in

sensoring and artificial intelligence are being used in speech recognition systems, which are very important to reproduce human capabilities.

1.4.3 Cognition and control system

An important aspect in mobile robot design regards the control of the mechanical structure of the robot itself which is needed to perform tasks and achieve specific objectives. It is thus necessary to combine the hardware and software parts of the robot. This can be done exploiting the control system which is responsible for coordinating all the input data and planning the robot's motion. The control system can be generally divided into three different pillars: perception, processing and cognition, and action. Once the information is received from the perception system (i.e., information about the environment, the robot itself and the relationship between them) it is then processed. Subsequently, in order to move the mechanical structure, the appropriate commands are sent to the actuators. Once the robot's direction, destination or purpose and the environment are known, the cognitive architecture of the robot must plan the path that the robot must take to achieve its objectives. Therefore, the cognitive level of the robot represents the decision-making and execution part that the robot uses to achieve high-level objectives [8]. In conclusion, based on the information got from the sensors and the robot's objectives, the cognition and control system must decide how to act and what to do to meet its goals. Moreover, the robot may need a "cognitive" model that is intended to represent the robot itself, the environment, and the manner in which they interact. For example, computer vision and pattern recognition are used to track objects [15] while mapping algorithms are used to build maps of the environment [16]. Eventually, motion planning and other artificial intelligence algorithms might be used to determine how the robot should interact.

1.4.4 Navigation

The most important aspect in the design of a mobile robot is its navigation skills. The objective for the robot is to move from one point to another in a known or unknown environment, taking into consideration the data coming from the sensors to reach the desired target point. To this purpose, the robot must rely on its other systems, that are:

- perception, in which the robot uses its sensors to obtain valuable data.
- localization, through which the robot knows its position and configuration.
- cognition, that is used by the robot to decide what to do to achieve its goals.
- motion control, by which the robot calculates its input forces on the actuators to achieve the desired trajectory.

An observation must be made on the path performed by the robot. In fact, in most of the time, robots cannot move straightforward from the starting point to the end point for various reasons: presence of objects along the way, structural conformation of the environment in which the robot is located (doors, walls, stairs, ...) and so on. For these reasons, motion planning techniques must be used. To this regard, it is possible to observe that mobile robot navigation can be subdivided into the following tasks:

- generating a model of the world in the form of a map.
- computing a collision-free trajectory from starting position to target position.
- moving along the calculated trajectory avoiding collision with obstacles.

The combination of these tasks allows the robot to safely reach the target position where, eventually, it can start to interact with the environment executing specified commands.

1.5 Quadruped robots

Quadruped robots belong to the category of legged mobile robots, specifically each robot is made of four legs that make it able to move. Quadruped robots are the best choice among all the legged robots as concern mobility and stability of locomotion. The four legs of the robot are easily controlled, designed, and maintained as compared to two or six legs [17]. In the early 1900s, many scientists and researchers were devoting their time to study the leg mechanism of the four-footed robot. Chebyshev developed the first walking mechanism in 1870, which primarily converts rotational motion to translation motion with constant velocity based on the four-bar mechanism, as shown in the figure below (Fig. 11). This device could walk dynamically on flat terrain only and does not have an independent leg motion. Nowadays, a lot of progress has been made on quadruped robots and various applications involve their use such as inspection, surveying in construction sites, delivery, security and monitoring, search, and rescue. One of the most recent quadruped robots is Spot from Boston Dynamics (Fig. 12). Among its distinctive features there are its great stability and its ability to walk despite changes in type, slope, and height in the terrain.

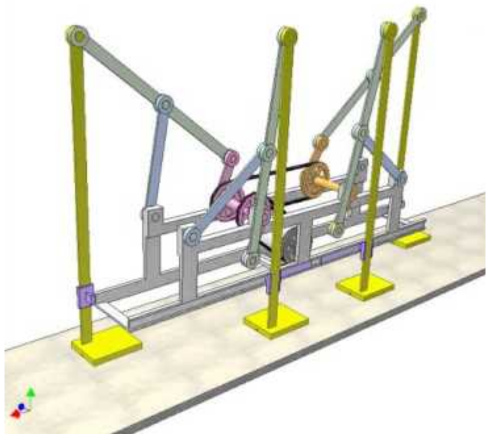


Fig. 11: Chebyshev's mechanism: the animation link is reported in [19].

Fig. 12: Spot from Boston Dynamics.

1.6 Spot

From 1992, Boston Dynamics started developing highly dynamic quadruped robots. After years of work, the company released different models of the four-legged canine-inspired robot known as Spot. In 2017 Spot Enterprise made its first appearance and then, in 2021, the robot was provided with a manipulator placed at the top of it, giving birth to the newest model known as Spot Arm.

1.6.1 Spot Enterprise

Spot Enterprise is one of the basic models of the quadruped robot and it is mainly used for inspection and patrolling. Some hardware peculiarities of Spot are its stability, its ability to avoid obstacles and to recognize differences in height (e.g., steps) and climb them. To better describe its general features, some macro categories are identified [19]:

- dimensions
- environment
- power
- cameras
- payload
- connectivity

1.6.1.1 Dimensions

From a dimensions' perspective, the robot is 1100 mm long and 500 mm wide while different considerations can be made about its height depending on how bent the robot's leg joints are. Specifically, Spot's height can span between 520 and 700 mm while walking (default walking height is 610 mm) and it reaches a height of 191 mm while in sitting position. Concerning other features of the robot, it is important to note that it has 12 degrees of freedom and it can reach up to 1.6 m/s of speed.

1.6.1.2 Environment

As concerns the environment, the properties that need a major attention regard the ingress protection IP, the temperature of operation, the maximum step height, and the slopes the robot can cross. In particular, Spot belongs to IP54 ingress protection category; this can be a limit for certain outdoor applications since it means that the robot can handle at most light rain and, consequently, it cannot be used when the weather conditions are bad (heavy rain, snow, etc.). Another feature of Spot is related to the operating temperature that spans between -20° C to +45° C. This allows a lot of applications both indoor and outdoor, but it can be also seen as a limitation since in certain country they reach colder temperatures or in some industrial indoor environment it is possible to reach hotter temperatures. An interesting capability of Spot is that it is autonomously able to cross and handle changes in the type of terrain as concerns different heights. In particular, the maximum step height it can cross is equal to 300 mm while the slopes are in a range between – 30 and + 30 degrees.

1.6.1.3 Power

Spot comes along with a plug-in battery which can be inserted from below i.e., from the belly of the robot; the battery weighs about 5 kg and has a typical runtime of 90 minutes, but the effective duration of its charge depends on the tasks in which Spot is employed. The battery of the robot can be charged stand-alone or while inside the robot:

- using the Spot charger and plug the battery into the robot in a second moment.
- from inside the robot using the shore power cable available with the Spot charger.
- from inside the robot exploiting the Dock station provided by its mother company.

The Spot charger is a case provided together with few cables (Fig. 13). The number of cables depends on the model of the charger: basically, there are a power cable and a shore power cable; if the charger is a legacy model, it has additionally a battery charging cable, otherwise there is a battery charging trail that is a slot in which the battery can be plugged in and powered. The power cable is used to supply the case with current while the shore cable is used to charge the Spot battery while it is inside the robot; this cable is connected on one end to the case and on the other end to the robot. In the legacy model the battery charging cable is used to connect the case with the battery in order to charge it (Fig. 14); in the other charger models, it is sufficient to place the battery inside the charging trail slot to charge it (Fig. 15).

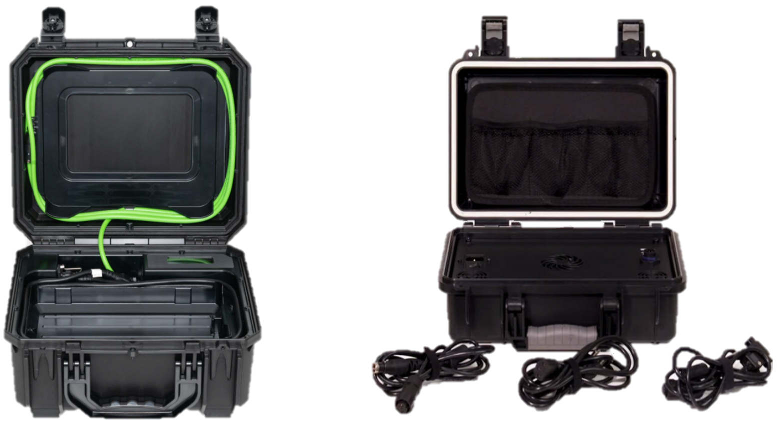


Fig. 13: Spot charger models: model with battery trail (left) and legacy model (right).

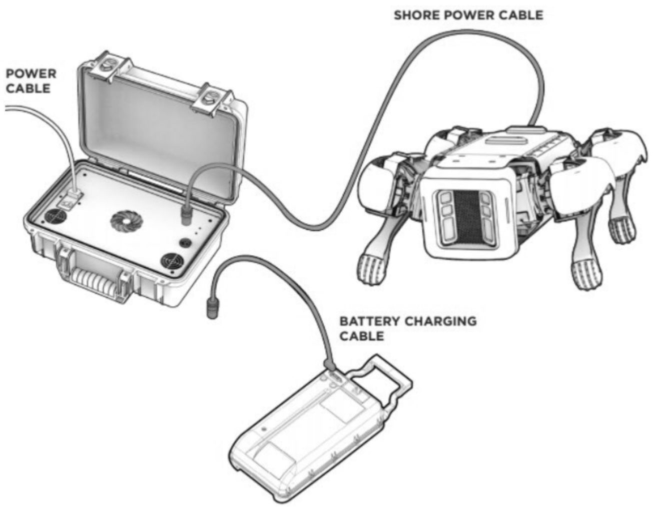


Fig. 14: two ways of charging the Spot battery with legacy charger model.

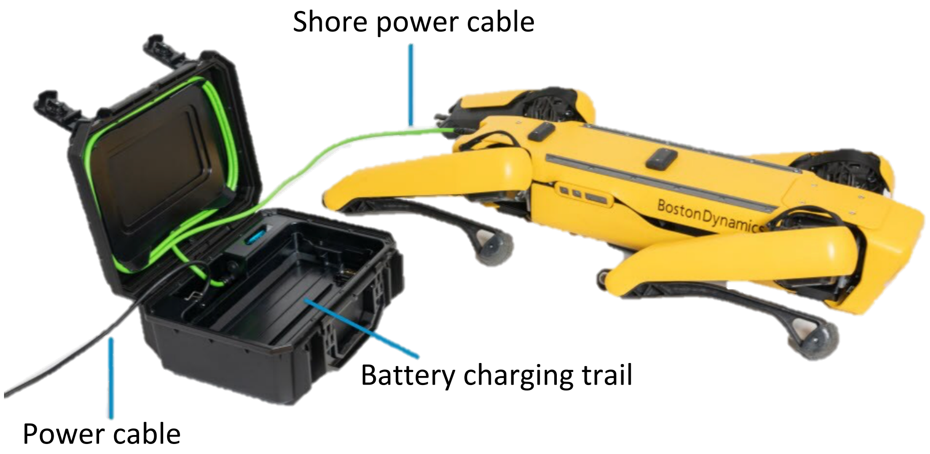


Fig. 15: two ways of charging the Spot battery with other charger models.

An alternative method through which is possible to charge the Spot battery while inside the robot is exploiting the Dock station (Fig. 16). The main difference between the two methods of charging is in the time needed to charge the battery, that is 2 hours for full charge (100%) with the Dock station while it is needed from 1 to 2 hours by using the Spot charger (1 for legacy model and 2 for the others).

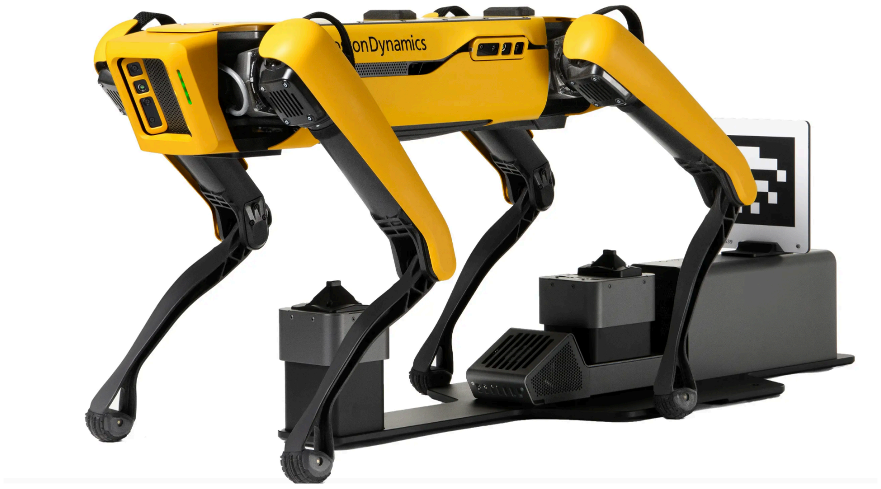


Fig. 16: Spot battery charging exploiting docking station.

1.6.1.4 Cameras and sensors

Spot is provided of cameras and sensors all over its body (Fig. 17). Different camera functions are available: black and white, color fisheye, range (depth), infrared. The robot's perception system consists of five stereo cameras with a 360° field of view and a detection range up to 4 meters. Exploiting this system, the robot is able to avoid obstacles and collision with them.

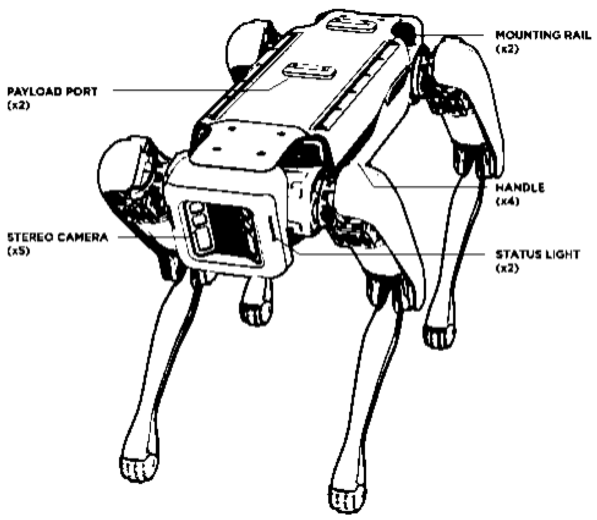


Fig. 17: Spot cameras and sensors.

1.6.1.5 Payload

Different payloads can be mounted on Spot and they can be both from the robot's mother company and/or from third-party companies [19]. Notice that the maximum weight Spot can carry is 14 kg, so this aspect must be kept in mind while mounting payloads on the top of the robot. As concerns the payloads, the ones from Boston Dynamics worth mentioning are Spot CAM+, Spot CAM+IR and Spot Core I/O. The Spot CAM+ payload turns Spot into a powerful inspection tool with purpose-built cameras (Fig. 18). It can be used to get eyes on remote or hazardous environments while focusing on inspection details that matter. Some main features are:

- spherical camera (360 x 170° view).
- Pan-Tilt-Zoom (PTZ) camera with 30x optical zoom.
- two speakers and microphones enable two-way audio.
- four pairs of LEDs provide illumination in dark environments.
- roll cage for crash protection.
- USB port available for saving image data.
- protected cabling and sealed electronics.
- configuration options for front or rear mounting.

Moreover, it has an ingress protection IP65 that is a bit higher compared to the one of the overall robot (IP54).

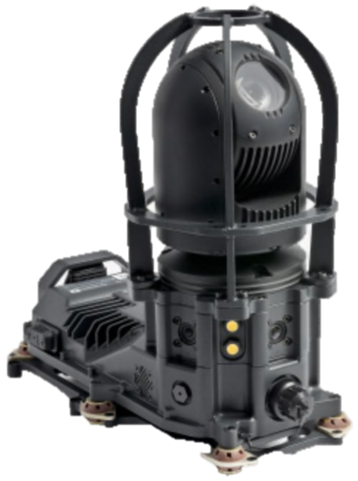


Fig. 18: Spot CAM+ payload.

To the basic version of the Spot CAM+ a thermal camera can be added to enable detailed thermal and visual inspections (Spot CAM+IR), as shown in the figure below (Fig. 19).

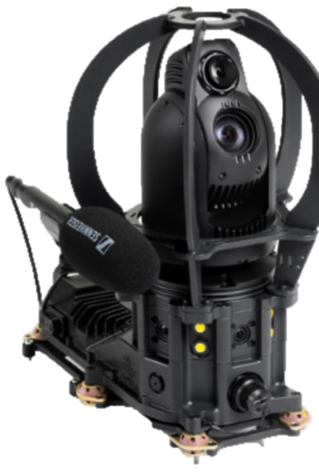


Fig. 19: Spot CAM+IR payload.

As regards the Spot Core I/O (Fig. 20), it is used to enhance both the computation and communications available on the Spot platform. It connects sensors, cameras, and other devices to Spot, process the data collected into actionable insights and relay those insights over 5G/LTE. Some main features are:

- compact CPU and GPU with customizable inputs and outputs.
- 5, 12 and 24 V regulated power output.
- RJ45 standard ethernet adapter.
- easy cable sealing to maintain IP54 rating.
- built-in 5G/LTE modem with CBRS support for private networks and option to use AT&T's public network.
- option to add lidar for enhanced autonomy.
- comes with sample computer-vision model.

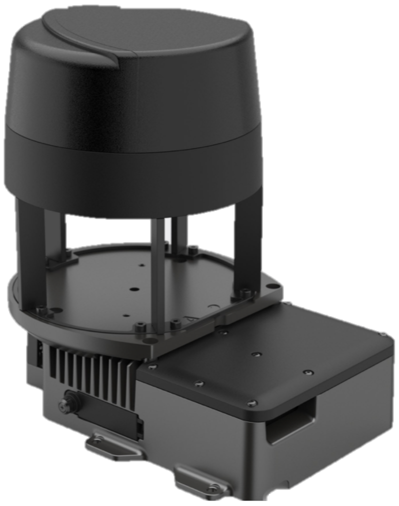
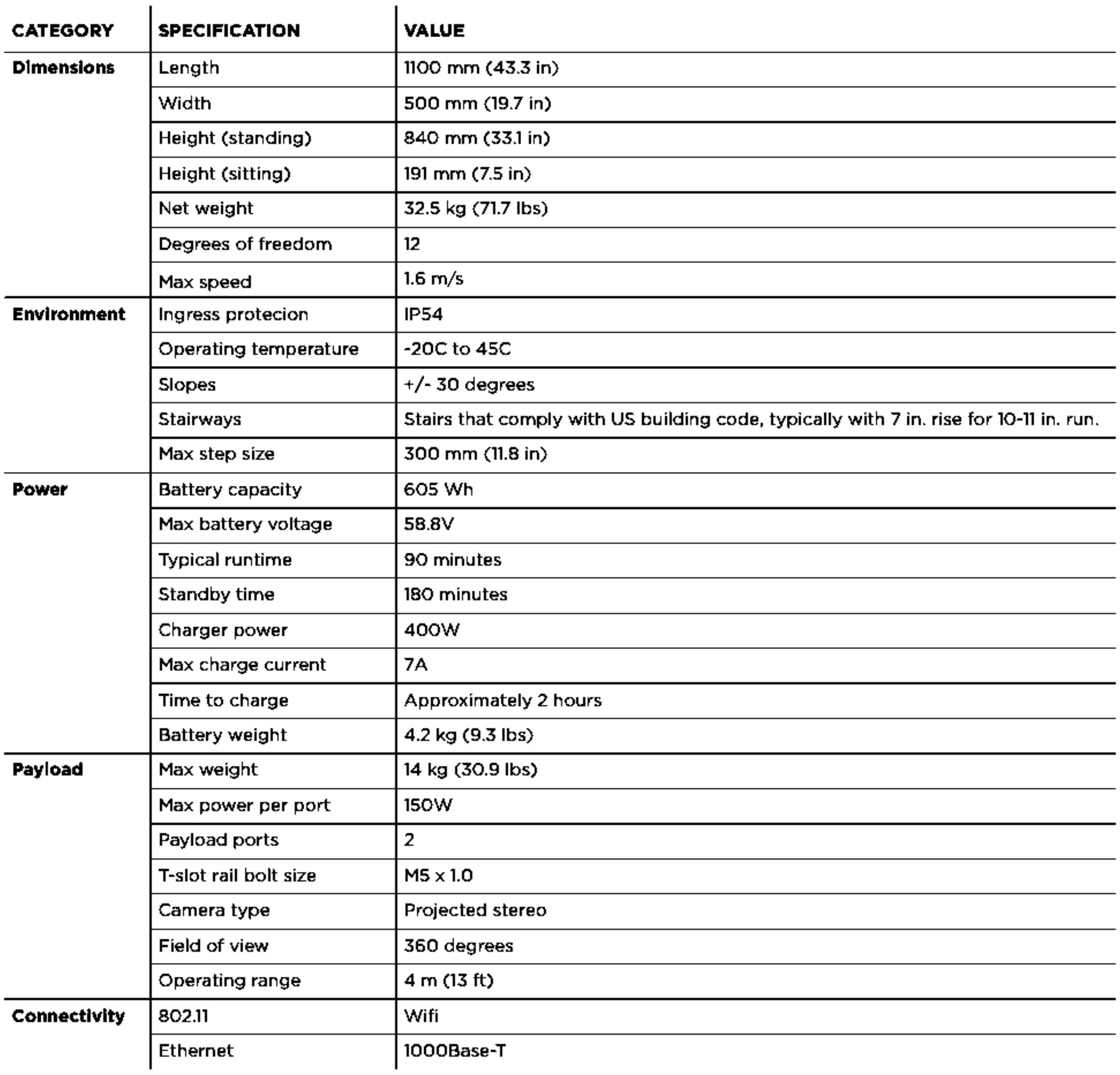


Fig. 20: Spot Core I/O payload.

1.6.1.6 Connectivity

As concern connectivity, Spot is provided by WI-FI that is used to teleoperate the robot via tablet and to collect data in the cloud in order to process them in a second moment. Other connections that the robot can host are 4G and 5G. In this regard, a third-party modem can be added to the robot as a payload and used to send data to the cloud infrastructures exploiting 4G or 5G technologies in the case in which the place where Spot is employed is not provided of a strong Wi-Fi connection.

A detailed overview of Spot's general features divided by macro categories is reported in the table below [Table 1][20].



CATEGORY	SPECIFICATION	VALUE
Dimensions	Length	1100 mm (43.3 in)
	Width	500 mm (19.7 in)
	Height (standing)	840 mm (33.1 in)
	Height (sitting)	191 mm (7.5 in)
	Net weight	32.5 kg (71.7 lbs)
	Degrees of freedom	12
	Max speed	1.6 m/s
Environment	Ingress protecion	IP54
	Operating temperature	-20C to 45C
	Slopes	+/- 30 degrees
	Stairways	Stairs that comply with US building code, typically with 7 in. rise for 10-11 in. run.
	Max step size	300 mm (11.8 in)
Power	Battery capacity	605 Wh
	Max battery voltage	58.8V
	Typical runtime	90 minutes
	Standby time	180 minutes
	Charger power	400W
	Max charge current	7A
	Time to charge	Approximately 2 hours
	Battery weight	4.2 kg (9.3 lbs)
Payload	Max weight	14 kg (30.9 lbs)
	Max power per port	150W
	Payload ports	2
	T-slot rail bolt size	M5 x 1.0
	Camera type	Projected stereo
	Field of view	360 degrees
	Operating range	4 m (13 ft)
Connectivity	802.11	Wifi
	Ethernet	1000Base-T

Table 1: Spot's general features divided by category.

1.6.2 Spot Arm

Spot Arm is the quadruped robot model with an arm fixed on top of it. This model is useful to inspect objects that are located in positions that Spot Enterprise is not able to reach and to physically interact with the environment. This latter functionality is not available during autonomous missions: the robot cannot interact autonomously with the environment yet, but only operated via tablet by a qualified human operator or exploiting custom code realized with the use of the Software Development Kit provided by the mother company.
Spot Arm has all the features mentioned previously with Spot Enterprise apart for the arm, so in the following only the characteristics of the arm are reported.

1.6.2.1 Arm general specifications

The arm can be seen as a standard manipulator with 6 degrees of freedom and a gripper, as depicted in the figure below (Fig. 21). As concerns the dimensions, the arm is 984 mm long at full extension and weighs 8 kg including the gripper. The maximum endpoint speed, lift capacity, and drag capacity are respectively: 10 m/s, 11 kg, and 25 kg. Moreover, with the arm extended above, Spot can reach a height of about 1800 mm and a weight of 39.7 kg. Table 2 shows a complete overview about arm's specifications. As regards the gripper, details are reported in Table 3. Notice that the hand position and orientation can be controlled independently of the body. In fact, the hand can be locked to the body or to the world. The first configuration can be selected on the tablet by the operator when it is desired to maintain the arm's position relative to the robot while walking it to a different location. Conversely, selecting the other configuration, the position of the gripping head is fixed with respect to the world, therefore the robot body moves around it. As the body moves, the arm automatically adjusts to compensate for the robot's body motion and it is able to keep the hand positioned in space.

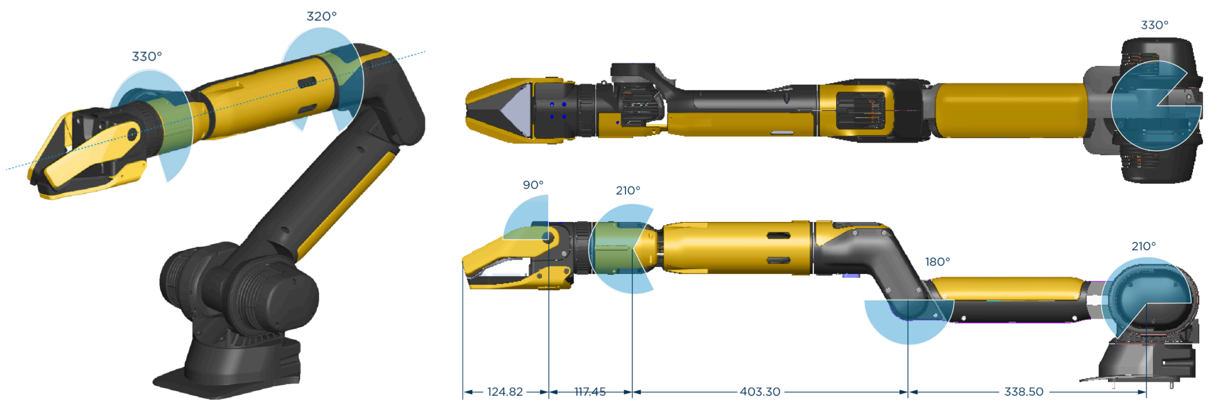
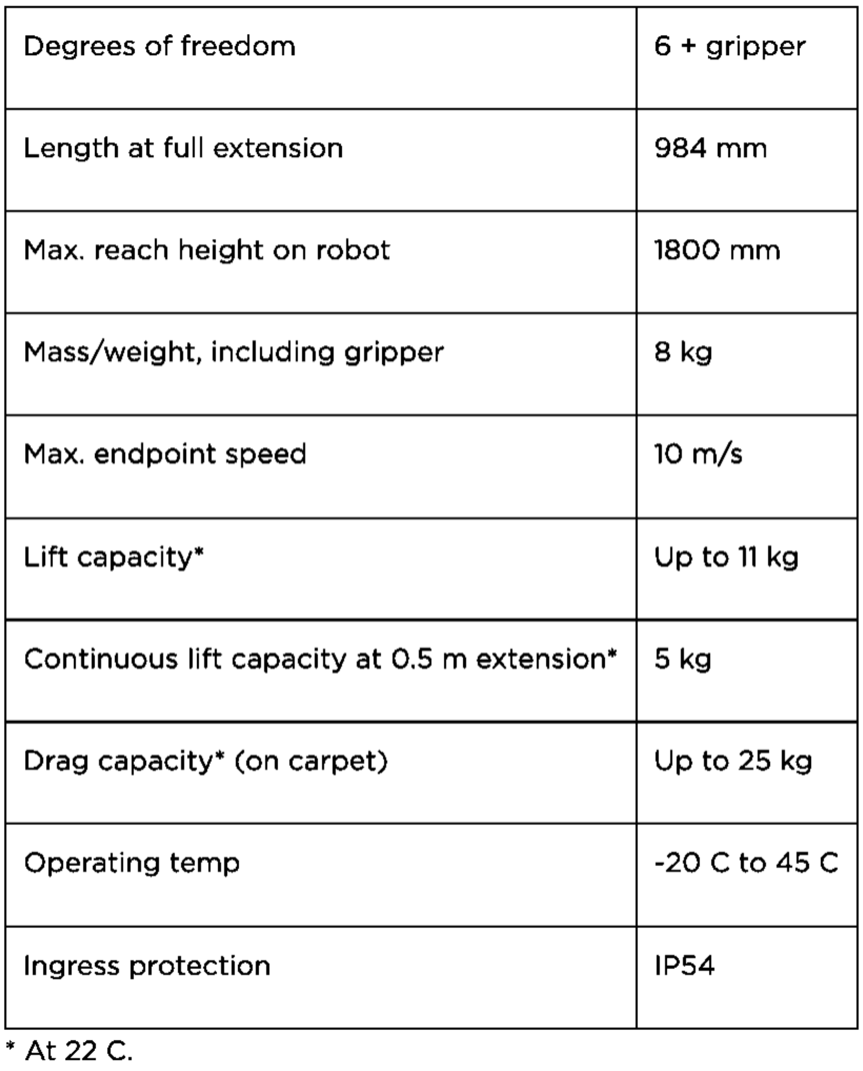


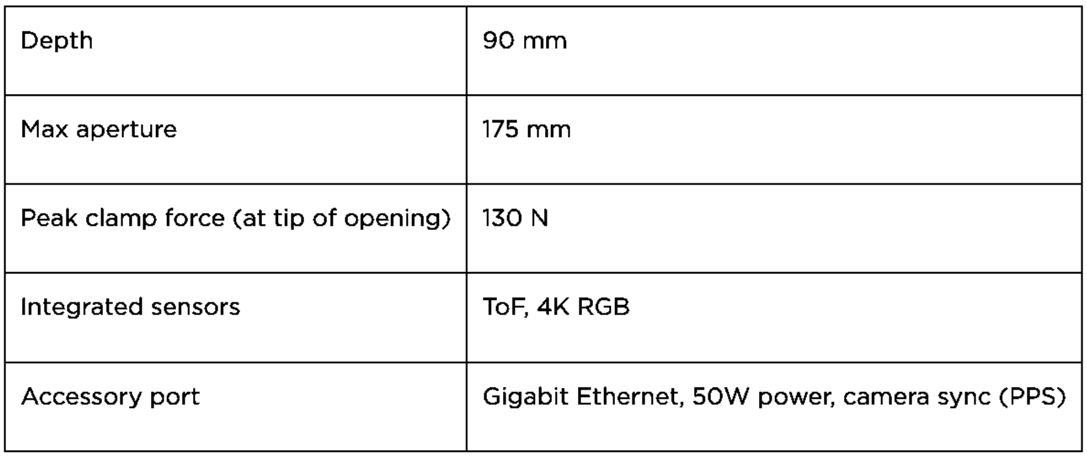
Fig. 21: Spot Arm's dimensions (mm) and degrees of freedom.



Degrees of freedom	6 + gripper
Length at full extension	984 mm
Max. reach height on robot	1800 mm
Mass/weight, including gripper	8 kg
Max. endpoint speed	10 m/s
Lift capacity*	Up to 11 kg
Continuous lift capacity at 0.5 m extension*	5 kg
Drag capacity* (on carpet)	Up to 25 kg
Operating temp	-20 C to 45 C
Ingress protection	IP54

* At 22 C.

Table 2: Spot Arm arm's specifications.



Depth	90 mm
Max aperture	175 mm
Peak clamp force (at tip of opening)	130 N
Integrated sensors	ToF, 4K RGB
Accessory port	Gigabit Ethernet, 50W power, camera sync (PPS)

Table 3: Spot Arm gripper's specifications.

A detailed and complete overview about the arm and its gripper is reported in [21].

1.7 Solution proposed

The objective of this thesis work is to overcome the inspection problems mentioned in chapter 1.1 and to provide a reliable and efficient solution to Reply's customer while meeting their needs. Therefore, a six-month project began in mid-June 2022. It is an extended two-phase pilot project carried out exploiting the quadruped robot dog from Boston Dynamics. The first phase involved data collection about meters to be analyzed, viability of the autonomous missions to be recorded, machine learning based model training and computer vision algorithms development. The second phase instead includes mission configuration and recording, testing, validation, and analysis of the results carried out. In the following chapters more details on these topics will be presented.

2. Project phase one

This preliminary and fundamental step of the project aims to collect data, label them, and subsequently use them to train the model exploiting machine learning techniques. All these activities have taken about a month of work. After this, computer vision algorithms development phase has begun. More details will be presented in the following.

2.1 Data collection

Data collection is a crucial activity to be done in order to build the image dataset i.e., the input on which subsequent activities can be based. The objects depicted in the images of interest are analog gauges, mechanical and digital water meters, digital indicators, expansion vessels, status indicator valves and liquid level indicators (Fig. 22). Those objects were found scattered in three different areas of the customer plant, namely area 01, area 02 and area 03.

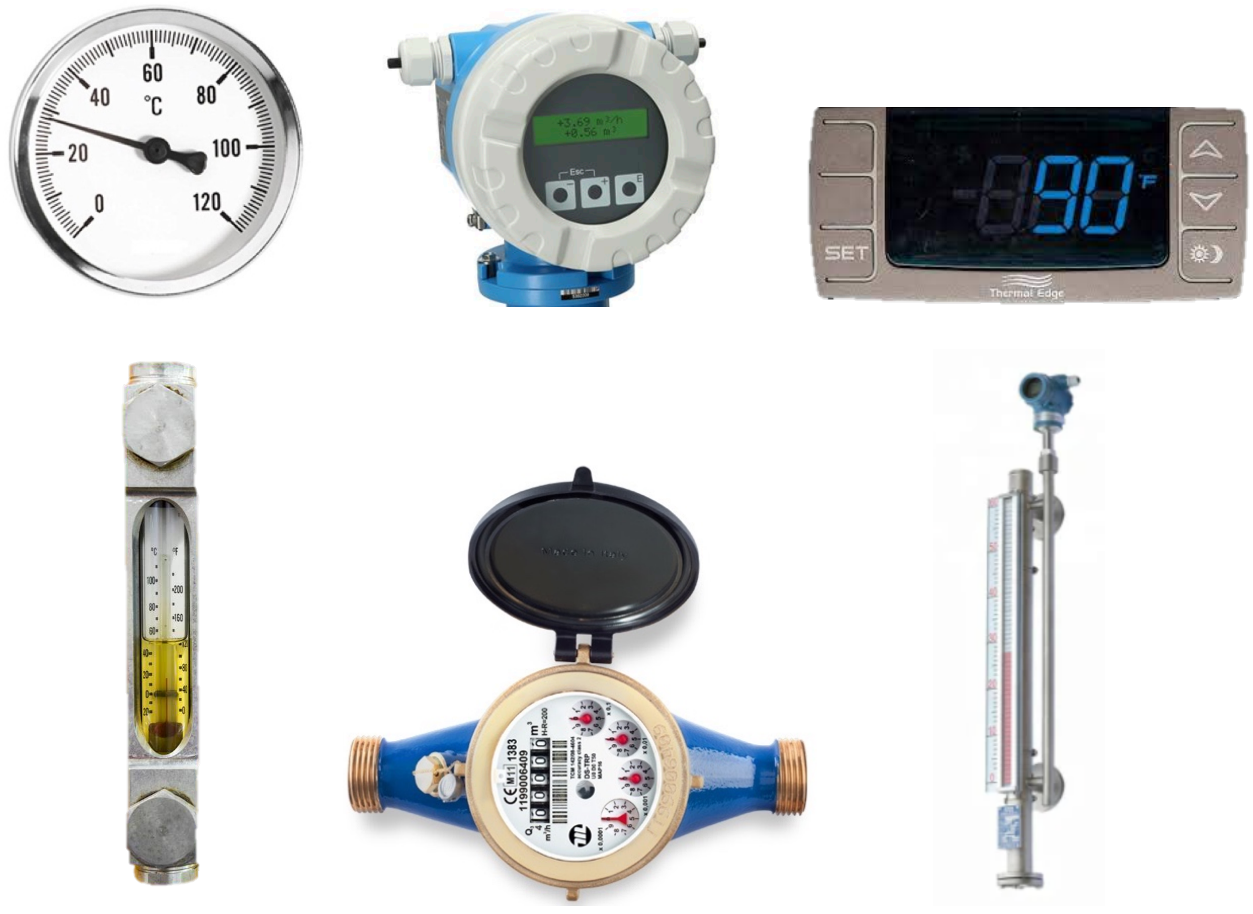


Fig. 22: examples of some of the objects of interest.

2.2 Data gathering

To build a heterogenous and varied dataset, both phone cameras and Spot PTZ has been used. Thousands of photos were collected across six site inspections from March to October. This made it possible to obtain photos with different camera quality and resolution and to get photos of the instruments with non-identical measurements. The latter will be useful for testing the robustness of computer vision algorithms.

2.3 Data labelling

Data labelling allows artificial intelligence and machine learning algorithms to build an accurate understanding of real-world environments and conditions. This is the reason why the dataset needs to be labelled. For the purposes of this project, the labelling activity has been accomplished manually by five people using LabelImg [22]. Ten classes were identified and used to label the dataset:

- analog_gauge
- mechanical_water_meter
- digital_water_meter_circle
- digital_water_meter_square
- general_water_meter
- expansion_vessel
- status_indicator_valve
- digital_indicator_circle
- digital_indicator_rect
- oil_level

2.3.1 LabelImg tool

LabelImg is an open-source tool for visual image annotation. It is developed in Python programming language and it uses Qt for its graphical user interface. The latter is a cross-platform software for creating graphical user interfaces as well as cross-platform applications that run on various software and hardware platforms. To the aim of this project, annotations in LabelImg have been saved as XML files in PascalVOC format which is the format used by ImageNet [23]. Additionally, CreateML and YOLO formats are supported. The figure reported below shows an example of how labelled images look like (Fig. 23).

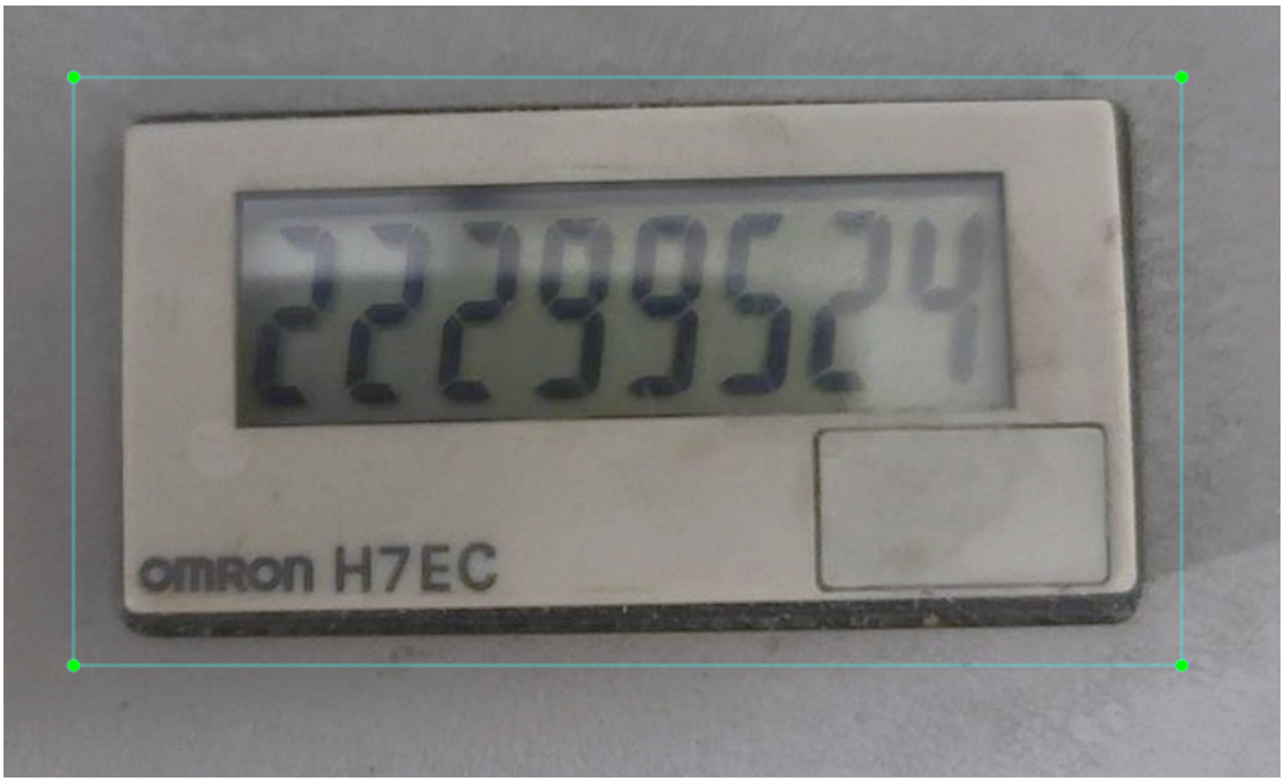


Fig. 23: example of a labelled image using LabelImg tool.

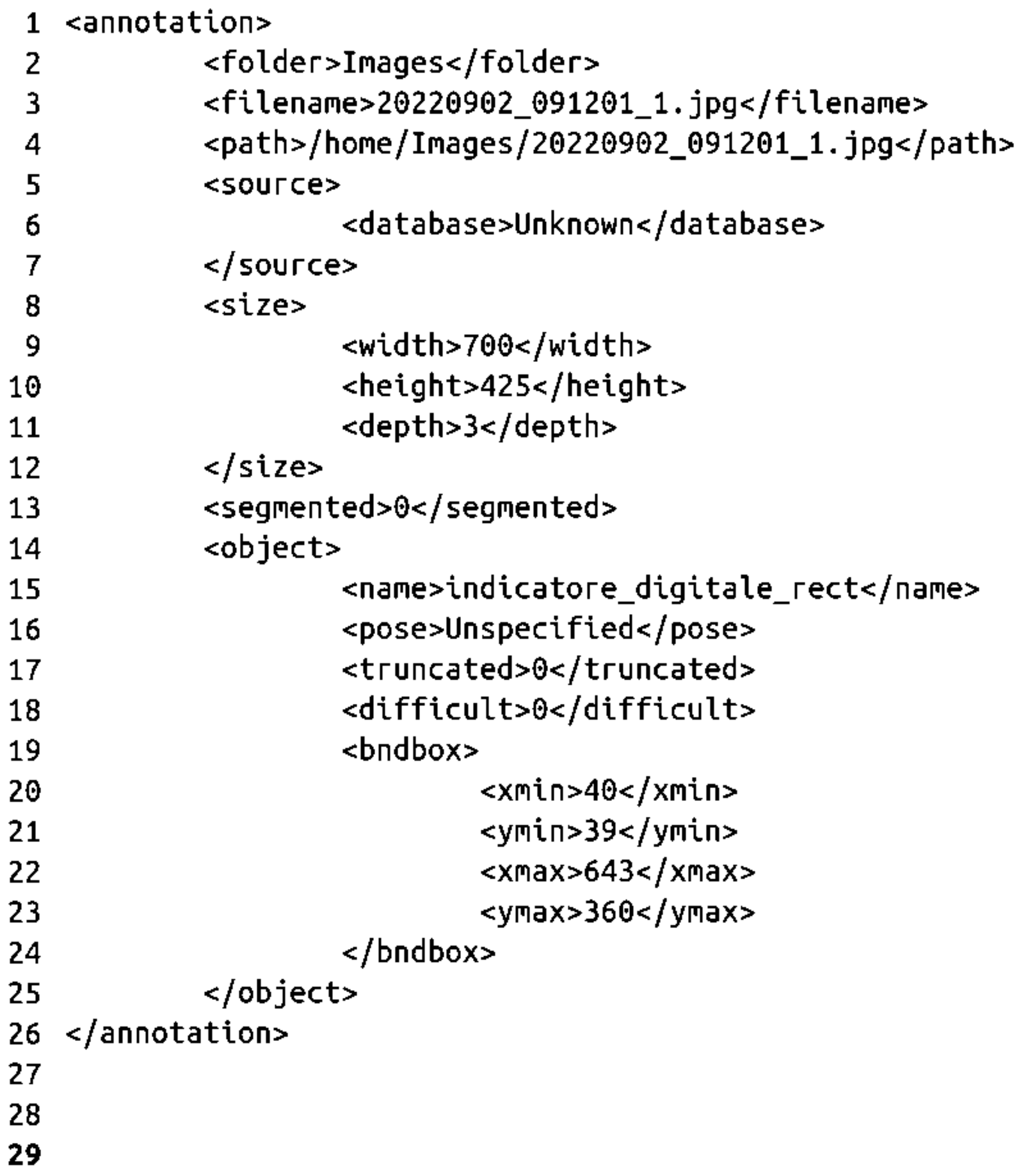
2.3.2 PascalVOC format

PascalVOC is an XML file used for annotation purposes. Specifically, a file is created for each of the image in the dataset considered. Initially, PascalVOC has been used as the annotation format of the Visual Object Classes (VOC) challenge, but recently it has evolved into a standard format for object detection labels [24]. In the challenge, two principal aspects were involved:

- classification that answers to the question *"does the image contain any instances of a particular object class?"*
- detection that instead answers to the question *"where are the instances of a particular object class in the image (if any)?"*

The main objective of the VOC challenge was to provide challenging images and high-quality annotations, together with a standard evaluation methodology.

An example of an XML file obtained from an object labelled using the PascalVOC annotation format is shown in the picture below (Fig. 24).



```xml
<annotation>
	<folder>Images</folder>
	<filename>20220902_091201_1.jpg</filename>
	<path>/home/Images/20220902_091201_1.jpg</path>
	<source>
		<database>Unknown</database>
	</source>
	<size>
		<width>700</width>
		<height>425</height>
		<depth>3</depth>
	</size>
	<segmented>0</segmented>
	<object>
		<name>indicatore_digitale_rect</name>
		<pose>Unspecified</pose>
		<truncated>0</truncated>
		<difficult>0</difficult>
		<bndbox>
			<xmin>40</xmin>
			<ymin>39</ymin>
			<xmax>643</xmax>
			<ymax>360</ymax>
		</bndbox>
	</object>
</annotation>
```

Fig. 24: XML file of a labelled object in PascalVOC format.

2.4 Model training

This part of the project focuses on the training of the model using machine learning techniques, in particular exploiting the YOLO tool [25]. This is a fundamental step as the goal is to make the robot able to recognize autonomously the target objects located in the customer's plant.

2.4.1 YOLO

You Only Look Once, also known as YOLO, is a real-time object detection algorithm. Its main feature is that it uses a single neural network to carry out both classification and prediction of the bounding boxes of the detected objects. In this way it substitutes what was previously a multi-step process, like in the R-CNN family of algorithms, making the whole process easier and faster.

2.4.1.1 YOLOv5

During the last few years, YOLO has been continuously updated by different authors until the seventh version released in 2022, which is currently the most recent version of this tool. The key ideas have been introduced in the first three versions while the following developments focused on performance improvements by adopting new architectures and algorithms published by the deep learning and computer vision community during the years. To the purpose of this project, the fifth version of YOLO, also known as YOLOv5, has been chosen as it has reached a stable version that can be easily trained and deployed compared to more recent versions.

2.4.1.2 YOLO algorithm

The YOLO algorithm divides the input image into a grid of cells and assigns a set of bounding boxes to each cell. Each bounding box represents a potential object in the image and it is associated with a class label and a confidence score. These confidence scores indicate the probability that the bounding box contains an object and the accuracy of the predicted box.

To classify the objects in the image, YOLO uses a multi-scale detection approach. First, it processes the input image at different scales and subsequently it combines the detections obtained from each scale to generate the final set of bounding boxes. This allows YOLO to detect objects of different sizes and scales in the image. This aspect has been very useful since the labelling phase has been done manually and so the labelled images varied in sizes and scales.

Instead, to improve the accuracy of object detection, YOLO uses a number of techniques such as anchor boxes, non-maximum suppression and hard negative mining. These techniques allow YOLO to handle occlusions, overlap and other challenges that can occur in real-world images. But the main change introduced in YOLOv5 compared to other versions is the porting of the backbone architecture, called darknet.

2.4.2 Training

To recognize all the classes of the objects labelled and presented in the section 2.3 of this paper, the model has been trained. From the results of this training, it has been deduced that the performance over the 'status_indicator_valve' class was rather poor, so a new model has been trained to recognize only this class. The dataset used to train this model has been split between the train and the validation sets, with a size of 682 and 146 respectively. Then, this model has been finetuned for 150 epochs starting from the YOLOv5s weights using an image size of 512, a batch size of 32, and a starting learning rate of 0.01. Stochastic Gradient Descent has been used as optimizer. Moreover, some data augmentation techniques have been used like gaussian blurring and CLAHE. The results obtained from the trained model is reported in the figure below (Fig. 25).

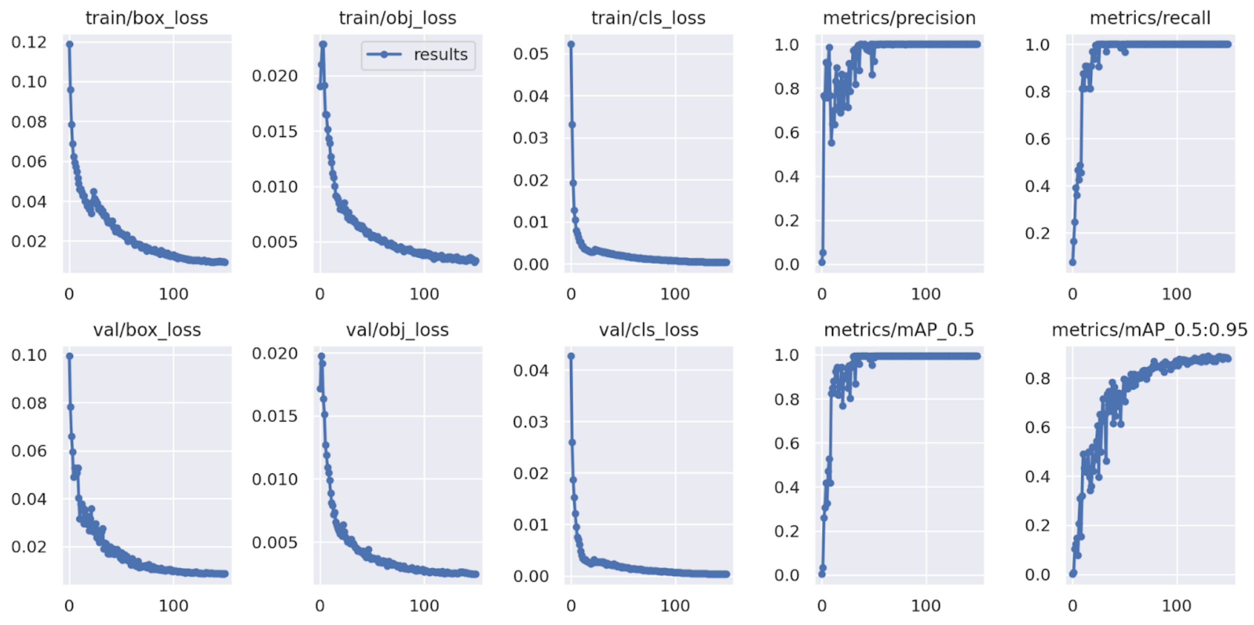


Fig. 25: results of the trained model on train and validation sets.

2.5 Computer vision algorithms development

Computer vision in robotics is defined as the process of extraction, characterization, and interpretation of information coming from images of a tridimensional world. Thus, in this project, the development of computer vision algorithms played an important role, besides the model training phase. In particular, it has been essential the processing of the image given in input in order to obtain as output the target numeric value represented on the meters or on the valves. After the photos of interest acquired by the robot have been uploaded on a webapp designed on purpose, the implemented codes have been associated with each measuring instrument. In this way the callbacks of the codes have been executed and the results of the readings through artificial intelligence have been carried out.

2.5.1 General approach and description

The images of interest can be split into two major classes: text and non-text. In the first case, the problem that has been faced takes both scene text detection and scene text recognition into account. The former required to isolate only one region of interest among many areas in the image, that is the one containing the text to be extracted. The latter regarded pure text recognition. At the beginning, this has been approached as an OCR task but, after few attempts, it has been decided to switch to scene text approach which led to better results. This change has been necessary due to several different aspects: difference in lighting conditions of the input photos taken, difference in font type of the displayed numbers and/or letters, and in the text format that needed to be extracted. As concerns the second class, the one not containing text, a case-by-case approach has been followed due to the rather negligible similarities between the considered components. Here only a digital image processing approach has been followed, where the features of interest have been isolated from the cropped detection result step-by-step. Reliability of the results under different lighting conditions and a wide range of image quality have been one of the key features kept in mind when designing the algorithms. When this was not possible, the parameters have been outsourced in a configuration file, written in json format, where they can be set for each input image.

2.5.2 Programming language

All the algorithms have been implemented in Python. The reason of this choice has been mainly due to the convenience of fast experiment iterations and due to the large number of production-tested libraries and frameworks that this programming language makes available. It has also been decided to dockerize the algorithms. Dockerizing is the process of packing, deploying, and running applications using Docker containers and so this has been done as the last step. This decision allowed us to use different Python versions depending on the single developer preferences. For all the components, the OpenCV library has been used as the main tool for digital image processing, whilst for the ones containing text, PyTorch library has been the tool chosen for the scene text recognition tasks.

2.5.3 Libraries

In the programming world, a library is a collection of pre-compiled codes that can be used later in a program for some specific well-defined operations. Other than pre-compiled codes, a library may contain documentation, configuration data, message templates, classes, and values. Specifically in the Python programming language, a library is a collection of related modules. It contains groups of code that can be used repeatedly in different programs. This makes programming simpler and convenient for the developer. Moreover, Python libraries play an important role in fields of machine learning, data science, data visualization, etc.

2.5.3.1 OpenCV

OpenCV (Open-Source Computer Vision) is a library of computer vision algorithms and related software tools. It is designed to help developers to build applications that can analyze and interpret visual data, such as images and videos. The OpenCV library is written in C++ and includes interfaces for other programming languages, such as Python. This allows developers to use the functionalities provided by the OpenCV library in their Python applications. The Python wrapper makes it easy to use OpenCV in Python and provides a high-level, intuitive interface for accessing the features of the library. It is widely used in a variety of applications, including image processing, computer vision, and machine learning.

2.5.3.2 PyTorch

PyTorch is a free and open-source deep learning platform that provides a continuous path from research to production. It is designed to enable researchers to prototype and build models quickly and to allow developers to create and deploy deep learning applications easily. PyTorch is based on a dynamic computational graph, which allows users to change the structure of the graph instantly and to perform computations in a memory-efficient manner. It also provides support for distributed training allowing users to train large models across multiple GPUs and machines.
In the field of computer vision, PyTorch can be used to build and train deep learning models for a variety of tasks such as image classification, object detection, and segmentation thanks to the wide range of tools and libraries that it makes available. Researchers and developers working on computer vision tasks often use the PyTorch platform as it has various applicability.

2.5.4 Pipeline

In computing, the pipeline refers to a set of data processing elements connected in series where the output of one element is the input of the next one. It has been necessary to use this approach to get the best possible artificial reading. In particular, given that for each of the ten classes obtained from the labelling phase there may be more than an occurrence, that is different models of meters that detect the same measure, it has been decided to develop a different algorithm for each occurrence. In some cases, several Python methods have been implemented in the same algorithm and they have been used if it is so specified in the relative configuration file. This allowed to reduce the number of different algorithms developed still considering all the cases. An example is given by the code implemented for the round digital water meters. It is also to be noted that all the photos given in input have been taken with the same angle and perspective. This derives from a trial-and-error procedure that lasted about three months. The generic pipeline for a computer vision task is based on pre-processing, segmentation, description, recognition, and interpretation processes. Those steps will be detailed in the following, analyzing in particular the development of the codes which I contributed to this project with i.e., round digital water meters and mechanical water meters.

2.5.4.1 Round digital water meters

Two main occurrences have been identified for this type of meter differentiated based on the display color: orange displays and green displays (Fig. 26). To take this difference into account, a color parameter has been initialized and set through the configuration file. In particular, if that parameter is set to orange, some Python methods will be executed; otherwise, if it is set to green, some other methods will be run. After this premise, the first step concerns the pre-processing of the image given in input. Basically, the original input image has been converted into its grayscale version and, in some cases, other pre-processing functions have been applied such as bilateral filtering, gaussian blur, median blur [26], and morphological operations [27]. The next step in the pipeline has been to use the obtained pre-processed image for the segmentation stage. This is the process of partitioning a digital image into multiple image segments. Its goal is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. For the purposes of this algorithm, a mask based on the orange and green colors in their HSV color spaces has been used. For each color two boundaries have been defined: a lower and an upper limit. This means that the color is detected if it is between those limits; moreover, to optimize the code and make it reusable, the parameters related to those color boundaries have been set through the configuration file so that also the color can be changed if needed. In addition to color masking, Canny edge detection [28] has been used to isolate the contours of the display since they enclose the region of interest (ROI). The input image passed at this step has been the one got from color masking, so the one containing the area of the display. The objective for a correct artificial reading is to obtain the image of interest cropped as precisely as possible. To achieve this result in a robust manner, an iterative loop has been built before in order to detect circles and rectangles. To this aim, contours must be retrieved from a starting image. In particular, for the meters with orange displays, the contours obtained from color masking have been used for the circle detection, while the contours obtained from the Canny edge detection have been used to detect rectangles. Conversely has been done for the meters with green displays. The reasons of this choice are due to the segmentation capabilities of the used OpenCV functions. The subsequent step in the pipeline regards the description process which is responsible to compute characteristics, such as dimensions and shapes, that are useful to differentiate one object from another. From this point of view, the mentioned iterative loop has been essential to detect the circle (Fig. 27) and the rectangle of interest which areas are comprised between specific numbers of pixels, set in the configuration file. In particular, the target rectangle is the one that lies inside the detected circle within the specified areas limits (Fig. 28). The use of this loop is compulsory because the artificial intelligence normally identifies all the circles and rectangles in the image provided, but the goal for a correct reading is to isolate the circle and the rectangle of interest. The last meaningful process in the pipeline concerns recognition that identifies the objects obtained from the description process. In fact, that image has been passed to an open-source optical character recognition (OCR) model that has already been trained on texts of various fonts, sizes, orientations, text shapes, illumination, amount of occlusion, and inconsistent sensor conditions [29]. In particular, it is a Permuted Autoregressive Sequence model (PARSeq) i.e., a unified Scene Text Recognition model with a simple structure, trained using Permutation Language Modeling. PARSeq follows an encoder-decoder architecture. The encoder is a 12-layer ViT without the classification head and the [CLS] token while the decoder is made of one layer only. Compared to the standard ViT, all the output tokens are used as input to the decoder. The decoder follows the same architecture as the preLayerNorm Transformer decoder but uses twice the number of attention heads. It has three required inputs consisting of position, context, and image tokens and an optional attention mask. In the figure below it is shown the logics behind this OCR model (Fig. 29). Finally, the outcome of the OCR has been then post-processed if needed.

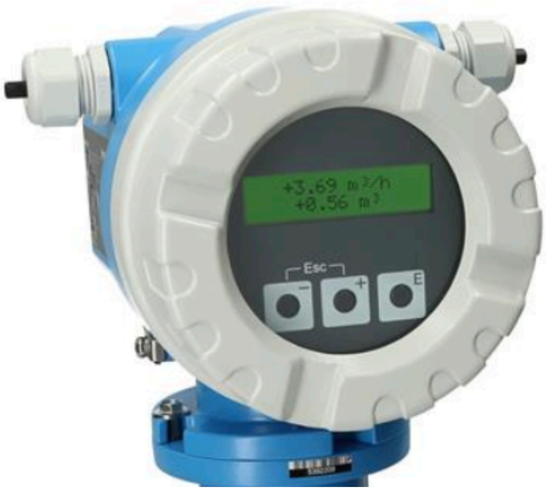


Fig. 26: round digital water meter with green display.

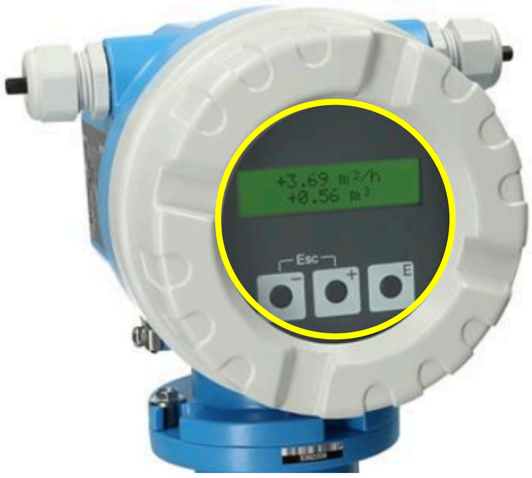


Fig. 27: circle detection of a round digital water meter.

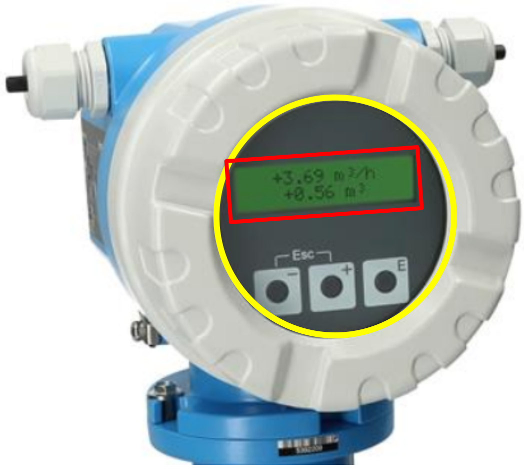


Fig. 28: rectangle detection of a round digital water meter.

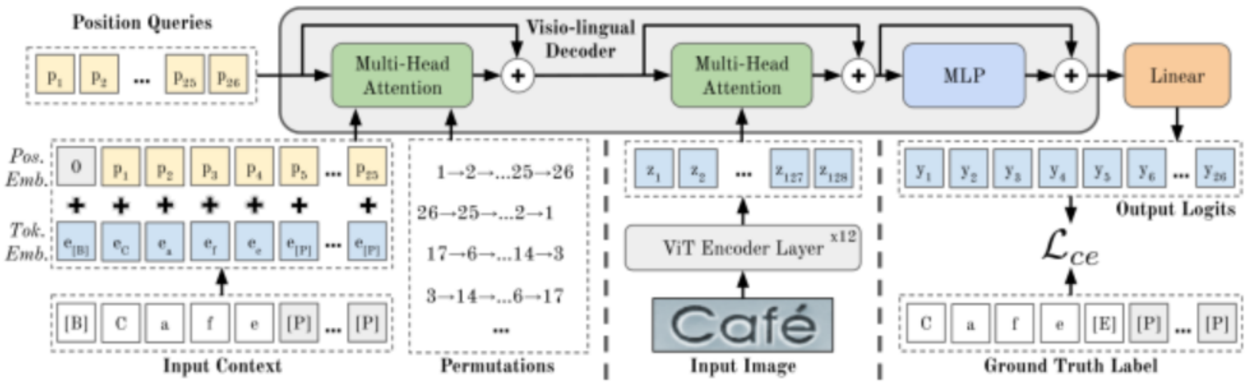


Fig. 29: logics of the Permuted Autoregressive Sequence Models (PARSeq).

2.5.4.2 Mechanical water meters

In the development of this algorithm, many occurrences have been identified. Some of them have been later defined as 'out of scope' since, while registering the missions, the robot was not able to reach those meters and to get a decent photo acquisition. In particular, the occurrences taken into consideration are five and they are different models similar to the one shown in the figure below (Fig. 30). For each occurrence a different configuration file has been created. Here the parameters concerning the upper and lower limits of the color in its HSV color space, the maximum and minimum limits of the areas, the rotation angle, and some coefficients have been reported. Analyzing the pipeline used in the development of this computer vision algorithm, the first step regards the pre-processing of the image given in input. In particular, the main operation performed regards the conversion of the original image in its grayscale version and the application of the bilateral filter and of the median blur [26]. The subsequent step in the pipeline regards the segmentation of the image, that is the process of partitioning a digital image into multiple image segments. The isolation of the region of interest (ROI) happened thanks to the application of a color-based mask. Specifically, since all the mechanical water meters considered had the display area enclosed by a white circular shape, the color-based masking has been done on the white color (Fig. 31). The color limits used to delimitate the desired range of white have been set in all the configuration files of the water meters. Next step in the pipeline regards the description process, so the computation of characteristics, such as dimensions and shapes, that are useful to differentiate one object from another. To this aim, an iterative loop on all the contours found on the mask has been implemented. Specifically, it has been noticed that the biggest area found by the contours on the mask, enclosed the circle of interest. Thus, if that contour lied between a specific range set in the configuration file, the area enclosed in it had been isolated (Fig. 32). After this, the obtained image has been rotated, if needed, by an angle specified in the configuration file (Fig. 33). In addition, if necessary, the image can be flipped. This option arose from the cases where Spot was not able to reach the position of the meter to acquire, but it was still able to take the photo accurately. Therefore, it has been decided to use a mirror for the photo acquisition and flip the image in a second moment. Specifically, the direction of flip of the image can be set in the configuration file and it has been possible to choose between horizontal, vertical or both axes flip. Lastly, the image obtained from the previous step must be cropped properly for a correct artificial intelligence-based reading. To this purpose, the algorithm allows the choice between what has been called dynamic crop and static crop. This choice can be made by setting a true or false parameter in the configuration file. In this way, the same algorithm can be exploited for all the models of mechanical water meters that have been considered. The dynamic crop function has been implemented to crop the image got from the previous step in a more robust manner. A color-based mask on the black color has been applied. Then the number of contours that have been found on that mask have been used to implement an iterative loop. Here the bounding rectangle of the contour has been computed and chosen depending on the contour's dimensions and area, giving birth to the final and further cropped image (Fig. 34). As far as static crop is concerned, the isolation of the region of interest i.e., the display, has been made in a slightly rough way compared to the dynamic crop. In particular, two edges of the image obtained from the last step has been taken as reference: the upper and the left. Taking this premise into account, some coefficients have been set in the configuration file; those coefficients represent percentages of the width and height of the image obtained from the rotation or flip stages. The precise values of the mentioned percentages have been obtained from a trial-and-error procedure. The result obtained is similar to the one got using the dynamic crop. Finally, the last step in the pipeline concerns recognition. In fact, the image obtained from either the dynamic or the static crop stage has been passed to the same open-source optical character recognition (OCR) model mentioned in the previous algorithm. Furthermore, to the outcome of the OCR model a regular expression (RegEx) function [30] has been applied to make aware the OCR model that it only needs to identify numbers.

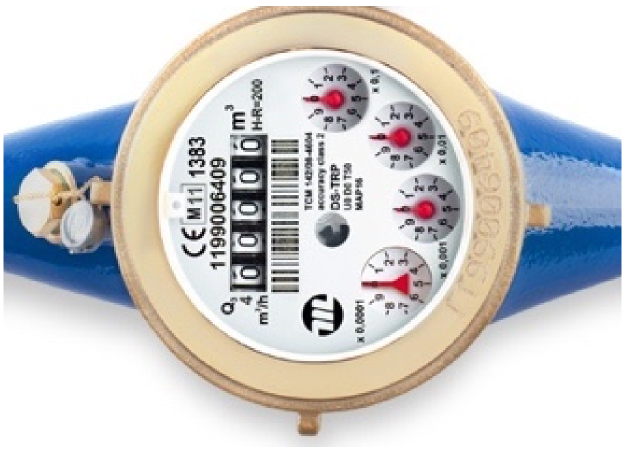


Fig. 30: example of an occurrence of a mechanical water meter.

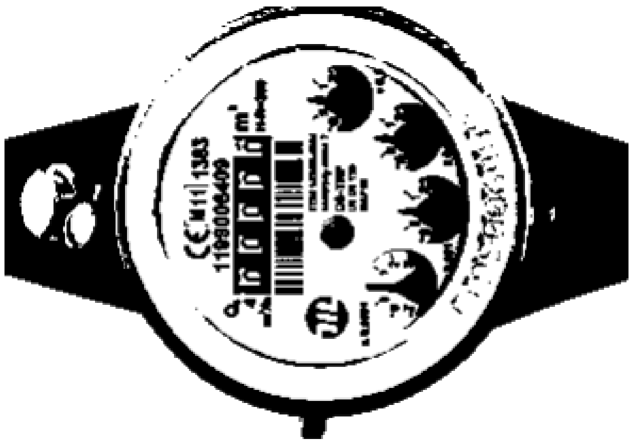


Fig. 31: example of result from the application of a color-based mask on a mechanical water meter.

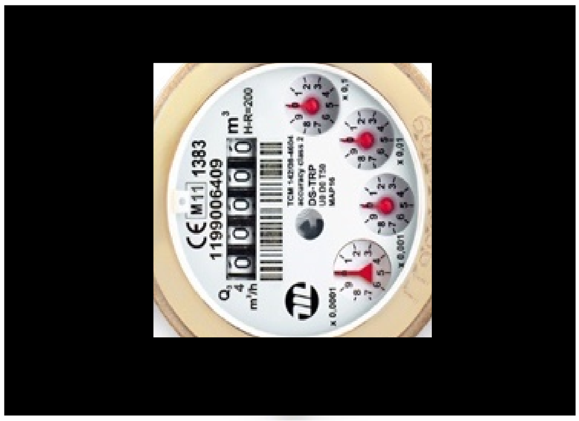


Fig. 32: example of result of the iterative loop of a mechanical water meter.

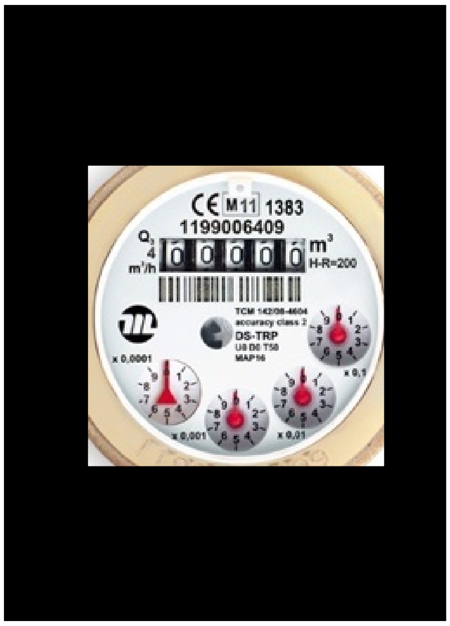


Fig. 33: example of result of the rotation of a mechanical water meter.

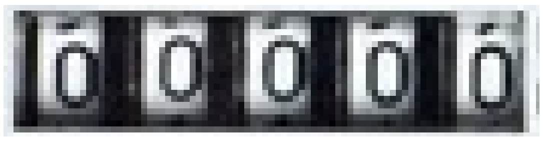


Fig. 34: example of result of the dynamic crop of a mechanical water meter.

2.5.4.3 Other algorithms

The remaining algorithms have been developed by the other members of the team.
As regards the circular meters, the general approach followed involved the detection of the circle by averaging among all the circles found applying the Hough Transform function of the OpenCV library [31] and selecting the circles that have a center close to that of the image and a diameter close to its width. Then the region of interest i.e., the display, has been isolated and passed to the Parseq OCR for the text recognition step. Also, for the analog gauge, the Hough Transform has been used to find the pointer inside the outer circle. Only the circles that were tangent to the center of the outer circle and that had a diameter at least equal to its radius were considered (Fig. 35). Subsequently, the perimeter of the inner circle has been split into periodic sections and the position of the section containing the pointer was mapped in the relative configuration file. In the interpretation step the developer can observe in which section the pointer is located and use the mapping to extract the real value.
As concerns the liquid level indicators, two types have been considered. In both cases, first the upper and lower extremes of the indicator were found by applying masking and thresholding processes and then the liquid level was detected exploiting the color difference of the liquid compared to the background (Fig. 36).
For the status indicator valve (Fig. 37), the approach followed focused first on finding the red lever by color masking and then finding the two black spots that define the extremes of the valve. Subsequently, by using a mapping of the relative position of the lever with respect to the two black spots, defined in the configuration file, the developer was able to extract the result.
Lastly, for the digital indicators (Fig. 38), an approach similar to the liquid level indicators has been followed. Therefore, masking and thresholding processes have been the starting point of the algorithms; then, few operations concerning the isolation of the display has been performed. Finally, the extracted region of interest has been passed to the OCR which was responsible for the text recognition aspect.

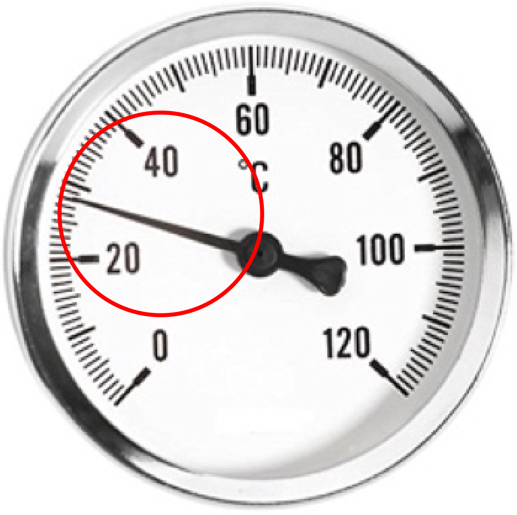


Fig. 35: detection of the pointer in the analog gauge.

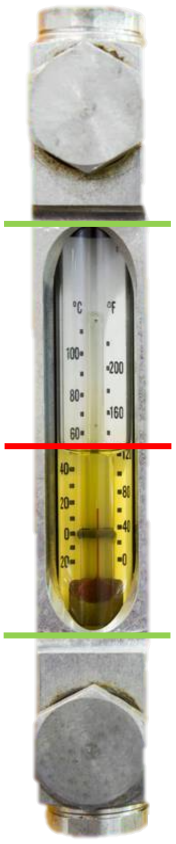


Fig. 36: green lines show the detection of the upper and lower extremes of the liquid level indicator, while the red one indicate the detection of the level of liquid.

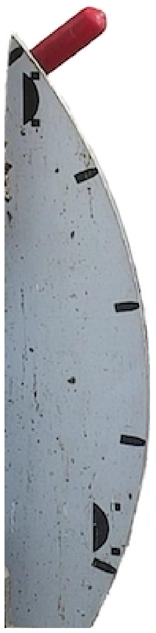


Fig. 37: example of the status indicator valve.

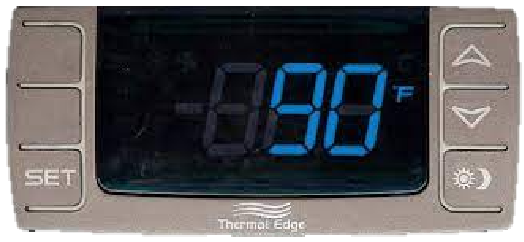


Fig. 38: example of the digital indicator.

2.6 Docker

Docker is an open platform that is used for developing, shipping, and running applications. It enables to separate the application from the infrastructure, in this way the software can be delivered quickly. With Docker, it is possible to manage the infrastructure in the same ways the application is managed. By taking advantage of the methodologies that this platform makes available for shipping, testing, and deploying code quickly, the delay between writing code and running it in production can be significantly reduced. In this project, Docker has been used to bring the code implemented in an upper abstraction level, that is to make the codes implemented independent from a specific Python version. In this way, it has been possible for the developers to work with their own preferred Python version. Another advantage obtained by using the Docker platform came from its ability to package and run an application in containers. Details about this topic will be presented in the following.

2.6.1 Docker platform

Docker provides the ability to package and run an application in a loosely isolated environment called container. The isolation and security allow the developer to run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application; thus, it is not necessary to rely on what is currently installed on the host. Also, containers can be easily shared allowing the people receiving them to get the same container that works in the same way. In addition, Docker provides tooling and a platform to manage the lifecycle of the containers. Specifically, first the application and its supporting components must be developed using containers: the container becomes the unit for distributing and testing the application. Then, the application can be deployed into the production environment, as a container or an orchestrated service. This works the same whether the production environment is a local data center, a cloud provider, or a hybrid of the two.

2.6.2 Usage benefits

The main advantage that is possible to obtain by using the Docker platform regards fast and consistent delivery of the applications.
Docker simplifies the development lifecycle by allowing developers to work in standardized environments using local containers which provide applications and services. Moreover, containers are great for continuous integration and continuous delivery (CI/CD) workflows. Another benefit of this container-based platform comes from the highly portable workloads it provides. In fact, Docker containers can run on developer's local laptop and on physical or virtual machines. Furthermore, Docker's portability and lightweight nature make it easy to dynamically manage workloads, scaling up or tearing down applications and services, in near real time.
Lastly, Docker's lightweight nature allows to run more workloads on the same hardware. It provides a viable, cost-effective alternative to hypervisor-based virtual machines and it is perfect for high density environments and for small and medium deployments where there is the need to do more with fewer resources.

2.6.3 Docker architecture

Docker is based on a client-server architecture. The Docker client communicates with the Docker daemon, which is responsible of building, running, and distributing the Docker containers, exploiting REST API, over UNIX sockets, or using a network interface. The Docker client and daemon can be run on the same system, or the Docker client can be connected to a remote Docker daemon. Docker Compose is another

Docker client that allows to work with applications consisting of a set of containers. The figure below shows a schematic of this architecture (Fig. 39).

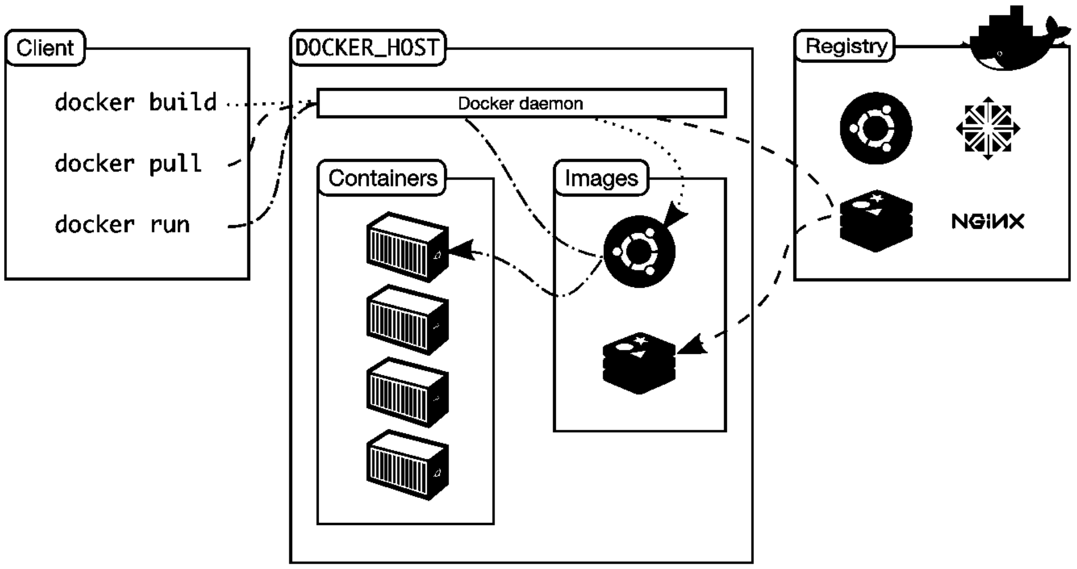


Fig. 39: Docker architecture schematic.

2.6.3.1 Docker daemon

The Docker daemon listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

2.6.3.2 Docker client

The first interactions many users have by using Docker are through the Docker client. The latter is responsible to send commands, such as *docker run,* to the Docker daemon which carries them out. The Docker client can also communicate with more than one daemon.

2.6.3.3 Docker registries

A Docker registry's job is to store Docker images. By default, Docker is configured to look for images on Docker Hub, which is a public registry that anyone can use. It is also possible to run private registries. As regards the functioning, when the *docker pull* or *docker run* commands are used, the required images are pulled from the configured registry. Instead, when the *docker push* command is used, the image is pushed to the configured registry.

2.6.3.4 Docker objects

While using Docker, images, containers, networks, volumes, plugins, and other objects are created and used. In the following a brief overview of some of those objects is reported.

2.6.3.4.1 Images

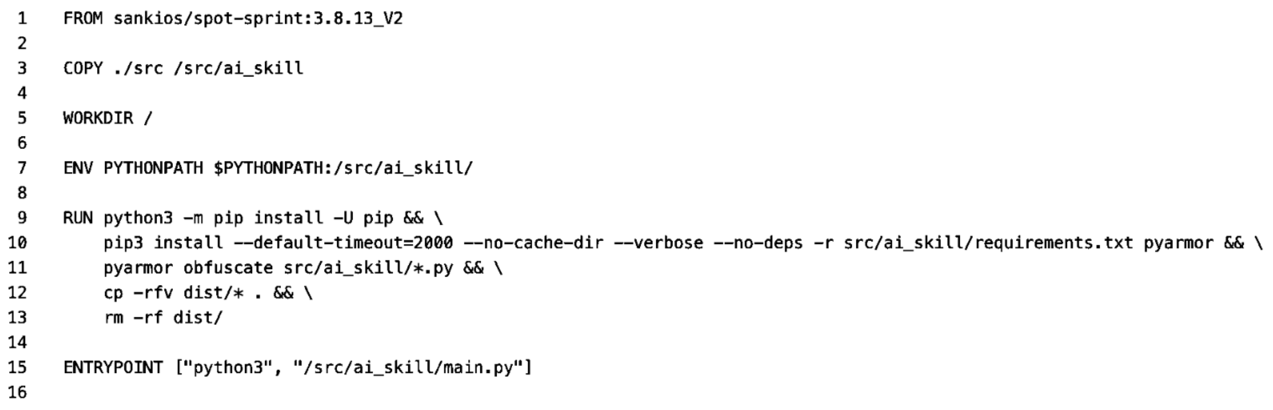
An image is a read-only template with instructions for creating a Docker container. Often, an image is based on another image with some additional customization. It is possible to create images or to use only those created by others and published in a registry. In order to build a new image, it is necessary to create a Dockerfile, which is a file with a simple syntax used for defining the steps needed to create the image and run it. Each instruction in a Dockerfile creates a layer in the image. When the Dockerfile is modified and the image is rebuilt, only the layers which have changed are rebuilt. This is part of what makes images so lightweight, small, and fast compared to other virtualization technologies.

2.6.3.4.2 Containers

A container is a runnable instance of an image and it can be created, started, stopped, moved, or deleted using the Docker API or CLI. A container can be connected to one or more networks, storage can be attached to it, or a new image based on its current state can be created.
By default, a container is relatively well isolated from other containers and its host machine. The isolation level of a container's network, storage, or other underlying subsystems from other containers or from the host machine can be controlled by the developer. Notice that a container is defined by its image as well as by any configuration options provided to it by the developer when it has been created or started. Thus, when a container is removed, any changes to its state that have not been stored in persistent storage disappear.

2.6.4 Packing of computer vision algorithms into Docker containers

As the use of containers simplifies running an application and since they are easily sharable, it has been decided to dockerize the computer vision algorithms that have been implemented. Before doing this, a requirements txt file has been created for each algorithm. It is a file that includes all the Python libraries and packages used during the development of the relative code, together with all its dependencies. To avoid conflicts between libraries due to different versions, each library present in the txt file has been accompanied by the exact version used. Moreover, a virtual environment has been used to create the requirements file. This has been done to prevent the Docker container's receiver from downloading more Python libraries than needed. As far as the Docker container creation is concerned, an image has been created and built for each different algorithm. The instructions needed to do this has been written in a Dockerfile, as reported in the example below (Fig. 40). Based on this, a container has been created for each image exploiting Docker Desktop, which is an easy-to-install application that enables to build and share containerized applications and microservices. Furthermore, the use of Docker Compose has been essential as the final application is made up of a set of containers i.e., all the computer vision algorithms necessary for the recognition of the objects of interest by the robot.



```
FROM sankios/spot-sprint:3.8.13_V2

COPY ./src /src/ai_skill

WORKDIR /

ENV PYTHONPATH $PYTHONPATH:/src/ai_skill/

RUN python3 -m pip install -U pip && \
    pip3 install --default-timeout=2000 --no-cache-dir --verbose --no-deps -r src/ai_skill/requirements.txt pyarmor && \
    pyarmor obfuscate src/ai_skill/*.py && \
    cp -rfv dist/* . && \
    rm -rf dist/

ENTRYPOINT ["python3", "/src/ai_skill/main.py"]

```

Fig. 40: example of Dockerfile.

3. Project phase two

This second phase of the project has been taken on site, in the customer plant. The goal was first to configure and register missions and then test and finetune the computer vision algorithms in order to get the best possible results i.e., high percentage of correctly read meters by artificial intelligence. More than five weeks of hard work has been necessary to carry out the optimal results. In the following more details about this second phase of the project will be presented.

3.1 Autonomous mission configuration

Mission configuration is intended as the process by which Spot is instructed on the paths it must take. The configuration can be done by means of the controller that has been provided together with the robot (Fig. 41). Specifically, the Spot Autowalk feature enables the recording and the replaying of autonomous behaviors. The Autowalk function consists of two parts:

- **Missions recording**: this allows to drive Spot through a route and make it able to perform certain actions in specified points along the way.
- **Missions replaying:** by clicking a button on the tablet, Spot will execute the movements and actions that have been recorded, while adapting to minor changes in the environment.

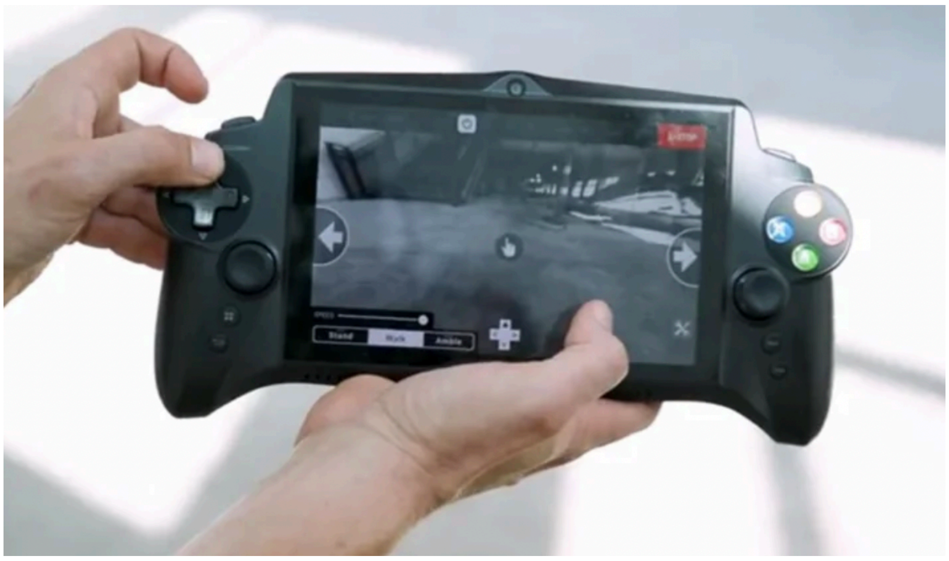


Fig. 41: Spot controller through which mission configuration is possible.

Before going into the details of how the robot's missions have been configured in this project, a premise on the terminology and definitions used must be made.

3.1.1 Terms and definitions

3.1.1.1 Missions

A mission defines the robot's movements and location-based actions in the environment. Missions can be used to automate repetitive tasks with Spot.
The robot tracks its position as it walks through the environment during mission recording and it follows the same path during mission replay. If the environment changes between recording and replay, Spot will adapt its route to avoid obstacles and attempt to complete the mission. Notice that missions are behavior trees that are recorded on Spot and are transferred to the robot controller when saved.

3.1.1.2 Actions

An Action is a data gathering activity or a behavior that assists in a data gathering activity for an Autowalk mission or for a manual activation. Actions are a key part of the robot's autonomous site inspection capability. For example, during a mission the robot can be configured to take a picture with Spot CAM or to pose in a specific way to aim a custom payload. Actions are added to Autowalk missions during the recording process by using the controller. When the mission is replayed, the Actions are performed as the robot reaches the locations where they have been placed. In addition, customized Actions can be created by the users.

Spot is able to perform several Actions. Below some examples are reported:

- take pictures using onboard cameras or the Spot CAM for Computer Vision model analysis, classification, and inspection.
- call a custom callback function to download data, command a payload, or power-cycle a payload.
- add a docking behavior to the end of an Autowalk mission to recharge the robot's batteries and download data.
- use ‘Go here’ to mark a waypoint to navigate to on playback.
- inspect the status of site properties and features such as doorways or fire extinguishers.
- detect the presence of gases or radiation.
- monitor environmental sound levels or temperature.
- lidar scans of an industrial space.
- ‘Pose’ to position the robot making it able to capture data.
- use an attached Spot Arm to perform Arm Sensor Pointing inspection tasks.

3.1.1.3 Maps

Maps are topological graphs consisting of waypoints and edges and they are recorded using the Autowalk feature from the controller. Once a map is recorded, Spot can autonomously move to any waypoint described by the map. Also, multiple missions can be associated with a single map.

3.1.1.4 Navigation

The data acquired from Spot's sensors are compared with the data saved in the recorded mission by the robot itself. This allows Spot to autonomously navigate through the environment during mission replay. Observe that mission replay relies on at least one navigation fiducial and multiple waypoints defining the robot's path.

3.1.1.5 Fiducials

Spot needs to match its internal map with the world around it and, to this purpose, fiducials are used (Fig. 42). These are specially designed images similar to QR codes and are required at the beginning of every mission. Moreover, as the Spot Dock comes with an associated fiducial, the latter can be used both at the end of an Autowalk mission. Fiducials that are recognized by the robot are of AprilTag type and they have to meet the following requirements:

- AprilTags in the Tag36h11 set.
- default image size of 146mm square.
- it must be printed on white non-glossy US-letter size, preferably rigid sheets.

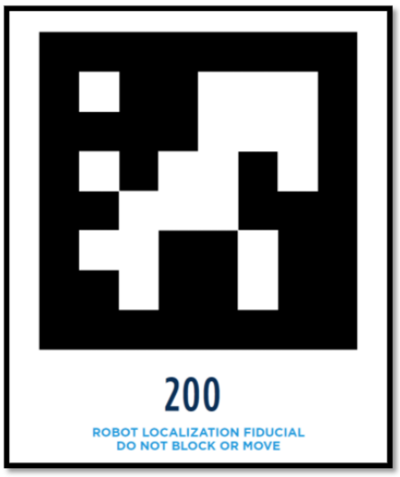


Fig. 42: example of fiducial used by Spot.

3.1.1.6 Waypoints

Navigation waypoints are location-based points automatically placed by the robot during mission recording to help it follow a specific path (Fig. 43). During mission replay, Spot walks from a waypoint to the following. Waypoints positioning is done as follows:

- at 2 meters intervals along straight paths.
- when either event occurs within a 0.3 meters path segment:
    - Spot turns more than 30 degrees.
    - elevation of Spot changes more than 0.3 meters.
- when an Action is recorded.

As regards the robot's orientation in the environment, the basic robot i.e., without payloads, can track visual features within 2 meters from its position by exploiting its stereo cameras. Moreover, if it is outfitted with a lidar, it can track features within that sensor's range that typically is over 50 meters. The added sensing range of lidar enables the robot to travel farther from features and through more dynamic environments.

During mission replay, Spot calculates its position by comparing features in its current sensor data with features in the data snapshots taken at each waypoint during mission recording. Deviations in Spot's path and small changes in the environment are automatically compensated by the robot itself, but large discrepancies may require the intervention of a human operator.

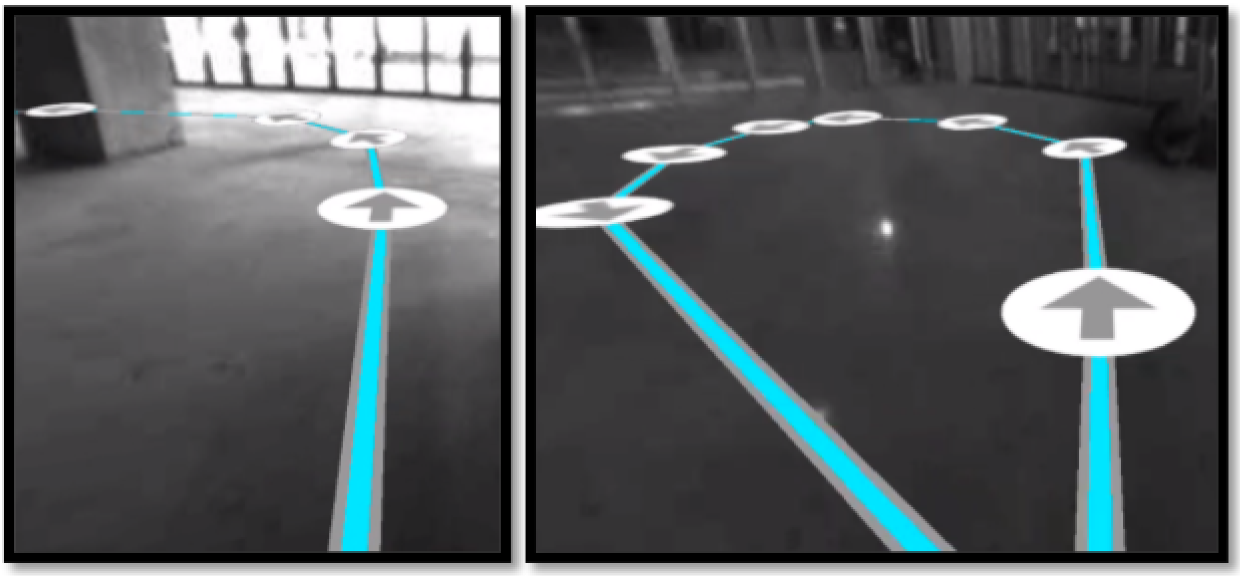


Fig. 43: example of Spot's navigation waypoints.

3.1.1.7 Payload services

When payloads are mounted on the robot, the availability of Actions in general increases. This is a consequence of the correct execution of the service that supports a given Action; the latter may not be running because the physical payload is not connected properly to Spot or the service is not properly registered with the robot. Payloads can be divided into three categories based on the weight:

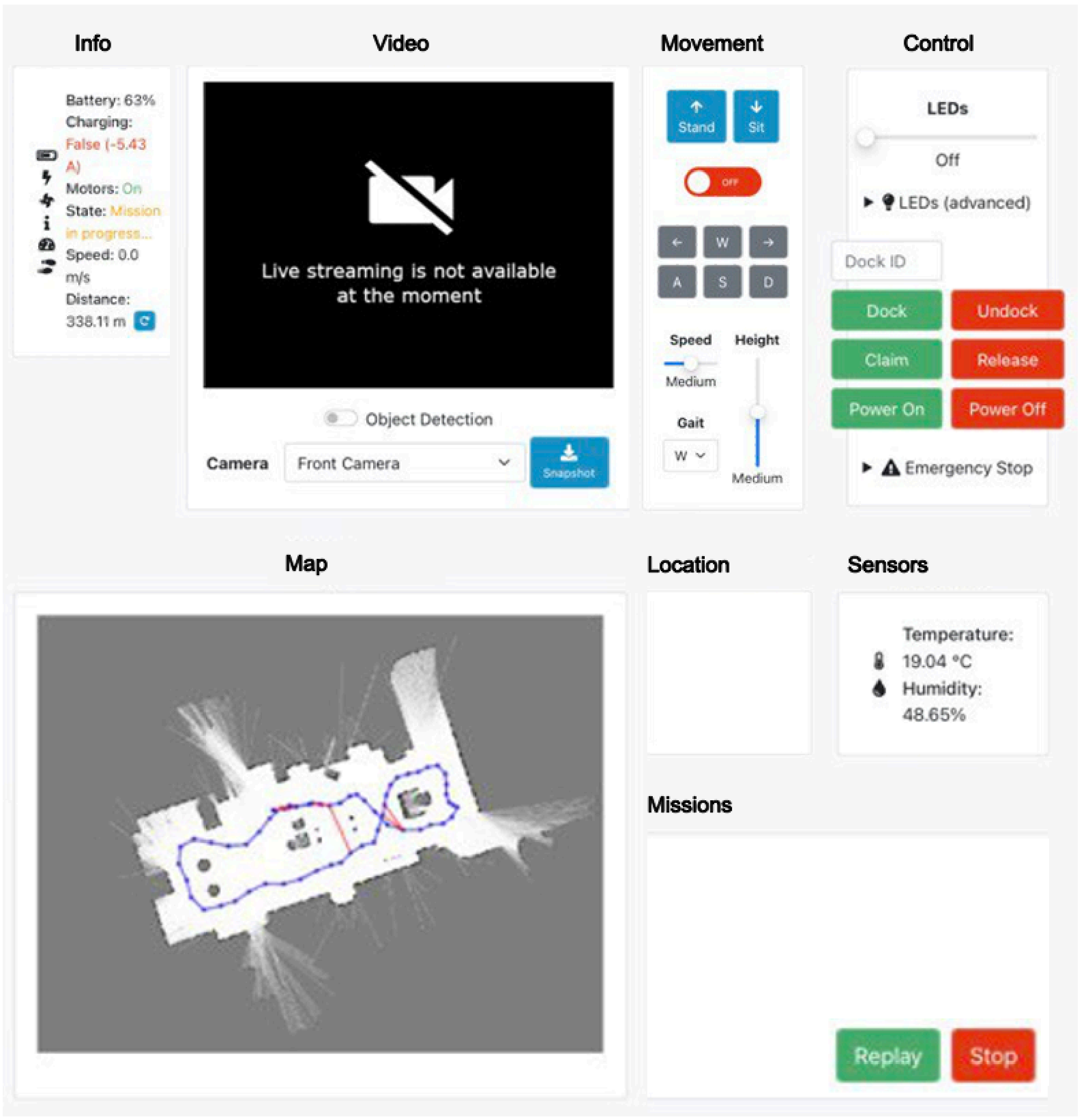
- light payloads, which have no associated Actions.
- medium payloads, which use pre-existing services on Spot.
- heavy payloads, which register their own services with the robot, either directly when attached or more frequently via a payload computer like Spot CORE.

Details about the creation and registration of payload services can be found in the article [32].

3.1.2 Autonomous mission configuration on site

The on-site setup of the developed solution started with a detailed configuration and planning for five consecutive weeks of work. Three different patrol tours have been identified and missions have been configured for each of them. Also, some on-field integration tests have been run i.e., computer vision algorithms combined with robotics. Subsequently, the collection of more on-field photos according to standard patrol rounds and the check over the optimal positions with which to take the photos followed.
During the mentioned five weeks of work, some members of the team worked on-field and some in the lab. After the first week, autonomous missions in the field with the relative collection and interpretation of the first results started being carried out. In particular, the photos taken during missions replaying have been uploaded on a webapp developed on purpose (Fig. 44). Then, the part of the team in the lab has taken care of downloading the photos from the webapp, verifying their quality and the correct execution of the patrol tours. It has been an iterative process of performance verification of the algorithms over an ever-growing dataset. Also, continuous fixes have been necessary when the results were not satisfactory. At the same time, there were regular meetings with the representatives of the client.
Starting from the third week, on-field experiments with Spot Arm have begun. Specifically, this included the opening and closing of doors and arm camera interaction with the components located in unreachable places by Spot Enterprise. It is important to notice that the robot-door interaction can take place only with the help of a human operator: this is the main activity on-field I contributed with.

In parallel with the introduction and the use of Spot Arm, the missions have been replayed continuously to reach the preset target number of daily autonomous patrol tours. This number has been decided in order to make future results reliable as the latter are expressed as success percentages compared to the total number of replayed missions.
During the last week all the data has been gathered and the results started being analyzed. Finally, they have been presented in a steering committee meeting with the client's representatives.



Battery: 63%
Charging: False (-5.43 A)
Motors: On
State: Mission in progress...
Speed: 0.0 m/s
Distance: 338.11 m

Live streaming is not available at the moment

Camera: Front Camera

Temperature: 19.04 °C
Humidity: 48.65%

Fig. 44: appearance of the webapp developed on purpose.

3.1.2.1 Patrol tours definition

Three different areas of interest have been identified in the customer plant. For this reason, three different patrol tours have been defined in the project: MISS01, MISS02 and MISS03. Before going into the details about those patrol tours, some general considerations are necessary. Since only one docking station has been available, it has been decided to define loop patrol tours. This means that the starting and ending point of the robot is the same for all the rounds. Also, it has been decided to make the robot stop to perform the specified Actions only during the outward path; while, in the return path, the robot must come back to the location where its docking station has been placed without intermediate stops. This choice is due to a better management of Spot's battery duration.

3.1.2.1.1 MISS01

This patrol tour is the first that has been configured and it is the longest among the recorded missions in terms of distance traveled: overall, its path covers 1153.84 meters of length. All but one of the meters to be detected were in an outdoor environment; this simplified the mission recording phase as there were no major obstacles for the robot to avoid. On the other hand, exposure to light and changing weather conditions limited the daily time frame in which this round could be repeated. Also, the presence of one door in the path made this mission only partially autonomous. As concerns the mission configuration, during the forward path the robot had to perform twelve stops: eleven for meters' detection and one to upload data on the cloud from which the webapp can retrieve the data. The robot started the mission from the indoor location in which its docking station was placed and then it followed the path represented with a solid green line in the figure below (Fig. 45). Once it reached the area 01 of the plant, represented in the top left corner of the image reported below (Fig. 45), it started performing all the stops to acquire the photos needed exploiting the PTZ camera of the Spot CAM+ payload mounted on top of it. In the figure, the stops performed by Spot and their order of photos acquisition are identified by numbers. To acquire the meters present in the eleventh stop, the robot had to cross a door and then climb some stairs. As for the stairs, there were no problems as Spot is able to cross them if they are wide enough, if the slopes range within 30 degrees and if the height of each step does not exceed 300 mm. The only other precaution was to make it go down the stairs in reverse: this had been done to prevent Spot from losing its balance and falling down the stairs. As regard the door, the presence of a human operator was essential to open and close it as the robot passed. Subsequently Spot, right after walked out of the door, performed the last stop in which all the acquired photos and relative data have been sent to the cloud. Finally, it performed the return path without intermediate stops, reaching the plant location where the docking station had been placed to dock to it and autonomously recharge the battery.

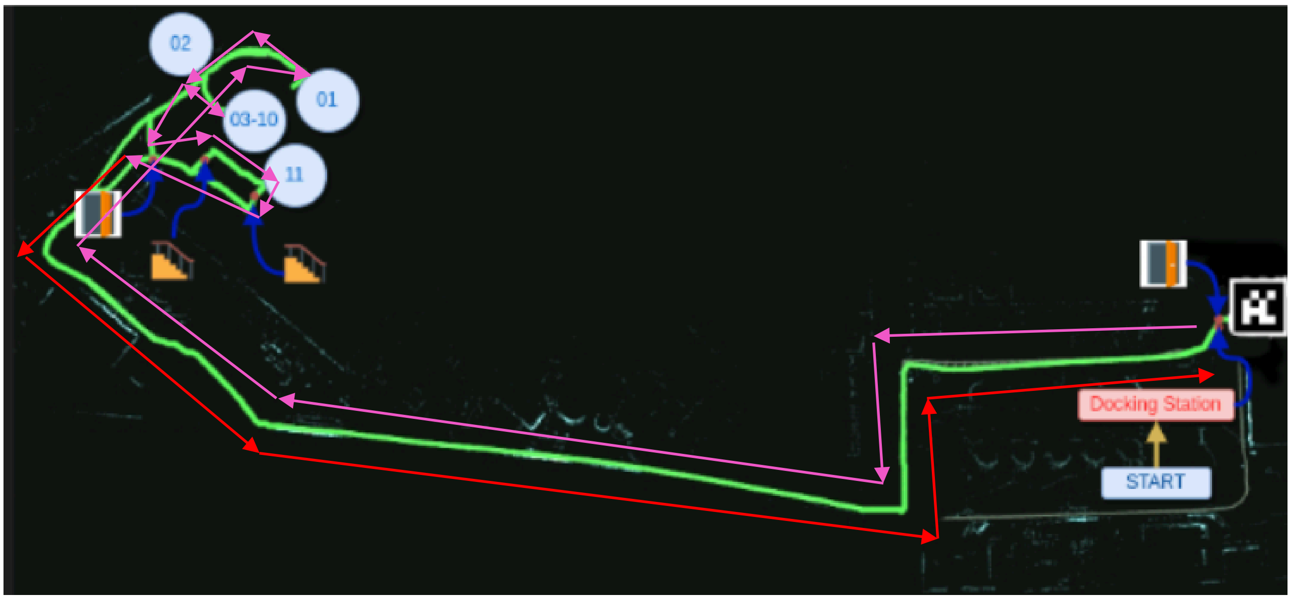


Fig. 45: green solid line identifies the path crossed by Spot in the MISS01 patrol tour; pink arrows depict the direction of the forward path while red arrows depict the direction of the return path.

3.1.2.1.2 MISS02

This is the second patrol tour that has been configured as it required further evaluation for the management of access to indoor environments. Compared to the previous mission, this one is shorter even if it included fifteen stops: fourteen for acquiring photos of the objects of interest and one to upload the data on the cloud. The overall length of the path was 865.53 meters and the objects to be detected were located both in inside and outside environments. Also, stepping platforms and doors were present along the path; the former was used to make Spot reach meters located in positions that were difficult for the robot to reach i.e., the meter in the second and third stops. Also in this case, the mission was started from the same initial point of the previous patrol tour, that is from the place where the robot's docking station was placed. Subsequently, Spot started performing the stops and acquiring photos of the objects of interest. The latter is represented by numbers in the picture below (Fig. 46) and the order of the numbers indicate the order in which the photos have been taken. Notice that there are some missing numbers in the image because those meters have been declared 'out of scope' as they were in positions unreachable by the robot. As concerns the door present before performing the sixth stop, it has been decided to keep it open whenever possible; otherwise, a human operator needed to open and close it to allow the passage of the robot. Once Spot performed all the stops, it uploaded the photos acquired and relative data on the cloud and started the return path, represented by red arrows in the image below (Fig. 46), until it reached the docking station location.

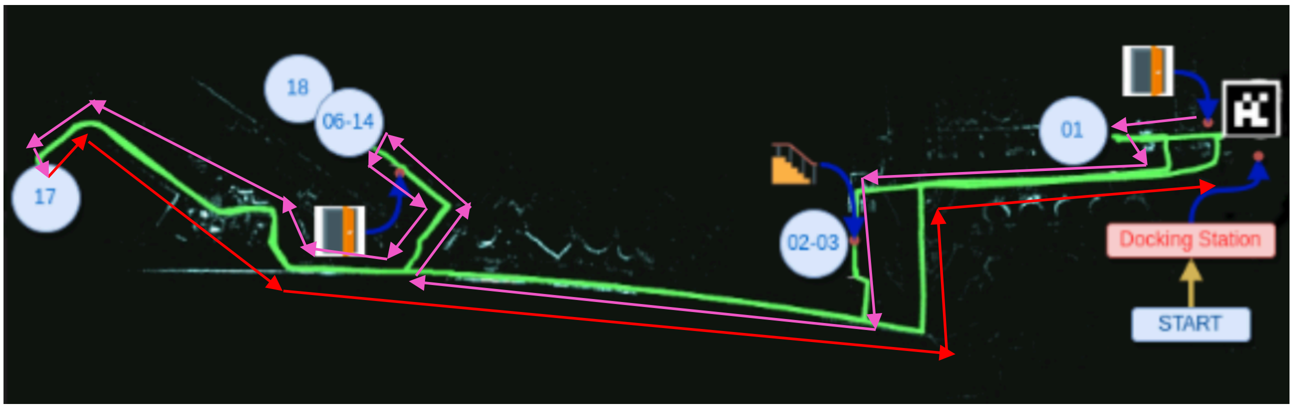


Fig. 46: green solid line identifies the path crossed by Spot in the MISS02 patrol tour; pink arrows depict the direction of the forward path while red arrows depict the direction of the return path.

3.1.2.1.3 MISS03

This patrol tour has been the last to be configured due to the presence of a construction site that prevented the access for about a couple of weeks. Compared to the other missions, this is the shortest both in terms of distance covered, having a path length of 272.22 meters, and of stops performed that are six in total. Four of these were used to detect the meters of interest, one has been used by the robot to upload data on the cloud, and the last stop has been added to the path to allow Spot to wait for the opening of the second door. In this case, in fact, there were two doors that needed to be opened and closed by a human operator: this made the mission only partially autonomous. Moreover, this mission is the one that could be replayed at any time as all the components to be detected are independent from lighting and changing weather conditions since they were placed in an indoor environment. As the previous patrol tours, the robot started the mission from the place where its docking station was located. Then, it began to perform the stops needed to acquire the photos of the objects of interest and to wait for the doors to be opened by a human operator when necessary. Once all the stops were accomplished, Spot uploaded the data on the cloud and started the return path to its docking station. The picture below (Fig. 47) shows both the forward and return paths followed by Spot with a solid green line; to distinguish between them arrows have been added: pink arrows refer to the direction of the forward path while red ones depict the direction of the return path. Also, as in the other missions, the numbers show the stops performed by the robot along the path and the order by which the photos have been acquired. The missing number i.e., the third stop, is not present in the final path configuration as it has been declared 'out of scope' due to unreachability of the meter's position by the robot. In particular, to reach the meter of interest, the robot would have had to climb a stair that was not wide enough for it to traverse.

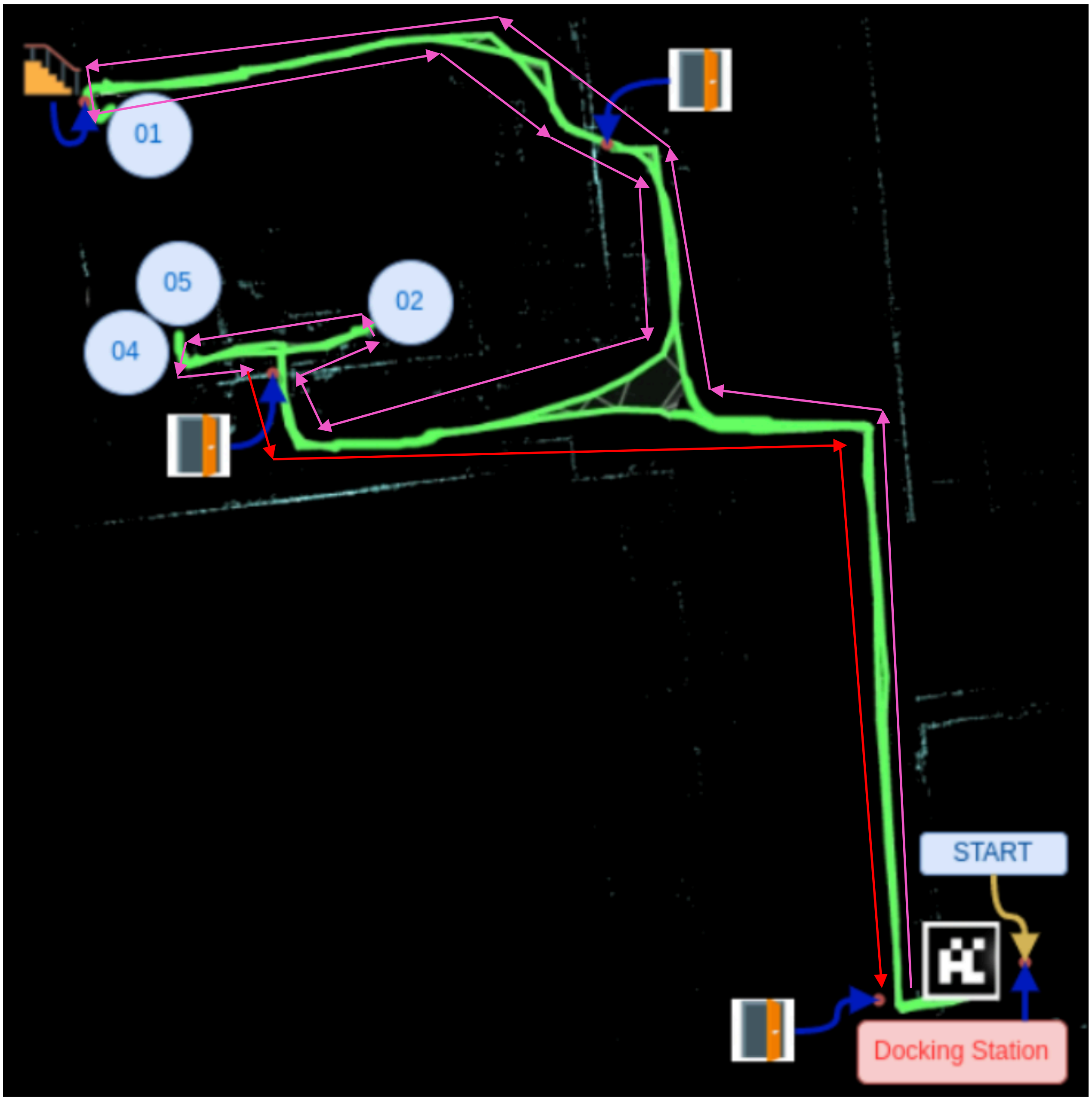


Fig. 47: green solid line identifies the path crossed by Spot in the MISS03 patrol tour; pink arrows depict the direction of the forward path while red arrows depict the direction of the return path.

3.1.2.2 Mission recording

Mission recording is the process by which the robot is instructed on the path to cross and the actions to perform. For the missions configured on site, it has been decided to register loop paths due to the presence of a single docking station. Specifically, the recordings must be done by a human operator who first makes the robot scan a fiducial placed in the same indoor environment where the docking station was located and then, once that fiducial has been successfully scanned, he moves Spot along the

desired path by using the robot’s controller (Fig. 48). In doing this, some precautions have been considered. In particular, the operator had to drive Spot as steadily as possible, that is avoiding abrupt and zigzag movements. In addition, to not interfere with the work of the other stakeholders in the plant and for safety reasons, the robot has been moved along paths suitable for pedestrians. Also, few actions have been added along the paths. In particular, for each component to be detected, at least one Action to perform the photo acquisition has been set. Moreover, actions such as ‘Go Here’ and ‘Pose’ have been added to allow the robot to move to a specified position or to stop itself in order to upload on the cloud the acquired data. Lastly, at the end of each patrol tour, the pre-set action ‘Dock Here’ has been added; in this way, Spot will be able to scan the fiducial placed on the docking station and autonomously dock itself.
It is worth mentioning that Spot's walking speed during the mission recording phase does not affect how fast it walks in the mission replay phase. In fact, in this last phase, the robot's gait could be adjusted via its controller. Also, if the robot is occasionally stopped during mission recording without having configured specific actions, for example due to interferences from the external environment, this stop will not be performed during the mission replay. In some cases, alternative paths have been defined and recorded to allow the robot to avoid obstacles. In particular, in the MISS01 patrol tour, there was often a car in front of the door that Spot had to walk through; for this reason, an alternative path has been recorded in which the robot has been driven outside of the perimeter of the parking lot that was there.

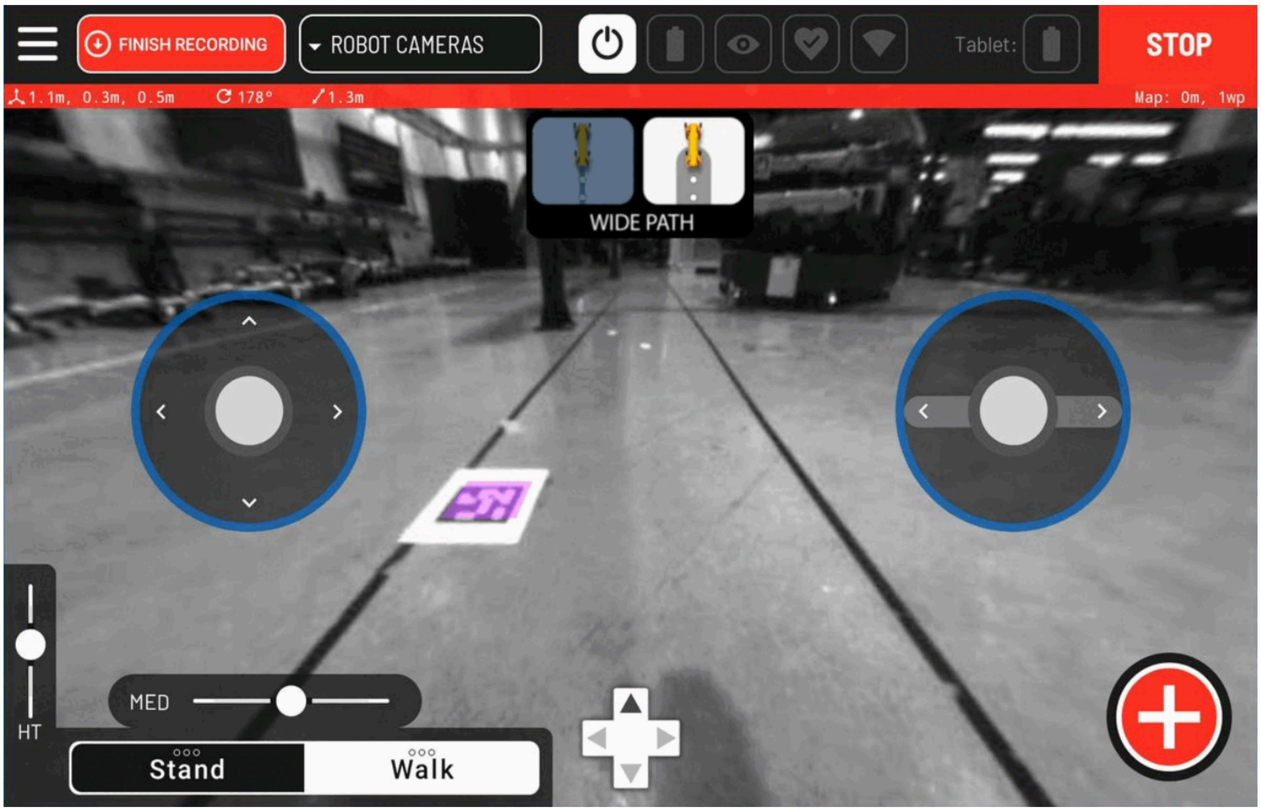
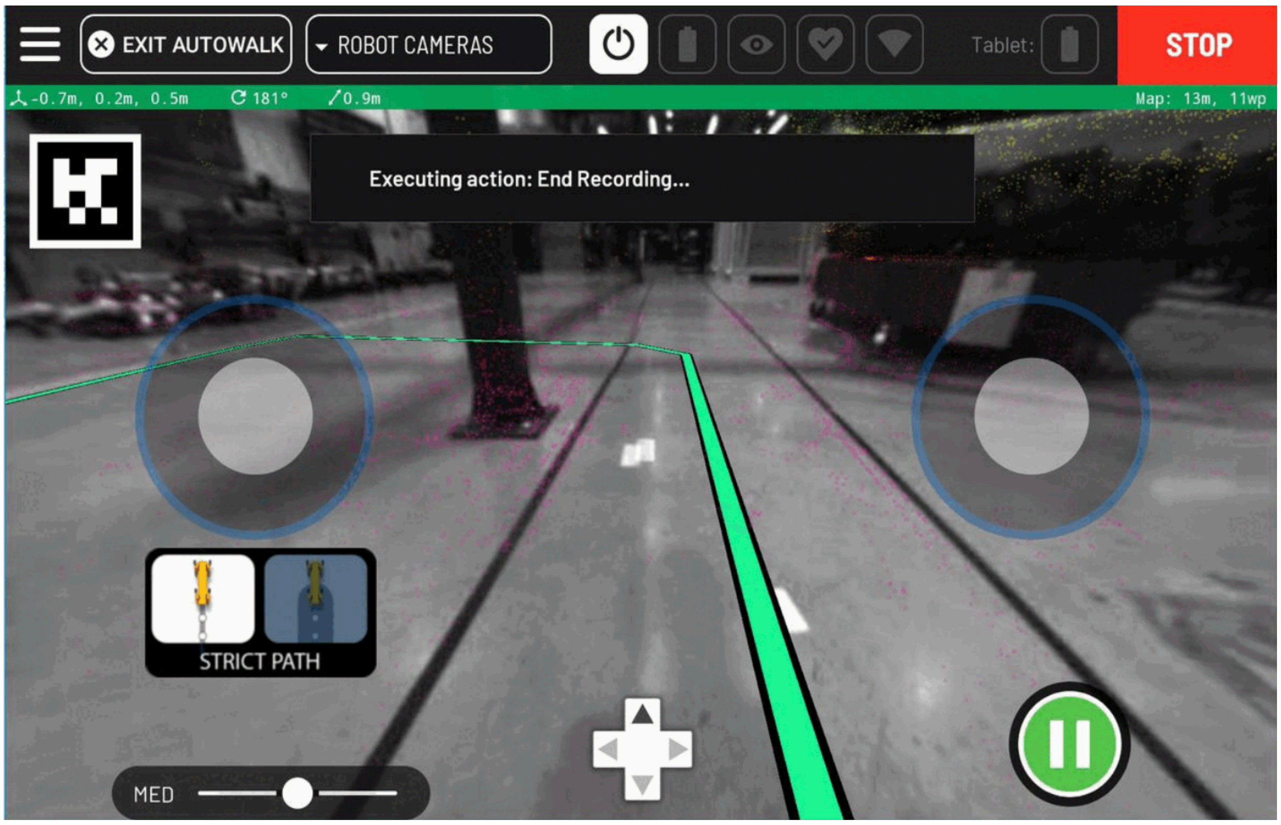


Fig. 48: appearance of the controller during mission recording.

3.1.2.3 Mission replay

After all the missions have been recorded, they could be replayed. In doing this, the operator must select the mission to be performed on the controller and then press the play button. At this point, it is asked to undock the robot and move it sufficiently close to the fiducial to be scanned. Subsequently, Spot starts walking on the configured path and performing the chosen mission (Fig. 49). If needed, the missions can be paused; an example of application of this is the crossing of roads by the robot since it has not been configured to stop when there were oncoming vehicles. Moreover, in case anything changes in the path compared to the one configured during the mission recording phase, such as presence of obstacles, the robot will try to avoid them autonomously; if the obstacle is too big to be overcame and if there are no alternative paths recorded in the same mission, Spot will sit down. In this case, the operator will be prompted to take the control of the robot via the tablet and, if there is no response within the set time interval, the robot will autonomously follow the path taken up to there in reverse, coming back to its docking station. It is also possible to set a number of attempts in which Spot tries to overcome the obstacle: in the case of this project three attempts has been set up.



Executing action: End Recording...

Fig. 49: appearance of the controller during mission replay.

3.2 Testing

To test the precision level of the AI-based readings, two aspects were taken into account: the quality and the accuracy of the photos acquired during missions replay and the effectiveness of the developed computer vision algorithms. As regards the mission replay phase, during the first week of on-field work only the first two patrol tours were recorded i.e., MISS01 and MISS02. Consequently, the tests started on those tours. In doing this, some problems due to weather conditions occurred. In particular, since Spot has an IP54 ingress protection level, it must not walk in outdoor environments while adverse weather conditions such as heavy rain and snow; at most, it can withstand light rain. This caused a reduction in the number of missions replayed. The only mission that could have been replayed even in adverse weather conditions was the third one i.e., MISS03, because its path was defined mostly in an indoor environment; unfortunately, at that time, this mission was not recorded yet because of the presence of a construction site in that area of the plant. Thus, for the first two weeks the MISS01 and MISS02 patrol tours have been replayed whenever possible and then, starting from the third week, mission replay of MISS03 has been joined. Thanks to the prediction of better weather conditions in the remaining weeks of on-site work, it has been decided to fix a minimum number of missions to be replayed each day, that is eight. This allowed to improve the reliability of the results having more data available. Moreover, to ensure the quality and the accuracy of the photos acquired, some parameters of the PTZ camera has been gradually redefined. This allowed to get pictures that were more suitable for the developed algorithms to perform well.
Other problems that occurred include the robot, its firmware, the payloads, and the tablet. Specifically, during the mission replay, the robot had to walk near a wall and, even if the registration was rather smooth, the robot showed a zigzagging gait. This problem was solved by substituting the robot with a newer one that had an updated firmware. Also, the tablet often showed connection faults which led to a sudden stop of the mission when replayed in supervised mode. This problem was solved by replacing the tablet with a newer version. Lastly, the PTZ camera showed some random faults probably due to degraded connectivity to the robot. This was not solved during the on-site pilot, but it will be solved in future by replacing the camera with a newer model.
As concerns the testing of the developed computer vision algorithms, adjustments and finetuning have been performed in parallel to the five weeks of on field work. In particular, those adjustments regard mainly changes of parameters in the configuration files. For example, in the 'round digital water meters' algorithm, changes occurred in the upper and lower bounds of the display color to be identified. This was necessary to take different lighting conditions into account since the acquisition of the photos of interest depended on the changing weather conditions. Other refinements concerned the cropping area of the region of interest which in some cases was too small with respect to the reading to be extracted. Thus, exploiting a

trial-and-error procedure, the parameters in the configuration files were set in such a way that more and more meters and gauges were read correctly. This allowed to also test the robustness of the implemented codes and to reach higher percentages of success rates.

3.3 On-field manipulation with Spot Arm

The three missions recorded and replayed were only partially autonomous due to the presence of major obstacles, such as doors, that Spot Enterprise was not able to overcome. The solution adopted until now included the presence of a human operator that needed to open and close the doors in order to allow the passage of the robot. Starting from the third week of on-site work and thanks to the availability of the new robot from Reply's side, Spot Arm was joined to Spot Enterprise on an experimental level; it was employed in substitution to the human operator in order to open and close the doors that were present along the paths of the configured missions. In particular, the operator had to maneuver the robot and its arm by using its controller. To do this, he had to enter in the Manipulation Menu and drive Spot Arm in front of the door to be opened with a distance of 1.5 meters from it. Once the robot was positioned, the operator had to select the 'Operate arm' under Drive Behaviors (Fig. 50): this made the robot be in standing position and the arm be in stow position (Fig. 51) so as to prevent accidents with the operator and the surrounding environment. Subsequently, the operator had to select the 'Grasp Wizard' button under Grasp Behaviors; in this way, he will be prompted to indicate the position of the door's handle by tapping it on the screen (Fig. 52). After this, the Grasp Wizard Menu will appear (Fig. 53) and the operator needs to adjust the grasp parameters to reflect the best approach angle, gripper location, and gripper orientation to approach the door's handle. The robot may re-position itself while modifying those parameters, but it will not attempt to grasp the door's handle yet. By clicking on 'next' on the controller, a new screen will appear (Fig. 54) and the operator can choose if the grasp behavior has to be done with the palm or with the fingertip; in the case of this project, generally a fingertip range within 40% to 60% has been set. Finally, the operator tapped on the 'Auto Grasp' button which made Spot Arm to execute the grasp based on the pre-defined parameters set by the operator. At this point, if the grasp behavior is successful, the operator is prompted to select a post-grasp action (Fig. 50). For all the types of door present in the customer's plant, it was necessary to select the 'Twist, turn, pull' button which enable the robot to manipulate objects that are constrained in how they move. In particular, the 'Turn Knob' option was selected. A new screen appeared on the controller and the operator configured the basic parameters such as selecting the direction around which to apply torque and the intensity of the force to apply (Fig. 55). Lastly, the operator made the door handle turn clockwise or counterclockwise by moving the joystick up or down. After performing all those actions, the door's handle will be grasped and turned by the robot, but the door will be still closed because one last step is missing. This regards the 'Drag' action among the post-grasp actions. By clicking on that button, the operator will see on the controller's screen the rear camera of the robot and using the joystick he can move Spot Arm backward to make the door open. The explained sequence of operations has been done for each door present along the path of the configured missions with slight modification if needed. It is worth noticing that Spot Arm is provided with an 'Open Door' pre-configured grasp behavior but the reason why it has not been used for the purposes of this project is because, with this latter functionality, the robot will open the door and enter inside provided that the operator indicates to it via tablet the position of the door's handle and of the hinge and whether it has to push or pull it. To the aim of this project, it was needed to open the door with Spot Arm allowing the passage of Spot Enterprise inside (Fig. 56). The limit of Spot Arm to not perform the mentioned actions completely autonomously is due to safety reasons; maybe it will be possible to make those missions fully autonomous in the near future, but the outcome of this depends on the development plans of Spot's mother company. Maneuvering Spot Arm and making it open/close doors is how I contributed to the project during the on-field work.

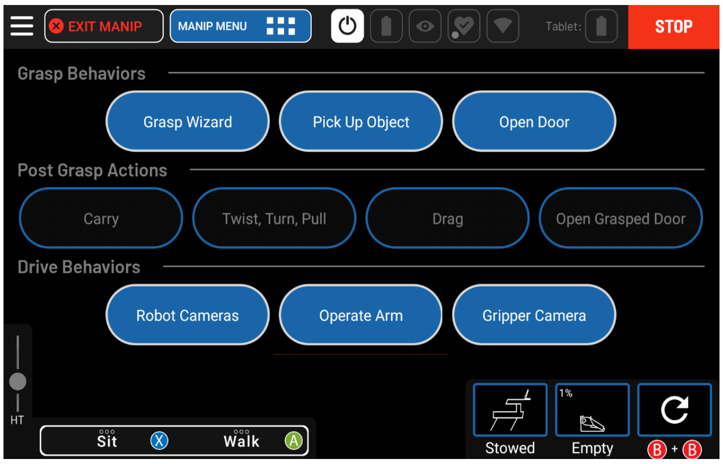


Fig. 50: appearance of the Manipulation Menu.

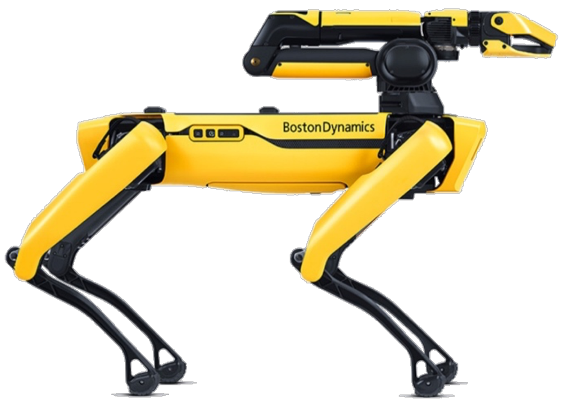


Fig. 51: Spot Arm in stow position.

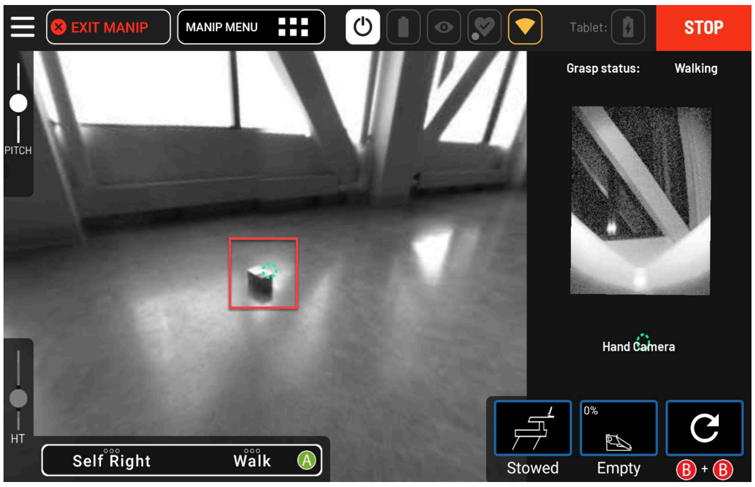


Fig. 52: appearance of the controller when it is used to indicate the position of the door's handle by tapping it on the screen.

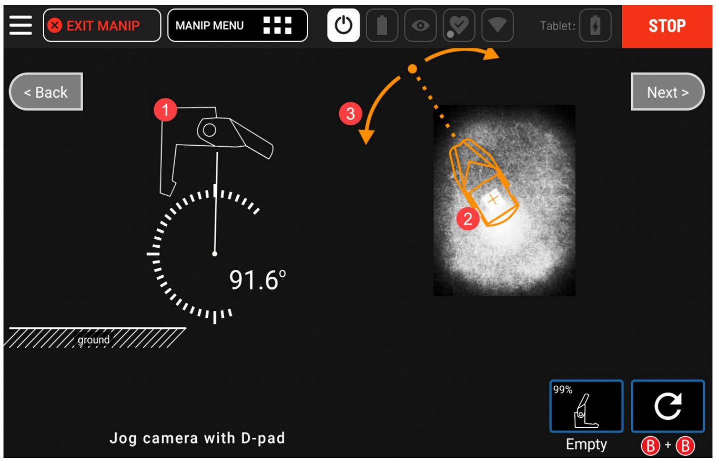


Fig. 53: screen view showing how to set the approach angle (1), the gripper location (2) and gripper orientation (3) to approach the door's handle.

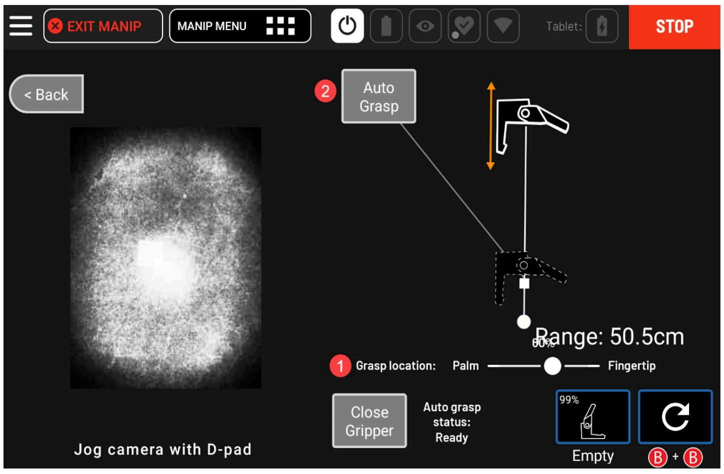


Fig. 54: screen view showing how to set the grasp location (1) and the Auto Grasp button (2).

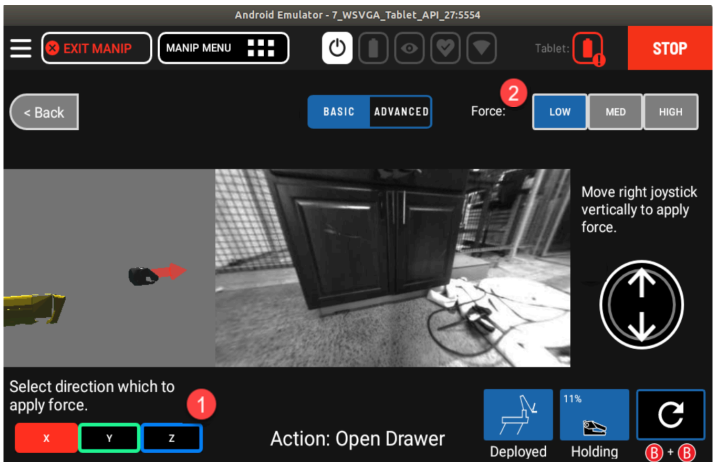

Android Emulator - 7_WSVGA_Tablet_API_27:5554
EXIT MANIP
MANIP MENU
Tablet:
STOP
< Back
BASIC
ADVANCED
Force:
2
LOW
MED
HIGH
Move right joystick vertically to apply force.
Select direction which to apply force.
1
X
Y
Z
Action: Open Drawer
11%
Deployed
Holding
B + B

Fig. 55: screen view showing how to set the direction in which to apply torque (1) and the intensity of the force to apply (2).

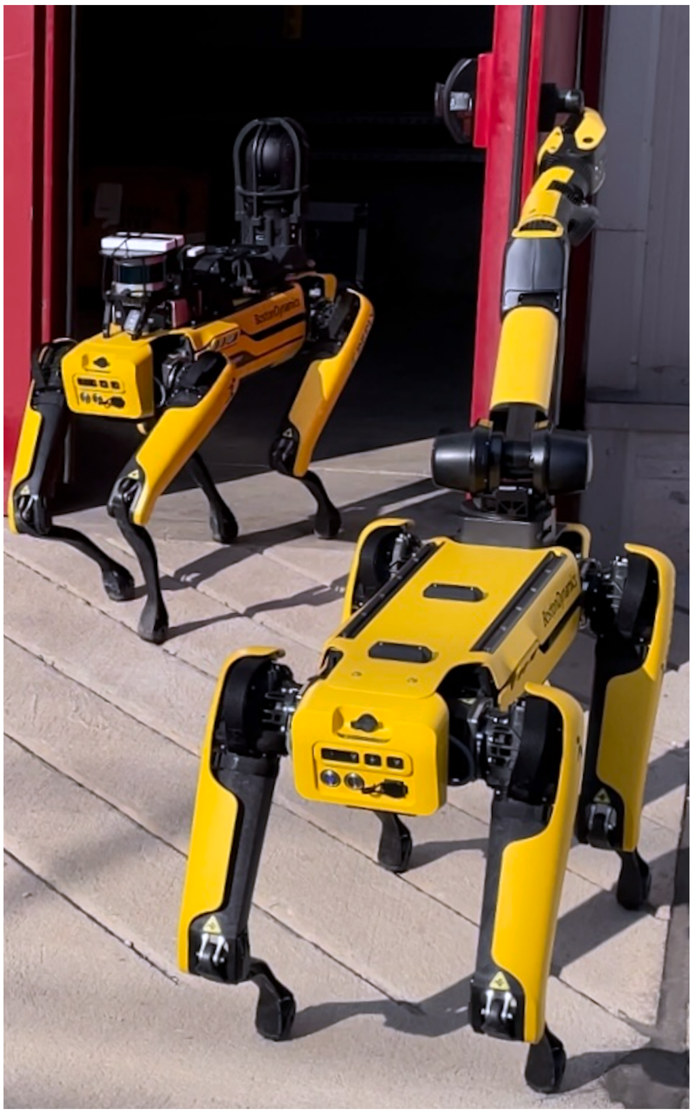


Fig. 56: on-field collaboration between Spot Enterprise and Spot Arm.

3.4 Results and discussion

The AI-based readings of the meters' photos obtained from the missions configured in the customer's plant have been collected in an Excel performance file in order to be analyzed and validated. The validation has been done comparing the human eye and the artificial readings and, for each measurement, a label has been associated depending on the cases. From here the success rates for each component has been extrapolated and finally the average success rate for each mission has been estimated. In the following, more details will be presented.

3.4.1 Results

All the obtained results have been carried out during the on-field test phase that lasted about a month. Several missions have been replayed but some of them have not been taken into consideration for the results purposes. This is because the first repetitions carried out for each mission were used for fine-tuning some parameters, for example adjusting the pointing of the PTZ camera. The objective of these tests was to collect statistics about the accuracy of the AI-based readings of the objects of interest and evaluate their reliability. In the table below [Table 4], it is reported the number of components that were present along each path, the number of repetitions performed, and the kilometers traveled by Spot during each mission. In addition, their total values are also outlined. It can be observed that 141 overall missions have been replayed. Among these, 57 belong to MISS01 patrol tour which has been replayed the most because it was the first to be recorded; therefore, several repetitions were needed in order to fix some robotic bugs faced in the early days. The remaining 84 repetitions are almost equally split between the MISS02 and MISS03 patrol tours. The latter is the mission with the smaller number of repetitions because it was the last to be recorded due to the presence of a construction site in that area of the plant; thus, all the main faults occurred in the first days had already been fixed when this patrol tour has been configured.

Mission ID	**Number of components**	**Number of repetitions**	**Km travelled**
MISS01 MISS02 MISS03	11 16 5	57 44 40	54.2 38.1 10.9
Total	32	141	103.1

Table 4: counting of the number of components, of the tests carried out, and of the kilometers traveled divided by mission.

In the figure below (Fig. 57), the total number of tests performed each day is reported. Evidently, the number of tests has grown over time. Also, two time frames are clearly distinguishable by the number of tests performed i.e., before and after day 13. In the first twelve days, a total of 40 mission replays has been carried out with an average of only 3.63 iterations per day. Instead, in the remaining period of time, it has been reached a total of 101 repetitions that led to a higher average of daily iterations, that is 8.42. The substantial difference of results obtained depends on the additional activities that may have been accomplished. In particular, during the first period, some time was spent in fine-tuning general aspects of the missions and in improving the computer vision algorithms. This caused slowdowns and led to a reduction in the number of tests performed. Subsequently, once all the main faults had been solved, most of the attention focused on the execution of on-field tests. The presence of some minimum points or totally missing days in the figure below (Fig. 57) could be due to unfavorable weather conditions which prevented the robot from walking outdoors.

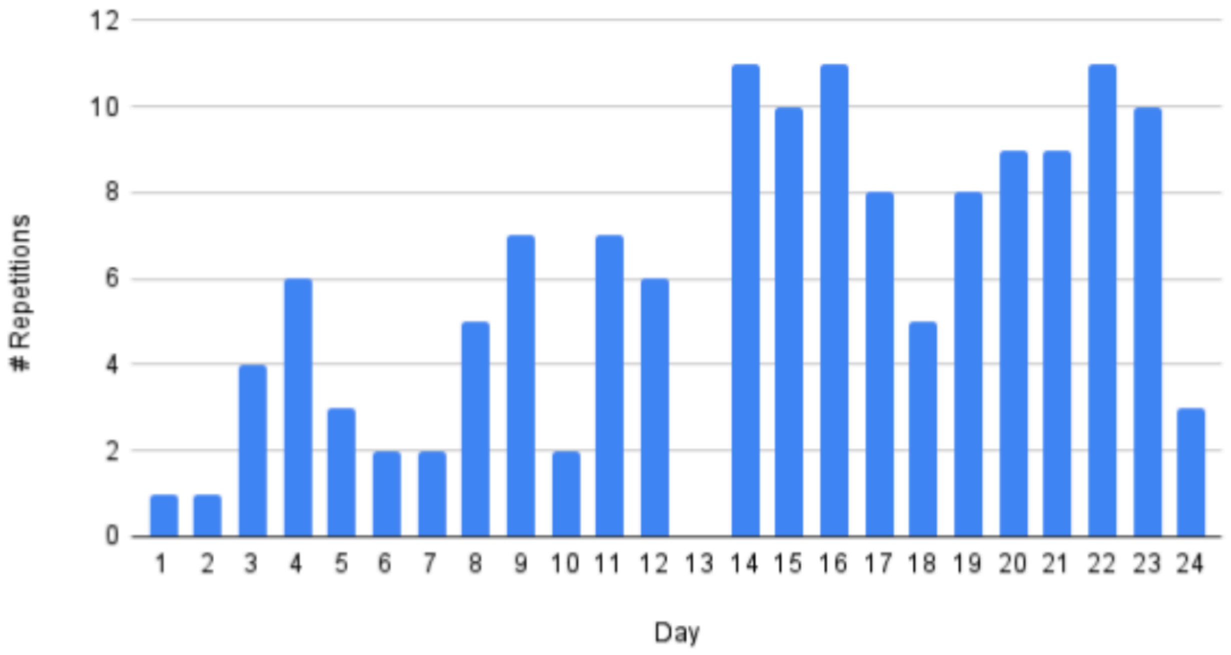


Fig. 57: total number of tests carried out each day.

The results obtained have also been collected separately for each mission to analyze them better. In the figures below (Fig. 58)(Fig. 59)(Fig. 60), it is depicted the number of daily repetitions for all the configured missions. It is worth noticing that, since the presence of a construction site slowed down the configuration of the MISS03 patrol tour, most of the mission repetitions made in the last few days of on-site work involved this patrol tour (Fig. 60). This choice has been done on purpose in order to reach a number of tests that is comparable with the other missions and to collect enough data to get significant and accurate statistics.

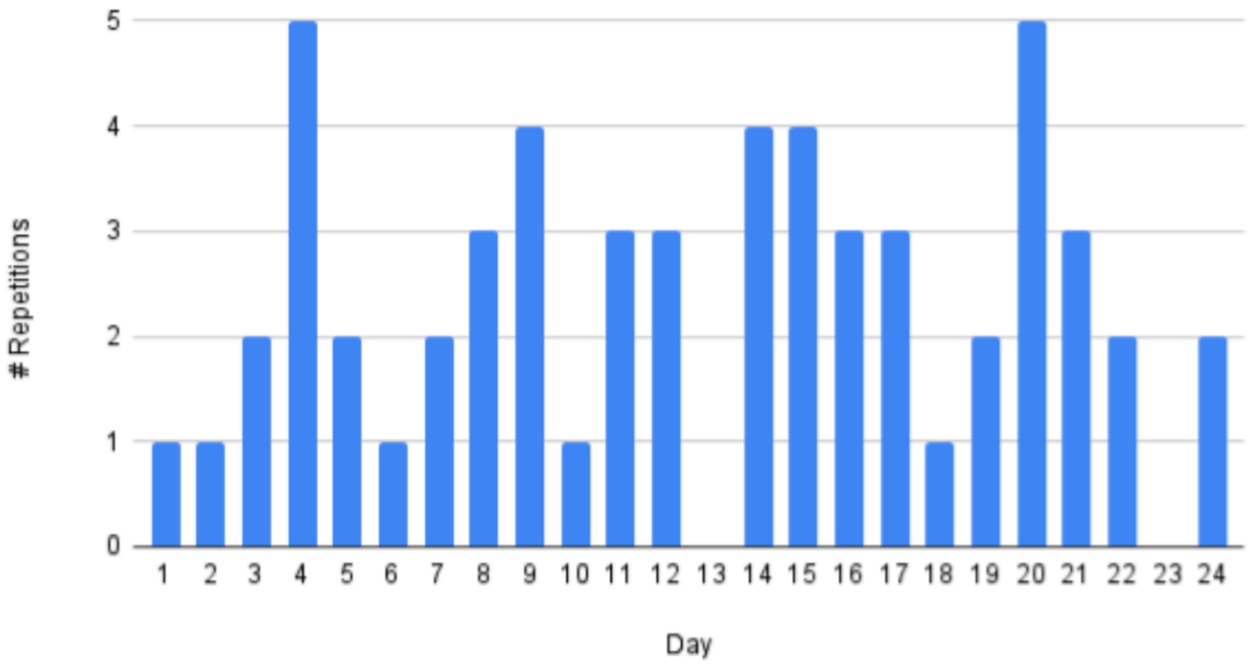


Fig. 58: daily repetitions for mission MISS01.

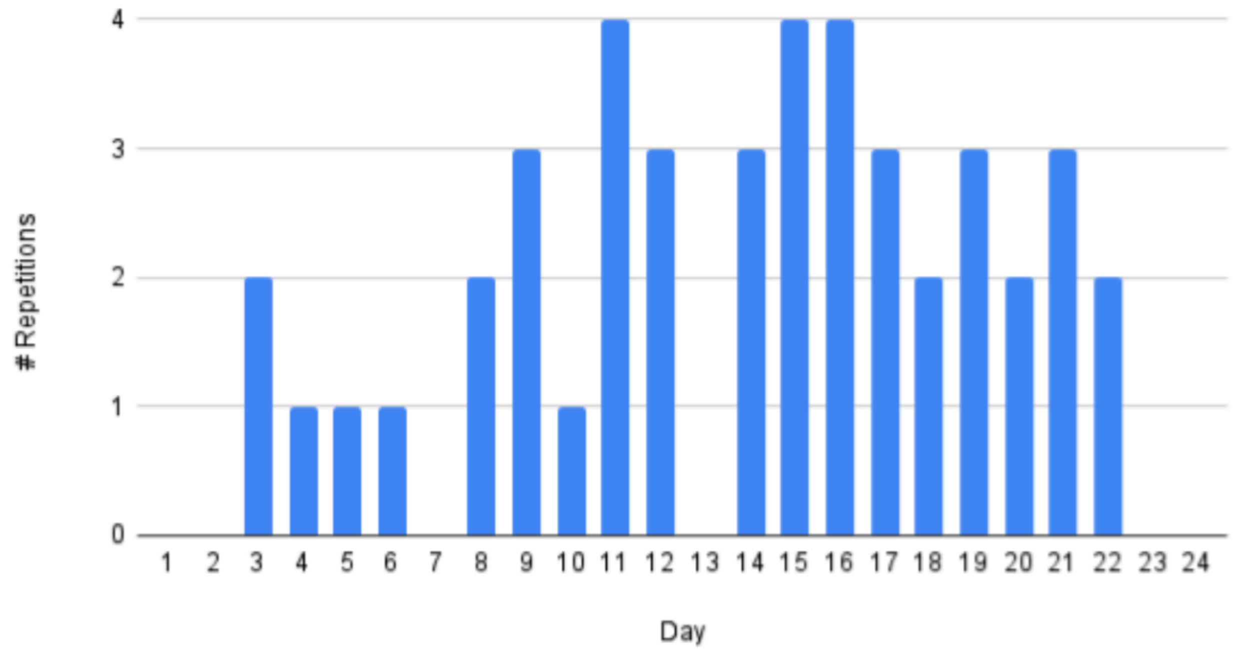


Fig. 59: daily repetitions for mission MISS02.

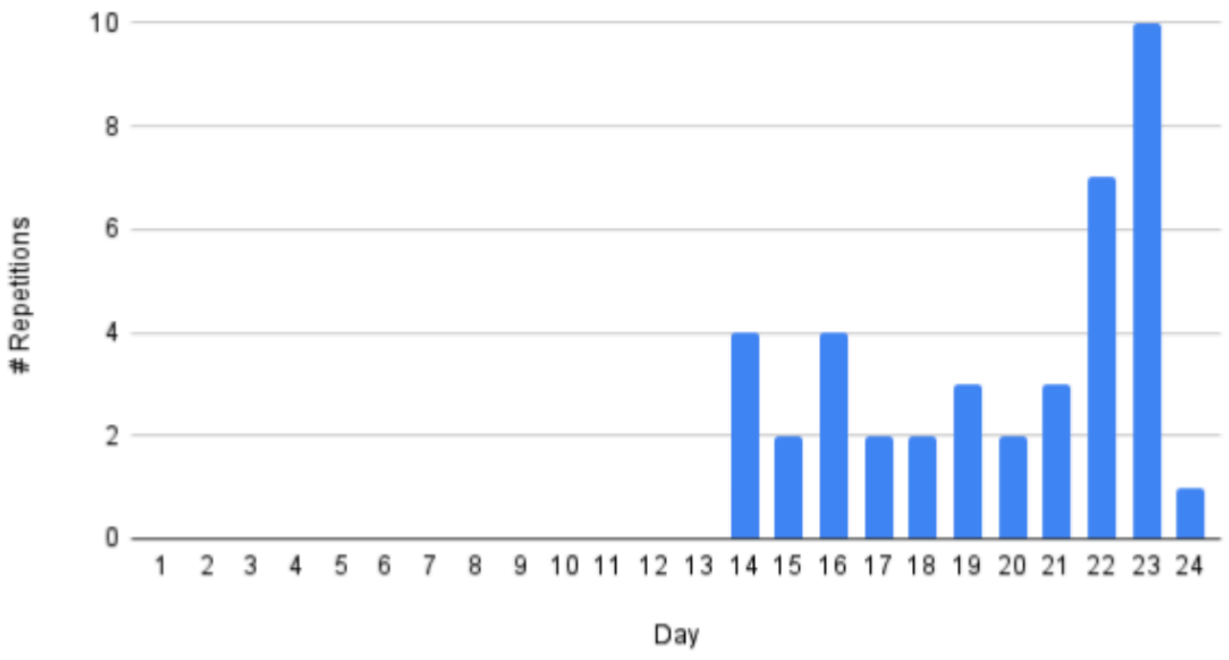


Fig. 60: daily repetitions for mission MISS03.

3.4.2 Validation

Validation refers to the action of checking or proving the validity or accuracy of something. For the purposes of this project, the outcome carried out from the AI-based readings of each component has been collected in an Excel performance file. Subsequently, by comparing artificial and human eye readings, a label has been associated for each of them. Depending on the success or failure of the readings, three main labels could be associated: 'WORK IN PROGRESS', 'OK', 'KO'. The first label has been mostly used in the case in which the AI reading was not yet present in the performance sheet during the testing period. Sometimes, it has been associated when there was no image to be processed at all. This could have been happened for two main reasons: due to failures of the camera or due to the robot skipping the callback. Note that the latter has been done intentionally for the components that have been previously defined as 'out of scope'. The remaining two labels i.e., 'OK' and 'KO', have been used in antithesis; therefore, if the artificial and human readings differ by less than a certain tolerance the result has been labelled as 'OK', otherwise it has been labelled as 'KO'. Precisely the result has been considered successful if:

$$\text{AI\_reading} \in [0.8 * \text{true\_reading}, 1.2 * \text{true\_reading}]$$

This constraint has been decided in agreement with the customer mainly for simplicity reasons, however a better idea would have been to set different tolerance percentages for each measuring instrument; in this way, it would have been possible to take into account the singular problems or faults, which may not be resolvable in the short term, that have emerged for each component.

It is worth mentioning that the ‘KO’ label included different types of faults, resumed in the table below [Table 5]. Specifically, this is a terminology, not disclosed with the customer, adopted to help the developer to understand in detail the source of the error. In the following a brief overview with detailed explanation is presented:

- ROBOT STOP MISSING: the robot skipped the callback because it was not able to reach the target meter due to both accidental causes, such as presence of construction sites or closed doors, and intentional causes, such as temporary exclusion from the mission due to technical reasons.
- AUTOFOCUS KO: the PTZ camera mounted on the robot was out of focus which resulted in a blurry picture of the target component.
- DETECTION KO: the computer vision algorithm associated to the target measuring instrument has partially or not at all captured the component.
- UNREADABLE KO: presence of light reflections or inconsistency in the digits, for example in the mechanical water meters in which there were numbers that scrolled rapidly and therefore in some cases it was difficult to read the measurement even by the human eye.
- DARK KO: lack of a sufficient level of brightness which made it critical or impossible to read the measurement both by the human eye and/or by the AI algorithm.
- DIRTY KO: presence of dirt on the component that made difficult to read the measurement.
- DECIMAL SEPARATOR KO: the decimal separator was misplaced by the algorithm, even if the latter correctly read the measurement. This is a common error that occurred only for digital indicators. In some cases, when the number of decimals was fixed, the solution has been to insert the decimal separator in a programmatic way.
- AI READING KO: the algorithm failed in reading the measurement even if the input image has been correctly taken.

Internal KO status	**KO type**
ROBOT STOP MISSING	Robot KO
AUTOFOCUS KO	Point Engine KO
DETECTION KO	Point Engine KO
UNREADABLE KO	External KO
DARK KO	External KO
DIRTY KO	External KO
DECIMAL SEPARATOR KO	AI KO
AI READING KO	AI KO

Table 5: resume of the different types of ‘KO’ label.

3.4.3 Discussion

3.4.3.1 Overview

The missions replayed during the on-site work resulted in 141 repetitions, with a total of 1531 images used to feed the computer vision algorithms. Among these, 1007 AI-based readings have been classified as 'OK', meaning that the algorithms correctly extracted the reading according to the success criteria specified in the previous section. Of the remaining 524 reading, 505 instances have been labelled as 'KO' due to one or more failures specified in the previous table [Table 5]; while the other 19 have been left in the 'WORK IN PROGRESS' state. The latter refers to the photos that the algorithms were not able to analyze for reasons still under investigation. The table below [Table 6], reports a resume of the number of components associated with each label and the relative percentage values. It can also be seen that the overall success percentage of the proposed solution is 65.77%.

Outcome	**Count**	**Percentage**
WORK IN PROGRESS	19	1.24 %
KO	505	32.98 %
OK	1007	65.77 %
Total	1531	100 %

Table 6: counting of the number of components and relative percentage associated with a specific label.

3.4.3.2 Error analysis

As regards the measurements associated with the 'KO' label, it is possible to classify them into different sub-labels based on their cause. This allowed the developers to extract insights on the performances of the computer vision algorithms, on the behavior of Spot, and on the PTZ camera mounted on the robot. Specifically, among the 505 'KO' instances, only 439 involved the shooting of a photo of some point of interest by the PTZ Camera. There are different reasons why they had been labelled as 'KO', such as the environment, the pointing/focus of the PTZ camera, the robot, and the problems with the AI-readings. In fact, 44 instances out of 439 were due to environmental conditions as reflections over the screen, presence of dirt on the indicators, poor lighting conditions. In 110 cases out of 439 instead, there were problems in the information interpretation by the algorithms because they require substantial changes to the solution proposed or because the image provided as input is simply not readable. Of the remaining 285 instances, 58 were due to a robot fault and 227 were actually ‘KO’ due to the AI-based algorithms. The former can be in general considered as isolated cases that should not affect the evaluation of the overall performance of the system; in some cases, it can be exploited as an indication on the adaptations that should be made to the environment to prevent the failure from happening again in the future. Excluding the ‘KO’ due to the AI-based algorithms, their actual outcome is obtained. In numbers, there are 227 results labelled as ‘KO’ due to interpretation problems by computer vision algorithms compared to 1007 results labelled as ‘OK’; respectively they represent 15.70% and 69.64% of the total checkpoints as shown in the table below [Table 7]. The sum of the remaining ‘KO’ i.e., due to the environment, the pointing/focus of the PTZ camera, and the robot, constitute 14.66% of the total. Consequently, it is possible to conclude that the external and internal factors to the proposed solution have the same impact on its feasibility, therefore improvements must be made in both directions.

Outcome	**Count**	**Percentage**
KO – Environment	44	3.04 %
KO – AI	227	15.70 %
KO – Pointing/Focus	110	7.61 %
KO – Robot	58	4.01 %
OK	1007	69.64 %
Total	1446	100 %

Table 7: counting of the number of components and relative percentage for success and failures.

3.4.3.3 Missions error analysis

Focusing the error analysis on the outcome of each configured mission, the success and failure percentages mission per mission can be seen from the table below [Table 8]. In particular, MISS01 registered the highest success percentage among the missions, that is of 73%. Instead, MISS02 and MISS03 scored a success rate respectively of 67% and 68%. As concerns the counting evaluations, the first two patrol tours show a major weight compared to the last one: this is because they involved more checkpoints. Also, mission replays of MISS03 started later than the other missions so, to fit within the five-week time constraints, attempts were made in vain to match the number of mission replays, but this led to the three tours being unbalanced. In fact, MISS03 only represents around 15% of the total gathered data.

Mission ID	**OK Count**	**KO Count**	**Total**	**OK Percentage**	**KO Percentage**
MISS01 MISS02 MISS03	446 430 131	162 213 64	608 643 195	73 % 67 % 68 %	27 % 33 % 32 %
Total	1007	439	1446	70 %	30 %

Table 8: success and failure counting and percentages mission per mission.

To the performance evaluation purposes, the outcomes labelled as 'OK' and the ones labelled as 'KO', but only due to problems with the computer vision algorithms, have been considered. The combination of these represents a total of 1234 readings. As specified previously, the number of successes is 1007, which represents a rate of 81.6%. From the table reported below [Table 9], it can be noted that the highest success rate is on the mission MISS01, which reached a percentage of 85%; instead, the highest failure rate is given by mission MISS03 which obtained a success rate only of 74%.

Count	**OK**	**KO – AI**	**Total**	**OK Percentage**
MISS01 MISS02 MISS03	446 430 131	76 105 46	522 535 177	85 % 80 % 74 %
Total	1007	227	1234	81.6 %

Table 9: success rates for each mission.

In order to have a more in-depth understanding of the previously explained numbers, the breakdown of all the checkpoints of the configured missions has been reported in the following tables [Table 10][Table 11][Table 12].

Stop number	**Type of meter**	**OK**	**KO – Environment**	**KO – Robot**	**KO – AI**	**KO – Focus**
01	Analog Gauge	51.8 %	1.8 %	0.0 %	42.9 %	3.6 %
02	Vessel Level Indicator	73.7 %	5.3 %	5.3 %	12.3 %	3.5 %
03	Round Digital Gauge	25.0 %	28.6 %	10.7 %	10.7 %	25.0 %
04	Round Digital Gauge	69.1 %	18.2 %	0.0 %	10.9 %	1.8 %
05	Round Digital Gauge	96.5 %	0.0 %	0.0 %	1.8 %	1.8 %
06	Status Indicator Valve	90.6 %	0.0 %	3.8 %	3.8 %	1.9 %
07	Round Digital Gauge	60.0 %	14.5 %	0.0 %	16.4 %	9.1 %
08	Analog Gauge	100 %	0.0 %	0.0 %	0.0 %	0.0 %
09	Analog Gauge	80.4 %	1.8 %	0.0 %	16.1 %	1.8 %
10	Round Digital Gauge	73.7 %	0.0 %	0.0 %	21.1 %	5.3 %
11	Status Indicator Valve	100 %	0.0 %	0.0 %	0.0 %	0.0 %
Total	–	74.1 %	6.5 %	1.8 %	12.6 %	5.0 %

Table 10: breakdown of all the checkpoints of mission MISS01.

Stop number	Type of meter	OK	KO – Environment	KO – Robot	KO – AI	KO – Focus
01	Mechanical Water Gauge	25.0 %	2.3 %	4.5 %	45.5 %	22.7 %
02	Mechanical Water Gauge	17.9 %	0.0 %	0.0 %	66.7 %	15.4 %
03	Mechanical Water Gauge	31.8 %	6.8 %	2.3 %	31.8 %	27.3 %
06	Mechanical Water Gauge	68.2 %	0.0 %	2.3 %	27.3 %	2.3 %
07	Square Digital Water Gauge	81.8 %	0.0 %	4.5 %	4.5 %	9.1 %
08	Square Digital Water Gauge	88.6 %	0.0 %	4.5 %	6.8 %	0.0 %
09	Square Digital Water Gauge	88.6 %	0.0 %	4.5 %	0.0 %	6.8 %
10	Round Digital Water Gauge	72.7 %	0.0 %	2.3 %	6.8 %	18.2 %
11	Round Digital Water Gauge	93.2 %	0.0 %	2.3 %	2.3 %	2.3 %
12	Square Digital Water Gauge	84.1 %	0.0 %	4.5 %	0.0 %	11.4 %
13	Square Digital Water Gauge	75.0 %	0.0 %	4.5 %	0.0 %	20.5 %
14.1	Osmosis Display	84.0 %	0.0 %	8.0 %	8.0 %	0.0 %
14.2	Osmosis Display	48.0 %	0.0 %	8.0 %	36.0 %	8.0 %
14.3	Osmosis Display	88.0 %	0.0 %	8.0 %	4.0 %	0.0 %
17	Mechanical Water Gauge	47.6 %	2.4 %	9.5 %	21.4 %	19.0 %
18	Rectangular Digital Display	83.3 %	0.0 %	4.8 %	7.1 %	4.8 %
Total	–	67.3 %	0.8 %	4.4 %	16.4 %	11.1 %

Table 11: breakdown of all the checkpoints of mission MISS02.

Stop number	**Type of meter**	**OK**	**KO – Environment**	**KO – Robot**	**KO – AI**	**KO – Focus**
01	Analog Gauge	5.0 %	0.0 %	0.0 %	77.5 %	17.5 %
02	Round Digital Gauge	80.0 %	0.0 %	5.0 %	12.5 %	2.5 %
02-bis	Round Digital Gauge	69.4 %	0.0 %	5.6 %	25.0 %	0.0 %
04	Liquid Level Indicator	94.7 %	0.0 %	2.6 %	2.6 %	0.0 %
05	Rectangular Digital Display	92.3 %	0.0 %	5.1 %	0.0 %	2.6 %
Total	–	67.9 %	0.0 %	3.6 %	23.8 %	4.7 %

Table 12: breakdown of all the checkpoints of mission MISS03.

4. Conclusion

Three different patrol tours have been configured successfully exploiting Spot, the quadruped canine-inspired robot from Boston Dynamics. In particular, the robot captured high-resolution photos of 29 measuring instruments across 141 missions' repetitions, averaging 5875 replays per day. The collected data have been uploaded on the cloud at the end of every mission in order to be processed by the respective computer vision algorithm, which have been implemented in the Python programming language. Overall, 1531 photos constituted the dataset and, of these, almost 70% have been successfully interpreted; although, there is still room for improvement regarding the performance of some algorithms. Considering only the failures that did not depended on the customer's plant conditions, a total of 1234 photos remained in the dataset; these led to an interpretation success rate of the algorithms of 81.6%. During the missions' repetitions, the only human intervention involved regarded the opening and closing of doors to access indoor environments. This has been later substituted with Spot Arm, on an experimental level, despite it still needed the help of a human operator to be maneuvered. To improve this functionality, it is possible to develop algorithms making use of the SDK provided by the robot's mother company; in this way, the configured missions become fully autonomous. Alternatively, improvements may be made in the future when Boston Dynamics will provide the ability for Spot Arm to open and close doors by itself without the robot entering inside. Also, improvements can be done concerning the robustness of the configured missions. For example, adding from the webapp the 'Take Shortcut' functionality to the mission replay might be useful in reducing the length of some mission path sections. This should be accompanied with the addition to the mission of more alternative paths so that it becomes more robust to unexpected obstacles on its way. Furthermore, to make the missions more robust to unexpected events, an additional feature could be to set the possibility to take multiple shots of the same component from different positions and angles.

5. Bibliography and sitography

[1] Pallardy, Richard. "Deepwater Horizon oil spill". Encyclopedia Britannica, 23 Aug. 2022, https://www.britannica.com/event/Deepwater-Horizon-oil-spill Accessed 11 January 2023.

[2] https://www.hqts.com/oil-gas-quality-inspections/

[3] L. Yu et al., "Inspection Robots in Oil and Gas Industry: a Review of Current Solutions and Future Trends," 2019 25th International Conference on Automation and Computing (ICAC), 2019, pp. 1-6, Doi: https://doi.org/10.23919/IConAC.2019.8895089

[4] Fisher, M.; Cardoso, R.C.; Collins, E.C.; Dadswell, C.; Dennis, L.A.; Dixon, C.; Farrell, M.; Ferrando, A.; Huang, X.; Jump, M.; Kourtis, G.; Lisitsa, A.; Luckcuck, M.; Luo, S.; Page, V.; Papacchini, F.; Webster, M. An Overview of Verification and Validation Challenges for Inspection Robots. *Robotics* 2021, *10*, 67. Doi: https://doi.org/10.3390/robotics10020067

[5] Deepak Trivedi, Christopher D. Rahn, William M. Kier, and Ian D. Walker, "Soft Robotics: Biological Inspiration, State of the Art, and Future Research," Applied Bionics and Biomechanics, vol. 5, no. 3, pp. 99-117, 2008. Doi: https://doi.org/10.1080/11762320802557865

[6] Alici, Gursel. (2018). Softer is Harder: What Differentiates Soft Robotics from Hard Robotics?. MRS Advances. 3. 1-12. Doi: https://doi.org/10.1557/adv.2018.159

[7] Lee, C., Kim, M., Kim, Y.J. *et al.* Soft robot review. *Int. J. Control Autom. Syst.* **15**, 3–15 (2017). Doi: https://doi.org/10.1007/s12555-016-0462-3

[8] Rubio, F., Valero, F., and Llopis-Albert, C., 2019, "A Review of Mobile Robots: Concepts, Methods, Theoretical Framework, and Applications," *Int. J. Adv. Robot. Syst.*, 16(2). Doi: https://doi.org/10.1177/1729881419839596

[9] Siegwart R, Nourbakhsh IR. *Introduction to autonomous mobile robots*. Massachusetts London, England: A Bradford Book, The MIT Press Cambridge, 2004. ISBN 0-262-19502-X.

[10] Bajracharya M, Maimone MW, Helmick D. Autonomy for mars rovers: past, present, and future. *Computer* 2008; 41(12): 44–50. Print ISSN: 0018-9162. Doi: https://doi.org/10.1109/MC.2008.479

[11] Chiang JS, Hsia CH, Chang SH, et al. An efficient object recognition and self-localization system for humanoid soccer robot. Proceedings of the SICE Annual Conference 2010; 18–21: 2269–2278.

[12] Wolf DF, Sukhatme G. Semantic mapping using mobile robots. *IEEE Trans Robot* 2008; 24: 245–258.

[13] Cruz JPN, Dimaala ML, Francisco LGL, et al. Object recognition and detection by shape and color pattern recognition utilizing Artificial Neural Networks. In: Proceedings of the international conference of information and communication technology (ICoICT), Bandung, Indonesia, 20–22 March 2013, pp. 140–144. IEEE. Doi: https://doi.org/10.1109/ICoICT.2013.6574562
[14] Pandey, Anish, Shalini Pandey, and D. R. Parhi. "Mobile robot navigation and obstacle avoidance techniques: A review." *Int Rob Auto J* 2.3 (2017): 00022.
[15] Chen, Chi Hau, ed. *Handbook of pattern recognition and computer vision*. World Scientific, 2015.
[16] Thrun, Sebastian. "Robotic mapping: A survey." *Exploring artificial intelligence in the new millennium* 1.1-35 (2002): 1.
[17] Priyaranjan Biswal, Prases K. Mohanty. "Development of quadruped walking robots: A review." Ain Shams Engineering Journal, Volume 12, Issue 2, 2021, pp. 2017-2031, SSN 2090-4479.
Doi: https://doi.org/10.1016/j.asej.2020.11.005
[18] https://www.youtube.com/watch?v=ISfVS4mDTKs
[19] https://www.bostondynamics.com/products/spot/payloads
[20] https://support.bostondynamics.com/s/article/Robot-specifications
[21] https://support.bostondynamics.com/s/article/Spot-Arm-specifications-and-concepts
[22] https://github.com/heartexlabs/labelImg
[23] https://www.image-net.org/
[24] Everingham, M., Van Gool, L., Williams, C.K.I. et al. The PASCAL Visual Object Classes (VOC) Challenge. Int J Comput Vis 88, 303–338 (2010).
Doi: https://doi.org/10.1007/s11263-009-0275-4
[25] https://github.com/ultralytics/yolov5
[26] https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html
[27] https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html
[28] https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html
[29] https://huggingface.co/spaces/baudm/PARSeq-OCR/tree/main
[30] https://levelup.gitconnected.com/wtf-is-regex-really-4dd563ee5ce0
[31] https://docs.opencv.org/3.4/d4/d70/tutorial_hough_circle.html
[32] https://dev.bostondynamics.com