Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Electronics and Telecommunication Engineering (35^{th}cycle)

Safe navigation and human-robot interaction in assistant robotic applications

By

Pangcheng David Cen Cheng

******

Supervisor(s):
Prof. Marina Indri

Doctoral Examination Committee:
Prof. Antoni Grau, Referee, Technical University of Catalonia, Spain
Prof. Antonio Visioli, Referee, Università degli Studi di Brescia, Italy
Prof. Lucia Pallottino, Università degli Studi di Pisa, Italy
Prof. Paolo Valigi, Università degli studi di Perugia, Italy
Prof. Alessandro Rizzo, Politecnico di Torino, Italy

Politecnico di Torino
2023

Declaration

I hereby declare that the contents and organization of this dissertation constitute my own original work and do not compromise in any way the rights of third parties, including those relating to the security of personal data.

Pangcheng David Cen Cheng
2023

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I would like to dedicate this thesis to my loving parents

Acknowledgements

I have met wonderful people during the three years I spent during my PhD, and without their -academic, emotional and especially inspirational- support, it would be hardly possible to conclude all the research work I am presenting here. It is quite difficult to express here my gratitude to all those people, but I will give it a try.

First of all, I would like to thank Prof. Marina Indri for giving me the opportunity to start my research experience at the university, be able to play with robots in the laboratory and participate in memorable international conferences.

I am also very grateful to Corrado Possieri and Mario Sassano from Università Tor Vergata, especially to Corrado during his period in Politecnico di Torno since he first collaborated on my very first conference paper. There is still so much that I wanted to learn from him.

Abstract

In recent years, robotic systems have been employed widely in many fields. In an industrial context, robots are generally used to automatize manufacturing processes in hazardous environments for human beings or to relieve humans from repetitive and rough tasks. Nowadays, where the presence of human operators is becoming more relevant, the production focus is aimed to have robotic systems supporting human activities. In this context, the aim of this PhD thesis is to study and illustrate different approaches that enable safe human-robot interaction in an industrial context.

The main research focus is the improvement of the performance of existing robotic systems in the industry, such as mobile robots, manipulators and mobile manipulators, to enhance their perception of their surroundings as well as the presence of human operators in the same working space. In order to do so, the research work proposes feasible methodologies for mobile agents and manipulators to work safely in the presence of humans, exploiting open-source frameworks along with affordable sensory systems for sensor data fusion. In particular, among the various research topics that were investigated and developed are (i) safe path planning algorithms for autonomous mobile robots with human-obstacle avoidance, (ii) a sensor data fusion algorithm to improve the robot's awareness of the environment and human detection, and (iii) alternative frameworks to enable safe interaction between human and robots.

For what concerns the robot's navigation algorithms, most path planners generate an optimal solution in terms of distance or time, but they do not consider the safety concerning humans. The developed path planning algorithms provide an additional safety layer within the robot's navigation system to safely avoid humans in an environment where humans and robots coexist. Different planning levels were designed, for instance, a supervisory planner for computing a deterministic reference path and a local planner able to deal with human obstacles. The algorithms have

been developed employing MATLAB and open-source frameworks such as ROS (Robot Operating System). MATLAB was mainly used to compute the waypoints of the reference path while ROS was exploited for managing the data coming from the sensors and commanding the mobile robot.

The robot's navigation system can be also improved by upgrading the perception equipment to detect specific obstacles, especially humans moving too close to the robot in motion. An example of this is combining the data coming from a camera, a laser range finder and an object recognition system to identify an obstacle and have the corresponding distance associated with that obstacle, so the mobile robot can execute ahead of the path replanning. The sensor calibration has been performed using the MATLAB tools, while the data coming from the visual sensor is processed using the YOLO (You Only Look Once) object detection system, so the robot's perception system is able to differentiate between general obstacles and humans.

In the case that the robot has the capability to autonomously perform low-level tasks, such as path planning and sensing the environment, it is possible to build an organized framework to provide support and assistance to human operators, in which the robot can continuously learn about the human actions and execute high-level commands. Such a framework takes into account state-of-the-art solutions for human perception as well as object manipulation considering object affordance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robotic systems have been employed widely during the last decades in many fields, such as the manufacturing industry, medicine, aerospace, agriculture, military and customer services. Also, robots are often used in hazardous environments for human beings or to relieve humans from repetitive and rough tasks. In the industrial context, robotics applications evolved quickly in recent years. In particular, there is currently a transition trend from Industry 4.0 to Industry 5.0, which in other words means that there is a migration from fully automatized manufacturing operations to human-centred applications. The former has the main focus of optimizing the production line for mass production while the latter aims at optimizing mass or product customization but is rather centred on human reasoning in order to satisfy the customer requirements. This is because the robotic systems are not yet intelligent enough to make critical decisions by themselves or there is a lack of capabilities that only humans have: creativity and expertise for problem-solving. In this way, the robotic system is supposed to be a support agent to human work. Nevertheless, in order to facilitate the migration, existing operative systems developed in Industry 4.0 could be exploited during the transition, for instance, those functionalities that allow the robot to safely coexist with human operators, either for collaboratively working or just cooperating/interacting [2].

For instance, the robotic system should have good perception equipment to detect, recognize and plan actions accordingly to the requests. To foster and speed up the development of a framework in which robots can support human activities, it is possible to leverage existing systems to enable safe interactions, even in contact

with the human operators, or at least, find a feasible and safe solution that can take advantage of the already existing machinery by integrating low-cost sensory system along with a sensor data fusion algorithm software to deal with complex tasks. Despite the fact that low-cost sensors may not be able to guarantee safety if they are used individually, it is possible to design a safe sensory system that processes and merges relevant data coming from different sources. Note that in this way it is possible to enlarge the field of vision of the robots while lowering the costs, which can be also advantageous to Small and Medium Enterprises, since they may not have the same budget as larger firms.

Generally, in an industrial environment, there are different types of robots: mobile robots, fixed-base manipulators and mobile manipulators, where each one is in charge of executing particular tasks. Mobile robots are usually employed for material transportation and monitoring, while fixed-base manipulators are mainly exploited for assembly and manufacturing tasks. On the other hand, mobile manipulators are the most flexible robotic systems that can ease most of the collaborative applications with human operators since they provide mobility to navigate in an environment but also have the capability to manipulate objects. Therefore, in order to enable safe operations along with human operators, the robots should include the human variable within the perception system and control architecture. In this way, the robots are to be able to perform safe autonomous navigation and safe object manipulation.

For what concerns the navigation system, it is important to have a safe path planner, that is able to plan a deterministic route for the robot but also to react to dynamic obstacles, particularly if the dynamic obstacle is a human. Path planning problems become more relevant and complex in a dynamically-changing environment that involves the presence of human operators working in a shared space with other robots. Although there are many algorithms that generate an optimal path for the mobile robot, most of them are not deterministic and therefore, not safe.

Safety, in this case, means that exists a path that can be easily recognizable for a human worker. As a result, the human awareness of a lane dedicated to mobile robots, even if it is a virtual one, decreases the risk of a collision with the mobile platforms navigating in the shared working place. Moreover, the reference path followed by the robot is considered safe because the human knows that if he/she crosses the trajectory near the robot in motion, the robot is in any case able to avoid him/her.

Even so, computing a safe path and making the robot follow that generated path is not enough to guarantee safety, so that is the perception system that can change the game. A good perception system should be able to sense and “understand" its surroundings. Interestingly, the understanding part depends on programming and the application requirements. Usually, in an environment shared with humans, one might want to optimize the path computation by differentiating general obstacles from humans, and by doing so, the robot is able to move in narrow spaces while ensuring collision avoidance with humans and other obstacles. One way to distinguish the human from other obstacles can be achieved by combining the perception system with an Artificial Intelligence system able to classify the features within an image frame. An alternative way to ensure safety is to virtually enlarge the area that the human is occupying on the map so that the robot cannot get too close to the human and collide. In this case, it is possible to redundant the information about the humans by detecting them within the image frame and increasing their inflation radius, considering also their speed and orientation, so the prediction of the human motion can enhance further the path replanning. However, in the scenario in which it is required to supervise the robot but due to the hazardousness of the environment the human cannot coexist in the same working space, one might design an interface that provides high-level commands to the robot and still be compliant with the safety measures of the operation.

Another feature that is required for optimizing safe operations with human workers could be the development of a learning system for the robot, so as to be able to acquire knowledge about human activities and plan actions accordingly. In fact, according to the authors in [3], by combining some enabling technological pillars, e.g., Autonomous Robots and Artificial Intelligence, in an organized framework, it is possible to foster higher productivity and manage better the available resources.

This thesis aims to study and illustrate various methodologies that enable safe human-robot interaction in industrial environments. The target is to enhance the performance of existing robotic systems in the industry, such as mobile robots, manipulators and mobile manipulators, to improve their perception of their surroundings as well as the presence of human operators in the same working place. In order to do so, the research work presented here proposes feasible approaches for mobile agents and manipulators to work safely in the presence of humans, exploiting open-source frameworks along with affordable sensory systems for sensor data fusion.

During the three years of the PhD, research topics related to safe autonomous navigation for mobile robots and approaches for safe interaction in Human-Robot shared environments have been studied and developed. The achievements have been published in international conferences and journals. The research work related to the path planning algorithms for mobile robots in the industrial context are represented in the scheme shown in Figure 1.1.

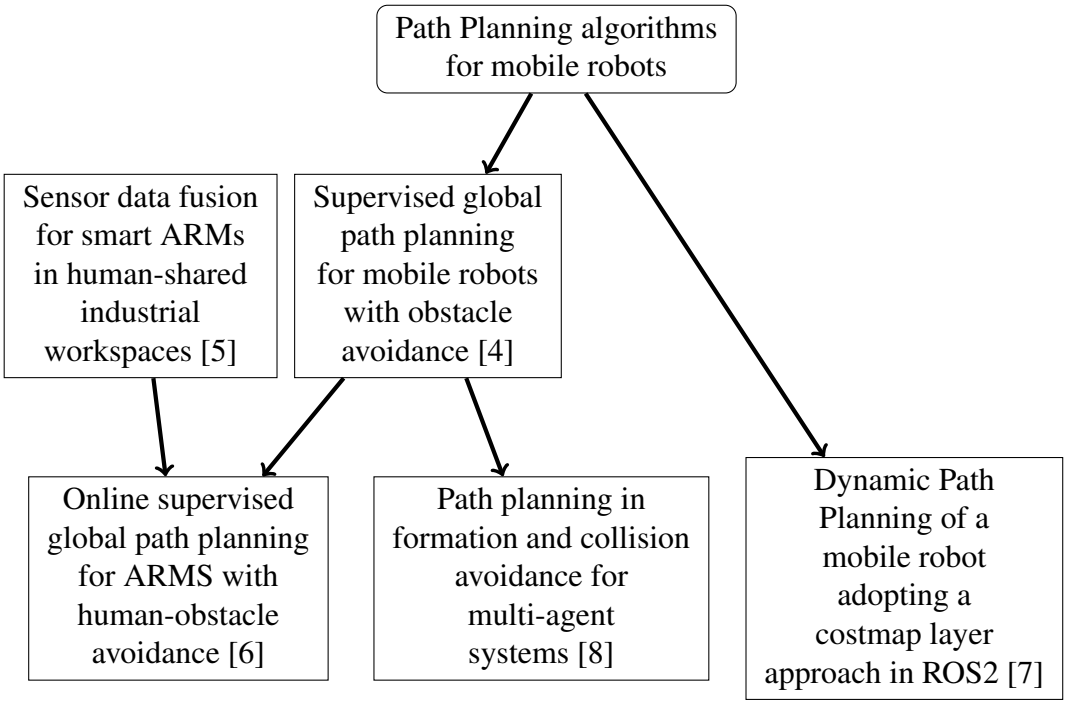


Fig. 1.1 Conceptual scheme of the relations among the topics of the developed path planning algorithms.

On the other hand, research work related to the methodologies that can be adopted in an environment where humans and robots coexist and cooperate is depicted in Figure 1.2.

In this thesis, the main contributions of each research work are presented, in particular, they have been grouped as follows:

- Safe path planning algorithms for mobile robots
- Improvement of the robot's perception system employing sensor data fusion algorithms

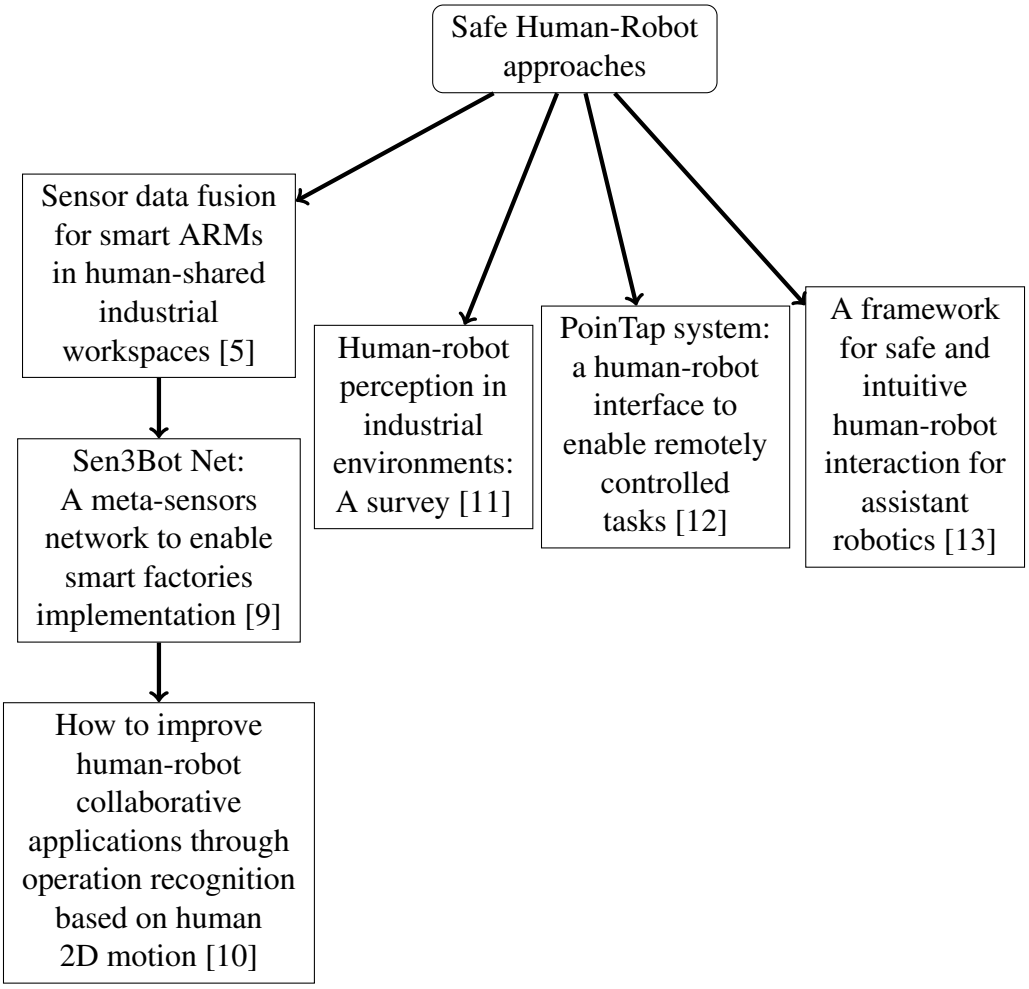


Fig. 1.2 Conceptual scheme of the relations among the topics of the proposed safe human-robot approaches.

- Framework that enables safe interaction between human and robot

The developed path planning algorithms have the aim of providing an additional safety layer within the mobile robot's navigation system. As an example, in [4], a supervisory planning level is added on top of the existing planning hierarchy of the mobile robot. The goal is to provide a deterministic path considered safe, whenever the robot deviates due to the presence of the human obstacle, the mobile agent reaches as soon as possible the reference path computed by the supervisory planner. This algorithm is improved later in [6], in which the reference path is updated online after the robot overcomes human obstacles, guaranteeing the existence of a deterministic virtual safe course in an environment where humans and robots coexist.

The dynamic path planner developed in [7] enlarges the area occupied by moving obstacles in a particular way, so the robot is able to predict the obstacle motion and take that information to replan more efficiently the path to reach the final goal. The supervisory planning algorithm has been also extended to the multiagent case [8], in which the robots can perform patrolling or monitoring tasks while avoiding collision among the agents and with other obstacles.

For what concerns the robot perception system, as already illustrated in Figure 1.2, the sensor data fusion algorithm has been developed to enhance the existing perception system of a mobile robot [5]. This algorithm combines the information coming from a visual sensor, a laser range finder and an object recognition system, so as to detect human obstacles when the robot is navigating in an industrial environment. It can be noted that it has been used to upgrade the navigation system presented in [6] and to create a network of intelligent sensors in [14]. The latter proposes the deployment of a fleet of mobile robots that have been enhanced with the proposed sensor data fusion algorithm, to support the sensory system of pre-existing mobile platforms within a manufacturing environment shared with human workers. The data obtained from the monitoring functionalities can be exploited to create a database for the robot, so it can learn the motions executed by the human operator during different operation phases in a working place [10]. The information about the human trajectory can be used as an a-priori constraint or an additional input to optimize the navigation algorithms, since they already consider the presence of the human as well as his/her trajectory. Nevertheless, there are several approaches in the literature for human-robot perception. The most used sensors and algorithms have been collected and analysed in [11], while considering also the robot type and the application context within industrial environments.

On the other hand, other safe approaches can be studied when considering the interaction with manipulators. A framework for safe Human-Robot interaction is proposed in [13], in which different functional blocks have been proposed to boost the performance in operations that assist or support the human worker. In the case that the environment is hazardous to the human worker, but still it is needed the human intervention for commanding the robot, an alternative interface is proposed in [12]. In particular, the robot user's interface is an LCD screen transformed into a large touchscreen, where the human can point and tap the screen to provide high-level commands to the robot.

Nonetheless, since the proposed algorithms have the aim to be developed in a smart industrial environment, it should be noted that they have been only implemented and tested in research demonstrators within a semi-structured laboratory environment.

1.1 Outline

The remainder of the PhD thesis is structured grouping the research topics with the objective to provide a clear but exhaustive description of the main developments of the research activity. The first chapters deal with mobile robots' path-planning methods in order to ensure safety while they are navigating in an environment shared with humans.

Then, some methodologies related to robot perception of the human are presented, in particular, sensor data fusion algorithms for detecting and avoiding both static and dynamic obstacles, with the main focus on human detection. Finally, alternative solutions for a safe human-robot interaction are proposed. In particular, in Chapter 2, a Supervised global path planning for mobile robots with obstacle avoidance is introduced, which is one of the main works related to safe navigation for Autonomous Mobile Robots (AMRs). Chapter 3 presents the Online supervised global path planning for AMRs with human-obstacle avoidance, which is an improvement of the supervised global planner, dealing with the dynamic changes of the environment. After dealing with the supervised global planner, the path planning in formation and collision avoidance for multi-agent systems is presented in Chapter 4. The local planner problem is dealt with in Chapter 5 through a Dynamic Path Planning of a mobile robot adopting a costmap layer approach in ROS2.

The following chapters propose some improvements to the robot's perception system through data fusion algorithms. Chapter 6 reports a camera-laser sensor data fusion algorithm for smart AMRs in human-shared industrial workspaces, while Chapter 7 sums up the sensors and algorithms commonly used in industrial robotics to perceive the human operator. Then, in Chapter 8 the proposed sensor data fusion algorithm is employed in different scenarios to enable smart factories.

Finally, in Chapter 9, a framework enabling safe and intuitive human-robot interaction is proposed, while Chapter 10 presents an example of a remote-controlled

application to command a manipulator. Last but not least, the conclusions are drawn in Chapter 11 as well as some open issues and future works.

Chapter 2

Supervised global path planning for mobile robots with obstacle avoidance

This chapter introduces a supervised global path planning for mobile robots with obstacle avoidance in industrial-like environments, specifically highlighting the main contributions presented in [4]. The aim is to propose an algorithm that supervises the overall planning hierarchy in order to obtain a deterministic path for a mobile agent. The curve computed by the supervisory planner takes into account the kinematic model of the mobile agent and the static information of the map, whose obstacles are encircled within ellipses of minimum radius. The integration of a supervisory planning level to the existing ones available in the robot enhances the planning procedure since it computes the waypoints corresponding to a deterministic and safe curve, while the global planner interpolates the waypoints to guide the mobile robot and the local planner enables the robot to avoid collision with unexpected obstacles and then attempt to reach the original route as soon as there are no other obstacles intersecting its path.

The chapter is organized as follows: Section 2.1 gives a background related to path planning algorithms in industrial contexts. The proposed procedure is presented in Section 2.2, in which relevant theoretical information about the supervisory planner is provided and then the algorithm implementation with obstacle avoidance capabilities is described. The experimental results are shown in Section 2.3, where three different test cases are studied and reported. Finally, the algorithm is discussed in Section 2.4.

2.1 Background

In recent years, mobile robots have been used in many applications, such as transportation, surveillance/monitoring and automation in industry. Within the industrial environment, path planning with obstacle avoidance is required to guarantee the safety of both mobile agents and human workers. While Automated Guided Vehicles (AGVs) are constrained to follow predefined paths, often guided by a magnetic tape or limited in a specific lane, the introduction of Autonomous Mobile Robots (AMRs) in an industrial context implies that additional requirements for path planning should be taken into account, such as the existence of preferred areas or safer paths that consider the presence of other types of machinery, working stations, robots and humans in the plant, and also a more efficient organization of the working process itself.

Usually, the path planning algorithm for mobile robots is executed at two levels: global planning and local planning. On one hand, the global planner computes a path for a given well-known environment map and allows the robot to move from an initial point to a goal point avoiding static obstacles while optimizing specific requirements, e.g., searching for the shortest path, so as to reduce energy consumption and travelling time. On the other hand, the local planner updates the global path of the robot considering the information of a local window coming from its sensors. The aim of the local planner is to allow the robot to follow the trajectory of the global planner and modifies the original path in order to avoid unexpected obstacles.

Depending on the task constraints and the environment, there are several path-planning algorithms available in the literature. Among the planners that compute the path based on a heuristic search method, there are the Dijkstra, A* and D* algorithms. The Dijkstra algorithm [15] computes the shortest path connecting two nodes in a map; nevertheless, it has an overall low-efficiency process since its execution needs to evaluate too many nodes. This problem is fixed in the A* algorithm [16], a planner based on Dijkstra, that considers a function that evaluates the distances between the nodes and their cost to reach the final point. However, the main drawback of A* is that it takes a relatively long computational time to execute, so it is not suitable for carrying out sequential tasks in real-time [17]. Alternatively, response time-constrained tasks can be executed by using D* [18], a dynamic re-planning algorithm based on the A*. The algorithm considers the direction of the robot

position as expressed by a series of states, whose values are updated according to the temporal evolution of the elements in the map. Nonetheless, the D* algorithm requires high computational effort, since the re-planning phase needs to compute twice each state on the environment [19].

Despite the computational effort required to execute an A* algorithm, it has been preferred over the years, since most of the researchers developed alternative versions of the algorithm, each one solving one aspect of the algorithm and adapting it according to environment/hardware requirements and the mobile platform characteristics. As an example of this, the Hybrid A* algorithm is an upgrade of the original one and has been specifically designed for non-holonomic mobile platforms, first introduced in [20] and then improved further in [21].

The computational burden can be mitigated by adopting probabilistic-based methods, e.g., Probability Roadmap (PRM) [22] and Rapidly-exploring Random Tree (RRT) [23]. In PRM, free spaces on the world map are randomly sampled, and the planner tries to connect the generated nodes in order to find a feasible path. This algorithm is commonly used to cover larger areas, where the use of other algorithms may require a higher computational cost, but it does not guarantee the shortest path [24] since the generated nodes' location changes between multiple iterations. In RRT, obstacle-free trajectories are obtained growing expanding *trees* from a starting point, going through random points called *seeds*, until the *tree* reaches to the destination point. Over the years, researchers have proposed a variety of improved methods since the release of the original RRT algorithm. In [25], the method consists of "planting" a tree in both the initial and the final points, and then both trees are expanded in the whole world map until an intersection point is found; the algorithm proposed in [26] controls the tree edges growth direction and density of the RRT* (an RRT variant converging to the shortest path presented in [27]). Nevertheless, the main drawback of sampling-based stochastic searches is that path cost is not taken into account, so the resulting solution is not guaranteed to be optimal one [28].

Alternative path planning algorithms are inspired by natural phenomena. The Genetic Algorithm (GA) [29] is based on the natural selection theory and applied as a path-searching method in robotics. The cost function for computing the best path is structured similarly to a chromosome, where each location is considered as a gene [30].

Another widely used approach involves computing offline a desired safe path and controlling the robot so as to it goes to the destination point following a set of waypoints. For instance, it is possible to apply the feedback linearization technique in order to design a control system for path following, such as the one proposed in [31], where a fuzzy logic-based control architecture is employed. Similarly, there are navigation functions based on artificial potential fields that solve the robot motion planning [32], and a feedback control law ensuring that the robot reaches the destination point while avoiding obstacles [33].

This chapter exploits the features of the algorithm developed in [34] (and validated there only in a purely theoretical context) to solve the path planning problem in an industrial-like scenario. A possible implementation is proposed by employing a differential mobile robot, within a complete planning procedure based on three levels. In particular, the proposed solution integrates the classical two-level planning hierarchy with a third level, which aims to guide the mobile robot to a given goal position traversing a virtual path previously identified as safe. The innovation lies in what the new priority is with respect to the former one (shortest path), which is getting as soon as possible on the safest path, in order to guarantee safe behaviour in the industrial-like environment.

2.2 Supervisory algorithm for path following with obstacle avoidance

This section goes through the description of the developed and tested path planning procedure for path following with obstacle avoidance.

The scenario is that of a mobile robot roaming within a closed space, for example, a warehouse or factory plant, where it is assumed that there are no unexpected obstacles and some predefined routes are considered safe for the mobile platform motion: the a-priori desired path computed by the algorithm, proposed in [34] and recalled in Section 2.2.1, tends by construction to a curve representing a safe route, which role is similar of what guide tapes and wires represented in the most traditional industrial mobile navigation set-up. The path planning algorithm proposed in this chapter has a hierarchical structure based on three levels: the Supervisory Global Planner (SGP), the Global Planner (GP) and the Local Planner (LP). The planning

hierarchy is shown in Figure 2.1, and enables an enhanced robot environment awareness, improving the path planning only based on the static map to a dynamic-obstacles aware system.

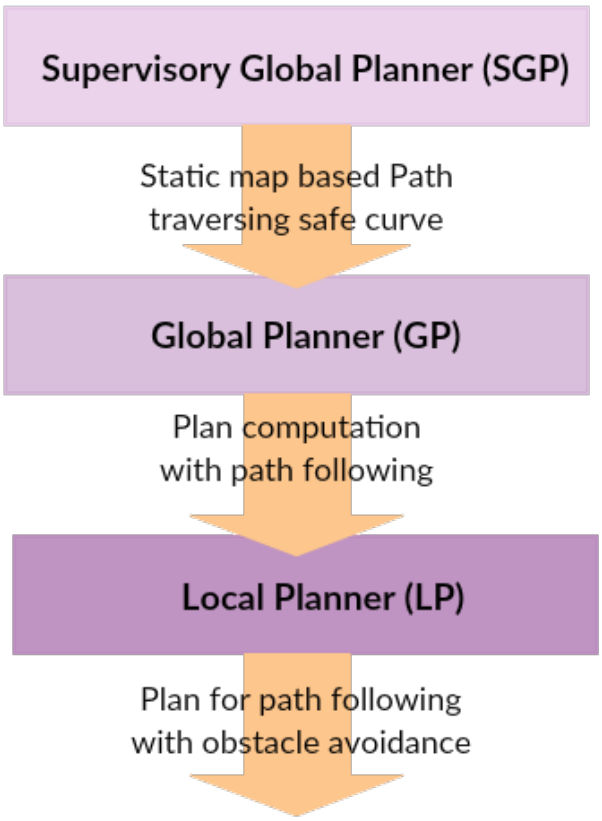


Fig. 2.1 Planning hierarchy schematics.

Due to the fact that the static map of the environment used at the supervising level could be not fully updated, the planned waypoints are integrated at a lower level within the GP, which lets the robot actually follow the planned trajectory (if possible) thanks to a cost-based algorithm. The avoidance of unexpected obstacles that may appear during the robot motion is left by the LP, which further modifies online the path previously computed by the GP, so the robot is able to reach the destination point.

2.2.1 Supervisory algorithm theoretical development

In this section, the algorithm proposed in [34] is reviewed in order to design the path of a mobile robot in a known environment. The aim of the algorithm is to generate a *collision free* path, i.e., it does not intersect the boundaries of the obstacles and tends to a preassigned algebraic curve, which is considered safe for motion.

In particular, assume that the aim of the supervisory planner algorithm is to obtain a path that tends to the planar curve

$$\mathscr{V} := \{x \in \mathbb{R}^2 : p(x) = 0\}, \tag{2.1}$$

where $p(x)$ is a given polynomial function. By applying the algorithm presented in [35] (omitted here for brevity), it is possible to compute two vector fields $\phi(x)$ and $\psi(x)$, whose entries are polynomials in x, such that the curve $\mathscr{V}$ given in (2.1) is attractive and invariant with respect to the dynamical system

$$\dot{\xi} = \phi(\xi) + \psi(\xi)\alpha(\xi), \tag{2.2}$$

where $\alpha(\xi)$ is an arbitrary function. Although the trajectories of system (2.2) converge to the set $\mathscr{V}$, they are not necessarily collision free. Thus, in order to ensure collision avoidance, the results given in [35] are coupled with classical navigation functions [32].

Let $b(x)$ be a function that describes the boundaries of the obstacles (i.e., if $b(\xi) = 0$, it means that the point ξ belongs to the boundary of one of the obstacles) and let

$$\mathscr{W} := \{x \in \mathbb{R}^2 : b(x) > 0\}$$

be the workspace of the mobile robot, which is assumed to be a connected set. Furthermore, we assume that

$$\mathscr{V} \cap \{x \in \mathbb{R}^2 : b(x) = 0\} = \emptyset,$$

i.e., that $\mathscr{V}$ is actually safe for motion. Thus, define

$$r(\xi) := \frac{p^2(\xi)}{b(\xi)}, \qquad \eta(\xi) := \left(\frac{\partial r(\xi)}{\partial \xi}\right)^\top.$$

Note that, by construction, the function $r(\xi)$ is nonnegative for all $\xi \in \mathscr{W}$, it is zero if $\xi \in \mathscr{V}$, and it tends to $+\infty$ if ξ tends to $\mathscr{W}$. This implies that, letting $\beta(\xi)$ be a nonnegative function such that $b(\xi) = 0 \implies \beta(\xi) \neq 0$, the trajectories of

$$\dot{\xi} = -\eta(\xi)\beta(\xi) \tag{2.3}$$

are collision free (see [34, Prop. 2]).

Therefore, the supervisory planner algorithm is obtained coupling systems (2.2) and (2.3). In other words, letting $\zeta(\xi)$ be an arbitrary function (which is amenable for further optimization, see [33]) and k be a positive constant, it is possible to guarantee that the trajectories of the dynamical system

$$\dot{\xi} = b^2(\xi)\phi(\xi) + b^2(\xi)\psi(\xi)\zeta(\xi) - k\,p^2(\xi)\eta(\xi) \tag{2.4}$$

tend to $\mathscr{V}$ while avoiding collisions with the obstacles.

It is worth noticing that, if the objective is to steer the robot to $\mathscr{V}$ and, additionally, to let it stop for some $\bar{\xi} \in \mathscr{V}$, then such a goal can be pursued by designing the function $\zeta(\xi)$ in (2.4), following [35, Alg. 2].

2.2.2 Integration with ROS Global and Local Planners

In this section, some high-level software details related to the implementation of the SGP and the integration of the dynamic obstacle avoidance capability are given. In order to adapt the theoretical algorithm to a physical implementation, the supervisory algorithm has been executed using a real environment map, considering also the physical dimensions of the performing robot. Note that, at this level of description, information related to the map and parameters tuning/adjustment specific to the chosen robot will be omitted (more details about the hardware set-up are given in Section 2.3) to highlight the general validity of the approach.

The SGP algorithm has been developed in MATLAB, while the navigation driving the robot exploits ROS (Robot Operating System) [36] tools. In particular, the ROS *Navigation Stack* [37] provides the user with off-the-shelf packages ready for mapping, localization, navigation and reference frames tracking. To ensure a correct interpretation of the supervisory algorithm output (i.e., a set of 2D waypoints converging to the preassigned safe curve to be traversed on the static map), the ROS `/world` frame origin (reference frame for the whole ROS coordinate frames transform tree, managed by the `tf` package) has been aligned to the frame origin used in MATLAB. Then, the output plan has been conveniently packed in an array of desired positions stored on the ROS Parameter Server, exploiting the MATLAB Robotics System Toolbox™[38]. The ROS framework provides the `actionlib` library stack [39], which allows the user to interact through a standardized interface with preemptable tasks, which correspond to the desired poses for the mobile robot to reach. A custom Python ROS node is in charge of sending through a ROS Action Client a request to the Action Server, via a message containing the information of the next goal position to be reached. It is worth noting that those ROS actions represent the ideal Client-Server-based mechanism for goal achieving since while the whole operation is brought on, a feedback message about its status can be sent to the client node. Figure 2.2 shows a sketch of the software setup.

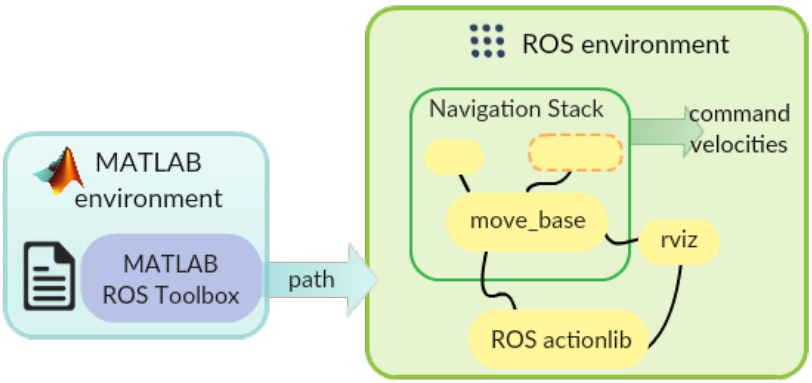


Fig. 2.2 Software setup for implementation of the proposed algorithm.

ROS package `move_base`, already included in the ROS navigation stack, provides several ready-to-use global planners. Within the proposed three-level planning hierarchy, the main role of the GP is to generate a plan as close as possible to the path computed by the SGP. On the other hand, the role of the LP algorithm is to locally update the robot trajectory to avoid the collision with unexpected obstacles

that may appear during the robot motion, computing a short-term plan based on a *local cost map* and taking into account a predefined portion of the GP-generated long-term plan. The *local cost map* uses the data coming from sensors in a local window and guarantees the avoidance of unexpected obstacles (either moving or fixed). Moreover, the whole planning performance depends on a set of *cost map parameters*, which can be accordingly tuned based on the planning requirements and on the mobile robot's physical characteristics.

The ROS global planner adopts Dijkstra's algorithm by default, which finds the shortest path from the initial pose to the desired goal. Alternatively, another widely used algorithm for path-finding is the A* algorithm. The A* algorithm is a combination between Dijkstra's algorithm and the Best First Search algorithm, meaning that it is an informed algorithm or a weight-based process, where the graph is explored and expanded only in those nodes that results convenient, based on an assigned cost with heuristic content. In fact, A* inherits the benefits of a uniform-cost search from Dijkstra's algorithm while adding heuristics to efficiently find an optimal solution in less time [40].

The GP algorithm has been built upon the A* cost-based structure: in order to have access to a simpler standalone piece of code, the default global planner provided by the `navfn` package has been substituted exploiting the ROS plugin mechanism. As the ROS global planner plugin implementation guidelines recommend, the ROS *cpp* library has been used to adhere to the `nav_core::BaseGlobalPlanner` C++ interface provided by the `nav_core` package (by overriding some specific methods) and employed by the `move_base` package to drive the mobile platform. With the aim of forcing the mobile robot to follow the set of positions computed by the SGP within the GP, the proposed algorithm assigns low costs to the preferred waypoints to be traversed.

Before addressing the description of the proposed algorithm, the cost mechanism of the A* search algorithm will be briefly described. Let's consider a path-finding problem, where an optimized path must be found between two points on a 2D cost map: at each main loop, starting from the source cell the algorithm expands the most suitable, and for construction most "convenient", cell depending on a function $f(c)$, defined as

$$f(c) = g(c) + h(c) \tag{2.5}$$

where c is the currently expanded cell, $g(c)$ represents the cost from the source position to the current one, and $h(c)$ corresponds to the heuristic estimated cost from the current cell to the destination cell. At each iteration, the expanded cell, namely, the cell whose neighbour cells are visited and their costs computed, is the one having the lowest $f(c)$. The cell expansion behaviour is shown in Figure 2.3

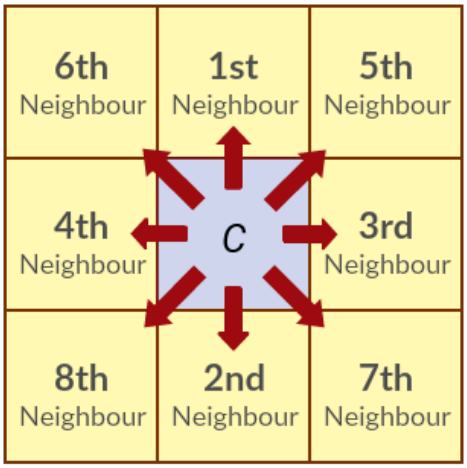


Fig. 2.3 Expansion of the most convenient cell.

Hence, $f(c)$ contains an estimated path cost from the source cell to the destination cell, traversing cell c. While $g(c)$ can be easily computed, the estimation of $h(c)$ usually involves an approximation method, due to the fact that exact heuristics computation (calculating the value attained by the cost-to-go function at cell c) would take a huge amount of time. There are several approximated heuristic functions available in the literature, such as the Manhattan distance, Euclidean distance and diagonal distance. The function $h(c)$ has been defined according to the Euclidean distance as follows

$$h(c) = \sqrt{(c.x - dest.x)^2 + (c.y - dest.y)^2} \tag{2.6}$$

where *dest* is the destination cell, while *.x* and *.y* are the fields corresponding to the x and y components of the considered cell, respectively.

In order to perform path following, the idea is to use a cost-based algorithm that acts as A* when the mobile robot is far from the path generated by the SGP, while the positions computed by the SGP are associated to a lower cost, so as to guarantee their presence in the output plan. Moreover, the passage on these positions

is favoured by including a new cost function h_{WP} defined as

$$h_{WP}(c) = \sqrt{(c.x - WP.x)^2 + (c.y - WP.y)^2} \tag{2.7}$$

where WP is the next expected waypoint while $.x$ and $.y$ are the fields corresponding to the x and y components of the considered cell, respectively. The value calculated in (2.7) is the Euclidean distance from the next expected waypoint, and assigns penalizing costs to those positions that are far from the desired path, depending on the following possible cases:

1. The currently visited neighbour cell n is a waypoint.
2. The currently expanded cell c is a waypoint.
3. All waypoints have been already traversed.
4. None of the previous conditions is valid.

It is worth noting that all discrete points making up the supervisory algorithm output path are considered as waypoints that the robot should traverse when is in their proximity.

Further details on how the costs have been assigned to achieve an overall balanced and feasible cost system are given in Algorithm 1, in which the pseudo-code for the proposed algorithm is depicted. Handling of the above enumerated possible cases can be referred at lines 14, 16, 18 and 20, respectively. It should be noted that highlighted lines indicate the modifications with respect to the original A*, to have a direct comparison. The inputs to the GP algorithm are the environment map and the initial and goal cells, while the output is represented by the computed path. In this way, the proposed GP boasts the mobile robot the capability of navigating to a goal position as safely as possible but that is not necessarily in the shortest way.

On the other hand, the employed LP exploits the `TrajectoryPlannerROS` wrapper, which adheres to the `nav_core::BaseLocalPlanner` interface. This local planner version deploys the so-called *trajectory rollout* algorithm: the mobile robot's control space is discretely sampled and, for each sampled velocity a forward simulation is performed using the current robot state as the starting point, to predict the robot motion if the considered velocity were applied for a restricted period of time. Each simulated trajectory is examined on the basis of some evaluation parameters,

Algorithm 1: Proposed algorithm with the path following.

```
Input: map, start position, goal position
Output: path
/* Set-up cell expansion lists                                          */
openList ;                                          // Declare the open list
closedList ;                                      // Declare the closed list
/* Insert starting cell in the open list                                */
openList.insert(start)
while openList is not empty do
    /* Pop cell c with lowest f(c) off the openList                     */
    c = openList.pop()
    /* Push cell c into the closedList                                  */
    closedList.push(c)
    foreach neighbor n of cell c do
        if neighbor n is the destination cell then
            n.parent = c ;                                // Assign parent node
            Stop searching
        else if neighbor n ∉ closedList and is not blocked then
            if n is a waypoint then
                Assign minimum values to n.g, n.h and n.h_WP
            else if c is a waypoint then
                Assign minimum values to n.h and n.h_WP
            else if waypoints have been traversed all then
                Assign minimum values to n.h_WP
            else
                if ( n ∈ openList) and ( new f(n) > old f(n) ) then
                    Ignore n
                else
                    n.g = n.g + distance from c to n
                    n.h = distance from n to destination cell
                    n.h_WP = distance from n to next waypoint
                    /* Compute f(n)                                     */
                    n.f = n.g + n.h + n.h_WP
                    /* Assign parent cell                               */
                    n.parent = c
                    /* Insert cell n in the openList                    */
                    openList.insert(n)
/* Trace the output path from the destination cell                      */
p = dest
while p.parent is not null do
    Path.push(p)
    p = p.parent
Path.push(p)
```

such as the proximity to the global path, to obstacles, to the goal, and the speed. Among the evaluated trajectories, those that are not collision-free are discarded, while the trajectory achieving the highest score is selected and the relative velocities are sent to the mobile platform. This procedure is looped at each motion step [41]. The LP belongs to the lower planning level and enhances the robot's awareness of its surroundings. In this way, it is possible to ensure a quick reaction to obstacles that are not considered in the static map, or are unknown when the path of the GP is already computed.

Additionally, to ensure that the resulting path tracks as precisely as possible the desired algebraic curve, the information of the LP about the GP long-term path is limited, so as to influence the short-term plan generation, due to the fact that a wider overview on the tracked plan would result in local path optimization.

2.2.3 Hardware/Software setup and testing

In order to test the supervisory algorithm with the path following and obstacle avoidance, the Pioneer-3DX differential drive mobile robot [42] has been employed and it is equipped with a SICK LMS200 laser range finder [43] with $10m$ range, scanning angle of 180° and scanning steps of 0.25°@75 Hz, a Raspberry Pi 3 Model B [44] mounting an ARM Cortex-A53 (x4 core) CPU (1.2 GHz) and 1 GB RAM, which is the processing unit in charge of receiving data and controlling the robot. The hardware setup is shown in Figure 2.4.

It is worth noting that the mobile robot's physical specifications are fundamental for deriving the equations of the kinematic model needed to execute the supervisory algorithm, to suitably adapt ROS visualization and map inflation parameters, and to generate proper steering commands taking into account other mechanical constraints such as the maximum speed and acceleration. The robot's dimensions are shown in Figure 2.5 and it can reach a maximum speed of $1.2m/s$ while its maximum rotation speed is $300^\circ/s$.

Figure 2.6 illustrates the interactions between hardware components: the Raspberry Pi receives the measurements related to the environment via laser scan data and sends proper commands to the robot. Simultaneously, the robot states and environment information are exchanged between the Raspberry Pi board and a computer; both processing units are running specific nodes of the ROS framework in

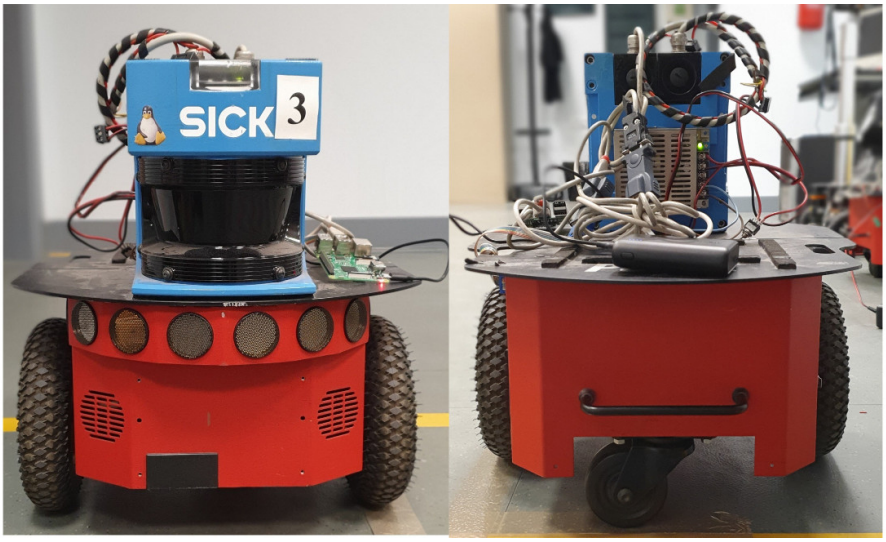


Fig. 2.4 Complete robot set-up used for the practical execution of the algorithm.

compliance with a typical ROS distributed system. In particular, the ROS master node and sensor drivers nodes run on the Raspberry Pi while the navigation and visualization nodes are on the computer.

As already mentioned in previous sections, the first step for the implementation of SGP from a theoretical to a practical point of view has been that of feeding the algorithm implemented in MATLAB with the real map of the environment where the robot can move, while conservatively enclosing fixed obstacles by describing their boundaries. The map has been built by ROS by employing the `gmapping` [45] package, which provides laser-based SLAM (Simultaneous Localization and Mapping) generated upon the OpenSlam's Gmapping algorithm [46]. The resulting map, which is saved as a PGM (Portable Gray Map) file, describes the environment according to the binary occupancy grid format, in which white or empty cells represent the free space for motion while the black-coloured cells indicate the occupied areas (obstacles). The described map is interpreted by the ROS *rviz* tool [47] to build up the cost map that inflates costs, based on the occupancy grid information and physical features of the performing robot. For the sake of simplicity, the original map of the whole research laboratory has been restricted to an area where unexpected obstacles are less probable, in order to be able to test the algorithm within the assumed conditions, leading to a 160x190 cells grid with a resolution of 0.05 $m/cell$, as can be seen in Figure 2.7.

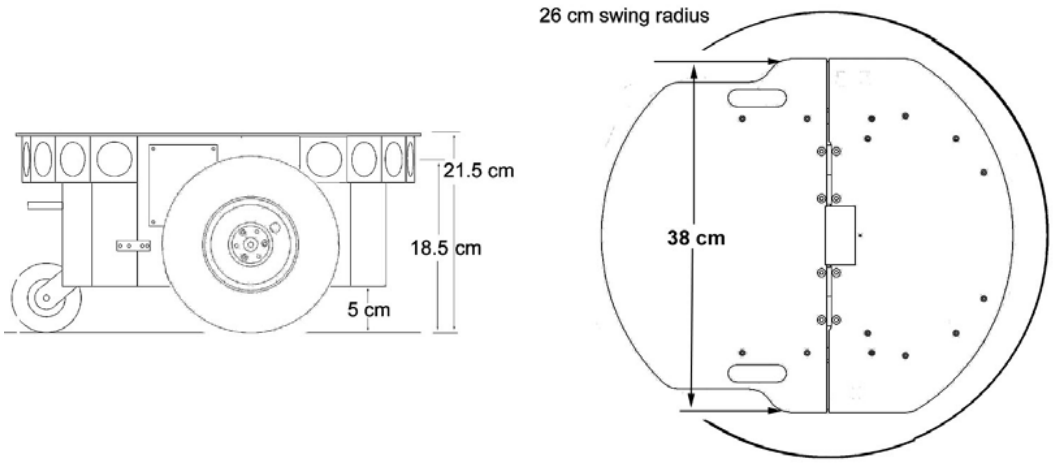


Fig. 2.5 The Pioneer3-DX mobile base dimensions [42].

2.3 Experimental results

This section presents the obtained results through the analysis of three test cases:

1. In the first one, no unexpected obstacles are involved, for instance, no additional obstacles are introduced in the static map.
2. The second one, the algorithm is executed in the presence of an unknown object, which is detected before the GP path computation but it was not initially included in the static map.
3. The third one considers the scenario in which the GP has already computed the path and then an obstacle appears during the robot's motion.

The explored test cases represent a preliminary set of experiments: the idea is to demonstrate the capability of the robot of reaching the desired goal position by traversing the safest path as soon as possible. Notice that, in general, the overall algorithm execution time depends on (i) the chosen GP and LP, (ii) the computational capability of the computer executing the navigation and planning instructions, and (iii) the goal to be reached. The given execution times (relative to each specific test case) are to be considered for our specific choice of planners, for reaching a specific goal position and running the heavier algorithm nodes on a high-performing computer.

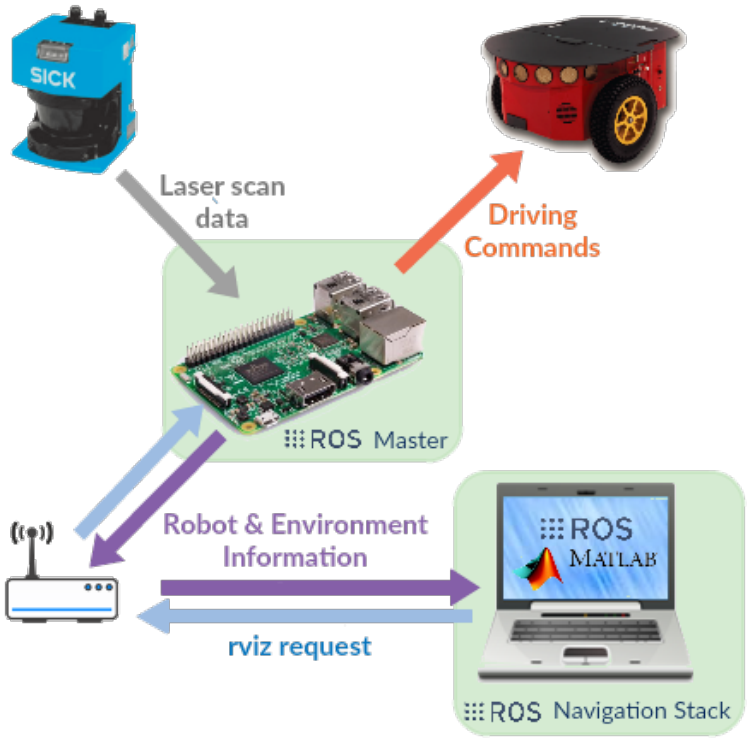


Fig. 2.6 General schema mapping hardware to the software distributed system.

All the considered test case results exploit the *rviz* visualization tool for directly comparing the planned path with the desired SGP trajectory. As mentioned, the LP local cost map, i.e., the portion of the global plan of which the LP has knowledge has been reduced. The chosen values for the cost map dimensions represent a trade-off for ensuring a faithful tracking of the GP plan and obstacle avoidance. For all the tests, the initial and final positions (expressed in meters) with the format (x, y) have been set to $(4, 7)$ and $(2.5, 2)$, respectively. Figure 2.8 shows the testing trajectory, in particular, the blue dashed line indicates the desired safe curve to which the path planned by the SGP (green solid line) tends, and avoids any intersection with the boundary curves (red lines) limiting the obstacles.

Furthermore, the execution of the algorithms in all test cases has been recorded in a video that can be found in [48], which demonstrates the achieved results.

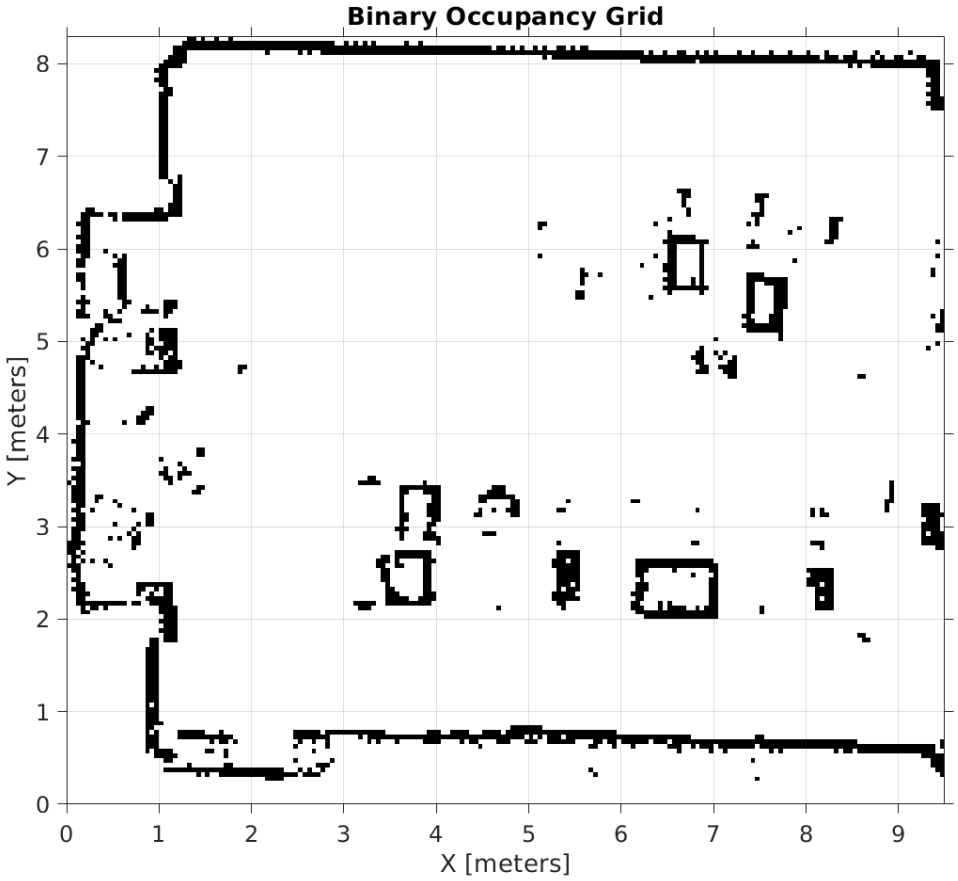


Fig. 2.7 Employed portion of the mapped laboratory in Politecnico di Torino.

2.3.1 Test Case 1: absence of unexpected obstacles

The results obtained from the execution of the proposed algorithm in the first test case are reported in Figure 2.9.

As it can be seen, the GP (green line) faithfully tracks the SGP generated path, and the LP also follows precisely the GP as a consequence of the ad-hoc tuning performed on the local cost map parameters. Note that, if the A* algorithm were used, it would generate a plan going straight to the goal position, since it is the shortest distance. However, the optimal solution considered in this work, in terms of safety, is to lead the robot to reach a safe virtual track as quickly as possible, making it part of the plan. This expected behaviour can be viewed within the “Test Case 1" section of the video: the time needed to reach the goal position is 43.05 *s*.

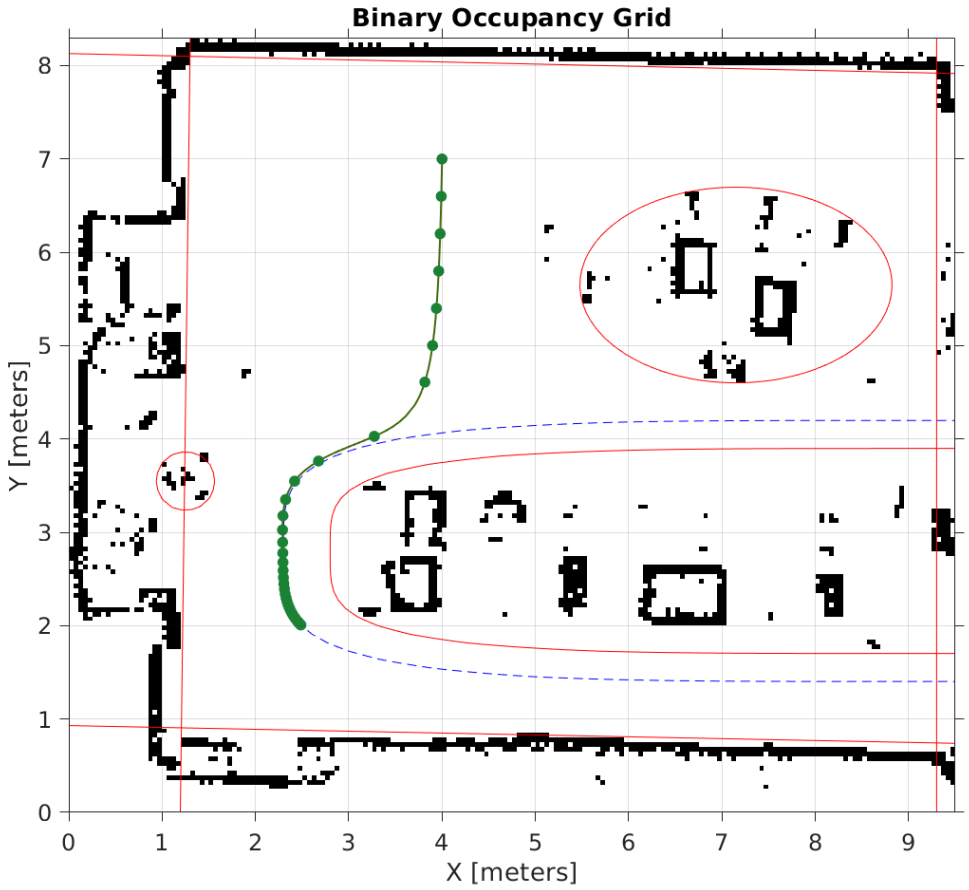


Fig. 2.8 MATLAB simulation plot for the SGP algorithm. A subset of the computed waypoints is highlighted with green dots.

2.3.2 Test Case 2: behaviour with obstacles not in static map

The resulting behaviour generated by the execution of the proposed algorithm in the second test case is reported in Figure 2.10.

It can be seen that the path generated by the GP deviates from the desired path since the laser range finder detected an unexpected obstacle, but the curve to be traversed is similar to the one computed initially by the SGP; being the global cost map updated with this new information, the traversed path going backwards, i.e., from the destination point to the starting point, will be the same. The test case execution is shown in the “Test Case 2" video section, in which execution time is 57.34 *s*.

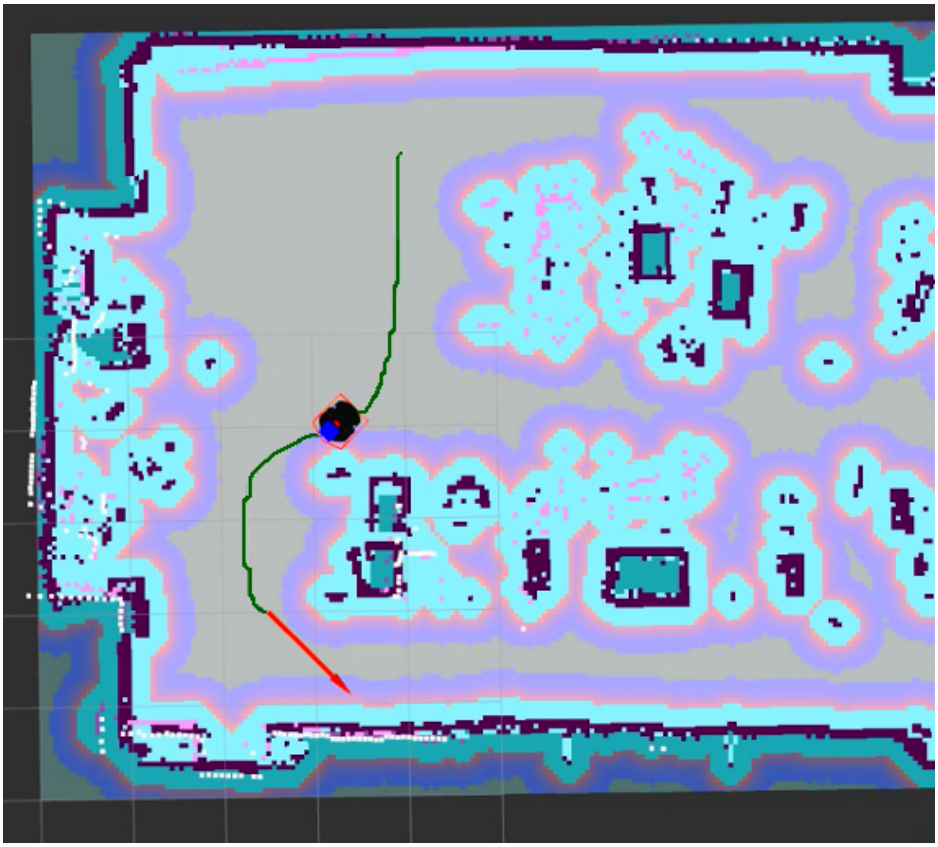


Fig. 2.9 *rviz* view during Test Case 1 execution.

2.3.3 Test Case 3: behaviour with dynamic obstacles

The execution results of the proposed algorithm in the third test case are reported in Figure 2.11.

As can be seen, due to the absence of new obstacles from the beginning, the GP computes a path coherent with the desired one. Instead, when an unexpected obstacle is detected by the sensor data during the robot motion, the LP re-plans the path in order to avoid it while trying to go back to the desired curve after avoiding successfully the obstacle. The present test case has been taken into account in order to demonstrate that, while the robot moves towards the safety curve, if something or someone enters the scene, safe behaviour is preserved, as shown in the “Test Case 3" section of the video, in which a human worker appears at 05:55 and the reaction of the LP allows the robot avoid him. The overall execution time, in this case, is 52.39 *s*.

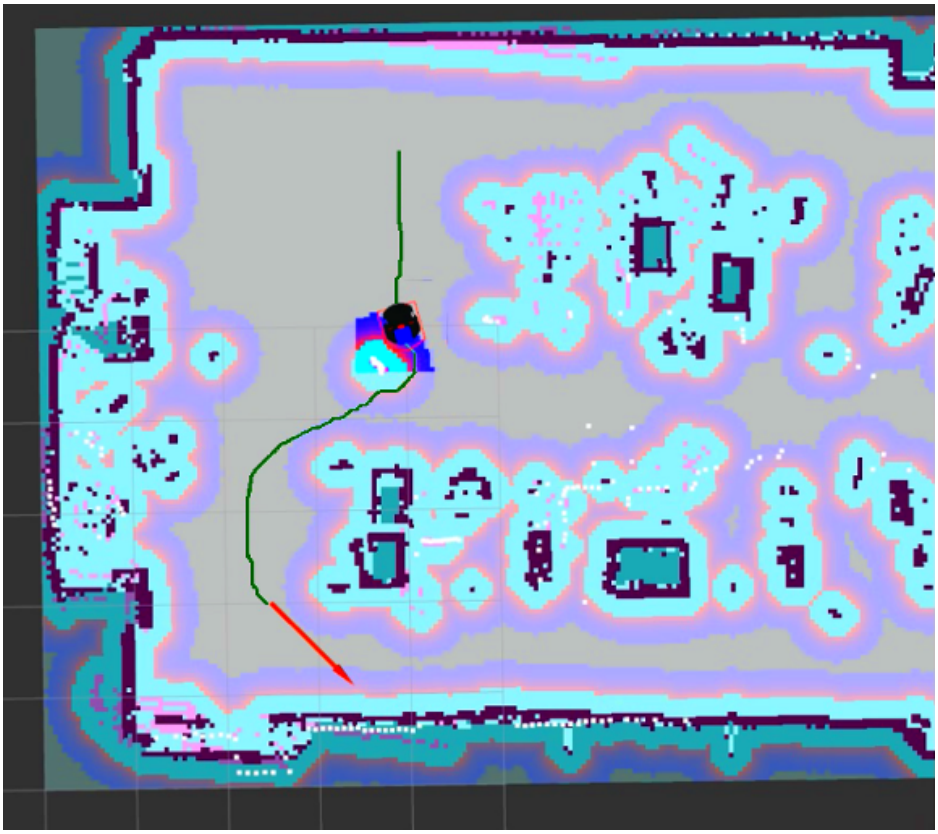


Fig. 2.10 *rviz* view during Test Case 2 execution.

2.4 Discussion

This chapter presents a global planning algorithm ensuring path following and obstacle avoidance. The algorithm was implemented by customizing the well-known A* algorithm to follow a set of waypoints computed by a supervisory planner, tending to a path considered safe. Thanks to a three-level planning procedure, the goals of the supervisory planner are imposed, while taking into account possible differences between the real scenario and the a-priori map used by the SGP, as well as unexpected obstacles to be avoided online. The overall planner algorithm has been experimentally tested by employing a differential robot to demonstrate its feasibility and usability in practice.

The developed algorithm can be seen as an upgrade of the global path planner in the sense that introduces the capability of following a set of pre-defined waypoints while ensuring a collision-free motion. In fact, the traditional global planning algorithms with path tracking do not include the feature of avoiding unforeseen

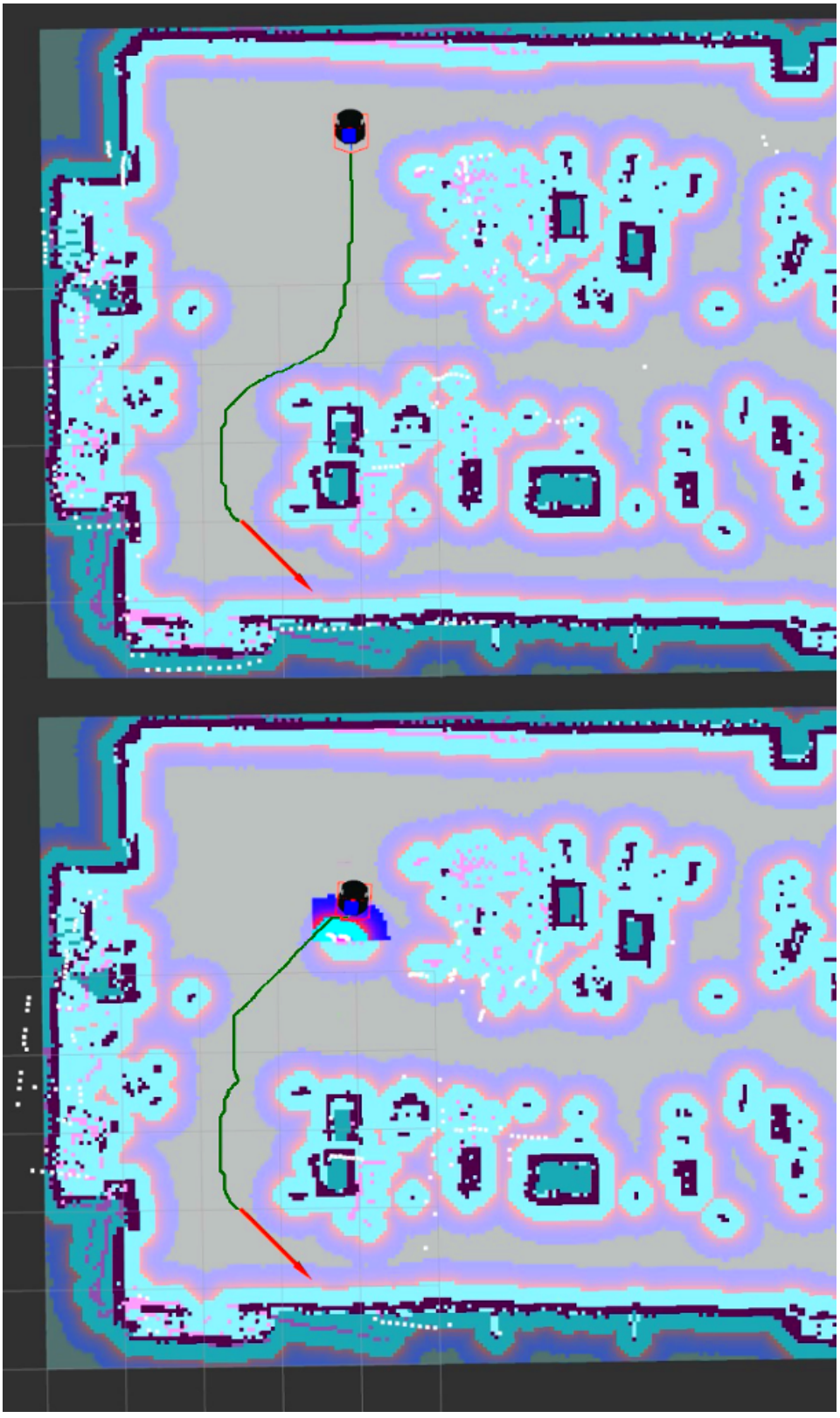


Fig. 2.11 *rviz* view during Test Case 3 execution. The top figure reports the plan before the obstacle appears, while the bottom one shows the re-planning action.

obstacles, which is a crucial requirement in the upcoming industrial scenario that features human operators and mobile platforms.

Moreover, the presented algorithm employs the A* path search with obstacle avoidance when the mobile robot's position is located significantly far away from the desired waypoints. In such a way, whenever the platform is sufficiently close to one of the SGP-computed waypoints, it is "attracted" by the path previously generated by the SGP. Nevertheless, one of the waypoints can be reached by simply sending a specific goal position to the mobile base, through the Action server provided by ROS.

The role of this research work is mainly the implementation of a laboratory real robot demonstrator executing the SGP algorithm. At this stage, the SGP plan computation is executed offline, but a possible improvement would include the algorithm execution with an online re-planning behaviour involving a new computation of the SGP as well. Indeed, by taking advantage of event-based systems, it is possible to trigger the replanning behaviour of the SGP, as will be presented later in Chapter 3.

Furthermore, one of the objectives for future works is to extend the proposed navigation paradigm to multi-agent scenarios. Indeed, by taking advantage of the results given in [33], the supervisory algorithm reviewed in Section 2.2.1 can be adapted to control and coordinate multiple mobile agents with the aim of patrolling selected paths with a prescribed formation, while avoiding collisions with each other and with static obstacles in the environment. This algorithm is presented later in Chapter 5, where the path planning for multi-agent systems performing patrolling and formation with collision avoidance is proposed.

Such an extension of the proposed planning algorithm is aimed at meeting the main requirements of the Smart Factories of the future, in which a growing presence of autonomous mobile robots is expected to enhance the flexibility of the production lines and to guarantee safety towards human workers for human-robot shared environments.

Chapter 3

An online supervised global path planning algorithm for AMRs with human-obstacle avoidance

This chapter presents an online supervised global path planning algorithm for AMRs with human-obstacle avoidance in industrial-like environments. The presented algorithm is an update of the one presented in Chapter 2 that supervises the path planning hierarchy in order to obtain a path considered safe for a mobile robot and updates online the path when unexpected obstacles intersect its trajectory when it is in motion. The aim is to provide deterministic path planning since the environment is shared with human operators, whose description is available in [6]. In particular, the algorithm makes the distinction between general obstacles and humans during path re-planning behaviour, so as to guarantee safety towards human workers while optimizing the navigation when general obstacles are encountered during motion.

This chapter is structured as follows: some relevant background information related to this research work is reported in Section 3.1. In Section 3.2 the proposed algorithm is presented, at first providing some details about the online supervisory planner, then describing the algorithm implementation. The hardware and software setup for testing the procedure is unfolded in Section 3.3, where the experimental results obtained from different test cases are analyzed. Finally, the research work is discussed in Section 3.4.

3.1 Background

Smart factories envisage the scenario in which mobile agents work in a space shared with human operators. In this context, the employment of AMRs is preferred due to their capability to process the information coming from their surroundings, thanks to the onboard intelligent sensory system. Nevertheless, planning the motion of a mobile robot is not an easy task. Path planning is a well-known problem for computing feasible obstacle-free paths that allow the robot to reach a desired destination. The complexity of the problem may increase and depends on specific optimization requirements and a-priori information about the environment, as well as the presence of dynamic obstacles, such as human workers in industrial working spaces.

Hence, there are several state-of-the-art algorithms developed for robot navigation. For instance, there are navigation algorithms based on probabilistic-search methods, such as the Rapidly-exploring Random Tree (RRT) [23], RRT* [27], and the Probability Roadmap (PRM) [22], that randomly sample and explore the free spaces on the map to find a feasible path connecting the starting and final poses. Despite several improvements of such methods and their wide use thanks to their simplicity, they do not guarantee the optimal solution due to the fact that the graph network is randomly generated at each iteration[28].

Alternative path planning algorithms are based on heuristic-search methods, such as Genetic Algorithms (GAs) [49], Dijkstra [15], A* [16] and D* [17]. The GAs are inspired by Darwinian evolution, by selecting the fittest sub-optimal solutions for reproduction, so as to produce offspring of the next generation. On the other hand, A* is an improvement of the Dijkstra algorithm to find the shortest path, whilst D* is an algorithm based on the A*, but with a dynamic cost behaviour.

The traditional A* algorithm has been widely employed in several path planning problems due to its simplicity and high efficiency in finding the optimal solution. Nevertheless, even though it does not provide a safe and smooth path, it is still used extensively due to its versatility, since specific requirements can be fulfilled by customizing and/or combining with other path-finding algorithms. Therefore, many variants of the A* algorithm were proposed. For example, in [50], a modified A* algorithm is presented, in which virtual obstacles are included in the environment in order to guarantee safety during the path planning process. The enhanced A*

algorithm proposed in [51] considers the safety and time cost in the objective function, so it can be executed also in complex terrain environments.

Another motion planning approach that is recently emerging is based on Artificial Potential Fields (APFs), in which the path is generated by attracting the robot towards the desired destination while it is repelled from obstacles [52]. A Membrane Evolutionary Artificial Potential Field (memEAPF) is proposed in [53]. The parameters for generating a feasible and safe path are obtained by combining three methods: APF, membrane computing and a genetic algorithm. In [54], the authors introduced a navigation system for dynamic industrial cluttered environments that merges and process the data coming from a sensor network and an APF-based navigation algorithm.

The scenario where human operators and mobile agents share the same working space in an industrial-like environment, safety is a crucial factor to be considered. Nonetheless, most of the navigation algorithms do not take into account the human factor, since: (i) a human moving within the environment is a dynamic obstacle whose behaviour is too complex to predict, or (ii) human operators are treated as generic dynamic obstacles without any specific distinction [55].

Instead, there are some safety concepts presented in [56], that consider the usage of mobile robots in different scenarios. Furthermore, there are currently no standards for safe navigation in the industry for AMRs. However, some guidelines are recently being developed (R15.08 Drafting Subcommittee presentation from the Autonomous Mobile Robot Conference, September 2019 [57]). The current working scenario synergistically involves human operators and fixed-base or mobile robots, making the latter ones part of the production line. This creates new questions arise about the operators' willingness to work closely with unpredictable machines (as usually the AMRs are) [58].

This chapter refers to an upgrade of the Supervisory Global Planner (SGP) architecture presented in [4] that integrates an online replanning mechanism for the computed path with the human detection capability introduced in [5]. Consequently, the updated supervisory algorithm enables a more conservative local behaviour around human obstacles. On one hand, when considering the case of quasi-static scenarios, the location of the obstacles in the environment are well known, so the path produced by the SGP can be considered as safe. On the other hand, when a human operator intersects the path while the robot is in motion, the local planner

deviates the robot from the human obstacle to avoid the collision. After that, the robot successfully avoided the human operator, and the computation of the SGP is triggered considering the current robot pose as a starting point.

The fundamental point is given by the definition of predefined safe paths, to be followed by the mobile agents whenever possible, in order to reduce the risk of collision with unexpected obstacles along their motion. The aim of the virtual safe paths is quite similar to the fixed paths followed by traditional AGVs. Moreover, this is enriched by the capability of handling the presence of dynamic obstacles at the same time, so as to guarantee a safer avoidance policy in case of the presence of humans and to catch up on the safe path as soon as possible. The resulting performance is especially applicable for industrial contexts, in which a proper level of autonomy must be left to the mobile agents to really exploit them in the development of efficient and flexible production lines while assuring simultaneously safety conditions adequate for the presence of human operators.

The path planning algorithm run by the supervisory planner has a repetitive and deterministic behaviour, namely, given the starting and goal poses, the computed path will always be the same. It is then considered as safe not only because it automatically avoids all the known static obstacles, but also because it provides a fixed virtual path followed by the mobile robots, in the case that there are no variations in the environment along such a path. As a result, the human operator who is working cooperatively with mobile agents is aware of the trajectory of the robot, and can eventually avoid intersecting the robot's safe path. However, in the case where a human operator unintentionally crosses the dedicated pre-defined route, the proposed algorithm is able to conservatively avoid obstacles classified as humans even though the latter ones are not expected.

The combination of the mentioned functionalities guarantees an overall safe behaviour since the robot has the capability to (i) follow a safe virtual path, (ii) re-plan its trajectory when a human obstacle is nearby and (iii) react accordingly to the environment's changes.

3.2 Online Supervisory algorithm for safe path traveling with human obstacle avoidance

In this section, a description of the developed online path-planning algorithm is provided.

Typically, the mobile robot path planning is executed at two levels: Global Planning and Local Planning. Given a starting point and the destination coordinates in an environment map, the Global Planner (GP) computes offline a feasible path taking into account only the static obstacles already included in the map, while the Local Planner (LP) updates the computed path within a local window and it is triggered when the sensors recognize the presence of unexpected obstacles, allowing the robot to avoid any potential collision while moving.

The integration of the SGP as the highest level within this planning hierarchy permits the robot to follow a set of deterministic waypoints, along a safe virtual path. The SGP is based on the collision-free motion planning algorithm introduced in [34], in which the output is a path that tends to an algebraic curve. In particular, the SGP takes into account the kinematic model of the unicycle robot, and the safe curve is obtained letting the intersection of the obstacle space and a set of polynomial functions describing the possible trajectories be empty. For the sake of briefness, the equations describing this curve will be omitted here, since they were already presented and experimentally validated in [4]. Nonetheless, in the latter, the SGP computed the waypoints offline and only once before the robot is put in motion, so whenever the LP deviated the robot due to the presence of unknown obstacles, it was not ensured to resume the motion along the SGP computed path. Actually, the modified A* algorithm (included in the SGP algorithm) is intrinsically attracted by the imposed safe curve when the robot is sufficiently near to it, while only the pure A* planning mechanism is kept otherwise. Also, a human obstacle was treated in the same way as a generic object since the obstacles were detected but not identified.

With the aim to achieve safer behaviour, in terms of human obstacle detection and avoidance, the features of smart AMRs introduced in [5] are integrated into this work, so the human worker can be identified and published as a virtual obstacle, whose inflation radius is wider than the one around generic obstacles. When it comes to safe path conservation, an event-based trigger is included in the SGP. This planner will

be reported in the next sections as the Online Supervisory Global Planner (OSGP). The updated hierarchical planning architecture is illustrated in Figure 3.1.

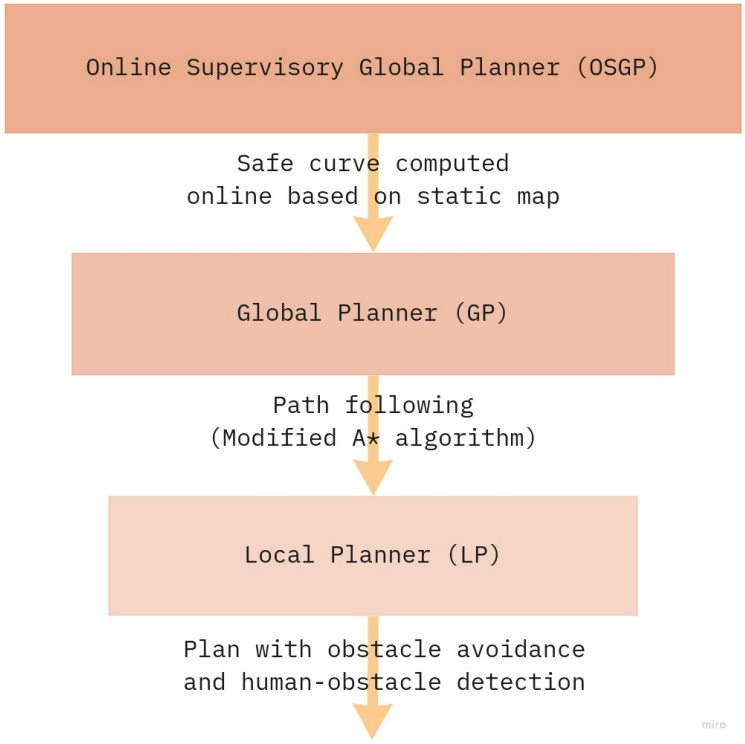


Fig. 3.1 Hierarchical structure of the online supervised global planning.

3.2.1 OSGP algorithm implementation

The program for calculating the SGP path is performed in MATLAB and provides as output a set of waypoints from an initial pose towards a destination pose. The MATLAB function also registers as a ROS (Robot Operating System) [36] node, in order to transmit the planning data onto the ROS Parameter Server. The communication with the robot ROS network is performed through the MATLAB ROS Toolbox, which enables the local machine to compute and send the waypoints and read the status of the robot in order to activate the re-planning mechanism.

The values of the waypoints are properly namespaced in order to ease potential multi-agent developments, and made available to the ROS system nodes handling the autonomous mobile robot navigation and vision-enhanced obstacle avoidance. It is worth noting that the whole MATLAB part is executed in *headless mode*, namely,

the code is run from the command line with specific options that suppress the display server, the splash screen display, and the desktop version modules. As a consequence, the CPU usage is significantly reduced each time that the MATLAB code is run.

Following a system of event-based triggers, the SGP algorithm generates a new path only when strictly necessary, so as to ensure deterministic and time-efficient behaviour. As previously mentioned, the path computed by the OSGP can be considered safe, since it is based on a binary occupancy grid map in which static obstacles are conservatively enclosed by ellipses of minimum radius. The GP implements a modified A* algorithm where waypoints passed by the OSGP are favoured in terms of cost, for the heuristic global plan computation. As a human obstacle is detected and sufficiently close to the AMR, the robot is deviated by the LP computed path and a trigger is sent to the OSGP for a new global path computation. A small delay is introduced during the replanning phase, in order to time correctly the trigger of the OSGP computation while the robot is already overcoming the human obstacle. The current pose of the mobile platform is taken into account and used as a starting pose for the collision-free motion planning algorithm.

The dynamic obstacles are avoided thanks to the LP and, in particular, human obstacles are identified using the real-time object detection system YOLO (You Only Look Once) [59]. The C++ YOLO code has been modified to filter the information about the bounding boxes enclosing the identified obstacles, so as to consider only the pixels labelled as “person”. These data are then written to a text file, which is fed to a ROS topic. The identified humans’ relative distances are subsequently computed, exploiting camera-laser data sensor fusion. The humans’ positions on the map are then published as virtual obstacles enclosed in virtual cages. In order to condition the local planner costmap, the inflation radius value assigned to humans is larger than other obstacles. This avoids the AMR from travelling too near to the operator when trying to surpass it. For further details about this functionality, refer to [5]. The overall algorithm flow diagram is shown in Figure 3.2.

To better emphasize the main differences from the SGP algorithm proposed in [4], the intended algorithm output is illustrated in Figure 3.3. In the case when the offline SGP version is used, a considerable deviation from the supervisory path could result in the activation of the pure A* search mechanism, which finds the shortest route (grey dotted line) to reach the final goal. Nevertheless, this is unintended, due to the fact that it is not compliant with the safety requirements, since there may

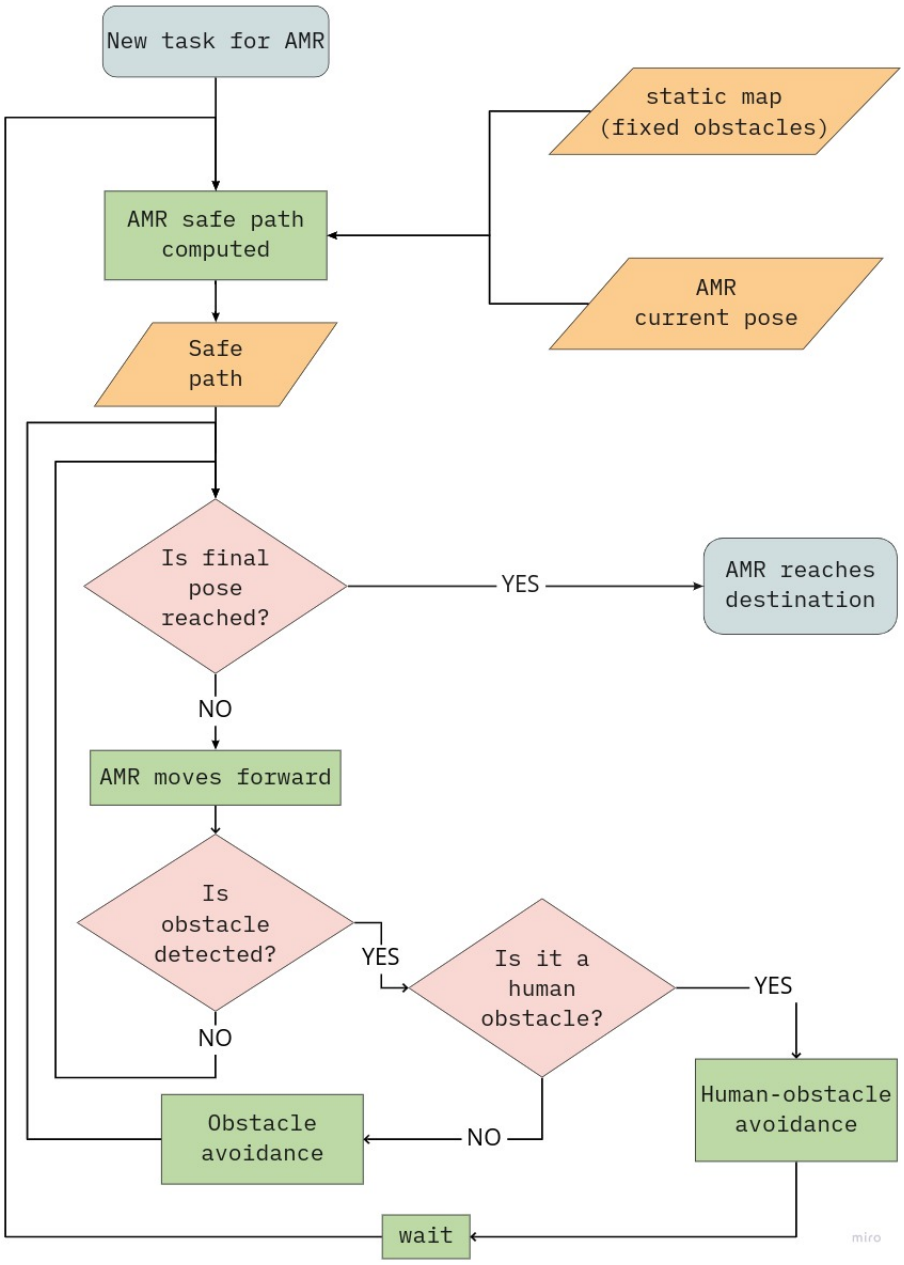


Fig. 3.2 Flowchart for the OSGP algorithm.

be other human workers outside the safe route. To make up for this undesirable behaviour, the SGP safe virtual path is re-planned online, taking as starting pose the current one (blue solid line) after avoiding the human.

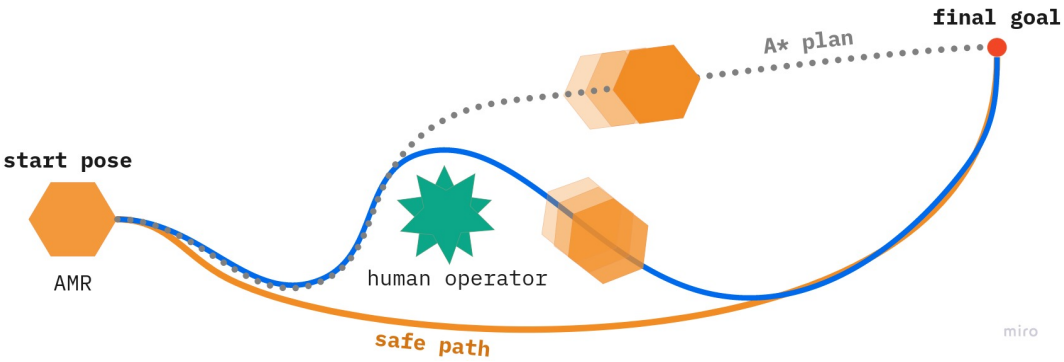


Fig. 3.3 SGP behaviour versus OSGP behaviour.

Furthermore, the proposed algorithm presents some development strategies that can improve the technology criticality level [60] and can be considered as a catalyst for simplified transferability to real industrial contexts.

- *Containerization as portability facilitator.* The OSGP algorithm with human obstacle detection and avoidance has been containerized using Docker [61]. There are some elements which can be considered robot-independent and they have been grouped based on their main task, for instance, AMR vision, AMR navigation, and SGP computations. Hence, to facilitate the migration to other software environments and hardware specifications, all the robot-specific elements and controllable parameters have been put together to enable suitable edits. The use of containers enables a quicker transfer process from laboratory demonstrators to commercially available AMRs. In fact, the Linux containers technology is considered a lightweight alternative to virtual machines [62], and lets the end-user execute the containerized application having the installation of Docker on the target machine as the only constraint.

- *Cost-efficient improvements as technology transfer enablers.* Using cost-efficient sensors boosted by deep learning algorithms can be considered a key factor for technology transfer in contexts that would, as a matter of principle, not consider it. By exploiting sensors which may not be classified as high-end and/or high-tech devices, the goal is moved from the economic value toward the innovative meaning of a resulting solution. Upgrading obsolete equipment can foster the adoption of new technologies to avoid exacerbating low technology transfer rates that may affect Small and Medium Enterprises [63].

3.3 Experimental results

In order to test and validate the OSGP algorithm in an environment similar to an industrial-like context involving a mobile agent navigating in a closed area shared with human operators, so as to emulate scenarios such as warehouse corridors with racks or assembly workstations with conveyors. The working space used for testing can be seen in Figure 3.4, reporting the MATLAB occupancy grid map, the plot of the OSGP path and the corresponding GP path plan visualization on ROS *rviz*.

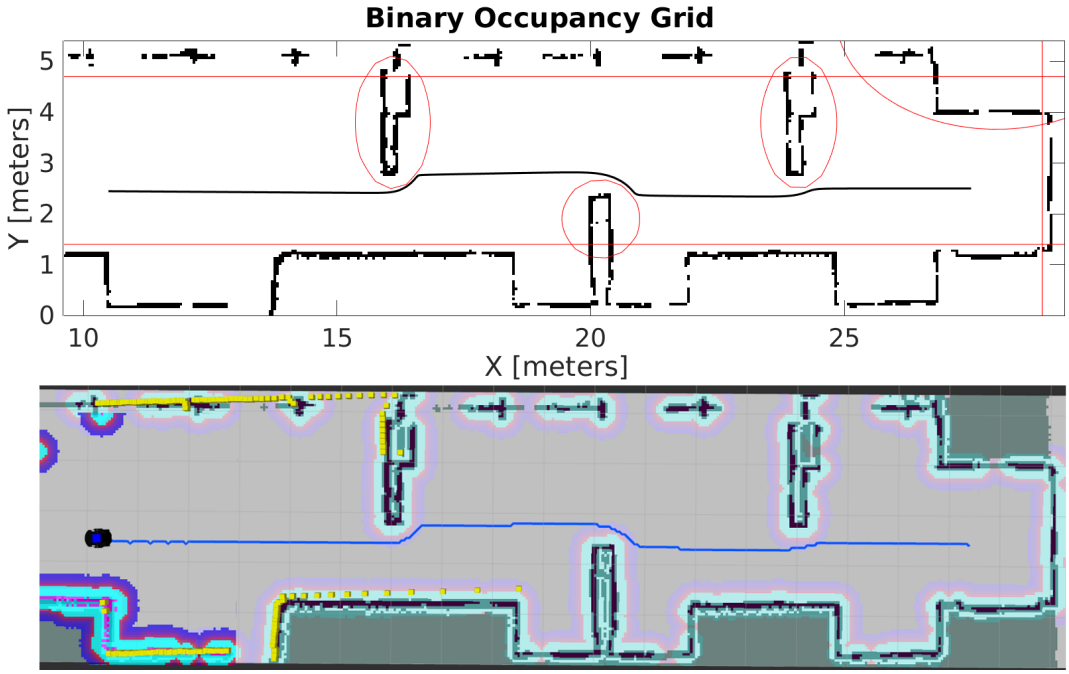


Fig. 3.4 Top: MATLAB OSGP path plot. Bottom: *rviz* visualization of the static map used for SLAM navigation.

.

To demonstrate the validity of the proposed algorithm, a Pioneer 3DX mobile robot has been employed, whose technical specifications have been already described in Section 2.2.3. The video streams from an entry-level IP camera and serves as a visual source, fed to the YOLO real-time object detection system. Specific setups for the camera-laser calibration and data fusion will be illustrated in Chapter 6. In addition, the core processes, such as the supervisory path computation and video processing, were performed on a desktop PC with an Intel Core i7-7700 CPU and a dedicated GTX1060/6GB GPU.

The overall high-level setting up diagram is represented in Figure 3.5, while the execution of the online SGP algorithm in different test cases is showcased in the video footage available at [64].

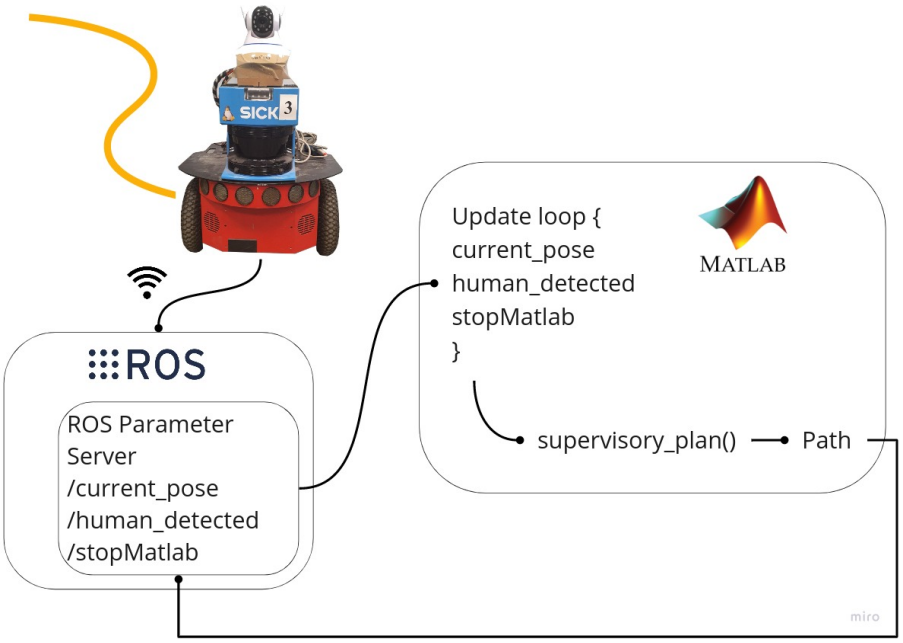


Fig. 3.5 High-level algorithm implementation schema.

It is worth pointing out that some open issues identified in the real-world implementation of the offline SGP algorithm have been solved. Among these, a smoother behaviour has been achieved by replacing the `TrajectoryPlannerROS` local planner with the `TebLocalPlannerROS`, based on Timed-Elastic Bands (TEB) evaluation [65], in which three alternative paths are generated online and the most feasible one is chosen to circumnavigate the obstacle. Also, the MATLAB code has been executed with its GUI switched off, meaning that it was run in *headless mode*, so as to get rid of the time necessary for the interactive program parts to load, since no further interaction is needed during the algorithm execution. In particular, the supervisory planner function is run according to specific trigger flags which permit MATLAB and the ROS system to interact and automate the re-planning process.

3.3.1 Test Case 1: OSGP re-planning behaviour

In this first test scenario, as shown in Figure 3.6, it has been tested the re-planning behaviour of the OSGP when a human obstacle is identified. In this case study, the person represents a human worker performing some operations in front of a workstation. As can be seen at 00:41, the AMR starts navigating along the safe route generated by the SGP and tracked by the GP plan (blue solid line) and when a person is detected, the LP (orange solid line) initiates its standard obstacle avoidance mechanism.

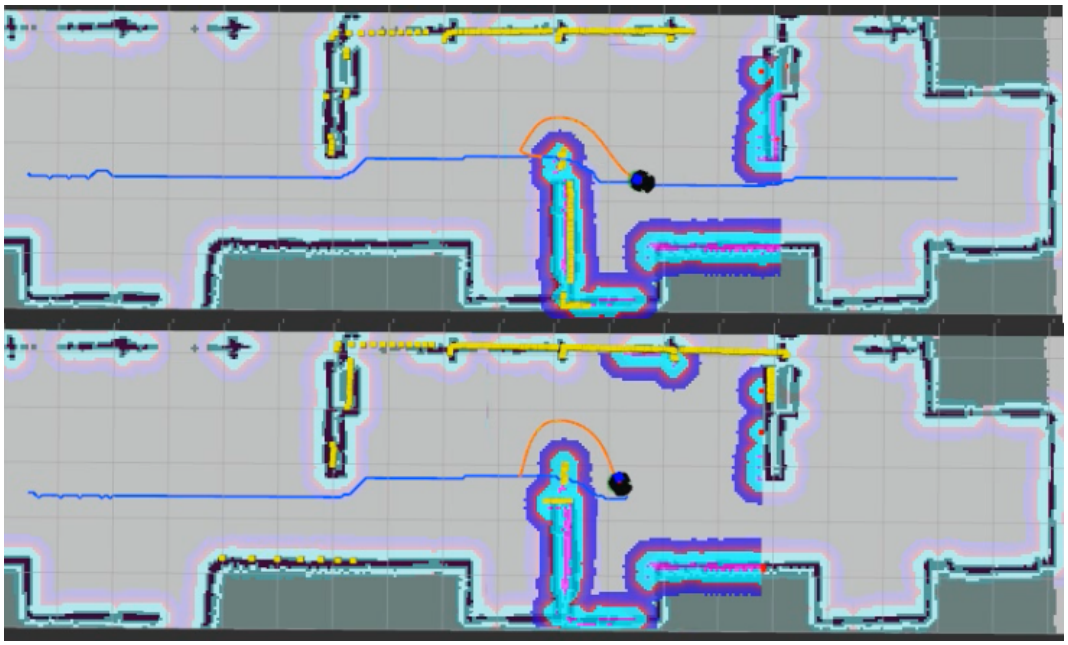


Fig. 3.6 Test Case 1: *rviz* view of the OSGP re-planning behaviour after human detection.

As the human gets closer to the robot in motion, the OSGP is eventually activated (00:58) and a new set of waypoints is provided to the GP. In this way, the mobile robot is always brought back towards the safe path even if its motion has been significantly deviated. Furthermore, it can be seen that the robot overcomes a person while maintaining a conservative distance defined by the virtual obstacle publication.

3.3.2 Test Case 2: human detection without re-planning

For the second test case, it can be appreciated at 01:15 of the video the scenario in which an operator is performing some quick checks along a rack is detected

but, being far enough from the current AMR location, the OSGP path re-planning mechanism is not triggered. Indeed, in this case, the human presence does not disturb the mobile robot activities and a re-planning would introduce unnecessary overhead or delay. It is worth noting that the operator location is published as a virtual obstacle on the map (red dots). A screenshot of the performed test is given in Figure 3.7.

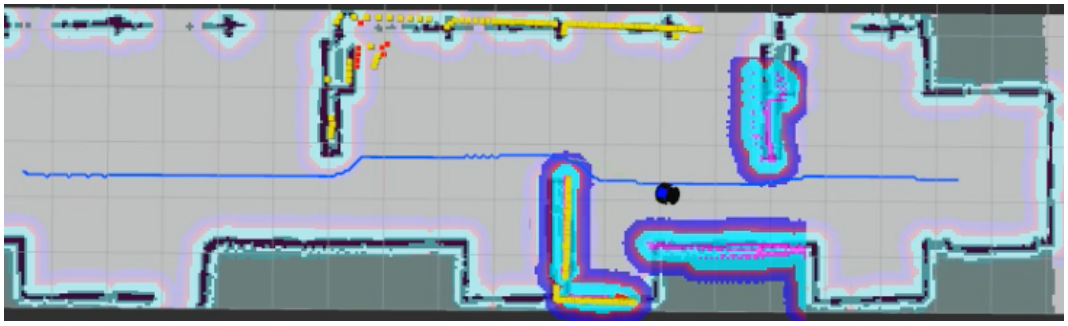


Fig. 3.7 Test Case 2: OSGP behaviour when a human obstacle is sufficiently far from the AMR.

3.3.3 Test Case 3: generic obstacle avoidance

The last test case, as shown at 01:37 of the video, takes into account the scenario where the robot encounters a generic obstacle, which is not originally present during static map building. The robot successfully avoids it without activating a new safe path computation. This is an intended behaviour so as to avoid further computations when it is not necessary. Namely, the absence of a human operator in the robot neighbourhood does not impose a reactive resume of the safe virtual path. The OSGP behaviour in this third scenario is shown in Figure 3.8.

Fig. 3.8 Test Case 3: OSGP algorithm reaction to generic obstacle detection.

3.4 Discussion

In this chapter, a supervisory planner algorithm that computes an online-updated safe path is presented. In a semi-structured environment, the motion of the AMR follows the safe virtual path that is generated by the OSGP, which can be repetitive and deterministic. This capability can potentially increase the workers' confidence in sharing the working space with mobile robots since their overall motion can be predicted. Besides, using YOLO, the AMR is able to differentiate if an obstacle is a generic object or a human operator, and estimate its distance through camera-laser data fusion. By publishing the human obstacle as a virtual one within the navigation map, its increased inflation radius enables a safer avoidance since the robot surrounds the human with a more conservative distance.

The pursuit of the reference safe virtual path is always guaranteed, due to the fact that the re-planning behaviour of the OSGP is triggered only when a human is near to the robot in motion. Additionally, thanks to the ROS namespace feature it is possible to adapt the algorithm to a multi-agent context, while the containerization using Docker fosters the technological transfer towards industrial processes by enabling an easier portability.

Even though some of the development decisions can ease the criticality level of the proposed algorithm, being the range of AMRs available on the market very wide, some custom tuning for integration would be necessary. For sure some testing procedure on a more realistic scenario would be more indicative of the usability and feasibility in an industrial context.

For the time being, the OSGP re-planning mechanism is activated after a small delay. As a future enhancement to make the system more efficient, the robot re-planning behaviour while it is overcoming the human obstacle could be ensured by selecting a different trigger mechanism, to make sure that the robot has indeed left the original safe virtual path. For instance, the trigger signal could be imposed if the distance from the initially defined path is higher than a certain threshold, so as to reduce the number of triggering events. Actually, the re-triggering mechanism after a specific waiting time does not guarantee that the robot has moved from its current pose, due to the fact it may be delayed by some internal computation lag or external disturbances.

It is worth noting that the online computation of the SGP could be fully implemented in ROS, for instance, by writing the code in Python and using a proper differential equation solver. Nonetheless, MATLAB has been chosen since its solvers provide good enough results and can be easily included within other systems.

Chapter 4

Dynamic Path Planning of a mobile robot adopting a costmap layer approach in ROS2

In this chapter, a Dynamic Path Planning of a mobile robot adopting a costmap layer approach in ROS2 is introduced. The main goal of this chapter is to present the Dynamic Obstacle Layer (DOL) approach, reported in [7], as a plug-and-play solution to enhance the handling of dynamic obstacles for mobile robot path planning. The idea is to include the DOL to the current ROS2 Navigation Stack with the information of moving obstacles in the environment, obtained through a generic 2D LIDAR sensor, whose preliminary results are available in [66].

The structure of this chapter is the following: first, a brief background and research motivation is introduced in Section4.1. Section 4.2 outlines some capabilities introduced in the Nav2 architecture, so as to highlight how the DOL can be integrated. Then, some related works, considered as a starting point for this study, are presented. Section 4.3 illustrates some fundamental concepts and the theory behind the DOL methodology, while Section 4.4 describes the main steps followed for the development in the ROS2 framework. The results are analysed in Section 4.5. Finally, the overall work is discussed in Section 4.6.

4.1 Background

Nowadays, mobile robots are frequently employed in many fields, such as industrial automation, transportation, monitoring, surveillance, personal, military and medical applications.

For this purpose, AMRs perform autonomous navigation, typically achieved by integrating several functionalities, such as perception data, localization, cognition and motion control. In particular, mobile robot navigation can be decomposed into the following tasks [67]: (i) modelling the environment as a grid map, (ii) computation of collision-free trajectories, and (iii) path pursuit while avoiding collision with obstacles.

The last two tasks are generally associated with the *motion planning* problem [68], which boosted over the years the design and development of several methodologies for its solution, based on different mathematical approaches and technologies [69]. Path planning algorithms can be performed within two hierarchy levels: global and local planners. The former exploits the information of the static map to generate a feasible obstacle-free route to navigate from one point to another. The latter instead, calculates new intermediate waypoints that consider the local information provided by the sensors. Such waypoints deviate the robot from those obstacles that were not known a-priori, while matching as much as possible the ones provided by the global planner after surpassing the detected obstacles. There are several dynamic obstacle avoidance techniques that recompute the trajectories by generating arcs, segments, clothoid lines, etc, whose outputs are intermediate waypoints that deviate the robot from dynamic obstacles [70].

Robot Operating System (ROS) is a well-established open-source framework for developing robotic applications. In particular, the ROS navigation stack metapackage [71] constitutes a set of software libraries widely employed for robot autonomous navigation. Among the local planners made available in ROS, there are the Dynamic Window Approach (DWA), Elastic Band (EBand) and Timed-Elastic Band (TEB).

The DWA is an online collision-free navigation algorithm that considers the motion dynamics of the mobile robot, so it is a local planner able to deal with the velocity limits and acceleration constraints of the robot. Its operating principle includes two main phases: first, it calculates a valid velocity search space, which is built from the set of velocities that generate feasible safe trajectories, and then

it selects the optimal solution through a cost function evaluating the trajectories scores [72]. EBand deforms the global path using an artificial force model [73] each time when new obstacles are spotted. In particular, an elastic band is created by a contraction force that pulls the robot towards the destination position, while a repulsive force pushes the path away from obstacles. Moreover, the TEB algorithm is an upgraded version of the EBand, in which the time information is included to the planning process, so there is a cost function that takes both the driving time and the distance to obstacles into account.

Nonetheless, scheduling the path re-planning process becomes a problem when the dynamic obstacle intercepts the recomputed path, so the TEB generates three elastic bands as alternative paths and the shortest one compliant with the planning requirements is selected [74], thus allowing the robot to navigate in the smoothest way between the obstacles. The choice of the most suitable planner depends on the navigation requirements, and it is generally a trade-off among precision, speed and performance. For instance, according to [75, 76], DWA planner stands out since it needs small computing power and the results are repeatable in consecutive tests, on the other hand, EBand provides more accurate results, and TEB has the quickest reaction to the dynamic obstacles, although it requires more computing resources due to the fact that it tries to optimize multiple trajectories.

Currently, there are two ROS versions and from now on the first one will be referred to as ROS1, while the second generation will be referred to as ROS2. Despite ROS1 usage has incremented a lot since its first distribution, it has some architectural limitations that prevent it from being competitive with other solutions. Indeed, ROS1 requires significant computing resources and cannot ensure fault tolerance, deadlines or process synchronization and, most importantly, it does not satisfy real-time execution requirements [77]. The first ROS2 distribution was released in 2017 with the design goals that are should be compliant with real industrial requirements, such as support teams of multiple robots, small embedded platforms, real-time control, non-ideal networks, and multi-platform support (Linux, Windows, RTOS) [78]. Also, alongside ROS2, the navigation stack underwent a substantial architecture upgrade as well, giving birth to the Navigation2 (Nav2) stack [79]. Nevertheless, Nav2 still leaves some open issues related to dynamic obstacle avoidance handling, as raised by the ROS2 community [80]. In fact, the existing Nav2 does not embed a strategy for managing dynamic obstacles. First, the navigation is executed according to a costmap representation of the environment, where static cells occupied in the

costmap are inflated by an exponentially decreasing cost decay rate, regardless of whether they are occupied by moving or still objects. Therefore, the robot plans a more pessimistic trajectory and maintains a wider gap from obstacles than actually needed to avoid the collision, since there is a lack of awareness of the time evolution of the occupancy grid. Consequently, the navigation performance of the AMR is compromised by unnecessary delay, which in dynamic factory environments and production departments results in a decrease in productivity and preventable downtimes.

4.2 ROS2 navigation stack

In this section, a brief overview of the ROS2 navigation stack is presented, along with a description of current methods for handling dynamic path planning.

4.2.1 Nav2: A navigation system

The design of the Nav2 are such that all the components are modular and run-time reconfigurable, including also multiple ready-to-use algorithms. The architecture of Nav2 exploits multi-core processors so as to be compliant with the low-latency and real-time features of ROS2. Its core consists of a Behavior Tree (BT) navigator [81] and task-specific asynchronous servers: *Planner*, *Controller* and *Recovery* servers. They are action servers hosting the environmental representation used by the algorithm plugins to compute their outputs, under the orchestration of the BT Navigator Server, as shown in Figure 4.1. In particular, *Planner*, *Controller* and *Recovery* servers are described as follows:

- The task of the *Planner* plugins is the computation of a feasible collision-free and potentially optimal path from a starting pose to a goal pose. This functionality is the correspondent to the robot's global planner in ROS1.
- *Controller* plugins replace the *local_planner* of Nav1. They calculate a feasible control effort to track the global plan, based on local environmental information gathered by the on-board sensors, thus performing local planning.
- The *Recovery* behaviours are plugins triggered by the BT when a navigation failure occurs. Among the available recovery mechanisms, there are:

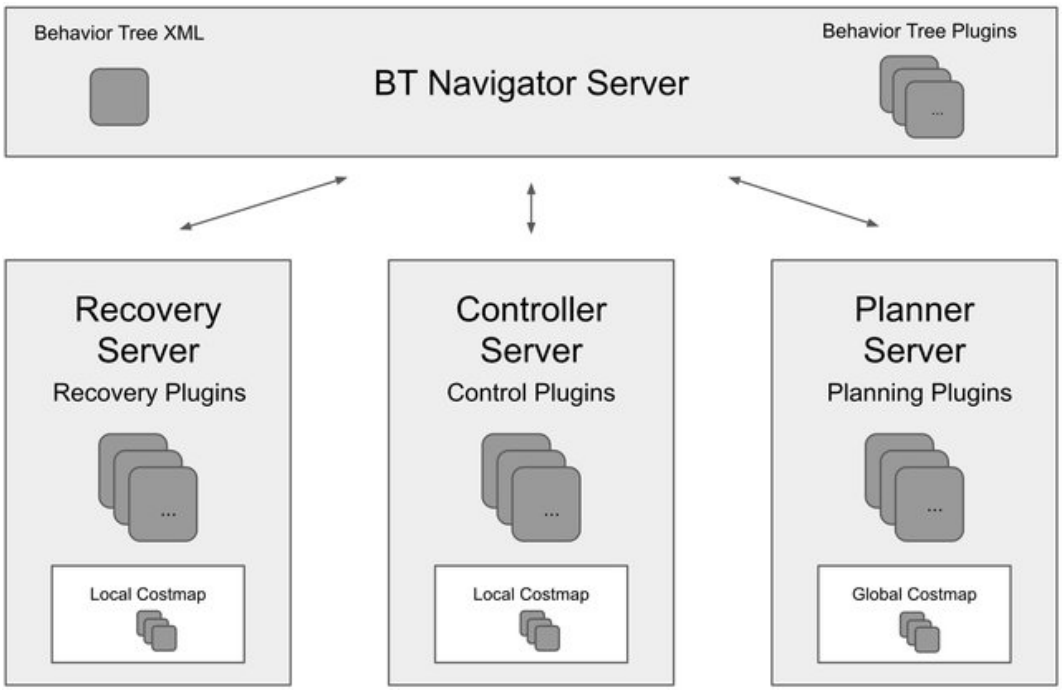


Fig. 4.1 Navigation2 stack architecture.

- Back Up: it backs out the robot whenever it is stuck and has a configurable distance.
- Clear Costmap: it clears a given costmap in order to clean incorrect measurements from the perception system.
- Spin: it rotates the robot with a configurable angle in order to relocate the robot and clear out the free spaces. It is used in the case that the robot perceives itself to be stuck and cannot perform Back Up.
- Wait: It stops the robot with a configurable time in order to update the sensor data or there is a traffic of time-based obstacles.

The *Planner* and *Controller* servers work on two different costmap representations of the environment, the *global_costmap* and the *local_costmap*, respectively. The former is built upon a pre-loaded static map and the latter is based on local sensor data. The huge potential of the costmap representation in Nav2 lies in the adoption of the costmap layers method [82], which are different from traditional monolithic costmaps, where all the data are saved in a singular grid of values. In the costmap layers approach, each layer tracks one type of obstacle or constraint

and then modifies a master costmap that is employed for path planning. Among the available costmap layers in Nav2, the base layers are the following:

- Static Layer: it stores the costs associated with the static map (occupancy information) provided at launch time.
- Obstacle Layer: it continuously marks and clears 2D costmap cells according to sensor data.
- Inflation Layer: it propagates the cost values out from occupied cells (obstacles) that decrease with distance, so as to define a safety margin for the robot navigation.

The dynamic obstacle handling is generally addressed in motion planning at a local level, which means that the Nav2 dynamic obstacle avoidance mechanism should be approached by the *Controller Server*. Indeed, among the Controller Server plugins for local planning implemented in Nav2 there are the TEB Controller [83] (*teb_local_planner*) and the DWB Controller [84] (*nav2_dwb_controller*). In particular, the latter is the successor to the DWA controller in ROS1.

The DOL has been designed as an additional costmap layer plugin to the *local_costmap*, which embeds the information of dynamic obstacles velocities and orientation into the occupancy grid, along with the existing controller plugins.

4.2.2 Dynamic path planning

Being the current flexible production plants highly demanding environments, research is brought on with the aim of enhancing the ROS1 Navigation Stack to make it compliant with industrial highly dynamic and ever-changing environments [85], along with the migration processes from ROS1 to ROS2 to improve and benefit from well-established frameworks, providing valuable capabilities, for instance, task planning [86]. Given its features, ROS2 is appropriate for industrial-grade mobile robots, making it a relevant choice for industrial applications to achieve safe dynamic obstacle avoidance. There are several works available in the literature about laser range finder data processing for dynamic obstacles detection and tracking [87–89]. Here, dynamic obstacle detection is performed by applying some heuristic algorithms directly on the LiDAR point clouds, clustering the obstacles based on

parameters such as the cluster radius or the distance between points. In this work, dynamic obstacles are detected and tracked making use of the occupancy information provided by the Nav2 packages, making it a modular solution such as the ROS1 *costmap_converter* package implemented in [90]. As a consequence, the DOL is less dependent on the type of sensor employed in the robot perception system.

Subsequently, a policy for translating the obstacles' velocities and orientation values into the costmap has been defined by leveraging the available Inflation Layer structure and applying a risk level set concept similar to the one presented in [91].

4.3 The DOL approach

The DOL approach designed in this work is structured in three steps:

- Object detection: given a costmap representation of the environment, dynamic obstacles are identified and differentiated from static ones by employing image processing algorithms and running average filters.
- Object tracking: identified dynamic obstacles are tracked and their velocity is estimated by applying a Kalman Filter.
- Cost assignment: a developed costmap layer assigns costs around each moving obstacle in the *local_costmap* according to a 2D Gaussian shape, where its variances are proportional to the obstacle velocity and oriented along its moving direction.

4.3.1 Object detection

Once the robot is ready to navigate, its costmap representation of the environment is treated as an image during the object detection step. In this way, the foreground indicates whatever is moving, while the background is everything that is static. *Background subtraction* is obtained by implementing two running average filters to each pixel, in particular, they correspond to a "fast" and a "slow" filter:

$$P_f(t+1) = \beta[(1-\alpha_f)P_f(t) + \alpha_f C(t)] + \frac{1-\beta}{8}\sum_{i \in NN} P_{f,i}(t)$$

$$P_s(t+1) = \beta[(1-\alpha_s)P_s(t) + \alpha_s C(t)] + \frac{1-\beta}{8}\sum_{i \in NN} P_{s,i}(t)$$

where $P_f(t)$ and $P_s(t)$ represent the output of the fast and the slow running average filters at time t, respectively. β defines the ratio between the contribution of the central cell filter and the effect of the neighbouring cells to $P_f(t)$ and $P_s(t)$, so as to capture the running average filter of the 8 Nearest Neighbour (NN) cells due to the fact that large objects form blocks of cells in the local costmap. The gains α_f and α_s denote the effect of the current costmap $C(t)$ on both filters. Hence, the two filter rates are chosen such that

$$0 \leq \alpha_s < \alpha_f \leq 1$$

Besides, two thresholding steps are performed to filter out the high and low-frequency noise and identify those pixels occupied by the dynamic obstacles:

1. The fast filter classifies a cell as foreground if it exceeds a threshold c_1:

$$P_f(t) > c_1$$

2. The difference between the fast and the slow filter has to exceed a threshold c_2 in order to remove the quasi-static obstacles with low-frequency noise:

$$P_f(t) - P_s(t) > c_2$$

The constant values c_1 and c_2 have been heuristically set, based on the range of values that cells can assume in a Nav2 costmap (from 0 to 255) and taking into account the same settings adopted in [90]. The output is a binary map where all the dynamic pixels are marked with "1", as it is reported in the general scheme of the DOL approach shown in Figure 4.2.

Afterwards, the *SimpleBlobDetector* heuristic algorithm, provided by the OpenCV library [92], clusters the dark pixels in the binary image into blobs representing each dynamic obstacle in terms of contours and a centroid. The former corresponds to a list of the cells that define the blob contours, and the latter is the coordinate of the cell (pixel) in the weighted centre of the blob.

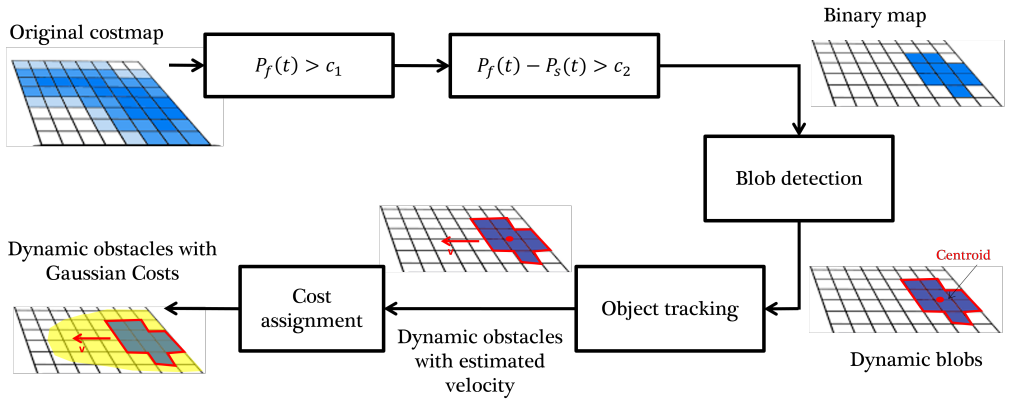


Fig. 4.2 Dynamic Obstacle Layer method steps.

4.3.2 Object tracking

The centroid of dynamic obstacles progresses with each costmap update and subsequent foreground detection. The assignment of blobs in the current map to obstacle tracks constitutes a data association problem. In order to disambiguate and track multiple objects over time, the current obstacles are matched with the corresponding tracks of previous obstacles. A new track is generated whenever a novel obstacle emerges that is not tracked yet. Tracks that are not assigned to current objects in the foreground frame are temporarily maintained. The track is removed if it is no longer confirmed by object detections over an extended period of time. The assignment problem is solved by the so-called Hungarian algorithm [93], which solves weighted assignment problems by minimizing the total Euclidean distance between the tracks and the current set of obstacle centroids.

Then, a Kalman filter estimates the current velocity of tracked obstacles assuming a first-order constant velocity model, since it is sufficient to capture the prevalent motion patterns of humans and robots in indoor environments.

4.3.3 Cost assignment

In order to add information about the dynamic obstacles in the costmap, the area enclosing each detected obstacle is inflated with a 2D Gaussian shape. In particular, the amplitude of the obstacle's velocity has been associated with the maximum value of the Gaussian, so as an outcome, faster obstacles have larger inflation sizes than

slower ones. Modifying the obstacle's Gaussian shape aims at designing a local planner more aware of the obstacle with sufficient forewarning for replanning.

Moreover, the information related to the obstacles' orientation is employed to inflate those cells along their moving direction. This is achieved by blending two 2D Gaussian shapes, one increasing the cells in front of the obstacle and the other inflating the cells on its back region.

Let us consider an obstacle O with centroid in position $c(x,y)$ in the map reference frame, a local coordinate system is defined whose origin is centred in c, with the X-axis oriented along the velocity vector direction, the Z-axis pointing outwards the costmap plane and finally, the Y-axis is set according to the right-hand rule. The relationship between the map and the local reference frames is represented in Figure 4.3a.

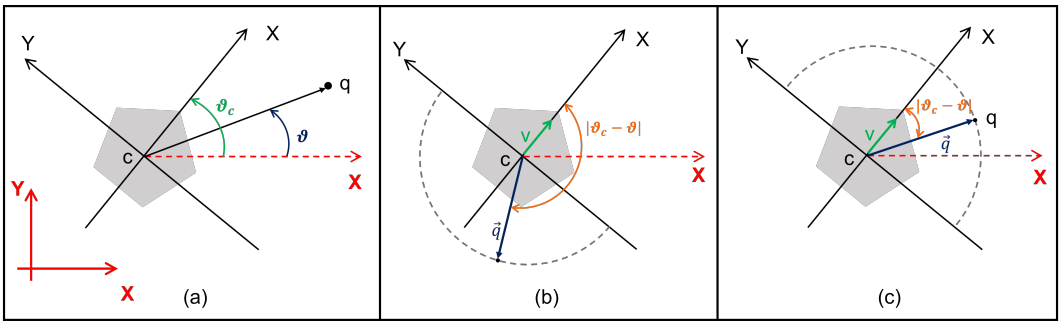


Fig. 4.3 a) Local obstacle reference frame (black) with respect to the global (map) reference frame (red). b) A point q within the back area. c) A point q in the frontal space.

Hence, the obstacle-inflated area is defined by:

$$\Phi_{c,\Sigma_f,\Sigma_b}(q) = \delta(q)\Phi_{c,\Sigma_f}(q) + [1-\delta(q)]\Phi_{c,\Sigma_b}(q)$$

where $q = (x_q, y_q)$ represents the coordinates of a point in the map reference frame, Φ_{c,Σ_f} and Φ_{c,Σ_b} are the Gaussian functions that inflate the frontal and the back area of the obstacle, respectively. While $\delta(q)$ selects the correct Gaussian function depending on whether the considered cell is in the frontal or back space of the obstacle (Figures 4.3b and 4.3c), and it is described as follows:

$$\delta(q) = \begin{cases} 1 & \text{if } \vec{v}\cdot\vec{q} \geq 0 \\ 0 & \text{if } \vec{v}\cdot\vec{q} < 0 \end{cases} \Rightarrow \delta(q) = \begin{cases} 1 & \text{if } cos|\vartheta_c - \vartheta| \geq 0 \\ 0 & \text{if } cos|\vartheta_c - \vartheta| < 0 \end{cases}$$

where $\vec{v}$ is the obstacle's velocity vector and the angles ϑ_c and ϑ are defined as in Figure 4.3a. Each Gaussian function is computed as:

$$\Phi_{c,\Sigma}(q) = A\,exp\left\{-\frac{[d\,cos(\vartheta-\vartheta_c)]^2}{2\sigma_x^2}-\frac{[d\,sin(\vartheta-\vartheta_c)]^2}{2\sigma_y^2}\right\}$$

where d is the Euclidean distance between q and $c(x,y)$, A is an amplitude parameter set to the maximum cost possible on the costmap, i.e., 255. σ_x^2 and σ_y^2 are the diagonal components of the Σ covariance matrix, which determines the shape of the inflation area. In particular, the two covariance matrices are defined as:

$$\Sigma_{front} = \begin{pmatrix} \sigma_{x\_f}^2 & 0 \\ 0 & \sigma_{y\_f}^2 \end{pmatrix}$$

$$\Sigma_{back} = \begin{pmatrix} \sigma_{x\_b}^2 & 0 \\ 0 & \sigma_{y\_b}^2 \end{pmatrix}$$

Consequently, σ_x and σ_y can be tuned to model a generic shape at will. Here, in order to take into account the obstacle velocity quantity, a maximum obstacle speed *max_speed* has been set. After that, in order to enlarge even more the front region, the *speed ratio* has been defined as:

$$r = \frac{vel}{max\_speed}$$

where *vel* is the obstacle's estimated speed. Finally, the variances are modified according to the following heuristics:

$$\begin{cases} \sigma_{x\_f}^2 = & (1+r)\sigma_{x\_f}^2 \\ \sigma_{y\_f}^2 = & (1-\frac{r}{2})\sigma_{y\_f}^2 \\ \sigma_{x\_b}^2 = & (1-r)\sigma_{x\_b}^2 \\ \sigma_{y\_b}^2 = & (1-\frac{r}{4})\sigma_{y\_b}^2 \end{cases} \tag{4.1}$$

As a result, the Gaussian shape is elongated in the obstacle's moving direction while the lateral area is more narrowed.

4.4 Implementation in ROS2 framework

The proposed DOL approach has been developed within ROS2 Foxy version on an Intel NUC8 with a Linux Ubuntu 20.04 environment. For what concerns the testing, during the development phase, a virtual environment has been created using the Webots simulator, while Rviz has been used for visualising the behaviour and debugging purposes. The TurtleBot3 has been chosen for the simulation, for which both Webots and Nav2 already provide a physical model and interface packages. Nonetheless, the DOL still remains independent of the mobile robot choice. On the other hand, regarding the Nav2, different configurations are possible thanks to its modular architecture and depend on the specific plugins that are used. Here, the default Nav2 configuration has been adopted, according to the dedicated TurtleBot3 navigation packages and including the *nav2_dwb_controller* (DWB) as the plugin for the Controller Server.

As can be seen in Figure 4.4, the *ros2_costmap_to_dynamic_ obstacles* package provides the nodes implementing the functionalities for obstacle detection and publishing the blobs corresponding to detected obstacles through a specific custom ROS2 message type on the */detection* topic.

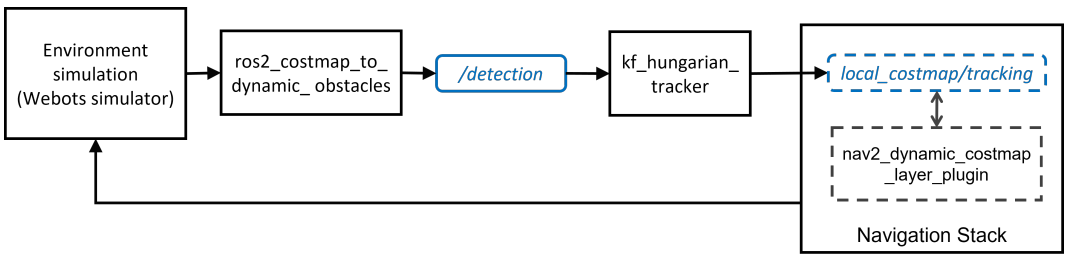


Fig. 4.4 Interaction scheme between internal SW functional blocks and the simulated external environment.

The *kf_hungarian_tracker* package provides a subscription to the */detection* topic in order to perform object tracking. Afterwards, it publishes the dynamic obstacles and their estimated velocities on the *local_costmap/tracking* topic, which is created when Nav2 is launched. Finally, the costmap layer is implemented through *nav2_dynamic_costmap_layer_plugin*. It processes the information related to the dynamic obstacles, so as to determine the Gaussian costs and updates the master costmap used for the robot navigation.

4.5 Simulation tests and analysis

In this section, the local path planning approach available in ROS2 (DWB Controller) and the DWB merged with the proposed Dynamic Obstacle Layer method (DWB + DOL) are compared through simulations for a DOL preliminary performance evaluation.

The simulations have been executed on Webots employing a TurtleBot3 burger robot, a two-wheeled robot equipped with an RPLIDAR A3 LiDAR sensor. In particular, the sensor provides a maximum distance range of $25\,m$, an angular resolution of 0.225° and a scan rate of $15\,Hz$.

4.5.1 Experimental setup

Webots is an open-source platform that allows also to create realistic 3D virtual worlds including the physical properties of each object. Moreover, it is possible to define the dynamic behaviour of robotic objects (*Robot Nodes*) through a Webots Controller, whose piece of code can be written in many different programming languages, e.g., C, C++, Java, Python or MATLAB. The world created for testing the algorithm is depicted in Figure 4.5. The scenario is an empty rectangular arena $10\,m \times 6\,m$ where dynamic obstacles are simulated as wooden boxes of $20\,cm \times 20\,cm$ base and $50\,cm$ tall, configured as *Robot Nodes*. Starting from a defined initial location, each box moves with a constant speed back and forth traversing the entire arena along the short side direction. A Webots Controller has been coded for each obstacle and the speed can be commanded when the world is launched so that simulation scenarios can be easily changed.

It is worth observing that, before launching the navigation tests, the *turtlebot3_cartographer* package has been exploited to perform Simultaneous Localization and Mapping (SLAM), so as to generate and upload the static occupancy map of the virtual environment without boxes.

In order to compare the performances of the baseline DWB algorithm with the DWB + DOL configuration, the TurtleBot3 is commanded to traverse the *dyn_env_1.wbt* world from one side of the rectangular arena to the opposite side, covering a total distance of $8\,m$. The robot navigation scheme can be seen in Figure 4.6 and it is repeated N times. The global path for navigating the robot to the destination

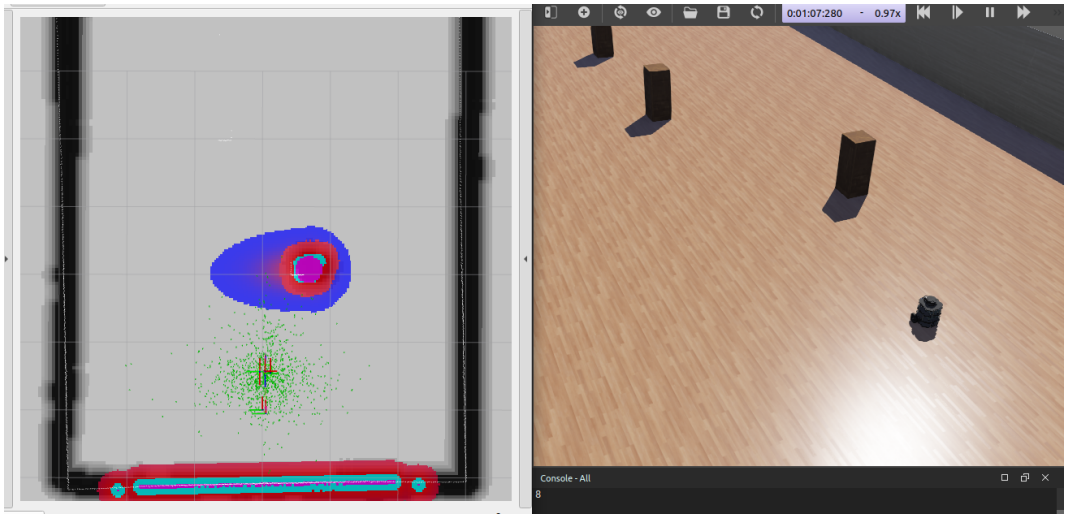


Fig. 4.5 *dyn_env_1.wbt* virtual world on the right and Rviz output on the left with DOL plugin.

pose is computed by the Planner plugin *nav2_navfn_planner/NavPlanner*. A short demo video displaying the experimental setup and behaviour using DWB alone and DWB + DOL can be seen at [94].

The parameters considered for evaluating the algorithm are: the travel time, the number of *wait* recovery behaviours triggered during travel and if any collision occurred. The results have been collected from experiments carried out in two different conditions where the boxes move at different speeds, in particular:

1. Test set 1: Obstacles moving at a constant speed set to $0.6 m/s$
2. Test set 2: Obstacles moving at a constant speed set to $0.8 m/s$

The performance indices considered for each test set are described as follows:

- Smooth navigations [%]: It indicates how many times the TurtleBot3 smoothly navigated to the destination position. It considers also the cases in which the robot stopped briefly to avoid collisions with a moving box in its proximity.
- Wait recoveries [%]: It is the number of times the *wait* recovery behaviour has been triggered, still successfully reaching the goal. A *wait* recovery is usually activated when obstacles come suddenly too close to the robot and the Controller Server cannot generate a valid and feasible replanned path by a given timeout interval.

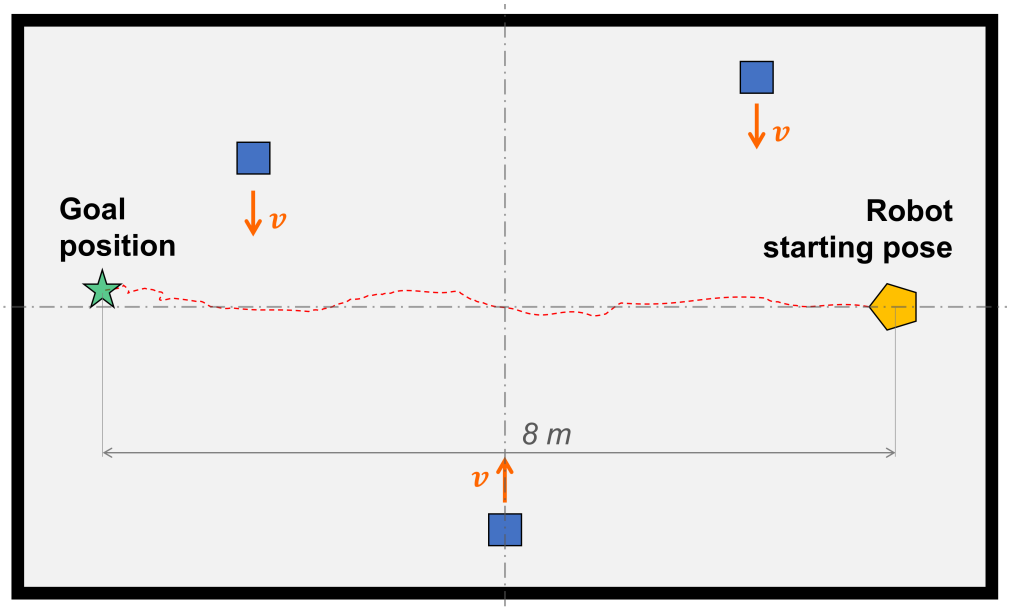


Fig. 4.6 *dyn_env_1.wbt* test scheme.

- Collisions [%]: It corresponds to the percentage of unsuccessful navigation due to a collision of the TurtleBot3 with a moving box.
- Successful navigations [%]: It is the total number of times in which the robot successfully reached the goal position, either performing smooth navigation or after triggering the recovery behaviour.

4.5.2 Results and discussion

For what concerns Test set 1, where the obstacles move at $0.6 m/s$, the simulated scenario has been executed $N_1 = 50$ times for both DWB and DWB + DOL configurations. In particular, the global navigation results are summarized in Table 4.1.

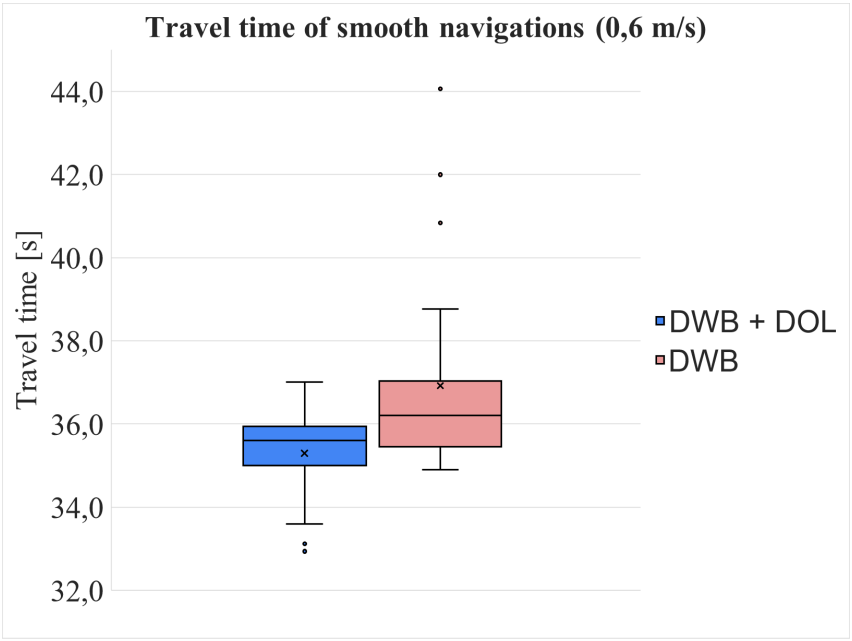
At first glance, it can be seen that the DWB + DOL approach presented a higher successful navigation rate than the simple DWB: 96% against 82%, respectively. Even though the "smooth navigations" percentage is quite similar with the two approaches, it is worth mentioning that the DWB + DOL solution reported fewer collisions due to the fact that the *wait* recovery was triggered more frequently. As a result, the dynamic obstacle layer provides safer navigation when it is combined with the actual DWB local planner. In fact, the Gaussian costs forewarn the TurtleBot

Table 4.1 Test set 1: Navigation results at $0.6 m/s$ obstacles speed.

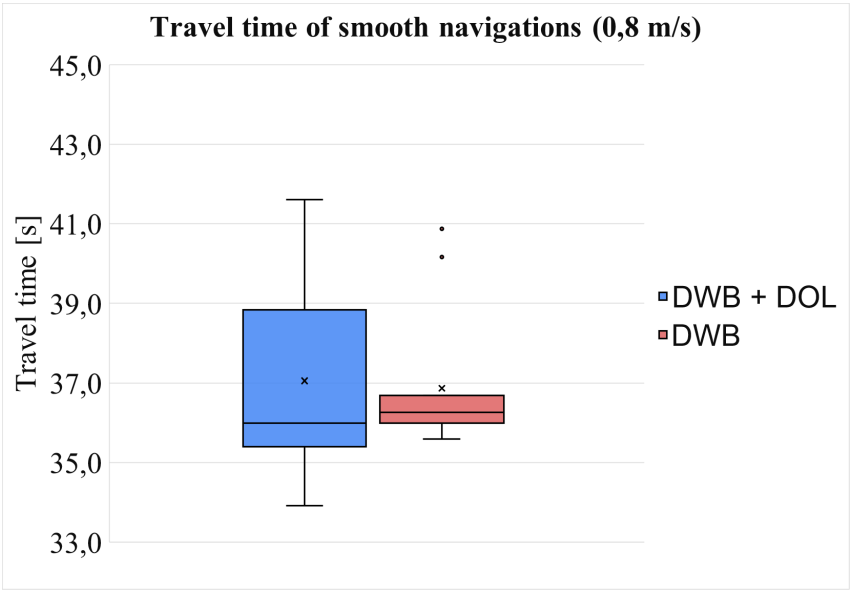
	DWB + DOL	**DWB**
Smooth navigations	86,0%	82,0%
Wait recoveries	10,0%	0,0%
Collisions	4,0%	18,0%
Total successful navigations	**96,0%**	**82,0%**

about an approaching obstacle and the Recovery Server is triggered on time if moving on would result in a collision. On the other hand, setting the obstacles speed at $0.6 m/s$ and maintaining the same Nav2 parameters configuration, DWB was not able to react on time when an obstacle suddenly approaches.

It can be observed that the number of "smooth navigations" for DWB + DOL and DWB over the total number of tests is similar, in particular, 86% and 82%, respectively. However, the travel time performances are different. Figure 4.7a depicts the box plots representing the travel time data of all the "smooth navigations" achieved by the robot with the two planning algorithms. First of all, it can be noticed that the DWB distribution is more asymmetric and for sure non-Gaussian. Secondly, the mean travel time reported in DWB + DOL is lower than that of DWB, even if only of $1 s$. Nevertheless, the most important result is that the interquartile range (IQR) for DWB + DOL is smaller than DWB box. Therefore, it seems that the proposed approach ensures to estimate a more confident travel time for a given environment and setting. Finally, it should be pointed out that outliers lay all below the box plot for the DWB + DOL reporting shorter travel times, while the DWB plot presented longer travel times. This suggests that increasing the number of tests might produce fewer overlapped box plots. Regarding Test set 2, the obstacle speeds have been set to $0.8 m/s$. Increasing the obstacles' speed value has the aim to push the available DWB Controller to its limits. In fact, in this case, only $N_2 = 30$ simulations are sufficient to clearly prove the poor performance of the DWB algorithm compared to the DWB + DOL approach. Actually, the main distinction in the setup of both the whole environment and the Nav2 parameters with respect to the previous test set is the obstacle speed. The same performance indices have been taken into consideration and the results are collected in Table 4.2.



(a) Test set 1



(b) Test set 2

Fig. 4.7 Box plot of the travel times during 'smooth navigations' for both test sets.

Table 4.2 Test set 2: Navigation results at $0.8\,m/s$ obstacles speed.

	DWB + DOL	**DWB**
Smooth navigations	50,0%	43,3%
Wait recoveries	36,7%	0,0%
Collisions	13,3%	56,7%
Total successful navigations	**86,7%**	**43,3%**

It is worth observing that the amount of collisions during the simulations launched with only DWB have significantly incremented, in particular, from 18,0% at $0.6\,m/s$ to 56,7%. A similar behaviour occurred for the DWB + DOL, in which the collisions increased from 4,0% at $0.6\,m/s$ to 13,3%. Nonetheless, for the DWB + DOL case, the percentage of triggered recoveries behaviour has increased as well, achieving 86,7% of "successful navigations", while the percentage of success in case of DWB has been decreased about a half with respect to Test set 1. Despite this, for the second test set, travel time data have been collected over $N_2 = 30$ data points (Figure 4.7b). Given that the "smooth navigations" indices are equal to 50% (DWB + DOL) and 43,3% (DWB), it is worth pointing out that no robust consideration can be made.

As shown in Figure 4.7, it can be noted that the median values of both test sets are very similar. At first glimpse, the most remarkable difference is that in Test set 2, the DWB + DOL box plot has a greater variance due to the small amount of data. This is also because the faster are the obstacles, the more corrective actions are executed by the robot, making the travel time more unpredictable.

For what concerns the collected data for DWB case, the variance has been considerably reduced with respect to the previous test set. Nevertheless, the data sample is too small, since the robot performed successfully a smooth navigation 13 times out of 30. Note that "smooth navigations" values are reported only when the robot starts the navigation at a random instant and follows the planned path while not being intercepted by any of the moving boxes.

As it is shown by the results obtained in simulation, the DOL approach definitely reports some performance improvements in terms of collisions rate, but still there are collisions, even at the lower obstacle speed of $0.6\,m/s$. A motivation behind this issue could be the not optimal communication between the Webots virtual environment and the Navigation Stack, that may cause lags in the obstacle detection and costs

computation. Nevertheless, this is strictly related to the computational performances of the whole ROS2 framework and Linux environment installed on the Intel NUC platform.

However, some parameters of Nav2 could be fine-tuned to diminish the collision rate with the adopted HW/SW setup. For instance, in reference to the *Configuration Guide* section of DWB Controller in the Nav2 documentation page, they are:

- *controller_frequency* (default $20\,Hz$): it is the controller server update rate. Higher values may lead to a faster reaction to obstacles since the local trajectory is replanned more frequently by the DWB Controller plugin, ensuring safer collision avoidance.
- *update_frequency* (default $5\,Hz$): updating more frequently the `local_costmap` allows the robot to read more recent Gaussian cost values. As a result, the obstacle avoidance mechanism is enhanced due to the fact that the local plan is calculated using more reliable data.
- *<dwb plugin>.sim_time* (default $1.7\,s$): it corresponds to the time in which the DWB plugin simulates looking ahead to generate feasible local trajectories before scoring them and choosing the best one. Slightly increasing this parameter, the DWB Controller should better discriminate colliding trajectories from non-colliding ones. However, since the DWB is a local planner, higher values may interfere with the correct performance of the Planner Server.
- *<dwb plugin>.BaseObstacle.scale* (default 0.02): it is the scale used by the DWB plugin to score a trajectory and it depends on the location of the path in the costmap. As the value raises, it is more likely that the robot will prevent passing through inflated cells. In this way, the navigation should be smoother but would take a longer time to reach the destination.

Editing the above-listed parameters, especially the first three, may be an effective way to improve the overall performance. Nonetheless, higher computing resources may be required, so the parameter values should be finely tuned depending on the application requirements and the actual target hardware.

Concerning the Gaussian cost assignment, the variances (Gaussian shape parameters), have been set empirically and based on a hypothetical obstacle maximum speed

as in (4.1). As an alternative, an additional function could be introduced to calculate the costs by combining the obstacle speeds with the robot speed. For instance, if the robot's maximum speed is too small with respect to an obstacle, it should not try to overcome it, so Gaussian costs should be modulated accordingly. On the other hand, the obstacle speed data could be directly included in the local planner, but the trade-off of achieving a much more complex approach is no more as modular as the proposed DOL.

Hence, the proposed Dynamic Obstacle Layer approach seems to provide safer navigation in presence of dynamic obstacles. At the same time, as shown in the majority of the test cases, the DOL allows planning a smoother trajectory than the DWB Controller could achieve alone, resulting in reduced travel times starting from the same conditions. Indeed, despite the TurtleBot having a maximum speed lower than the set obstacle speeds, it manages to adjust the trajectory when an obstacle is reported by the DOL, dodging it or passing behind it, even if there is still much room for enhancements in the travel time performance.

4.6 Discussion

The proposed Dynamic Obstacle Layer approach implements a strategy for managing dynamic obstacles that can be easily included with the current ROS2 Navigation Stack, thus being a flexible and modular solution for the problem of navigation in dynamic environments. The simulation tests carried out in a virtual environment have shown a relevant performance improvement in terms of collision rate and travel times, merging the DOL with the available DWB Controller. The two carried-out simulation tests pointed out how the combination of DWB and DOL can enhance navigation safety as the dynamic obstacles' speed increases.

Nonetheless, the DOL code takes into account many parameters, such as filter parameters, blob detection parameters or Gaussian costs scaling factors, some of which have been set based on experimentally and manual tuning in this initial algorithm development.

Hence, this chapter presents a starting point for various possible future works in different directions. Firstly, the DOL has been tested only in a virtual environment. Even though the Webots simulator has been set to provide as realistic simulations

as possible, only tests in the real world can validate the proposed planning method, for example, using a physical TurtleBot with real sensors and checking the actual performance of the robot under different scenarios. Beyond that, object detection is based on the thresholds set for the running average filter: using the values based on reference examples, however, does not allow to achieve the desired filtering accuracy for low obstacle velocities. Thus, a set of tests should be performed to finely tune these parameters in order to be compliant with the requirements of hypothetical future application scenarios.

Finally, the inclusion of video/image frames from a camera can be investigated to provide more detailed information about dynamic obstacles, for instance, differentiating a human walking in the environment from another robot, so that the robot could react accordingly with different planning strategies. Moreover, semantic information can be easily integrated with a multi-layer costmap on the base of the current DOL, and Planner and Controller plugins can be easily foreseen to implement different behaviours.

Chapter 5

Path planning in formation and collision avoidance for multi-agent systems

This chapter presents an algorithm for path planning in formation and collision avoidance for multi-agent systems. In particular, this research work has been developed in collaboration with researchers from Università Tor Vergata (Rome, Italy). It is worth mentioning that this chapter depicts the algorithm presented in [8] and is an extension of the one presented in Chapter 2, corresponding to the single robot case. However, for the sake of simplicity, the algorithm does not consider other planning hierarchies, as it was tested in simulation and validated in a testbed known as Robotarium, composed of multiple mobile robots.

The chapter is structured as follows: Section 5.1 provides an overview of related works, the notation used in this chapter as well as the problem statement. The Lyapunov-based hybrid strategy that allows autonomously tuning the parameters to guarantee convergence of the system is outlined in Section 5.2. The design of a vector field to achieve patrolling and formation control of the multi-agent system with obstacle avoidance is described in Section 5.3. Numerical simulations demonstrating the effectiveness of the proposed approach are provided in Section 5.4. The applicability of the strategy is then demonstrated by experimental tests, whose results are showcased in Section 5.5. Finally, the discussion of the obtained results is presented in Section 5.6 .

Moreover, in order to provide a clearer relationship between the sections, Figure 5.1 depicts a conceptual scheme summarizing the relations among the topics of each section.

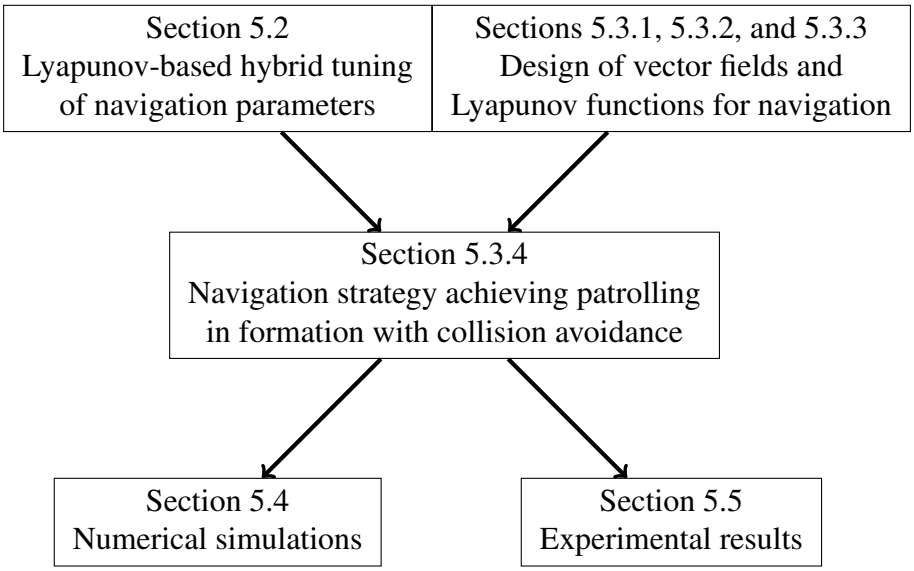


Fig. 5.1 Conceptual scheme of the relations among the topics of each section.

5.1 Background

The use of a group of mobile robots, cooperatively acting toward a common objective, has been significantly growing in the most recent years in several, different applications, from monitoring and target searching in automation and logistics scenarios, to surveillance and rescue missions. The actual advantages deriving from the employment of a robotic team instead of a single mobile agent rely on the possibility of achieving a robust, efficient collaboration among the members of the team, and on their ability to move in a proper way to carry out the assigned task, avoiding any collision with other elements in the environment and between themselves. The fundamental problems to be addressed from the research point of view are then related to path planning and formation control, to be solved together with collision avoidance.

Multi-robot path planning techniques are aimed at finding the optimal path for the members of the robotic team according to some criteria, for example, as in [95], where the problem is addressed on graphs over four minimization objectives: the

makespan (i.e., the last arrival time), the maximum distance (travelled by the single robot), the total arrival time, and the total distance. Here the computational efficiency of the developed optimization algorithm based on Integer Linear Programming is enhanced through some heuristics to allow handling of hundreds of robots at the expense of slight optimality loss. In [96], a coevolution-based particle swarm optimization method is instead proposed to cope with the multi-robot path planning issue, considering the total path length of the multi-robot system as the global objective function. The SplitAndGroup (SAG) algorithm developed in [97] provides constant factor makespan optimal solutions on average over all problem instances on grids and grid-like environments.

Such path planning algorithms, however, do not allow to impose a desired arrangement to the robotic agents while they are moving, as required in some applications, such as patrolling and surveillance. This requisite is addressed by the formation control approaches, which are often based on the definition of suitable navigation functions, originally introduced in [98] using artificial potential functions for a single robot, and subsequently extended to the multi-robot case (e.g., in [99], [100], [101]). The approaches of this kind exploit the knowledge of the environment topology and of the obstacles present in it to build control policies that guarantee the convergence of the team of robotic agents to the assigned formation scheme, while avoiding collisions [101]. Centralization and decentralization of the control strategy is a key issue: a centralized architecture, including a single control law, can be more complex from the computational point of view, but it can generally guarantee the completeness of the solution. Various decentralized solutions and approaches allowing some kind of scalability have been proposed in the literature, e.g., in [100], where the potential function is constructed in a way that facilitates the complete decentralization of the scheme, and in [99], where a decentralized cooperative controller is designed as the gradient of a proper navigation function, whose minimum corresponds to the desired configuration, guaranteeing the obstacle avoidance.

Obviously, computational burden and complexity are more significant aspects in applications involving a large or very large number of robots, for which specific solutions have been proposed in the literature, e.g., in [102] and [103]. In [102], the computational burden is reduced through a multi-agent flocking approach in which not all the agents are informed, but only the virtual leader and some of the agents that move with the desired constant velocity; the uninformed agents are able to move

with the same desired velocity thanks to periodical interactions with the informed ones. In [103], the problem is addressed by introducing an abstraction based on the definition of a map from the robots configuration space to a manifold, whose dimension is lower and independent of the number of robots involved; the approach allows to control the robots formation and the trajectory independently.

The control design is decoupled also in the hierarchical approach proposed in [104] to address the problem of making a group of unicycles converge to a common circle of assigned radius, and travel around it in a desired direction; the information exchange between the mobile agents is modelled by a directed graph, and the control scheme is designed decoupling the problem of making the unicycles converge to the common circle from the problem of stabilizing the formation. Modelling through formation graphs (i.e., graphs whose nodes capture the individual agent kinematics, and whose edges represent interagent constraints that must be satisfied) is specifically addressed in [105], focusing on the feasibility problem, i.e., the existence of nontrivial agent trajectories that satisfy the interagent constraints, given the kinematics of the agents.

Several formation control approaches are based on leader-follower strategies, mainly with the aim of achieving a simpler and easily scalable architecture, with reduced communication requirements; the downside of this kind of solutions is a potential weakness in practice, if the substitution of the predefined leader is not envisaged or it is even impossible in case of some fault. Among the first theoretical contributions relative to the leader-follower policy, it is worth recalling both the distributed control approach developed in [106], based on artificial potentials and virtual leaders, and the analysis of the stability properties of mobile agent formations based on leader following carried out in [107]. More recent control approaches can be found in [108], [109], [110], addressing specific issues. In particular, the leader-follower approach developed in [108] is focused on formation and tracking control along straight paths; here, the uniform global asymptotic stability of the closed-loop system is guaranteed by partially linear, time-varying controllers with the addition of a nonlinear term, within a strategy in which each robot is a leader to one robot and follower to another, with a unique swarm leader robot that receives the information of the reference trajectory (and a unique tail robot that is leader to none).

The decentralized adaptive formation controller proposed in [109] ensures uniformly ultimate boundedness of the closed-loop system with prescribed transient

and steady-state performances, taking into account the presence of external disturbances and uncertainties in the vehicle dynamics; the control objective is to make each vehicle follow its reference trajectory and to avoid collisions between each vehicle and its leader. In [110], prescribed transient and steady-state performances are achieved as well, under communication constraints among the agents, by a decentralized formation control in which only the leader has the trajectory information; a tan-type barrier Lyapunov function and a recursive adaptive backstepping procedure are adopted to guarantee the asymptotic stability of the closed-loop system.

Further recent contributions can be found in [111] and [112], developed under the consensus protocol; in the first approach, distributed kinematic controllers and neural network torque controllers are derived for each robot to let a group of mobile agents asymptotically converge to a desired geometric pattern along the given reference trajectory, while in the second one the bipartite consensus protocol is adopted under possible communication delays. A quite complete survey of multi-agent formation control approaches can be finally found in [113], where the types of sensed variables used in the different solutions are investigated, classifying the existing schemes into position-displacement and distance-based control approaches.

The centralized approach proposed in this chapter solves the patrolling and formation control problems for a group of *unicycle-like* mobile robots, guaranteeing that the trajectories of the multi-robot system are collision-free. Although, differently from the design strategies proposed in the above mentioned references [100]–[113], the approach proposed in this chapter is centralized, it possesses several desirable features that are difficult to achieve in a decentralized setting.

First, the motion planning architecture given in this chapter does not rely on a leader-follower hierarchy and, hence, it is intrinsically robust with respect to faults of one of the robots. Furthermore, the hybrid design proposed to tune online the navigation parameters can be used to optimize the trajectories followed by the team of mobile robots. Such an optimization cannot be easily carried out using a decentralized approach. Finally, this centralized design strategy allows also to guarantee robustness of the navigation scheme with respect to measurement and implementation errors, as formalized in Section 5.2.

The originality of the contribution is given by the use of a hybrid controller to ensure the convergence to a prescribed set, and the combination of algebraic techniques and methods inspired by the classical navigation functions to achieve

the convergence to the desired path in formation, avoiding collisions among the agents and with static obstacles in the environment. More in detail, the guarantee of convergence provided by the hybrid system approach ensures patrolling in formation, while the adopted barrier function prevents collisions during the transient time. Preliminary results were developed in [114] for the single-robot case and in [34] for the multi-robot case. The single-robot case was also experimentally validated in a laboratory industrial-like environment, considering an offline path planning as first implementation in [4], and then including a procedure for the online upgrade of the computed path in [6].

The main contribution of this work compared to the aforementioned preliminary results is that, here, the navigation strategies envisioned in [114, 34] are coupled with a hybrid framework that autonomously selects and updates navigation parameters so to guarantee the achievement of the desired formation and patrolling task while avoiding collisions with fixed obstacles in the environment and among agents. Furthermore, in this chapter, we report all the previously omitted proofs and we carry out a robustness analysis of the proposed navigation strategy, which ensures patrolling in prescribed formation even in the case of inaccurate measurements of the agents' positions and imperfect implementation of the nominal strategy. It is worth to be noted that, thanks to the centralization of the proposed approach, there is no predefined leader among the robots: the desired patrolling paths can be assigned to some or all the agents of the team, thus enhancing the robustness of the scheme.

The validity of the proposed approach is proved for groups of three or four mobile agents, both in simulation and in experimental tests, carried out using a remotely accessible robotic testbed (namely, `Robotarium` [115]). The results achieved in all the tests confirm the effectiveness of the proposed solution, which is expected to be applicable up to ten robots, or even more for simple agents like the ones of the remote testbed. Clearly, the scalability of the approach is related to the computational and communication potentialities of the available hardware/software architecture.

5.1.1 Notation

In this section, we introduce the notation used all throughout the chapter and we formalize the considered problem. Let $\mathbb{N}$, $\mathbb{R}$, $\mathbb{R}_{\geqslant 0}$, and $\mathbb{R}_{>0}$ denote the sets of natural, real, nonnegative real, and positive real numbers, respectively. The symbol I_n denotes

the n-dimensional identity matrix, whereas the symbol $\mathrm{diag}(a_1,\ldots,a_n)$ denotes the diagonal matrix whose diagonal entries starting in the upper left corner are $a_1,\ldots,a_n$. Given a function $V:\mathbb{R}^n\to\mathbb{R}$, the symbol ∇V denotes its gradient. Given two vectors $v,w\in\mathbb{R}^n$, the symbol $\langle v,w\rangle$ denotes their inner product. Given a set $\mathscr{V}\subset\mathbb{R}^n$ and a vector $x\in\mathbb{R}^n$, let $\|x\|_{\mathscr{V}}:=\inf_{y\in\mathscr{V}}\|x-y\|$. The symbol $\mathscr{B}$ denotes the closed unit ball in the Euclidean norm of appropriate dimensions. Let $\mathrm{col}(x,y):=\begin{bmatrix} x^\top & y^\top \end{bmatrix}^\top$. The symbols $f:\mathbb{R}^n\to\mathbb{R}^n$ and $F:\mathbb{R}^n\rightrightarrows\mathbb{R}^n$ denote functions and set-valued mappings from $\mathbb{R}^n$ to $\mathbb{R}^n$, respectively. A function $\alpha:\mathbb{R}_{\geqslant 0}\to\mathbb{R}_{\geqslant 0}$ is of class $\mathscr{K}_\infty$, denoted as $\alpha\in\mathscr{K}_\infty$, if it is continuous, zero at zero, strictly increasing, and unbounded. A function $\psi:\mathbb{R}_{\geqslant 0}\times\mathbb{R}_{\geqslant 0}\to\mathbb{R}_{\geqslant 0}$ is of class $\mathscr{KL}$, denoted as $\psi\in\mathscr{KL}$, if it is nondecreasing in its first argument with $\lim_{r\to 0}\psi(r,t)=0$ for each $t\in\mathbb{R}_{\geqslant 0}$ and it is nonincreasing in its second argument with $\lim_{t\to+\infty}\psi(r,t)=0$ for all $r\in\mathbb{R}_{\geqslant 0}$. Given a set $\mathscr{V}\subset\mathbb{R}^n$, a function $\rho:\mathbb{R}^n\to\mathbb{R}_{\geqslant 0}$ is positive definite with respect to $\mathscr{V}$, denoted as $\rho\in\mathscr{PD}(\mathscr{V})$, if $\rho(x)=0$ for all $x\in\mathscr{V}$ and $\rho(x)>0$ for all $x\in\mathbb{R}^n\setminus\mathscr{V}$. The ring of all the polynomials in x with real coefficients is denoted as $\mathbb{R}[x]$. Given $p_1,\ldots,p_\varkappa\in\mathbb{R}[x]$ the variety of $p_1,\ldots,p_\varkappa$ is $\boldsymbol{V}(p_1,\ldots,p_\varkappa):=\{x\in\mathbb{R}^n:p_i(x)=0,\,i=1,\ldots,\varkappa\}$. When dealing with hybrid systems, we use the same notation and nomenclature of [116], whereas when dealing with algebraic geometry concepts, we use the notation of [117, 118]. As an example, given $q\in\mathbb{R}^{n_1\times n_2}[x]$, the symbol $\mathrm{Syz}(q)$ denotes a presentation matrix for the set of all the polynomial vectors $p\in\mathbb{R}^{n_2}[x]$ such that $qp=0$.

5.1.2 Problem formulation

The interest of the proposed approach lies in designing paths of motion for *unicycle-like* mobile robots moving on the Euclidean plane (see Fig. 5.2), described by equations of the form [119]

$$\dot{X} = \cos(\Theta)v, \tag{5.1a}$$

$$\dot{Y} = \sin(\Theta)v, \tag{5.1b}$$

$$\dot{\Theta} = \omega, \tag{5.1c}$$

where $(X(t), Y(t)) \in \mathbb{R}^2$ denotes the Cartesian position of the center of mass, $\Theta(t) \in \mathbb{R}$ denotes the orientation with respect to the horizontal axis, $v(t) \in \mathbb{R}$ and $\omega(t) \in \mathbb{R}$ represent the linear and the angular velocity inputs, respectively.

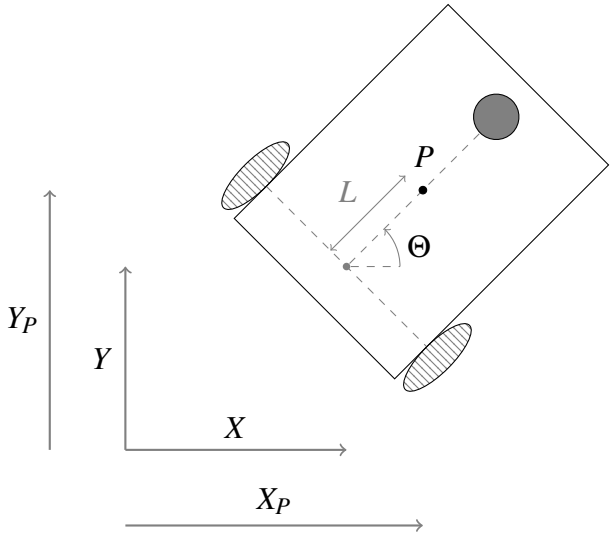


Fig. 5.2 Schematic representation of a unicycle-like mobile robot. The hatched areas represent the actuated wheels of the mobile robot, the filled gray area represents its passive wheel, the point P is the center of mass of the robot, and the length L is the distance of such a point to the line connecting the two wheels. In this two-wheels configuration, the input v represent the mean velocity of the two wheels whereas the input ω represents their differential velocity.

By hinging upon well-known techniques for instance illustrated in [120, 121] and recalled in [114], the relative dynamics of any point on the robot that does not belong to the segment connecting the two wheels (referred to as P in Fig. 5.2) can be feedback linearized and consequently arbitrarily assigned. Namely, the dynamics of the Cartesian position $(X_P, Y_P) \in \mathbb{R}^2$ of the point P are given by

$$\begin{aligned} \dot{X}_P &= v\cos(\Theta) - L\omega\sin(\Theta), \\ \dot{Y}_P &= v\sin(\Theta) + L\omega\cos(\Theta). \end{aligned}$$

In the following, we select P as the center of mass of the mobile robot.

Thus, assuming that $L \neq 0$, i.e., that the center of mass of the mobile robot does not belong to the line connecting the two wheels, and letting

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\Theta) & \sin(\Theta) \\ -\dfrac{\sin(\Theta)}{L} & \dfrac{\cos(\Theta)}{L} \end{bmatrix} \begin{bmatrix} u_X \\ u_Y \end{bmatrix}, \tag{5.2}$$

where $u = \begin{bmatrix} u_X & u_Y \end{bmatrix}^\top$ is an auxiliary input, the dynamics of the point P are described by *virtual* single integrators of the form

$$\dot{X}_P = u_X, \tag{5.3a}$$

$$\dot{Y}_P = u_Y. \tag{5.3b}$$

Therefore, letting $x_i = \begin{bmatrix} X_{P,i} & Y_{P,i} \end{bmatrix}^\top$, $i = 1,\ldots,N$, denote the position of the center of mass of the i–th mobile robot and letting its inputs v_i and ω_i be

$$\begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = \begin{bmatrix} \cos(\Theta_i) & \sin(\Theta_i) \\ -\dfrac{\sin(\Theta_i)}{L_i} & \dfrac{\cos(\Theta_i)}{L_i} \end{bmatrix} u_i, \tag{5.4}$$

the main objective of this research work is to design the input u_i so to guarantee that the corresponding solution constitutes a collision-free path for the i–th agent, with $i = 1,\ldots,N$, and the set of all solutions steers the mobile robots to patrol a preassigned path in controlled formation.

This problem is addressed by combining:

- A hybrid controller that autonomously tunes the weights of a parametric continuous-time system to ensure convergence to a prescribed set (see Section 5.2).
- An algebraic technique that allows us to design parametric vector fields such that the solution to the associated dynamical system achieves patrolling, formation control and obstacle avoidance (see Section 5.3).

5.2 Hybrid stabilization of parametric continuous-time systems

The main goal of this section is to propose a novel hybrid strategy that autonomously tunes the parameters of a parametric vector field to ensure that a given set is asymptotically stable for the corresponding dynamical system. Such a strategy is coupled in the subsequent Section 5.3.4 with an algebraic geometry technique, which allows to jointly design parametric vector fields having a given variety as attractive and invariant set and the corresponding Lyapunov function to solve the motion planning in formation with collision avoidance problem.

Consider the parametric nonlinear system

$$\dot{x} = f(x,k), \tag{5.5}$$

where $x(t) \in \mathbb{R}^n$ denotes the state, $k \in \mathscr{A}$ is a vector of parameters, $\mathscr{A} \subset \mathbb{R}^m$ is the compact set of admissible values for the parameters, and the mapping $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is assumed to be C^z for some sufficiently large $z \in \mathbb{N}$.

Definition 1. A compact set $\mathscr{V} \subset \mathbb{R}^n$ is *locally asymptotically stabilizable for* (5.5) if there exist $k^\circ \in \mathscr{A}$, an open set $\mathscr{W} \subset \mathbb{R}^n$ containing $\mathscr{V}$, a function $V \in \mathrm{C}^1$, functions $\alpha_1, \alpha_2 \in \mathscr{K}_\infty$, and a function $\rho \in \mathscr{PD}(\mathscr{V})$ such that, for all $x \in \mathscr{W}$,

$$\alpha_1(\|x\|_{\mathscr{V}}) \leqslant V(x) \leqslant \alpha_2(\|x\|_{\mathscr{V}}), \tag{5.6a}$$

$$\langle \nabla V(x), f(x,k^\circ) \rangle \leqslant -\rho(x). \tag{5.6b}$$

By classical Lyapunov arguments [122], if (5.6) holds, then the set $\mathscr{V}$ is locally asymptotically stable for the system $\dot{x} = f(x,k^\circ)$. Therefore, if the auxiliary inputs $u_1, \ldots, u_N$ appearing in (5.4) are designed as $\begin{bmatrix} u_1 & \cdots & u_N \end{bmatrix} = f(x,k^\circ)$, where $x = \begin{bmatrix} X_{P,1} & Y_{P,1} & \cdots & X_{P,N} & Y_{P,N} \end{bmatrix}^\top$, and the set $\mathscr{V}$ describes the target patrolling curve in formation, then the inputs $v_1, \omega_1, \ldots, v_N, \omega_N$ constitute a local *centralized* solution to the patrolling in formation problem for the considered team of mobile robots.

In this section, following constructions similar to those given in [123], we discuss a control design technique that guarantees global asymptotic stability of the set $\mathscr{V}$ for a hybrid implementation of system (5.5), under the following Assumptions 1 and 2.

Assumption 1. Set $\mathscr{V}$ is locally asymptotically stabilizable for (5.5) and functions V and ρ such that (5.6) holds are given.

A technique to design functions V and ρ such that (5.6) holds, hence ensuring that the set $\mathscr{V}$ is locally asymptotically stabilizable for (5.5), is proposed in the subsequent Section 5.3.

In order to streamline the exposition of our results, let

$$\ell(x,k) := \langle \nabla V(x), f(x,k) \rangle, \tag{5.7}$$

and, given a parameter $\mu \in (0,1)$, define the sets

$$\mathscr{C} := \{(x,k) \in \mathbb{R}^n \times \mathscr{A} : \ell(x,k) \leqslant -\mu\rho(x)\}, \tag{5.8a}$$
$$\mathscr{D} := \{(x,k) \in \mathbb{R}^n \times \mathscr{A} : \ell(x,k) \geqslant -\mu\rho(x)\}. \tag{5.8b}$$

Then, the proposed condition for stabilization of $\mathscr{V}$ via hybrid implementation of system (5.5) can be stated as follows.

Assumption 2. For all $x \in \mathscr{U} \subset \mathbb{R}^n$ there is $\kappa \in \mathscr{A}$ such that

$$\ell(x,\kappa) \leqslant -\rho(x). \tag{5.9}$$

Note that under Assumption 1, it results that $\mathscr{W} \subset \mathscr{U}$. Hence, consider the *hybrid implementation of system* (5.5) given by the following hybrid system

$$(x,k) \in \mathscr{C}, \quad \begin{cases} \dot{x} = f(x,k), \\ \dot{k} = 0, \end{cases} \tag{5.10a}$$
$$(x,k) \in \mathscr{D}, \quad \begin{cases} x^+ = x, \\ k^+ \in \operatorname{argmin}_{\kappa \in \Xi(x,k)} \Pi(x,k,\kappa), \end{cases} \tag{5.10b}$$

where $\Xi(x,k)$ is defined for each $(x,k) \in \mathbb{R}^n \times \mathscr{A}$ as

$$\Xi(x,k) := \{\kappa \in \mathscr{A} : \ell(x,\kappa) \leqslant -\rho(x)\},$$

and $\Pi(x,k,\kappa)$ is a lower semicontinuous function introduced to systematically select the most desirable k^+ according to some optimality criterion (*e.g.*, minimum norm,

minimum deviation from the current k, and so on); see [123] for a discussion on the relationship between the hybrid implementation (5.10) and the techniques given in [124–126]. It should be noted that under Assumption 2, the set $\Xi(x,k)$ is nonempty for all $(x,k) \in \mathscr{U} \times \mathscr{A}$. Hence, under the hypothesis that $\mathscr{U} = \mathbb{R}^n$, in the following theorem (whose proof is given in Appendix A.1), we show that the set $\mathscr{V} \times \mathscr{A}$ is globally asymptotically stable for the hybrid implementation (5.10).

Theorem 1. *Let the lower semicontinuous function $\Pi(x,k,\kappa)$ be level bounded in κ locally uniformly in (x,k) and let Assumptions 1 and 2 hold with $\mathscr{U} = \mathbb{R}^n$. Define $\varpi(x,k) := \inf_{\kappa \in \Xi(x,k)} \Pi(x,k,\kappa)$ and assume that it is locally bounded and continuous. Then the set $\mathscr{V} \times \mathscr{A}$ is globally asymptotically stable for the hybrid implementation* (5.10).

By weakening the assumptions of Theorem 1, we can still guarantee uniform local asymptotic stability of the set $\mathscr{V}$, as stated in the following corollary, whose proof is wholly similar to the one of [123, Cor. 1] and hence is omitted.

Corollary 1. *Let the lower semicontinuous function $\Pi(x,k,\kappa)$ be level bounded in κ locally uniformly in (x,k) and let Assumptions 1 and 2 hold for some $\mathscr{U} \subset \mathbb{R}^n$. Define $\varpi(x,k) := \inf_{\kappa \in \Xi(x,k)} \Pi(x,k,\kappa)$ and assume that it is locally bounded from above and C^0. Then the set $\mathscr{V} \times \mathscr{A}$ is locally asymptotically stable for the hybrid implementation* (5.10).

In view of Corollary 1, the main interest in the hybrid implementation (5.10) relies on the fact that the estimate of the basin of attraction of $\mathscr{V}$ for system

$$\dot{x} = f(x,k^\circ)$$

obtained by using V as Lyapunov function is a subset of the one for the hybrid implementation (5.10) since $\mathscr{W} \subset \mathscr{U}$.

The next remark provides insights on the solution to (5.10).

Remark 1. Differently from [123], solutions to the hybrid system (5.10) need not be eventually continuous. This is due to the fact that the state k of system (5.10) need not converge to the value k° satisfying (5.6), and hence the state of system (5.10) may persistently jump approaching $\mathscr{V} \times \mathscr{A}$. Nonetheless, if either

$$\ell(x,k) \leqslant -\rho(x), \quad \forall (x,k) \in \mathscr{W} \times \mathscr{K},$$

or $\Pi(x,k,\kappa)$ is designed so that $\operatorname{argmin}_{\kappa\in\Xi(x,k)}\Pi(x,k,\kappa)=k^\circ$ for all $(x,k)\in\mathscr{W}\times\mathscr{K}$, then, since the flow set is eventually positively invariant, solutions $x(t,j)$ to system (5.10) such that $\{(t,j)\in\operatorname{dom}(x):x(t,j)\in\mathscr{V}\}=\emptyset$ have a semi-global uniform dwell-time and are eventually continuous.

The next remark suggests a selection for the function Π.

Remark 2. A possible (trivial) selection for the function Π in (5.10b) that meets the hypotheses of Theorem 1 and Corollary 1 is $\Pi(x,k,\kappa)=1$ for all $(x,k,\kappa)\in\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^m$. In such a case, the jump map (5.10b) can be simply rewritten as

$$(x,k)\in\mathscr{D},\quad\begin{cases}x^+=x,\\ k^+\in\Xi(x,k).\end{cases}$$

It is worth pointing out that Theorem 1 and Corollary 1 establish robustness of local asymptotic stability of the set $\mathscr{V}\times\mathscr{A}$ for system (5.10). Namely, under the hypotheses of Corollary 1, by [116, Thm. 7.21] and the proof of Theorem 1, since the hybrid system (5.10) is well-posed and the set $\mathscr{V}\times\mathscr{A}$ is locally asymptotically stable, letting $\mathscr{I}_{\mathscr{V}\times\mathscr{A}}$ be its basin of attraction, then system (5.10) is robustly $\mathscr{KL}$ pre-asymptotically stable on $\mathscr{I}_{\mathscr{V}\times\mathscr{A}}$; see [116, Def. 6.27, 7.3, 7.10 and 7.18]. In particular, there exists a function $\varphi:\mathbb{R}^n\times\mathbb{R}^p\to\mathbb{R}_{\geqslant 0}$ that is positive on $\mathscr{I}_{\mathscr{V}}\setminus(\mathscr{V}\times\mathscr{A})$ such that $\mathscr{V}\times\mathscr{A}$ is $\mathscr{KL}$ pre-asymptotically stable (see [116, Thm. 3.40] for the characterization of asymptotic stability via $\mathscr{KL}$ functions) on $\mathscr{I}_{\mathscr{V}\times\mathscr{A}}$ for the system

$$(x,k)\in\mathscr{C},\quad\begin{cases}\dot{x}\in F_\varphi(x,k),\\ \dot{k}=0,\end{cases}\tag{5.11a}$$

$$(x,k)\in\mathscr{D},\quad\begin{cases}x^+=x,\\ k^+\in\operatorname{argmin}_{\kappa\in\Xi(x,k)}\Pi(x,k,\kappa),\end{cases}\tag{5.11b}$$

where, letting $\xi=\operatorname{col}(x,k)$, the inflated set-valued map $F_\varphi:\mathbb{R}^n\times\mathbb{R}^m\rightrightarrows\mathbb{R}^n$ is given by

$$F_\varphi(\xi):=\overline{\operatorname{con}}(f((\xi+\varphi(\xi)\mathscr{B})\cap\mathscr{C}))+\varphi(\xi)\mathscr{B}.$$

Note that the inflated set-valued map F_φ accounts for both measurement error (through the term $\xi+\varphi(\xi)\mathscr{B}$ at argument of f) and implementation error (through

the additive term $+\varphi(\xi)\mathscr{B}$). Hence, this robustness is particularly desirable when dealing with mobile robots, where position and actuation errors are unavoidable in realistic scenarios.

Furthermore, since $\mathscr{C} \cup \mathscr{D} = \mathbb{R}^n \times \mathscr{A}$ and $k(0,0) \in \mathscr{A}$ implies $k(t,j) \in \mathscr{A}$ for all $(t,j) \in \mathrm{dom}(k)$, trajectories of system (5.11) are complete. Therefore, the set $\mathscr{V} \times \mathscr{A}$ is $\mathscr{KL}$ asymptotically stable on $\mathscr{I}_{\mathscr{V}\times\mathscr{A}}$ for the inflated system (5.11), that is there is a function $\psi \in \mathscr{KL}$ such that every solution $\xi = \mathrm{col}(x,k)$ to system (5.11) satisfies, for all $(t,j) \in \mathrm{dom}(\xi)$,

$$\|\xi(t,j)\|_{\mathscr{V}\times\mathscr{A}}(\|\xi(t,j)\|_{\mathbb{R}^{n+m}\setminus\mathscr{I}_{\mathscr{V}\times\mathscr{A}}})^{-1} \leqslant \psi(\|\xi(0,0)\|_{\mathscr{V}\times\mathscr{A}}(\|\xi(0,0)\|_{\mathbb{R}^{n+m}\setminus\mathscr{I}_{\mathscr{V}\times\mathscr{A}}})^{-1}, t+j).$$

5.3 Design of a vector field to obtain patrolling, formation control and obstacle avoidance

The main goal of this section is the design of an algorithmic procedure to compute a parametric vector field $f(x,k)$, together with functions V and ρ satisfying Assumptions 1 and 2, such that the solutions to system (5.5) constitute a path for mobile robots that is collision-free and that steers them to patrol a preassigned path in controlled formation. This goal is pursued by combining algorithms that use tools borrowed from algebraic geometry (see Sections 5.3.1 and 5.3.2) with methods inspired by classical navigation functions (see Section 5.3.3). The former allow to automatically construct Lyapunov functions certifying the convergence to the desired path in formation, in the absence of obstacles, while the latter allow to avoid collisions among agents and with fixed obstacles.

5.3.1 Review on attractive affine varieties

In this section, we briefly recall some results given in [120, 127, 35] and partially summarized also in [114, 34], which allow to systematically design a vector field $h(x)$ whose associated dynamical system $\dot{x} = h(x)$, has a given affine variety $\mathscr{V} \subset \mathbb{R}^n$ as attractive and h-invariant set. Given $p_1, \ldots, p_s \in \mathbb{R}[x]$, the following Algorithm 2, taken from [127, 35], uses some tools borrowed from algebraic geometry to find a

set of vector fields $h \in \mathbb{R}^n[x]$ such that $\mathscr{V} := \boldsymbol{V}(p_1,\ldots,p_s)$ is attractive and invariant for $\dot{x} = h(x)$ (we refer the reader to [35] for the computational details of such an algorithm and for the feasibility of its solutions).

Algorithm 2:

Input: $p_1,\ldots,p_s \in \mathbb{R}[x]$ such that $\mathscr{V} = \boldsymbol{V}(p_1,\ldots,p_s)$.

Output: a class of vector fields $h \in \mathbb{R}^n[x]$ and a positively invariant set $\mathscr{I}_{\mathscr{V}} \subset \mathbb{R}^n$ such that $\mathscr{V}$ is h-invariant and attractive in $\mathscr{I}_{\mathscr{V}}$.

1: Compute the reduced Gröbner basis $\mathscr{G} = \{\theta_1,\ldots,\theta_\ell\}$ of the sub-module $\langle \nabla p \rangle : \langle p \rangle$.
2: Select $\lambda_i \in (\langle \nabla p \rangle : \langle p \rangle)$, $i = 1,\ldots,n$, such that, letting $\Lambda = [\ \lambda_1 \ \cdots \ \lambda_n\ ]$, the matrix $\Lambda + \Lambda^\top$ is positive semidefinite.
3: Let $c > 0$ be such that the polynomial matrix Λ is positive definite in $\mathscr{I}_{\mathscr{V}} := \{x \in \mathbb{R}^n : \sum_{i=1}^s p_i^2(x) < c\}$.
4: Solve the polynomial equation $\langle \nabla p, h \rangle + \Lambda p = 0$ in h.
5: **return** h and $\mathscr{I}_{\mathscr{V}}$.

In the following sections, firstly an ideal obstacle-free scenario is considered and Algorithm 2 is employed to ensure patrolling of a desired path having the robots maintaining a specific formation among them (Section 5.3.2), and then the obtained vector fields are modified in order to avoid collisions with obstacles and among the agents (Section 5.3.3).

5.3.2 Motion planning and formation control

Let $x_i = [\ x_{i,1} \ \ x_{i,2}\ ]^\top$ denote the Cartesian position of the i–th agent, $i = 1,\ldots,N$, let $n = 2N$, and let $x = [\ x_1^\top \ \cdots \ x_N^\top\ ]^\top$ denote the adjoint state of the *multi-agent model*. Thus, let $p_{i,j} \in \mathbb{R}[x_i]$, $i = 1,\ldots,s_i$, where s_i is a positive integer, be such that the affine variety

$$\mathscr{V}_i = \boldsymbol{V}(p_{i,1},\ldots,p_{i,s_i}) \subset \mathbb{R}^2 \tag{5.12}$$

is the desired patrolling path for the i–th agent. Furthermore, let $q_1,\ldots,q_\omega \in \mathbb{R}[x]$, where ω is a positive integer, be such that the affine variety

$$\mathscr{F} = \boldsymbol{V}(q_1,\ldots,q_\omega) \subset \mathbb{R}^{2N} \tag{5.13}$$

identifies the desired controlled formation of the agents. Essentially, the former set determines the absolute desired patrolling paths to be assigned to some or to

all the agents in the team, whereas the latter set characterizes the potential relative displacement of some of the robots (*followers*, if any), with respect to others (*leaders*, if any). Consider the following proposition, whose proof is given in Appendix A.2.

Proposition 1. *The output of Algorithm 2 with input*

$$p_{1,1},\dots,p_{1,s_1},p_{2,1},\dots,p_{2,s_2},\dots,p_{N,s_N},q_1,\dots,q_\omega \in \mathbb{R}[x],$$

is a family of vector fields $h(x)$ such that the corresponding dynamical system $\dot{x} = h(x)$ has the affine variety

$$\mathscr{V} = \mathscr{F} \cap \left(\bigcap_{i=1}^{N} \mathscr{V}_i \right) \subset \mathbb{R}^{2N}. \tag{5.14}$$

as attractive in $\mathscr{I}_{\mathscr{V}}$ and h-invariant set.

Note that if the affine variety $\mathscr{V}$ given in (5.14) is empty, then, by construction, Algorithm 2 returns $\mathscr{I}_{\mathscr{V}} = \emptyset$, thus showing unfeasibility of the considered problem. Interestingly, despite the fact that the statement of Proposition 1 merely guarantees the existence of the vector field h, indeed the results of [127] allow also to explicitly characterize, and especially *parameterize*, the structure of such vector fields. More precisely, it has been shown in [127] that the set of $h \in \mathbb{R}^{2N}[x]$ provided as outputs of Algorithm 2 with input $p_{1,1},\dots,p_{1,s_1},p_{2,1},\dots,p_{2,s_2},\dots,p_{N,s_N},q_1,\dots,q_\omega$ is described by

$$h(x) = g(x)\chi + f_b(x), \tag{5.15}$$

where

$$\left[\begin{array}{c|c} g & f_b \\ \hline 0 & 1 \end{array}\right] = \mathrm{Syz}([\ \nabla p \quad \Lambda p\ ]),$$

where $p = [\ p_{1,1} \quad \dots \quad q_\omega\ ]^\top$ and χ is an arbitrary vector, potentially function of the state variable, which might be employed to induce additional motions or to optimize desired optimality criteria, without hindering the achievement of the primary task of patrolling and formation control.

The results of Proposition 1 permit the design of patrolling paths for the entire *multi-agent* model, while also potentially establishing hierarchical ordering or formations among the agents. Namely, by [35], the affine variety $\mathscr{V}$ is invariant and

locally attractive with respect to the system

$$\dot{\xi}(t) = g(\xi(t))\chi + f(\xi(t)), \tag{5.16}$$

with basin of attraction containing the set $\mathscr{I}_{\mathscr{Y}}$. However, such techniques do not guarantee that collisions during motions are avoided. This is the objective of the following section.

5.3.3 Collision avoidance among agents and with obstacles

The main objective of this section consists in designing a vector field $\sigma(x)$ - to be combined with the family yielded by Proposition 1 as pursued in the subsequent Section 5.3.4 - such that the trajectories of the dynamical system $\dot{x} = \sigma(x)$ do not collide with the obstacles. Toward this end, the definitions of obstacle and collision are stated below.

Definition 2. *(Obstacle).* The region in the configuration space occupied by the w-th obstacle is denoted $\mathscr{O}_w$. There exists a polynomial $\varsigma_w \in \mathbb{R}[x_1, x_2]$ such that the boundary $\partial\mathscr{O}_w$ of $\mathscr{O}_w$ is given by

$$\partial\mathscr{O}_w = \boldsymbol{V}(\varsigma_w). \tag{5.17}$$

In this framework, the size of the agents is encoded in the polynomials ς_w, with $w = 1, \ldots, W$, that are suitably enlarged in order to take into account also the volume of each agent. It is worth noticing that, as it is customary in this context, it is possible to enclose obstacles whose boundary is not polynomial within an associated ellipse of minimal size.

Definition 3. *(Collision).* Suppose there are W obstacles described by the regions $\mathscr{O}_w$, $w = 1, \ldots, W$. Let r_i characterize the size of the i–th agent, $i = 1, \ldots, N$. A collision is said to occur if there exists $t \in \mathbb{R}_{\geqslant 0}$ such that either $x_i(t) \in \partial\mathscr{O}_w(x_i)$ for some $i \in \{1, \ldots, N\}$, or $|x_i(t) - x_j(t)| = r_i + r_j$ for some $i, j \in \{1, \ldots, N\}$, $i \neq j$.

Define, for $i, j \in \{1, \ldots, N\}$, $i \neq j$, the polynomial

$$c_{i,j} = (x_{i,1} - x_{j,1})^2 + (x_{i,2} - x_{j,2})^2 - (r_i + r_j)^2 \tag{5.18}$$

in $\mathbb{R}[x]$, and let

$$\mathscr{P}_{i,j} = \boldsymbol{V}(c_{i,j}) \subset \mathbb{R}^{2N}.$$

Note that $x \in \mathscr{P}_{i,j}$ if and only if $|x_i(t) - x_j(t)| = r_i + r_j$.

In the following, we suppose that the desired patrolling path is *feasible* in terms of obstacles and prescribed formations, as formalized in the following statement.

Assumption 3. The patrolling path, controlled formation and obstacles are such that

$$\mathscr{V} \cap \left(\bigcup_{w=1}^{W} \mathscr{O}_w \right) \cap \left(\bigcup_{i=1}^{N} \bigcup_{j=i+1}^{N} \mathscr{P}_{i,j} \right) = \emptyset,$$

where $\mathscr{V}$ is defined as in (5.14).

In order to ensure that Assumption 3 is met and to employ the collision avoidance strategy described in the remainder of this section, the shapes of the obstacles have to be known in advance. In the following lemma (whose proof is given in Appendix A.3), we propose a tool to assess whether a collision has occurred.

Lemma 1. *Define the polynomial in* $\mathbb{R}[x]$

$$b(x) = \left(\prod_{i=1}^{N} \prod_{w=1}^{W} \varsigma_w(x_i) \right) \cdot \left(\prod_{i=1}^{N} \prod_{j=i+1}^{N} c_{i,j}(x) \right).$$

A collision occurs if and only if there is $t \in \mathbb{R}_{\geqslant 0}$ *such that*

$$b(x(t)) = 0.$$

Assumption 3 does not depend on the initial configuration of the agents and it may be equivalently - but potentially less intuitive from the geometric point of view - stated by requiring that $\mathscr{V}$ defined as in (5.14) and $\boldsymbol{V}(b)$ possess an empty intersection, *i.e.* $\mathscr{V} \cap \boldsymbol{V}(b) = \emptyset$. This latter condition is, on the other hand, more prone than Assumption 3 to be systematically checked. By taking advantage of the result given in Lemma 1 and by considering a suitable adaptation of classical barrier functions, in the following we design vector fields $\sigma(x)$ such that the trajectories of the corresponding dynamical system $\dot{x} = \sigma(x)$ do not collide with the obstacles. Toward this end, let $p_{1,1}, \ldots, p_{1,s_1}, p_{2,1}, \ldots, p_{2,s_2}, \ldots, p_{N,s_N}, q_1, \ldots, q_\omega \in \mathbb{R}[x]$ be defined as

in Section 5.3.2 and let

$$a = p_{1,1}^2 + \cdots + p_{1,s_1}^2 + \cdots + p_{N,s_N}^2 + q_1^2 + \cdots + q_\omega^2.$$

Thus, letting b be defined as in Lemma 1, consider

$$r(x) = \frac{a(x)}{b(x)}. \tag{5.19}$$

By relying on Assumption 3, the function r goes to zero as x tends to the affine variety $\mathscr{V}$ given in (5.14) and tends to infinity as $b(x)$ tends to 0 (*i.e.*, if a collision is about to occur). Therefore, the positive invariance of sub-level sets of the function $r(x)$ with respect to the system $\dot{x} = \sigma(x)$ implies that the trajectories of such a system generate collision-free path for the multi-agent system. Hence, let

$$\eta(x) = \nabla r(x), \tag{5.20}$$

and consider the following (rational) system

$$\dot{x}(t) = -\eta(x(t))\beta(t), \tag{5.21}$$

where $\beta : \mathbb{R}_{\geqslant 0} \to \mathbb{R}_{\geqslant 0}$. The following proposition (whose proof is given in Appendix A.4) shows that the trajectories of system (5.21) are collision-free.

Proposition 2. *Let $x_0 \in \mathbb{R}^{2N}$ be given and assume that $r(x_0) \geqslant 0$. Suppose that Assumption 3 holds. If $r(x_0) < \infty$, then, letting $x(t)$ be the solution of system* (5.21)*, there does not exist a time $t \geqslant 0$ such that $b(x(t)) = 0$.*

Hence, by Proposition 2 and Lemma 1, the vector field $\sigma(x) = -\eta(x)\beta$ solves the collision avoidance problem. It is worth noticing that the main goal of such a vector field is just to ensure that the path of motion of the mobile robots, modelled by system (5.21), is collision-free. Such a goal can be pursued using a unified framework to deal with collision both among agents and with obstacles since the design is centralized, *i.e.*, the motion of all the agents is jointly designed toward this objective. Such a centralization avoids the issues that may arise from the disjoint design of the motion of all the mobile robots, which should account for the fact that the motion of each agent is influenced and influences the one of all the others.

5.3.4 Patrolling, formation control and collision avoidance

The control tools proposed in Section 5.3.2 constitute a solution to the patrolling and formation control goals alone, whereas the method given in Section 5.3.3 only guarantees that the trajectories of the multi-agent system are collision-free. The main objective of this section is to show how to suitably combine these two results together to solve the patrolling in formation and collision avoidance problem. Toward this end, consider the following assumption.

Assumption 4. Let $\mathscr{V}$ be defined as in (5.14) and let an initial configuration $x_0 \in \mathbb{R}^{2N}$ be given. There exists a continuous path $\mathscr{P} \subset \mathbb{R}^{2N}$ between x_0 and $\mathscr{V}$ such that $\mathscr{P} \cap \boldsymbol{V}(b) = \emptyset$.

Assumption 4 essentially guarantees that there exists a solution to the patrolling in formation and collision avoidance problem for a given initial configuration of the agents. The control task basically consists in determining such feasible path $\mathscr{P}$. Thus, let $p_{1,1}, \ldots, p_{1,s_1}, p_{2,1}, \ldots, p_{2,s_2}, \ldots, p_{N,s_N}, q_1, \ldots, q_\omega \in \mathbb{R}[x]$ be defined as in Section 5.3.2, let h (parametrized as in (5.15), with $f_b \in \mathbb{R}^{2N}[x]$ and $g \in \mathbb{R}^{2N\times\hbar}[x]$) be the output of Algorithm 2, and let η and b be defined as in Section 5.3.3. By Propositions 1 and 2 and by Lemma 1, the most intuitive way to combine the two previously designed vector fields would be to exploit the degrees of freedom hinted at in the discussion after Proposition 1 to *shape* the vector field as the one in (5.21) for some β. Hence, if there exist functions $\bar{\chi} : \mathbb{R}^{2N} \to \mathbb{R}^{\hbar}$ and $\bar{\beta} : \mathbb{R}^{2N} \to \mathbb{R}_{\geqslant 0}$ such that $f_b(x) + g(x)\bar{\chi}(x) = \eta(x)\bar{\beta}(x)$, then the solutions to the dynamical system

$$\dot{x} = f_b(x) + g(x)\bar{\chi}(x)$$

would solve the patrolling in formation and collision avoidance problems simultaneously. However, such functions need not exist in general, and alternative *combination* strategies must be envisioned. Thus, consider the following system

$$\dot{x} = \gamma_1 b^2(x) f_b(x) + \gamma_1 b^2(x) g(x)\zeta(x) - \eta(x)\mu(x)p(x), \tag{5.22}$$

where $p(x) := [\ p_{1,1} \ \cdots \ p_{N,s_N} \ \ q_1 \ \ \ldots \ \ q_\omega\ ]^\top \in \mathbb{R}^{\varkappa}[x]$ and $\gamma_1 > 0$, which has been obtained by adding the dynamics of system (5.16) multiplied by $\gamma_1 b^2(x)$ and (5.21), and by letting $\chi = \zeta(x)$ and $\beta = \mu(x)p(x)$, where $\zeta : \mathbb{R}^{2N} \to \mathbb{R}^{\hbar}$ and $\mu : \mathbb{R}^{2N} \to \mathbb{R}^{1\times\varkappa}$. The following theorem (whose proof is given in Appendix A.5)

states that, if the function μ and the parameter γ_1 in (5.22) satisfy some easily verifiable assumptions, then system (5.22) locally solves the patrolling in formation and collision avoidance problem.

Theorem 2. *Let Assumptions 3 and 4 hold and let the position and the shape of the obstacles $\mathscr{O}_w$, $w = 1,\dots,W$, be known. Thus, let $\mathscr{R} := (\bigcup_{i=1}^{N}\bigcup_{w=1}^{W}\{x \in \mathbb{R}^{2N} : \varsigma_w(x_i) \leqslant 0\}) \cdot (\bigcup_{i=1}^{N}\bigcup_{j=i+1}^{N}\{x \in \mathbb{R}^{2N} : c_{i,j}(x) \leqslant 0\})$ be the set of unfeasible states and let Λ be the matrix defined at Step 3 of Algorithm 2. Thus, if $\mu : \mathbb{R}^{2N} \to \mathbb{R}^{1\times\omega}$ and γ_1 are such that:*

(a) the symmetric part of the matrix $\gamma_1 b^2\Lambda + \frac{\partial p}{\partial x}\eta\,\mu$ is positive definite in $\mathscr{Y} := \{x \in \mathbb{R}^{2N} \setminus \mathscr{R} : p^\top(x)p(x) \leqslant d\}$, for some sufficiently small $d \in \mathbb{R}_{\geqslant 0}$;

(b) $\mu(x)p(x) \geqslant 0$ for all $x \in \mathbb{R}^{2N} \setminus \mathscr{R}$;

then system (5.22) *solves the patrolling in formation and collision avoidance problem for all $x_0 \in \mathscr{Y}$.*

Note that the choice $\gamma_1 > 0$ and $\mu(x) = \gamma_2 p^\top(x)$, with $\gamma_2 > 0$, guarantees that the requirements of Theorem 2 are satisfied for a sufficiently small $d \in \mathbb{R}_{>0}$ under Assumption 3. As a matter of fact, since, by Assumption 3, one has $\mathscr{V} \cap \mathscr{R} = \emptyset$, there exists a sufficiently small $\varepsilon^\star \in \mathbb{R}_{>0}$ such that $b(x) \neq 0$ for all $x \in \mathscr{V} + \varepsilon^\star\mathscr{B}$. Therefore, one has that $\gamma_1 b^2\Lambda + \gamma_2\frac{\partial p}{\partial x}\eta\, p^\top \simeq \gamma_1 b^2\Lambda$ for all $x \in \mathscr{V} + \varepsilon^\star\mathscr{B}$, thus implying that in $\mathscr{V} + \varepsilon^\star\mathscr{B}$ the dynamics of system (5.22) essentially matches the one of system (5.16) rescaled via the constant γ_1. Clearly, the domain $\mathscr{V} + \varepsilon^\star\mathscr{B}$ is a restrictive estimate of the set of initial conditions for which system (5.22) solves the patrolling in formation and collision avoidance problem. In particular, thanks to the use of the hybrid implementation (see the subsequent Theorem 3), the basin of attraction of the desired formation is usually almost all the workspace of the multi-agent system; see [121] for further details.

Therefore, letting $k = \mathrm{col}(\gamma_1, \gamma_2, \chi)$ and defining the set of admissible parameters $\mathscr{A}$ as any compact subset of $\mathbb{R}_{>0} \times \mathbb{R}_{>0} \times \mathbb{R}^s$, the parametric vector field

$$f(x,k) = \gamma_1 b^2(x) f_b(x) + \gamma_1 b^2(x) g(x)\chi - \gamma_2 \|p(x)\|_2^2 \eta(x), \tag{5.23}$$

locally solves the problem of designing a path for mobile robots that is collision free and that steers them to patrol a preassigned curve in controlled formation.

Furthermore, letting

$$\underline{\gamma}_1 := \min_{\mathscr{A}} \gamma_1, \qquad \underline{b} := \min_{x \in \mathscr{A} + \varepsilon^\star \mathscr{B}} b(x),$$

which are both positive under Assumption 4 since $\mathscr{V} \cap \boldsymbol{V}(b) = \emptyset$, the vector field $f(x,k)$, together with the functions

$$V(x) = \|p(x)\|_2^2, \qquad \rho(x) = \frac{1}{2}\underline{\gamma}_1 \underline{b}\, p^\top \Lambda p, \tag{5.24}$$

satisfies Assumptions 1 and 2, thus allowing us to implement system (5.22) via the hybrid implementation (5.10).

Therefore, the hybrid implementation (5.10), with f defined as in (5.23), locally solves the patrolling in formation and collision avoidance problem, as formally stated in the following theorem, whose proof is reported in Appendix A.6.

Theorem 3. *Suppose that Assumptions 3 and 4 hold. Let f be defined as in* (5.23)*, let $\mathscr{A}$ be a compact subset of $\mathbb{R}_{>0} \times \mathbb{R}_{>0} \times \mathbb{R}^s$, and consider the hybrid system* (5.10)*. If the function $\Pi(x,k,\kappa)$ is level bounded in κ locally uniformly in (x,k) and the function $\varpi(x,k) := \inf_{\kappa \in \Xi(x,k)} \Pi(x,k,\kappa)$ is locally bounded from above and C^0, then the hybrid implementation* (5.10) *solves the patrolling in formation and collision avoidance problem for all $x_0 \in \mathscr{Y}$.*

By Theorem 3, if the inputs of (5.3) are selected as

$$(x,k) \in \mathscr{C}, \quad u = f(x,k), \tag{5.25a}$$

$$(x,k) \in \mathscr{D}, \quad k^+ \in \operatorname*{argmin}_{\kappa \in \Xi(x,k)} \Pi(x,k,\kappa), \tag{5.25b}$$

then the closed loop trajectories constitute paths that are collision free and that steer the robots to patrol a preassigned path in controlled formation.

In view of the discussion given in Section 5.1.2, such a centralized control strategy can be implemented by the team of mobile robots by letting their inputs be

$$\begin{bmatrix} v_1 \\ \omega_1 \\ \vdots \\ v_N \\ \omega_N \end{bmatrix} = \begin{bmatrix} \cos(\Theta_1) & \sin(\Theta_1) & \cdots & 0 & 0 \\ -\dfrac{\sin(\Theta_1)}{L_1} & \dfrac{\cos(\Theta_1)}{L_1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cos(\Theta_N) & \sin(\Theta_N) \\ 0 & 0 & \cdots & -\dfrac{\sin(\Theta_N)}{L_N} & \dfrac{\cos(\Theta_N)}{L_N} \end{bmatrix} f(x,k), \quad (5.26)$$

where $x = \begin{bmatrix} X_{P,1} & Y_{P,1} & \cdots & X_{P,N} & Y_{P,N} \end{bmatrix}^\top$ and the navigation parameters k are updated following the hybrid dynamics (5.25).

5.4 Simulation results

In this section, the techniques outlined in Sections 5.2 and 5.3 are tested via numerical simulations. In particular, Example 1 shows how the proposed hybrid mechanism can be used to steer the agents to desired target positions, Example 2 shows how it can be used to let the agents patrol a selected algebraic curve, and Example 3 shows how it can be used to let the leader patrol a selected curve while the other agents keep a prescribed formation.

Example 1. Let $N = 4$ and suppose that the size of the 4 agents is $r_i = 0.3$, $i = 1, \ldots, 4$. The objective of this example is to steer the four robots to the target points

$$x_{t,1} = \begin{bmatrix} -1 \\ -0.5 \end{bmatrix}, \qquad x_{t,2} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix},$$
$$x_{t,3} = \begin{bmatrix} -1 \\ 0.5 \end{bmatrix}, \qquad x_{t,4} = \begin{bmatrix} 1 \\ -0.5 \end{bmatrix},$$

Thus, define the polynomials

$$\begin{aligned} p_{1,1} &= x_{1,1}+1, & p_{1,2} &= x_{1,2}+0.5,\\ p_{2,1} &= x_{2,1}-1, & p_{2,2} &= x_{2,2}-0.5,\\ p_{3,1} &= x_{3,1}+1, & p_{3,2} &= x_{3,2}-0.5,\\ p_{4,1} &= x_{4,1}-1, & p_{4,2} &= x_{4,2}+0.5. \end{aligned}$$

Algorithm 2 with input $p_{1,1},\dots,p_{4,2}$ returns a vector field h, parametrized as in (5.15) with $f_b \in \mathbb{R}^8[x]$ and $g=0$,

$$f_b(x) = \begin{bmatrix} -x_{1,1}-1 \\ -x_{1,2}-0.5 \\ 1-x_{2,1} \\ 0.5-x_{2,2} \\ -x_{3,1}-1 \\ 0.5-x_{3,2} \\ 1-x_{4,1} \\ -x_{4,2}-0.5 \end{bmatrix},$$

and the matrix $\Lambda = I_8$.

Assume that, in the workspace of the multi-agent system, there are four obstacles $\mathscr{O}_w$, whose boundary is $\partial\mathscr{O}_w = \boldsymbol{V}(\varsigma_w) \subset \mathbb{R}^2$, $\varsigma_w \in \mathbb{R}[x_1,x_2]$, $w=1,\dots,4$, where

$$\varsigma_1 = x_1 - 1.4, \qquad \varsigma_2 = x_1 + 1.4, \tag{5.27a}$$

$$\varsigma_3 = x_2 + 0.8, \qquad \varsigma_4 = x_2 - 0.8, \tag{5.27b}$$

so that the actual workspace of the multi-agent system is the interior of the rectangle $[-1.4,1.4]\times[-0.8,0.8] \subset \mathbb{R}^2$.

Following Proposition 2 and Lemma 1, let

$$\begin{aligned} a &= \sum_{i=1}^{N}\sum_{j=1}^{2} p_{i,j}^2 \in \mathbb{R}[x],\\ b &= \left(\prod_{w=1}^{4} \varsigma_w(x_1)\varsigma_w(x_2)\varsigma_w(x_3)\varsigma_w(x_4)\right)\left(\prod_{i=1}^{4}\prod_{j=i+1}^{4} c_{i,j}(x)\right) \in \mathbb{R}[x], \end{aligned}$$

where the polynomials $c_{i,j}$ are as in (5.18), let $r(x) = \frac{p(x)}{b(x)}$, and let $\eta(x) = \frac{\partial r(x)}{\partial x}$. Hence, defining $f(x,k)$ as in (5.23), the trajectories of the hybrid implementa-

tion (5.10) of system (5.5) are collision-free path of motions for the multi-agent system.

Figure 5.3 depicts the solution to system (5.10) with $\mu = 10^{-2}$, $x(0) = [\ -1.2 \quad 0 \quad -0.4 \quad 0 \quad 0.4 \quad 0 \quad 1.2 \quad 0\ ]^\top$, $\mathscr{A} = [10^{-3}, 10^3]^2$, $\Pi(k,x,\kappa) = \kappa_1^2 + \kappa_2^2$, and $k(0) = [\ 1 \quad 0.5\ ]^\top$.

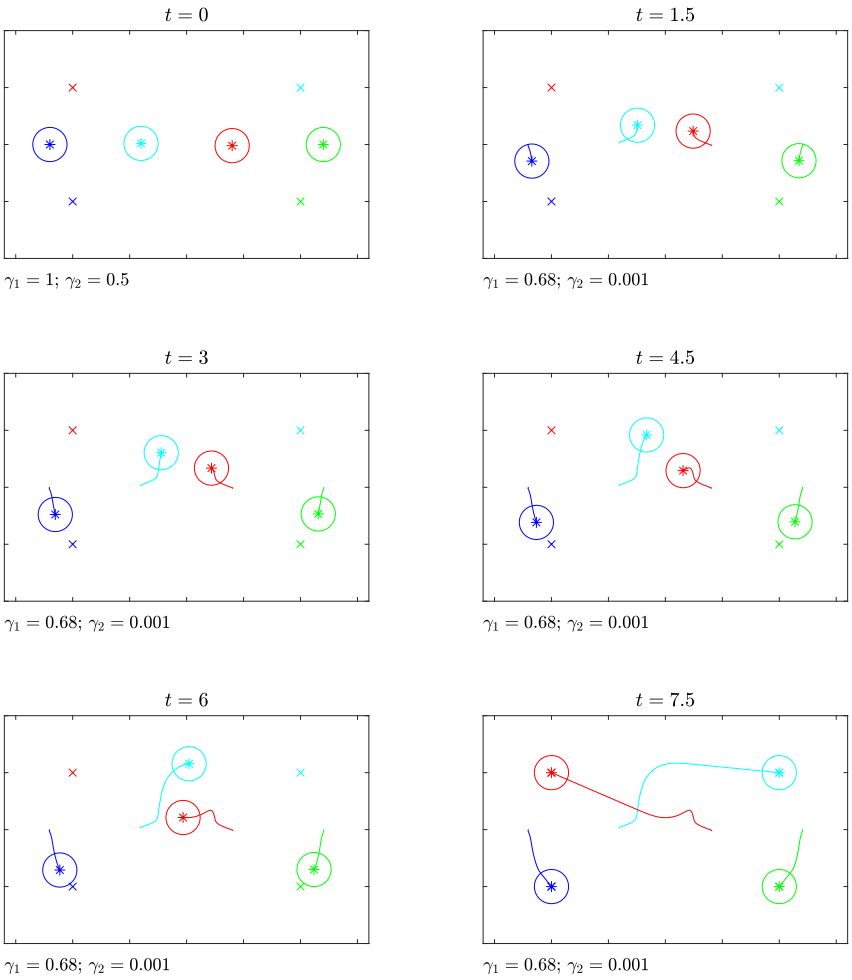


Fig. 5.3 Solution of the hybrid implementation (5.10) for the problem considered in Example 1.

As shown by such a figure, the hybrid implementation (5.10) generates collision-free path of motions for the mobile robots that avoid collision and that steers the agents toward the desired target points. It is worth noticing that the hybrid implementation triggers a change in the parameters k at time $t = 9.4818 \cdot 10^{-4}$ so to guarantee convergence of the path of motion to the desired target points. This emphasizes the

fact that the proposed navigation algorithm is capable of autonomously changing the initial values of the parameters k so to ensure convergence. $\triangle$

Example 2. Let $N = 3$ and suppose that the size of the 3 agents is $r_i = 0.3$, $i = 1,2,3$. The objective of this example is to let the three robots patrol the affine variety (see Figure 5.4)

$$\boldsymbol{V}\left(x_1^4 - x_1^2 + x_2^2 - \tfrac{1}{100}\right)$$

Thus, define the polynomials

$$\begin{aligned} p_{1,1} &= x_{1,1}^4 - x_{1,1}^2 + x_{1,2}^2 - \tfrac{1}{100}, \\ p_{2,1} &= x_{2,1}^4 - x_{2,1}^2 + x_{2,2}^2 - \tfrac{1}{100}, \\ p_{3,1} &= x_{3,1}^4 - x_{3,1}^2 + x_{3,2}^2 - \tfrac{1}{100}. \end{aligned}$$

Algorithm 2 with input $p_{1,1}, p_{2,1}, p_{3,1}$ returns a vector field h, parametrized as in (5.15) with $f_b \in \mathbb{R}^6[x]$ and $g \in \mathbb{R}^{6\times 3}[x]$,

$$f_b(x) = \begin{bmatrix} -x_{1,1}^7 + \frac{3x_{1,1}^5}{2} - x_{1,2}^2 x_{1,1}^3 - \frac{49x_{1,1}^3}{100} + \frac{1}{4}x_{1,2}^2 x_{1,1} - \frac{x_{1,1}}{200} \\ -\frac{1}{2}x_{1,2}^3 + \frac{1}{4}x_{1,1}^2 x_{1,2} + \frac{x_{1,2}}{200} \\ -x_{2,1}^7 + \frac{3x_{2,1}^5}{2} - x_{2,2}^2 x_{2,1}^3 - \frac{49x_{2,1}^3}{100} + \frac{1}{4}x_{2,2}^2 x_{2,1} - \frac{x_{2,1}}{200} \\ -\frac{1}{2}x_{2,2}^3 + \frac{1}{4}x_{2,1}^2 x_{2,2} + \frac{x_{2,2}}{200} \\ -x_{3,1}^7 + \frac{3x_{3,1}^5}{2} - x_{3,2}^2 x_{3,1}^3 - \frac{49x_{3,1}^3}{100} + \frac{1}{4}x_{3,2}^2 x_{3,1} - \frac{x_{3,1}}{200} \\ -\frac{1}{2}x_{3,2}^3 + \frac{1}{4}x_{3,1}^2 x_{3,2} + \frac{x_{3,2}}{200} \end{bmatrix},$$

$$g(x) = \begin{bmatrix} 0 & 0x_{1,2} \\ 0 & 0x_{1,1} - 2x_{1,1}^3 \\ 0 & x_{2,2} \quad 0 \\ 0 & x_{2,1} - 2x_{2,1}^3 \quad 0 \\ x_{3,2} & 0 \quad 0 \\ x_{3,1} - 2x_{3,1}^3 & 0 \quad 0 \end{bmatrix},$$

and the matrix

$$\Lambda = \operatorname{diag}\Big(x_{1,1}^2(1 - 2x_{1,1}^2)^2 + x_{1,2}^2, \\ x_{2,1}^2(1 - 2x_{2,1}^2)^2 + x_{2,2}^2, x_{3,1}^2(1 - 2x_{3,1}^2)^2 + x_{3,2}^2 \Big).$$

Letting the workspace be as in Example 1, define the polynomials $\varsigma_1, \ldots, \varsigma_4$ as in (5.27), let

$$a = \sum_{i=1}^{N} p_{i,1}^2 \in \mathbb{R}[x],$$
$$b = \left(\prod_{w=1}^{4} \varsigma_w(x_1)\varsigma_w(x_2)\varsigma_w(x_3) \right) \left(\prod_{i=1}^{3} \prod_{j=i+1}^{3} c_{i,j}(x) \right) \in \mathbb{R}[x],$$

where the polynomials $c_{i,j}$ are as in (5.18), let $r(x) = \frac{p(x)}{b(x)}$, and let $\eta(x) = \frac{\partial r(x)}{\partial x}$. Hence, letting $f(x,k)$ be defined as in (5.23), the trajectories of the hybrid implementation (5.10) of system (5.5) are collision-free path of motions for the multi-agent system.

Figure 5.4 depicts the solution to system (5.10) with $\mu = 10^{-2}$, $x(0) = [\ -0.5 \quad 0.5 \quad 0 \quad 0.5 \quad 0.5 \quad 0.5\ ]^\top$, $\mathscr{A} = [10^3, 10^6] \times [10^{-3}, 10^3]$, $\Pi(k,x,\kappa) = \kappa_1^2 + \kappa_2^2 + (\kappa_3 - 1)^2 + (\kappa_4 - 1)^2 + (\kappa_5 - 1)^2$ (where κ_1 and κ_2 have the role of γ_1 and γ_2, respectively, whereas $[\ \kappa_3 \quad \kappa_4 \quad \kappa_5\ ]^\top$ have the role of χ), and $k(0) = [\ 10^3 \quad 10^{-2} \quad 1 \quad 1 \quad 1\ ]^\top$.

As shown by such a figure, the hybrid implementation (5.10) generates collision-free path of motions for the mobile robots that avoids collision and that let the agents patrol the desired curve. Note that the trajectories attained by the agents are dependent just on the initial condition $(x(0), k(0))$, which uniquely determines the state-evolution of the (autonomous) hybrid system (5.10). Note that the hybrid implementation triggers a change in the parameters k at time $t = 1.4950$ so to guarantee convergence of the path of motion to $\mathscr{V}$. $\triangle$

Example 3. Let $N = 3$ and suppose that the size of the 3 agents is $r_i = 0.3$, $i = 1,2,3$. The objective of this example is to let the first robot (the leader) patrol the circle

$$\boldsymbol{V}\left(x_1^2 + x_2^2 - \tfrac{1}{8}\right),$$

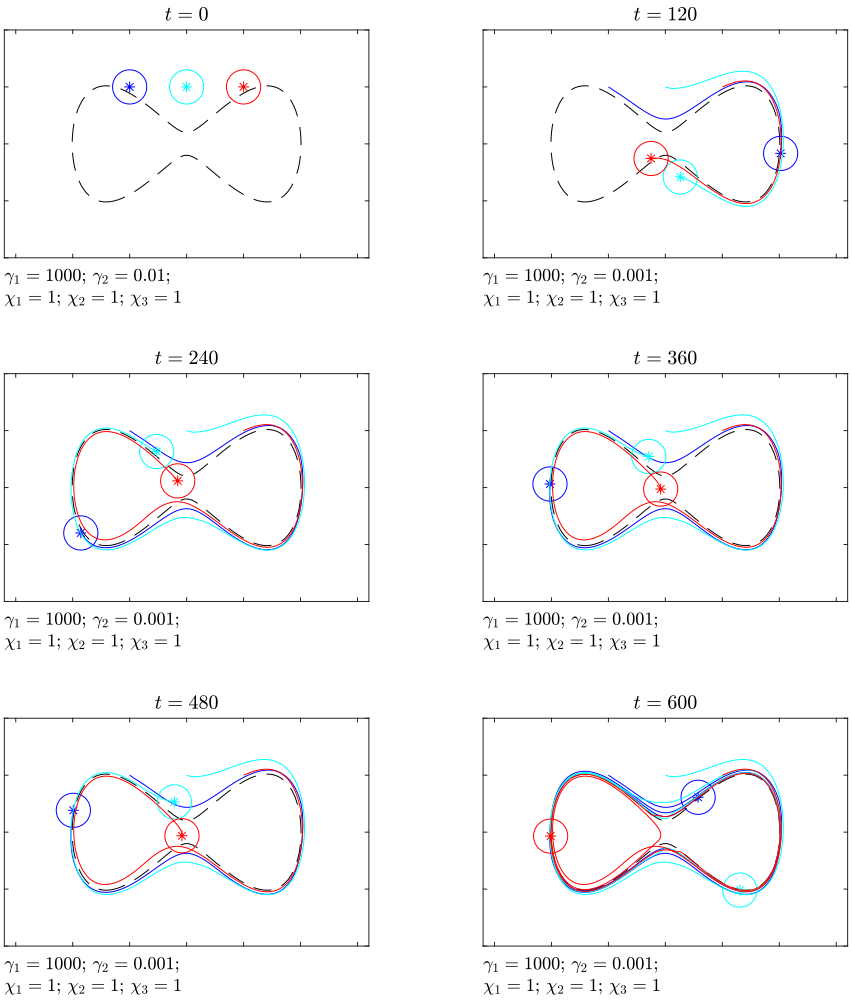


Fig. 5.4 Solution of the hybrid implementation (5.10) for the problem considered in Example 2.

while letting the other two agents be at prescribed distance, equal to 0.5. Thus, define the polynomials

$$\begin{aligned} p_{1,1} &= x_{1,1}^2 + x_{1,2}^2 - \tfrac{1}{8}, \\ q_1 &= (x_{1,1} - x_{2,1})^2 + (x_{1,2} - x_{2,2})^2 - \tfrac{1}{4}, \\ q_2 &= (x_{1,1} - x_{3,1})^2 + (x_{1,2} - x_{3,2})^2 - \tfrac{1}{4}. \end{aligned}$$

Algorithm 2 with input $p_{1,1}, q_1, q_2$ returns a vector field h, parametrized as in (5.15) with $f_b \in \mathbb{R}^6[x]$ and $g \in \mathbb{R}^{6\times 3}[x]$,

$$f_b(x) = \begin{bmatrix} -\frac{1}{2}x_{1,1}^3 - x_{1,2}^2 x_{1,1} + \frac{x_{1,1}}{16} \\ \frac{x_{1,2}}{16} - \frac{x_{1,2}^3}{2} \\ f_{b,3}(x) \\ -\frac{1}{2}x_{2,2}^3 + \frac{3}{2}x_{1,2}x_{2,2}^2 - \frac{3}{2}x_{1,2}^2 x_{2,2} + \frac{x_{2,2}}{8} - \frac{x_{1,2}}{16} \\ f_{b,5}(x) \\ -\frac{1}{2}x_{3,2}^3 + \frac{3}{2}x_{1,2}x_{3,2}^2 - \frac{3}{2}x_{1,2}^2 x_{3,2} + \frac{x_{3,2}}{8} - \frac{x_{1,2}}{16} \end{bmatrix},$$

$$g(x) = \begin{bmatrix} 0 & 0 & x_{1,2} \\ 0 & 0 & -x_{1,1} \\ 0 & x_{1,2}-x_{2,2} & x_{2,2} \\ 0 & x_{2,1}-x_{1,1} & -x_{2,1} \\ x_{1,2}-x_{3,2} & 0 & x_{3,2} \\ x_{3,1}-x_{1,1} & 0 & -x_{3,1} \end{bmatrix},$$

where

$$\begin{aligned} f_{b,3}(x) = &-\tfrac{1}{2}x_{2,1}^3 + \tfrac{3}{2}x_{1,1}x_{2,1}^2 - \tfrac{3}{2}x_{1,1}^2 x_{2,1} - x_{1,2}^2 x_{2,1} \\ &- x_{2,2}^2 x_{2,1} + 2x_{1,2}x_{2,2}x_{2,1} \\ &+ \tfrac{1}{8}x_{2,1} + x_{1,1}x_{2,2}^2 - \tfrac{1}{16}x_{1,1} - 2x_{1,1}x_{1,2}x_{2,2}, \\ f_{b,5}(x) = &-\tfrac{1}{2}x_{3,1}^3 + \tfrac{3}{2}x_{1,1}x_{3,1}^2 - \tfrac{3}{2}x_{1,1}^2 x_{3,1} - x_{1,2}^2 x_{3,1} \\ &- x_{3,2}^2 x_{3,1} + 2x_{1,2}x_{3,2}x_{3,1} \\ &+ \tfrac{1}{8}x_{3,1} + x_{1,1}x_{3,2}^2 - \tfrac{1}{16}x_{1,1} - 2x_{1,1}x_{1,2}x_{3,2}, \end{aligned}$$

and the matrix

$$\Lambda = \operatorname{diag}\Big(x_{1,1}^2 + x_{1,2}^2, (x_{1,1}-x_{2,1})^2 + (x_{1,2}-x_{2,2})^2, \\ (x_{1,1}-x_{3,1})^2 + (x_{1,2}-x_{3,2})^2 \Big).$$

Letting the workspace be as in Examples 1 and 2, define the vector η as in Example 2. Hence, letting $f(x,k)$ be defined as in (5.23), by Theorem 3, the trajectories of the hybrid implementation (5.10) of system (5.5) are collision-free path of motions for the multi-agent system.

Figure 5.5 depicts the solution to system (5.10) with $\mu = 10^{-2}$, $x(0) = [\ -1 \quad -0.6 \quad -0.5 \quad 0 \quad -1 \quad 0.6\ ]^\top$, $\mathscr{A} = [10^{-3}, 10^3]^2$, $\Pi(k,x,\kappa) = \kappa_1^2 + \kappa_2^2 + \kappa_3^2 + \kappa_4^2 + (\kappa_5 - 1)^2$ (where κ_1 and κ_2 have the role of γ_1 and γ_2, respectively,

whereas $\begin{bmatrix} \kappa_3 & \kappa_4 & \kappa_5 \end{bmatrix}^\top$ have the role of χ), and $k(0) = \begin{bmatrix} 10^3 & 10^{-3} & 0 & 0 & 1 \end{bmatrix}^\top$.

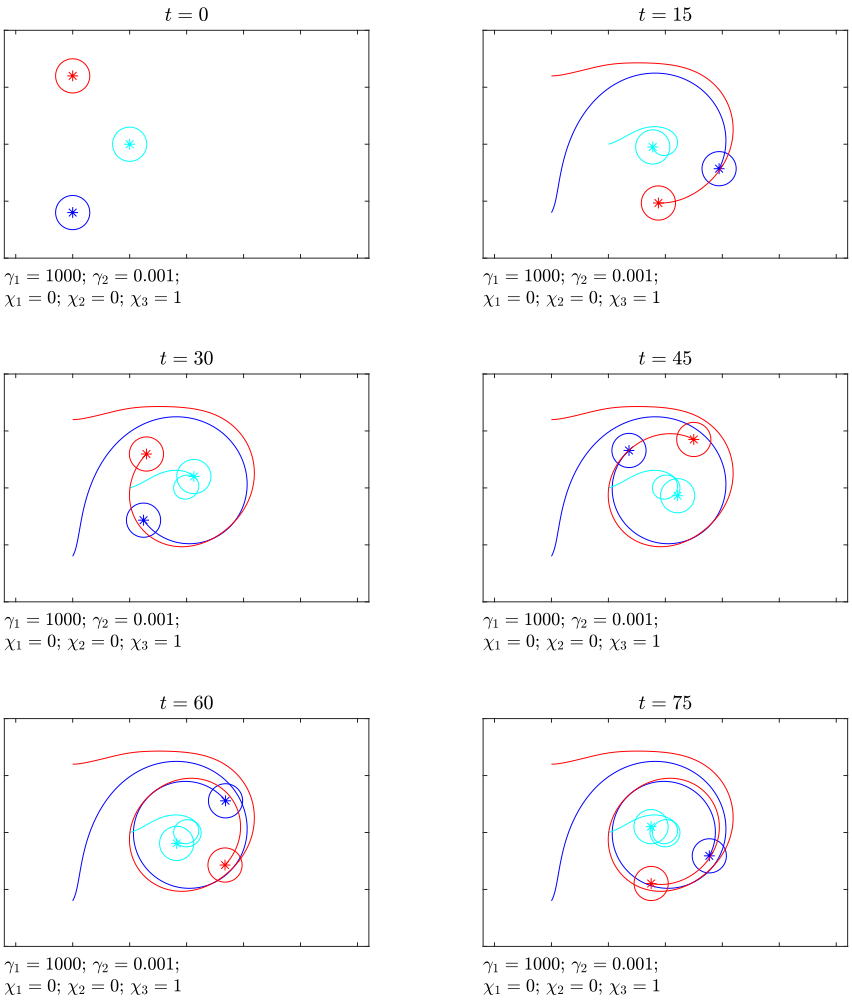


Fig. 5.5 Solution of the hybrid implementation (5.10) for the problem considered in Example 3.

As shown by such a figure, the hybrid implementation (5.10) generates collision-free path of motions for the mobile robots that avoid collision and that let the leader patrol the desired curve while letting the other agents be in formation. It is worth noticing that, in such a simulation, the hybrid implementation (5.10) does not trigger any change in the constants k. △

5.5 Experimental results

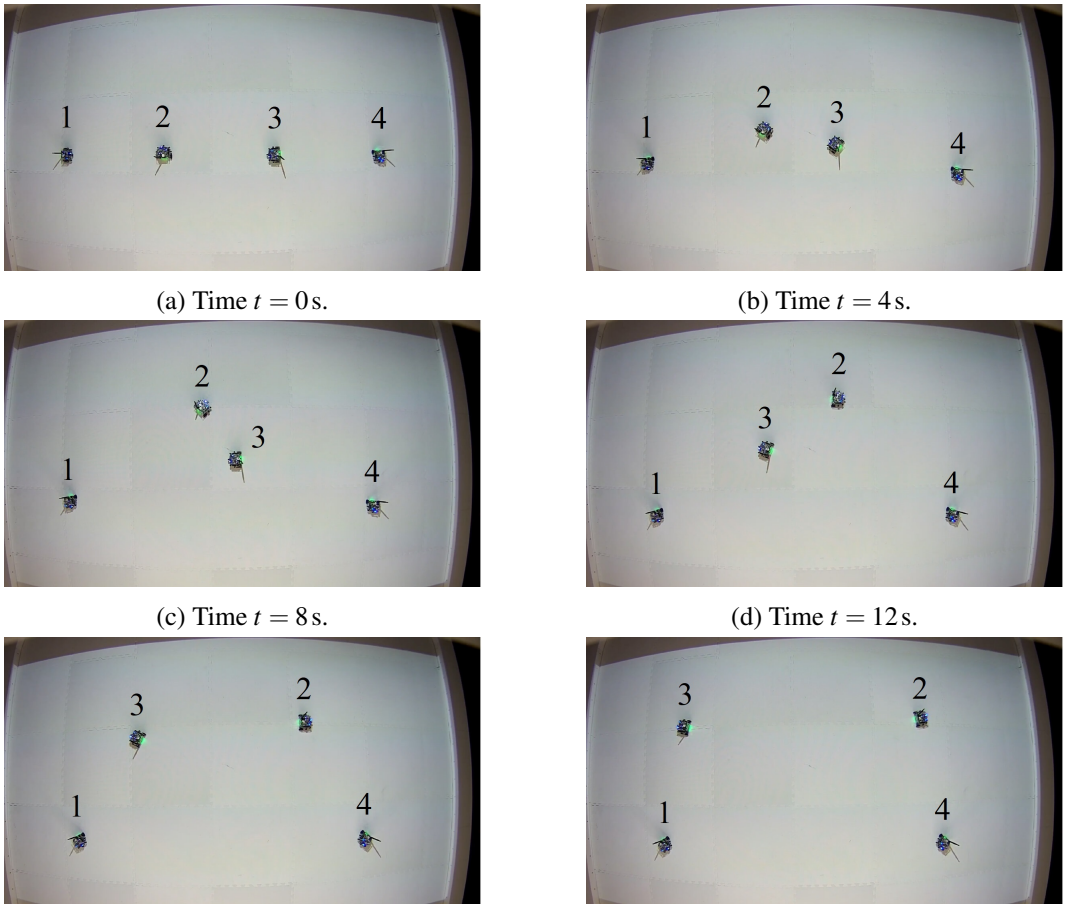
Experiments have been carried out to test the control law derived in Examples 1, 2, and 3 using the `Robotarium`, a remotely accessible robotic testbed, designed to help users quickly prototype and validate their control strategies through implementation on physical robots without the overhead of setting up their own hardware or high-fidelity simulation. In particular, Experiment 1 shows how the control law given in Example 1 can be employed to steer four mobile robots to a desired target point. In Experiment 2, the control method outlined in Example 2 is used to let three mobile agents patrol a given curve. In the last experiment, the control law reported in Example 3 is validated, letting the leader patrol a circle while keeping the other two agents at a prescribed distance. To ease the description of the experimental results, a set of relevant snapshots is presented, supported by a proper labeling of the involved robots.

The single-integrator dynamics corresponding to the hybrid implementation (5.10) have been converted to unicycle dynamics using the linearizing feedback given in (5.2), with $L = 0.1\,\mathrm{m}$, *i.e.*, the team of mobile robots has been controlled using the centralized control law given in (5.26). Further, since the trajectories of the system $\dot{x}(t) = F(t)f(x(t))$ are the same as those of the system $\dot{x}(t) = f(x(t))$, provided that $F(t) > 0$ for all $t \in \mathbb{R}$, the linear and angular velocities resulting from (5.2) have been uniformly scaled so to meet the actuation constraint. Namely, letting v_i and ω_i be the linear and angular velocities resulting from (5.2) for$i = 1, \ldots, N$, and letting $\bar{v}$ and $\bar{\omega}$ be the maximal admissible linear and angular velocities of the mobile robots, the parameter $F(t)$ above has been selected as

$$F = \min\left\{1,\ \frac{\bar{v}}{2\max_i\{|v_i|\}},\ \frac{\bar{\omega}}{2\max_i\{|\omega_i|\}}\right\}.$$

Experiment 1. The control law given in Example 1 has been used to steer four robots to a target position. The initial poses of the robots are $X_1(0) = -1.2\,\mathrm{m}$, $Y_1(0) = 0\,\mathrm{m}$, $\Theta_1(0) = 0\,\mathrm{rad}$, $X_2(0) = -0.4\,\mathrm{m}$, $Y_2(0) = 0\,\mathrm{m}$, $\Theta_2(0) = \frac{\pi}{2}\,\mathrm{rad}$, $X_3(0) = 0.4\,\mathrm{m}$, $Y_3(0) = 0\,\mathrm{m}$, $\Theta_3(0) = \pi\,\mathrm{rad}$, $X_4(0) = 1.2\,\mathrm{m}$, $Y_4(0) = 0\,\mathrm{m}$, $\Theta_4(0) = \frac{3}{2}\pi\,\mathrm{rad}$.

Figure 5.6 reports some snapshots of the corresponding experiment; see [128] for the full video.



(a) Time $t = 0\,\mathrm{s}$.

(b) Time $t = 4\,\mathrm{s}$.

(c) Time $t = 8\,\mathrm{s}$.

(d) Time $t = 12\,\mathrm{s}$.

(e) Time $t = 16\,\mathrm{s}$.

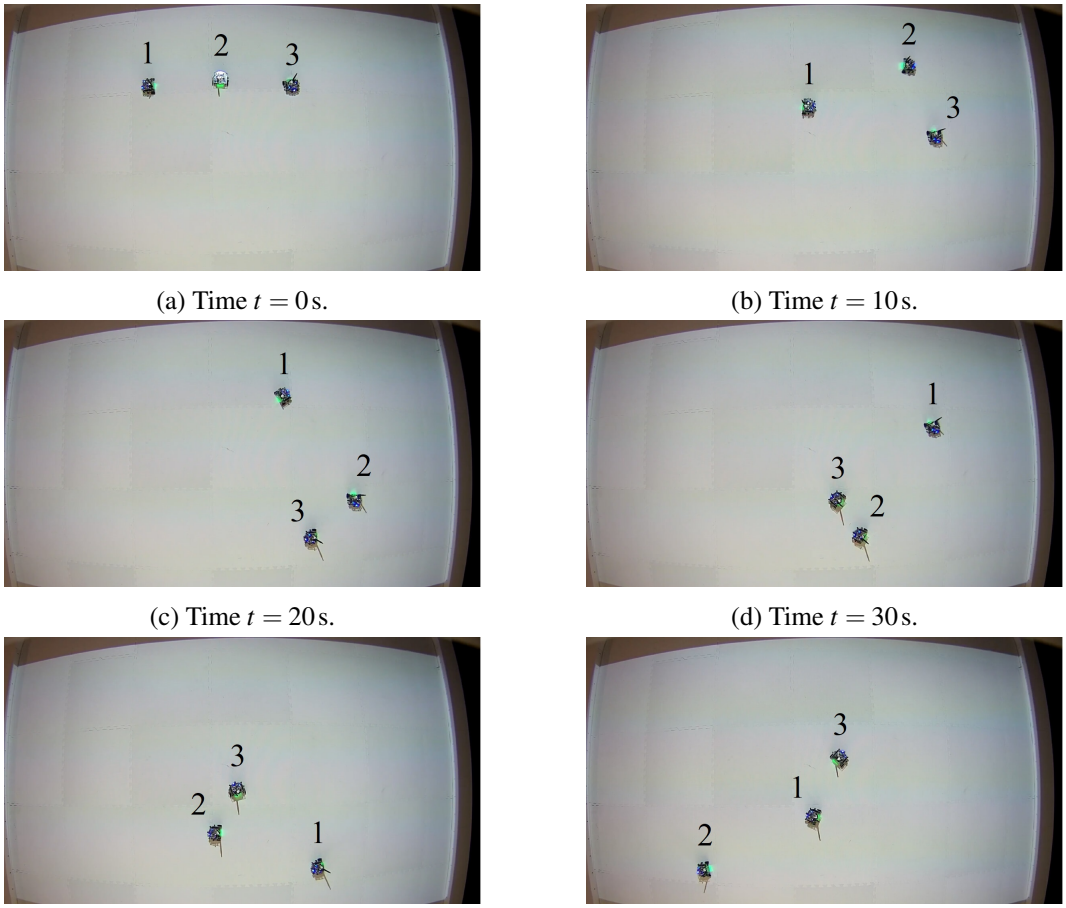
(f) Time $t = 20\,\mathrm{s}$.

Fig. 5.6 Result of the experiment with the control law given in 1.

As shown by such a figure, the proposed control method allows to steer the four agents to their target positions avoiding collisions among themselves and with fixed obstacles despite the actuators saturation and the robots nonidealities. However, by comparing Figures 5.3 and 5.6, it can be observed a slower convergence to the target points due to the saturation introduced to satisfy the actuators limitations. △

Experiment 2. The control law given in Example 2 has been used to let three robots patrol a given curve. The initial poses of the robots are $X_1(0) = -0.5\,\mathrm{m}$, $Y_1(0) = 0.5\,\mathrm{m}$, $\Theta_1(0) = \pi\,\mathrm{rad}$, $X_2(0) = 0\,\mathrm{m}$, $Y_2(0) = 0.5\,\mathrm{m}$, $\Theta_2(0) = \frac{\pi}{2}\,\mathrm{rad}$, $X_3(0) = 0.5\,\mathrm{m}$, $Y_3(0) = 0.5\,\mathrm{m}$, $\Theta_3(0) = \frac{3}{2}\pi\,\mathrm{rad}$.

Figure 5.7 reports some snapshots of the corresponding experiment; see [129] for the full video. As demonstrated by such a figure, the proposed control method let the three agents patrol the desired curve avoiding collisions among themselves and with



(a) Time $t = 0\,\mathrm{s}$.

(b) Time $t = 10\,\mathrm{s}$.

(c) Time $t = 20\,\mathrm{s}$.

(d) Time $t = 30\,\mathrm{s}$.

(e) Time $t = 40\,\mathrm{s}$.

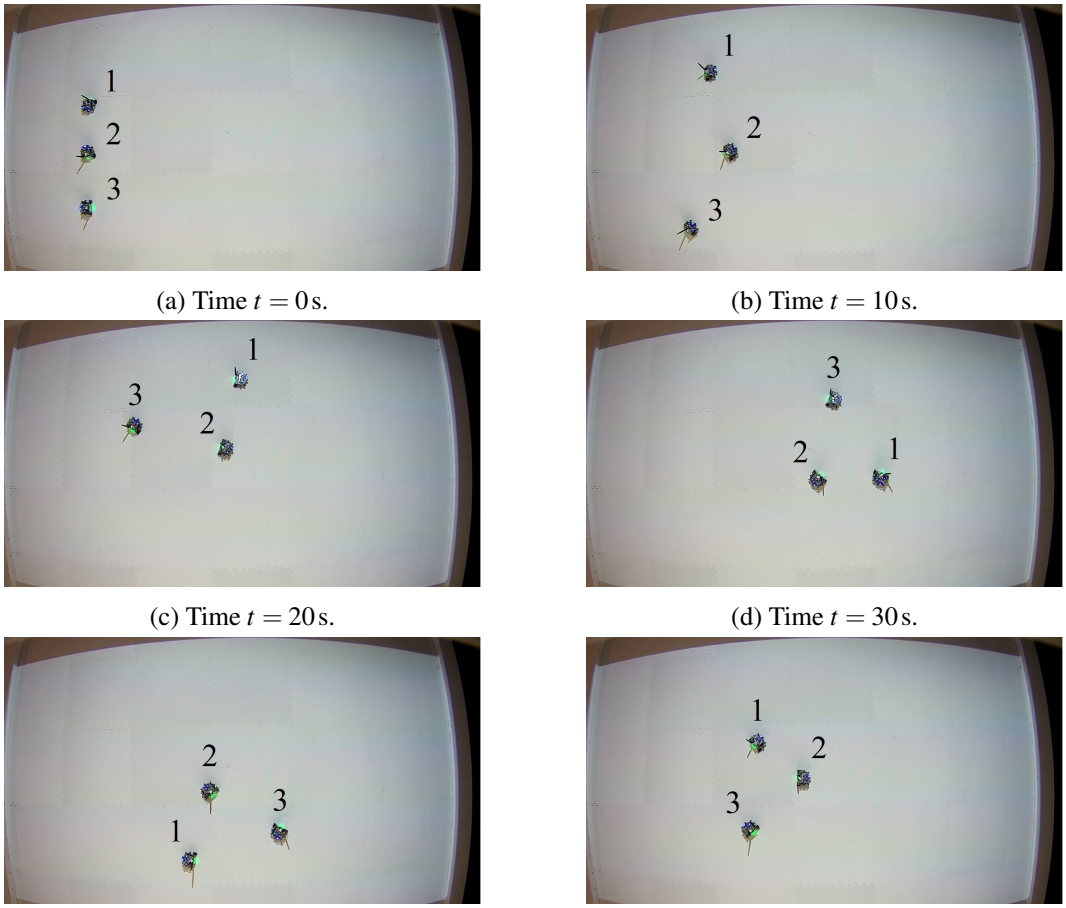
(f) Time $t = 50\,\mathrm{s}$.

Fig. 5.7 Result of the experiment with the control law given in Example 2.

fixed obstacles. However, note that the resulting trajectory is slightly different from the one depicted in Figure 5.4 due to position measurement errors, which, however, do not affect the convergence of the proposed navigation method by the robustness analysis reported at the end of Section 5.2. △

Experiment 3. The control law given in Example 3 has been used to let the leader patrol a circle while letting the other two robots be at a prescribed distance. The initial poses of the robots are $X_1(0) = -1\,\mathrm{m}$, $Y_1(0) = -0.6\,\mathrm{m}$, $\Theta_1(0) = 0\,\mathrm{rad}$, $X_2(0) = -0.5\,\mathrm{m}$, $Y_2(0) = 0\,\mathrm{m}$, $\Theta_2(0) = 0\,\mathrm{rad}$, $X_3(0) = -1\,\mathrm{m}$, $Y_3(0) = 0.6\,\mathrm{m}$, $\Theta_3(0) = 0\,\mathrm{rad}$.

Figure 5.8 reports some snapshots of the corresponding experiment; see [130] for the full video. As shown by such a figure, the proposed control method allows the leader to patrol the desired curve while keeping the other agents at a prescribed distance and avoiding collisions among the robots and with fixed obstacles. △



(a) Time $t = 0\,\mathrm{s}$.

(b) Time $t = 10\,\mathrm{s}$.

(c) Time $t = 20\,\mathrm{s}$.

(d) Time $t = 30\,\mathrm{s}$.

(e) Time $t = 40\,\mathrm{s}$.

(f) Time $t = 50\,\mathrm{s}$.

Fig. 5.8 Result of the experiment with the control law given in Example 3.

5.6 Discussion

A navigation strategy has been proposed to steer a team of unicycle-like mobile robots towards a desired path while maintaining a prescribed formation and avoiding collisions among the agents and with static obstacles in the environment. This goal has been achieved by employing some techniques borrowed from algebraic geometry to algorithmically construct a Lyapunov function that guarantees convergence to the desired formation and to the prescribed path if obstacles are absent. These techniques have been coupled with classical potential functions to guarantee that the path of motions of the agents are collision-free, and with a hybrid mechanism that autonomously selects the navigation configurable parameters, thus making the overall procedure completely automatized.

Indeed, given the prescribed path to be patrolled, the desired formation, the locations of the obstacles and the dimensions of the mobile robot, the results given in this work directly allow to algorithmically construct a navigation strategy that solves the problem, through the following steps:

1. Algorithm 2 is applied to determine functions f_b and g.
2. Equation (5.20) is used to determine η.
3. The formulas given in (5.24) are employed to compute V and ρ.
4. Equations (5.7) and (5.8) are used to define the flow and the jump set.
5. The system (5.25) can be finally, directly implemented by selecting a suitable function Π (possibly with $\Pi(x,k,\kappa) = 1$, as suggested in Remark 2).

The robustness analysis inherited by the hybrid framework shows that the proposed navigation strategy is robust with respect to inaccurate measurements of the positions of the agents and imperfect implementation of the nominal strategy.

The effectiveness of the proposed motion planning strategy has been validated both in simulation and in experimental tests, carried out using a remotely accessible robotic testbed.

Chapter 6

Sensor data fusion for smart AMRs in human-shared industrial working spaces

A sensor data fusion algorithm for smart AMRs in human-shared industrial working spaces is presented in this chapter. In particular, the objective of this chapter is to propose an improvement for the mobile robot perception system in order to be able to detect and avoid obstacles, particularly human workers that coexists with other mobile agents in an industrial environment. The proposed sensor data fusion algorithm processes and merges the data coming from a laser range finder, the video stream of a camera and an object detection system based on artificial intelligence. The combination of the three elements allows to classify a detected obstacle from the robot's field of view while extracting also the information related to distance-to-obstacle, with main focus to human detection.

The sections of this chapter are divided as follows: A background related to the research work is presented in Section 6.1. Section 6.2 presents the proposed solution, providing first a high-level description within the working scenario. Afterwards, the adopted implementation is briefly illustrated in Section 6.3. In Section 6.4 the achieved results are presented, showing the experimental testing the sensor data fusion algorithm to detect and safely avoid humans. Lastly, Section 6.5 discusses the research work as well as some open issues.

6.1 Background

In the last decades, mobile robots have been widely used in many different fields, thanks to their adaptability to a vast range of applications. In the industrial context, mobile robots are usually classified as Automated Guided Vehicles (AGVs) and Autonomous Mobile Robots (AMRs). The AGVs are mobile platforms that perform repetitive tasks while carrying the transportation of heavy materials, following a pre-defined path within industrial environments [131]. Human-accessible working spaces are traditionally separated from the AGV operational space due to safety reasons, since AGVs do not have a decision mechanism based on artificial intelligence, and most of them require a particular infrastructure setup [132]. On the other hand, AMRs are able to perceive their surroundings in order to create a model of the environment and locate themselves in it, leading to the capability of working in an unknown or partially known environment. Usually, an AMR is equipped with a heterogeneous set of sensors whose output data streams are processed by complex control systems [133]. It is well known that combining the data coming from different sources enhances the efficiency and robustness of the measurements, but on the other hand, it increases the complexity of the hardware and software required for merging and processing the information deriving from different sensors [134].

Depending on the application requirements, one may combine several types of sensors in order to cover a wider range of measurements or have redundant data to avoid false positives. For instance, AMRs working in industrial applications are commonly equipped with a combination of vision and distance sensors, e.g., cameras with Radio Detection And Ranging (RADAR) systems [135] or cameras with Laser Imaging Detection and Ranging (LIDAR) sensors [136],[137], due to the fact that a safe behaviour between machinery and human workers must be guaranteed.

On one hand, vision sensors are used for the recognition of objects or particular geometric patterns but are usually influenced by environmental lighting conditions. On the other hand, the distance sensors provide high accuracy on measurements, even though its performance may be affected by:

- The reflection properties of the object to be detected in the case of LIDARs
- External radio wave frequencies when using RADARs

Despite the limitations of each sensor type, by combining them it is possible to enhance the overall perception system since the detected object can be associated with its corresponding distance in the robot coordinate system.

A feasible approach for performing sensor fusion can be achieved by using *machine learning* techniques. A commonly used Neural Network (NN) for object classification in an image or video frame is the *You Only Look Once* (YOLO) [59] real-time object detection system. YOLO applies a Convolutional Neural Network (CNN) to the full image, dividing it into regions and performing the bounding boxes prediction and relative probabilities computation for each region. The bounding boxes that have high confidence scores are kept as final predictions, obtaining thus the considered detections.

For instance, the authors in [138] use the dataset coming from RGB-D cameras and compare the performance of several CNNs in order to robustly detect and localize a person. The work in [139] proposes a person detection algorithm based on the linear Support Vector Machines (SVM) learning process that extracts laser features and Histograms of Oriented Gradients (HOG) features from image data. Other approaches, e.g., in [140], combine a Kalman filter with the Global Nearest-Neighbour method, in order to predict and resolve the data association problem for people tracking. Furthermore, a considerable body of literature treats the LIDAR and vision data fusion as an extrinsic calibration problem: the two sensors' coordinate systems are put in relation through a rigid body transformation, so to align the data derived from both sensors [141]. For the purpose of computing this transformation, usually an external object is required, such as a checkerboard pattern [142],[143] or a trirectangular trihedron [144], to match the correspondences between the two sensors and obtain a mapping that transforms the points from the laser to the camera coordinate system, and thereafter to the image plane. Although most of the researchers use stereo cameras and 3D LIDARs to model the environment (since they can give a detailed representation of the surroundings), these devices are expensive, and most of the time it is possible to overcome this problem by using transformation matrices when using a 2D LIDAR and an entry-level camera [145].

If we take into account the problem from a higher level of analysis, a mobile robot's reaction to dynamically-changing environments has been handled in several ways. In [146], the mobile platforms have the capability of overtaking unforeseen obstacles appearing on (or near) the pre-defined route, exploiting local deviations fos-

tered by a centralized data fusion system, which takes in on-board sensing along with infrastructure-based environment perception system measurements. In [147], the mobile robot navigation in unknown dynamic environments implements a pedestrian-like behaviour. In particular, thanks to the human neural system, a pedestrian can estimate the time to collision with obstacles and changes its direction accordingly, while optimizing path length and safety distance. The navigation approach is similarly based on the motion model predictability of obstacles.

The scenario envisaged for the smart factories of the next future implies a significant presence of mobile agents, both having some level of autonomy or not, in working spaces shared with humans. This research work addresses, in particular, the system described in [14], where AMRs act as *meta-sensors*, namely, as mobile entities included in a wider concept of the sensor system, having the aim of supporting a net of traditional AGVs in order to increase their consciousness about their surroundings and to be compliant with safe collaborative operations between humans and robots. The main contribution of this work represents a preliminary development of one meta-sensor module, exploiting camera-laser data sensor fusion algorithms to ensure safe behaviour when specific objects are detected and shares relevant data with other robotic systems. Specifically, the attention is devoted to human detection and recognition, which has to be guaranteed in a safe way through a proper SW/HW architecture including both *safe* and *not safe* devices, from the industrial point of view.

Therefore, in contrast to other commonly adopted methods, where the perception system is trained at the starting phase to combine the data from different sources, we take advantage of a pre-trained NN for human identification. The outputs of the NN are the elements of the image already classified and labeled, which are subsequently used for the sensor fusion algorithm, avoiding the creation of a further dataset.

The aim of the proposed system is then to provide an affordable solution, which takes advantage of sensor fusion to integrate state-of-the-art object detection algorithms taking data from low-cost vision sensors to complement standard safety-compliant sensors, for instance, safety-rated laser scanning systems. It is worth pointing out that low-cost vision sensors may not be intrinsically safe when they are used alone. Still, they can cover a larger range of measurements when combined with other sources along with a data fusion algorithm. Also, with respect to the most recent concept that positions AMRs as an evolution of traditional AGVs, here the

former support the latter during their daily tasks, enabling the possibility of bringing back to light pre-existing obsolete systems. Furthermore, non-infrastructural sensors allow for a more flexible set-up and scalability with respect to other solutions, which associates centralized systems with infrastructural monitoring.

6.2 Sensor data fusion for a meta-sensor AMR

This work describes the first development phase of the entity called meta-sensor, which plays a fundamental role within the system whose specifications are set in [14]. In order to give a background and motivation to the features that have been built up for this entity, a brief description of the working scenario is provided hereafter. The system is thought for ideally any flexible production line, composed of traditional AGVs, cobots, and workstations, in spaces shared with human operators. Three macro-elements can be identified:

(a) A meta-sensor AMR fleet

(b) The Sensors Synergy Center (SSC)

(c) The AGV Coordination Center Interface

The work presented here covers points (a) and (b). Note that the developed model for the AMR meta-sensor entity can be replicated for other elements of the fleet. When going through the solution description, notice that our AMR is a feature-enhancer entity more than a mere evolution of the classical AGV; it is a part of the AGV net and it can be considered as the "brain" behind the system synergy, leveraging sensor data fusion to improve the AGV fleet awareness about the environment's dynamical changes. In this proposed system, the human operator is particularly considered as the target of interest to be detected and advertised to all agents moving within the system.

6.2.1 Solution overview within the meta-sensor system

After defining the role of the AMR within the system, it is possible to identify the ideal behaviour and capabilities the AMR element and SSC should have. In

order to achieve smart navigation in a human-shared workspace, informative and meaningful data from the surroundings should be gathered from the perception system. With this aim, but with the purpose of keeping cost low, too, a mobile robot has been equipped with a monocular camera and a 2D laser range finder in order to perform data association. So as to perform a correct mapping of information, the transformation between the laser and the camera must be computed, by applying the so-called extrinsic calibration.

The artificial intelligence needed for spotting any human obstacles is entrusted to the vision part of the sensor system, namely, it performs human-obstacle detection. Once a human obstacle is detected and its absolute position identified, the task of the AMR as a meta-sensor is to share this information with other agents and define a reaction rule in the presence of this particular kind of obstacle, i.e., human-obstacle avoidance.

Extrinsic calibration

In order to combine the information coming from a monocular camera and a laser range finder, an extrinsic calibration method is required to transform the laser points in the camera reference frame and project them onto the image plane. For that purpose, first, the internal and external parameters of the camera have been determined, and then computed the rigid body transformation between the laser and the camera coordinate systems.

- *Camera Calibration*. The camera calibration consists in estimating a relationship between the information of the camera coordinate system and the image frame, along with the relative pose of the camera with respect to the world reference frame [148]. The estimation process consist in finding distortion coefficients, and the camera's intrinsic and extrinsic parameters. Assuming the pinhole model of the camera, the transformation of the real-world 3D points $\boldsymbol{C_p} = [X, Y, Z, 1]^T$ to the image 2D points $\boldsymbol{c_p} = [u, v, 1]^T$ is defined as:

$$\boldsymbol{c_p} \sim \boldsymbol{K} \cdot \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \cdot \boldsymbol{C_p} \tag{6.1}$$

  where $\boldsymbol{K} \in \mathbb{R}^{3\times3}$ is the so-called camera intrinsic matrix, $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ and $\boldsymbol{t} \in$

$\mathbb{R}^{3\times1}$ are the extrinsic parameters of the camera, which relate the world 3D information to the camera coordinate system.

The intrinsic matrix $\boldsymbol{K}$ is a projective transformation of the 3D points from the camera coordinates into the 2D image coordinates and is defined as:

$$\boldsymbol{K} = \begin{bmatrix} \alpha_1 & s & c_x \\ 0 & \alpha_2 & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{6.2}$$

where α_1 and α_2 are the focal lengths in pixel units, c_x and c_y the coordinates of the principal point of the image in pixel units, and s the skew coefficient between the axis of the image.

Due to the fact that the ideal pinhole camera model does not have a lens and the image from a real camera may present some deformation, the distorted points have been corrected by estimating the radial and tangential distortion coefficients [149].

The radial distortion is related with those straight lines in space that are mapped to curves in the image, while the tangential one occurs when the image plane and the lens are not aligned. The relationship between those coefficients and the image points are defined as:

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} (1+k_1\cdot r^2+k_2\cdot r^4+k_3\cdot r^6)\cdot x \\ (1+k_1\cdot r^2+k_2\cdot r^4+k_3\cdot r^6)\cdot y \end{bmatrix} + \begin{bmatrix} 2\cdot p_1\cdot x\cdot y+p_2\cdot(r^2+2\cdot x^2) \\ p_1\cdot(r^2+2\cdot y^2)+2\cdot p_2\cdot x\cdot y \end{bmatrix} \tag{6.3}$$

where x_d and y_d are the distorted point coordinate, k_1, k_2 and k_3 are the radial distortion coefficients, p_1 and p_2 the tangential distortion coefficients, x and y the normalized image projection point, and $r^2 = x^2 + y^2$.

- *Camera-Laser calibration*. Once the camera is calibrated, a laser to camera extrinsic calibration algorithm based on [143] has been applied, taking into account the physical characteristics of the sensors to estimate the relative pose of the camera with respect to the laser range finder reference frame. Let's consider a point $\boldsymbol{C_p} \in \mathbb{R}^{4\times1}$ in the camera coordinate system, located in

$\boldsymbol{L_p} \in \mathbb{R}^{4\times1}$ in the laser reference frame. The rigid transformation between the two coordinate systems can be expressed as:

$$\boldsymbol{L_p} = \begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{\Delta} \\ 0\,0\,0 & 1 \end{bmatrix} \cdot \boldsymbol{C_p} \tag{6.4}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{3\times3}$ is the rotation matrix of the camera with respect to the laser range finder and $\boldsymbol{\Delta} \in \mathbb{R}^{3\times1}$ the relative translation vector.

Hereafter, the core of the centralized SSC features integrating the meta-sensors net is described.

Human-obstacle detection

Generally, object identification algorithms a has the aim to determine the presence of any instance of objects categorized into classes within some given image or video frame. Moreover, the recognized objects spatial locations within the image reference frame are returned, for example, via bounding boxes. In recent years, object detection performances have been upgraded with the introduction of deep learning techniques, which let a machine automatically learn feature representations from data. With deep learning, in general, patterns are classified using statistical techniques based on sample data and processing it with multi-layered neural networks [150],[151]. The CNNs are among the most popular architectures for deep-learning, since they are designed to receive multiple arrays data as input, such as the three-channel (RGB) image array structure. Many methods solve the object detection as a classification problem, namely, object proposals are produced and fed to a classifier. Nevertheless, some other methods formulate detection as a regression problem, having spatially separated bounding boxes and associated class probabilities as output [152]. Most of the recent methods are region-based, meaning that they perform selective research to obtain region proposals, despite this kind of approach often represents a speed bottleneck.

Therefore, regression-based methods getting rid of the region proposal step have represented a suitable choice for the proposed algorithm, since it is required to identify a specific object class with some index of confidence (or rather, the output class probabilities for the detected objects), and locate these objects in a suitable

spatial representation in a reasonable time. The scheme that represents the adopted human detection process at a high level is shown in Figure 6.1.

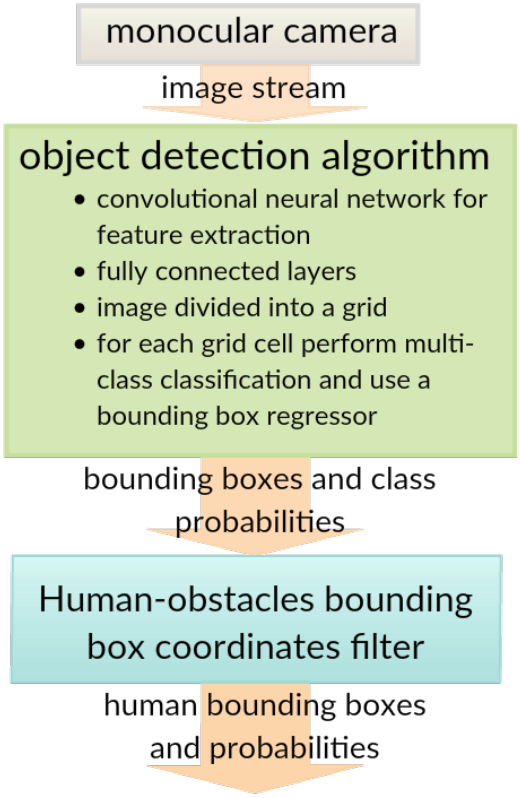


Fig. 6.1 Human obstacle detection process.

Human-obstacle avoidance

With regard to the mobile platform reaction to a human-obstacle, either advertised by other agents on the shared map or directly sensed by the considered robot, in this research work has been decided to apply a more conservative approach in terms of safety distance that separates the moving agent and the detected obstacle. By extracting the human location expressed in the shared map reference frame, it is possible to get target positions that can be enclosed in *virtual cages*. The virtual cages are areas that cannot be accessed by the network of mobile robots and it is characterized by a wider safety radius value with respect to other obstacles. Observe that, since relevant information is shared between the Sensors Synergy Center and the AGV Coordination Center, the presence of humans is made available to both

mobile robot fleets (AMRs and AGVs). In this way, the rules for obstacle avoidance applied in this case are not different from the ones adopted in the case that the robot encounters generic obstacles.

6.3 Solution implementation

6.3.1 Hardware setup

In order to test the sensor data fusion algorithm for human detection and avoidance, we have employed a Pioneer 3DX mobile robot, whose basic setup is previously described in 2.2.3. On top of that, it has been installed an entry-level IP camera (ONVIF [153] standalone unit, accessible via its IP address). The Raspberry Pi has been used for receiving data and controlling the robot while the code that required higher computational resources has been run on a desktop PC with an Intel Core i7-7700 CPU and a dedicated GTX1060/6GB GPU. The monocamera has been placed above the laser range finder and its orientation set such that the image plane intersects the laser plane as shown in Figure 6.2. In particular, the red, green and blue arrows represents the X, Y and Z axis respectively of each reference frame.

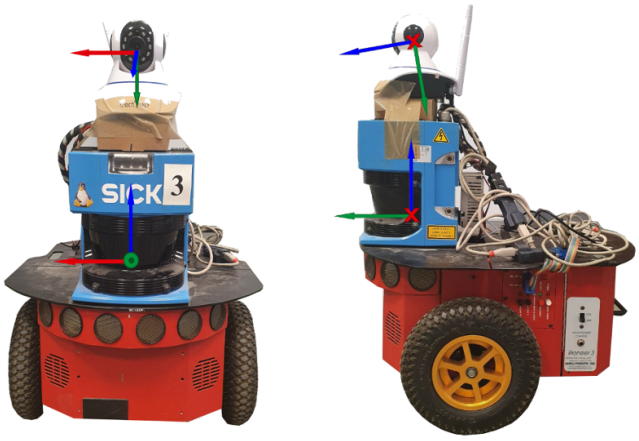


Fig. 6.2 Representation of the sensors reference frames.

6.3.2 Software implementation

The steps taken for the solution implementation are basically the following: sensors calibration, relevant data extraction, data association and creation of virtual obstacles, which are described hereafter.

Laser-Camera transformation computation

- **Intrinsic Calibration with MATLAB**: The intrinsic calibration of the camera was performed using the MATLAB Computer Vision Toolbox. For that purpose, it is required a set of 20 images of a checkerboard at some fixed distance from the camera and placed at different orientations, assuming that all inclination angles are kept below 45° with respect to the camera plane [38]. The camera provides an image resolution of 1280 x 720 pixels, and so the following intrinsic matrix is obtained:

$$\boldsymbol{K} = \begin{bmatrix} 1.3046 \cdot 10^3 & 0 & 727.7219 \\ 0 & 1.3064 \cdot 10^3 & 241.5278 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Extrinsic calibration and laser point projection**: The extrinsic calibration components was calculated by collecting simultaneously data from both the vision and laser range finder sensors. For the extrinsic calibration process, it has been taken 20 images of the checkerboard in several orientations and their corresponding laser readings. Taking into account the technical specifications of the sensors, a modified version of the extrinsic calibration algorithm proposed by Zhang and Pless in [143] has been employed, obtaining the following transformation matrices:

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.9848 & 0.1736 \\ 0 & -0.1736 & 0.9848 \end{bmatrix}, \boldsymbol{\Delta} = \begin{bmatrix} -0.0215 \\ -0.2065 \\ 0.0304 \end{bmatrix}$$

With the rotation matrix $\boldsymbol{\Phi}$ and translation vector $\boldsymbol{\Delta}$, it is possible to compute the laser points in the camera reference frame, and project them in the image plane. Indeed, by defining the matrix $\boldsymbol{H} \in \mathbb{R}^{3\times4}$ as:

$$\boldsymbol{H}(i,j) = \boldsymbol{\Phi}^{-1}(i,j), \text{ for } i = 1,...,3 \text{ and } j = 1,...,3$$
$$\boldsymbol{H}(i,4) = -\boldsymbol{\Delta}(i), \text{ for } i = 1,...,3$$

and combining (6.2) and (6.4) the vector $\boldsymbol{c_p}$ is obtained as:

$$\boldsymbol{c_p} = \boldsymbol{K} \cdot \boldsymbol{H} \cdot \boldsymbol{L_p} \tag{6.5}$$

It is worth noticing that the world 3D reference frame is coincident with the camera coordinates one, and so the resulting $\boldsymbol{R}$ matrix is the identity and $\boldsymbol{t}$ is a null vector. In order to have a correct representation of the points in the image, the vector $\boldsymbol{c_p}$ must be normalized with respect to the third component, leading to a vector in the form $\hat{\boldsymbol{c}}_{\boldsymbol{p}} = [u,v]^T$, so as to have the representation in pixels.

As a result, by applying the transformation matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Delta}$, the laser points are projected in the image plane, as can be seen in Figure 6.3, which is quite reasonable, since all the mapped points are coherent with the hit surfaces.

Fig. 6.3 Laser points projected in the image plane.

Relevant bounding box information extraction

Since the image processing requires a set of different tools, all the related software has been grouped within a Docker [61] container, which leverages the GPU computational capabilities and removes the burden of installing ad-hoc tools, fostering portability and scalability.

For what concerns the human detection system, the real-time object detection YOLO have been chosen, since it satisfies the proposed system requirements, as specified in Section 6.2.1). It first processes the IP camera stream, previously

interpreted as a generic webcam output using the *gstreamer* [154] tool. Then, the YOLO code that generates the bounding boxes on screen has been modified to save their coordinates information in a text file. The output file is used as a source for a ROS [36] node that reads it and wraps into ROS topic messages only the information we are interested in, which in this case are the lines identified by a "person" label. All communications between the container and the other ROS distributed nodes happen through the host network. Note that simply reading the stream output by a low-cost IP plug & play camera lightens up the already computationally heavy image processing step.

Mapping laser data to the image plane

The information published by the laser range finder is processed within a specific ROS node and projected on the image plane, exploiting the estimated calibration parameters. This same node is also in charge of comparing such information with the messages published on the topic related to the human-obstacles bounding boxes coordinates, in particular, data are synchronously gathered from topics, by exploiting the *ApproximateTimeSynchronizer* class that is included in the `message_filters` ROS package. If there is a correspondence among pixel coordinates, the relative points of interest are translated to be expressed in the global map reference frame, using the computed rigid transformation between the sensors and the robot model reference frames. All the transformations between different reference frames are made available thanks to the `tf` ROS publishing system.

Detected humans as virtual obstacles

With the aim of developing the desired safe reaction to human obstacles, it has been implemented the Timed-Elastic Bands (TEB) local planner [65] which generates three elastic bands attracting the robot towards a goal position while repelling it from obstacles. Its planning mechanism is based on a timed cost function and it chooses the shortest feasible path, taking into account dynamic obstacles and vehicle constraints [70]. Also, it allows to define custom virtual obstacles by specifying their location and shape. This local planner can be integrated with the global planner provided by the ROS navigation package.

Thereby, all points falling into each person's bounding box are published in the occupancy map as circular obstacles with a customized radius. Note that the detected human horizontal extension influences the obstacle shape. This mechanism temporarily enlarges the inflation radius of the map parameter for the region of interest. In this way, the presence of a human moving in the shared environment introduces an additional constraint for the path re-planning mechanism of the mobile robot, ensuring safety between humans and robots.

Figure 6.4 depicts an overview of the whole process. First of all, the YOLO C++ software has been edited in order to choose all the "person" labelled objects and append them to a *.txt* file, whose content is fed to a ROS node for its translation into ROS messages. Then, another node is in charge of filtering synchronized data to identify laser points falling into the image pixel ranges corresponding to humans. Finally, virtual obstacles are published according to the human location on the map, with a radius that is added to the *rviz* inflation one.

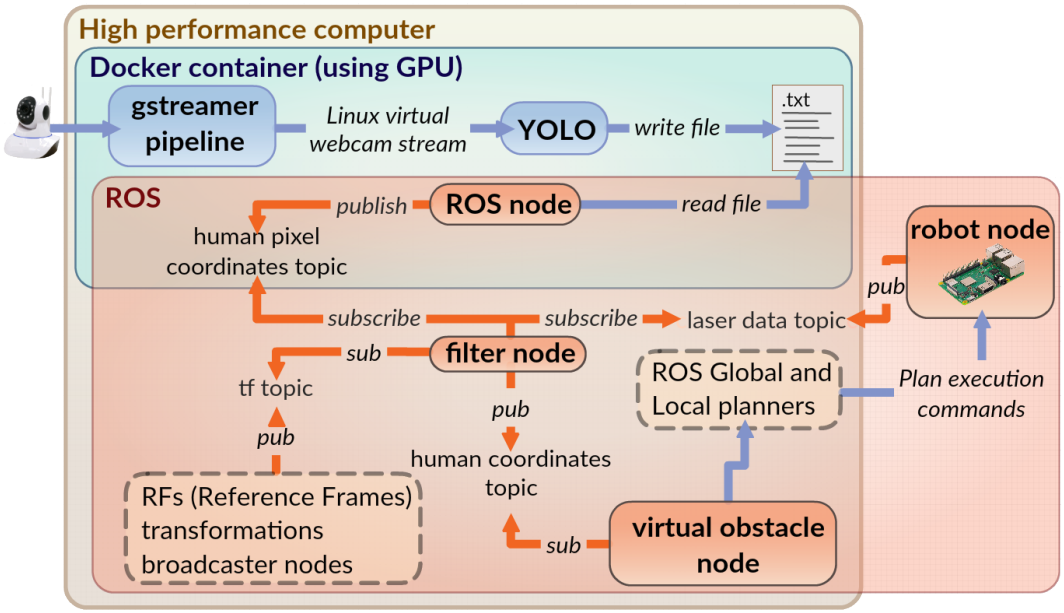


Fig. 6.4 Sketch of the process workflow.

6.4 Experimental results

This section presents the results obtained by employing the proposed sensor data fusion algorithm to enable a safe mobile robot reaction in the presence of humans. In particular. the sensor information is combined with the YOLO object recognition system, providing contextual information to the path planner. Furthermore, with the aim of demonstrating the achieved results, a sample working scenario is considered in the video available in [155].

The main features of our algorithm are highlighted in some salient points of the video that can be summarized afterwards.

The first experimental test is shown in Figure 6.5. It can be seen in the scenario in which a human is detected and its location is reinforced by the publication of a set of virtual obstacles, appearing as red squared markers on the map. Note that the human coordinates are perceived by the laser and inflated by ROS. The path computed by the TEB local planner conservatively remains outside of the inflated area imposed by ROS since the obstacle position is computed not only as detected by the laser range finder, but as the result of the latter and the YOLO processed information.

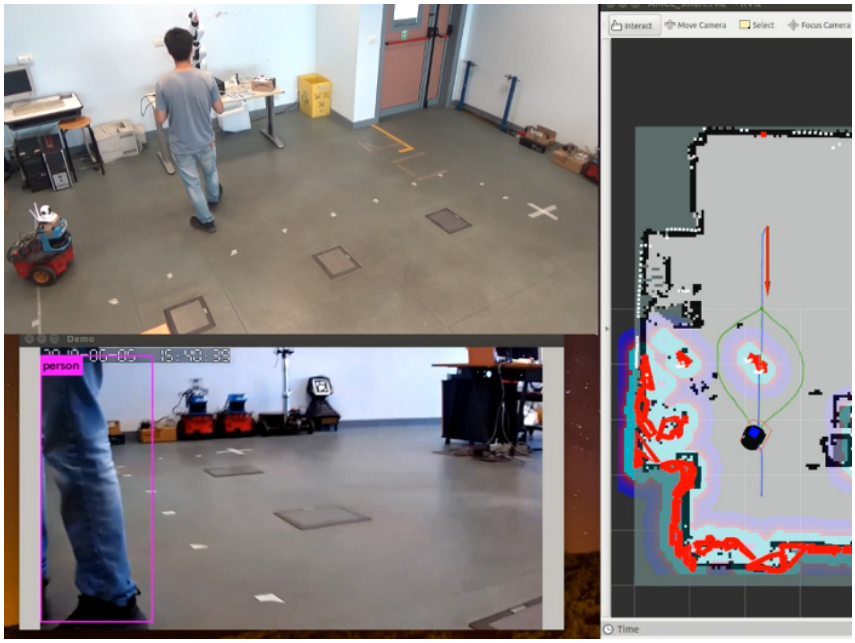


Fig. 6.5 Human detection and relative virtual obstacle publication at 02:37.

The second testing scenario, as depicted in Figure 6.6, aims at illustrating how the virtual obstacle publication depends on the size of the bounding box detected during the image processing stage. It can be noted that the robot's perception system is able to take into account the dynamic behaviour of a person and replan the path accordingly.

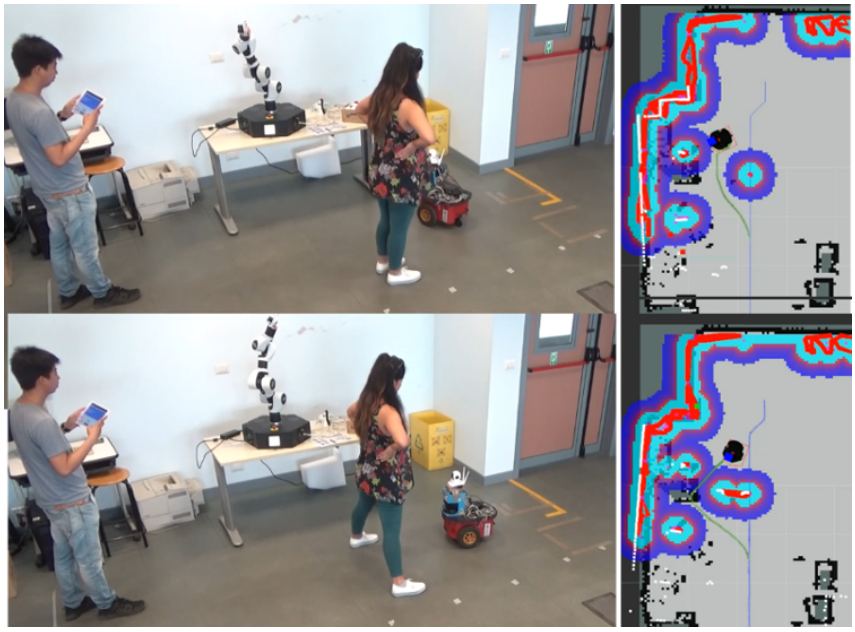


Fig. 6.6 The local planned path depends on the virtual obstacle, which covers the whole detected person bounding box.

The last experimental testing of the sensor data fusion algorithm is to demonstrate its capability to simultaneously identify and consequently publish multiple human obstacles. This behaviour is shown in Figure 6.7, where the algorithm is capable to deal with two human obstacles at the same time. As a result, the replanning mechanism of the mobile robot's local planner is influenced as well.

It is worth mentioning that the overall execution time of the whole system might depend on many factors, such as:

- The chosen local path planner. Different local path planners may have alternative methods to deal with dynamic obstacles, whose cost functions do a trade-off between quick reaction and accuracy.
- The adopted algorithms for implementing sensor data fusion and object detection.

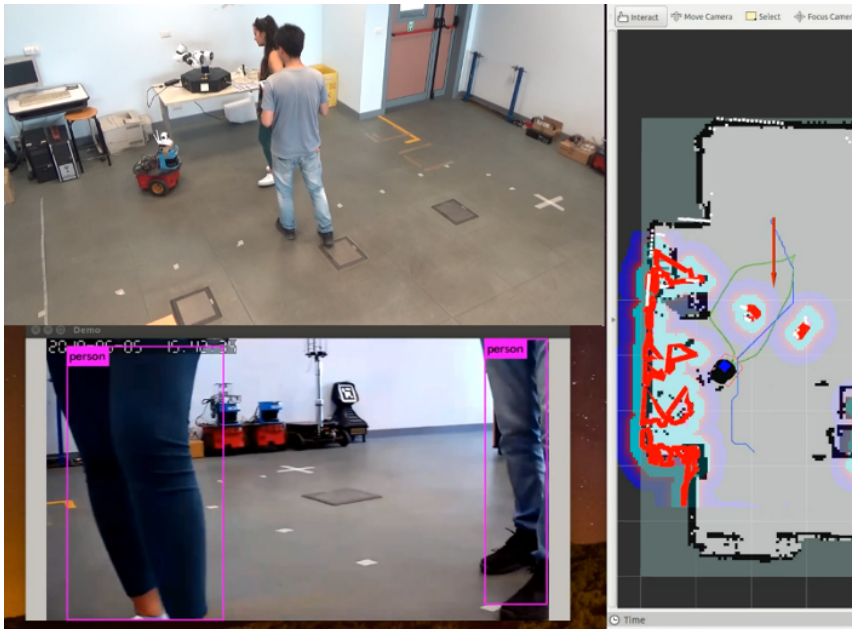


Fig. 6.7 Several human obstacles can be simultaneously detected fostering scalability (04:32).

- The computational capability of the computer executing the sensor data fusion code, the reaction of the robot might fall within the standard definition of soft real-time processes or not.

6.5 Discussion

This chapter presented an overall affordable and accessible sensor data fusion algorithm for mobile robots to ensure safety in an industrial environment where humans and robots coexist. The main contribution of this research work is to show how safe human detection can be achieved by the synergistic use of different sensors, even if they are low-cost, within an overall HW/SW architecture enabling information sharing within a fleet of AMRs, acting as meta-sensors to support the operating of standard AGVs. For the time being, the application is feasible when the relative motion between the robot and the human is slow, so that the robot has enough time to react and re-plan the trajectory, avoiding thus the human operator that is moving close to the mobile robot.

Due to the fact that the adopted calibration approach needs a sufficient amount of data in order to be accurate enough, it has been performed offline. Nevertheless, this

can be considered a relatively easier approach since the process has to be executed once, but it obviously relies on the assumption that the involved sensors are well and definitely fixed in place. Improved solutions would be given by the development of an online calibration procedure, or the application of deep-learning algorithms for data association that may compensate slight errors due to sensor displacement.

As future work, one of the objectives will be also to consider the proposed algorithm for the overall perception system composed of multiple mobile robots as specified in the original scenario, where all mobile platforms (AGVs and AMRs) share the relevant information about their surroundings. Furthermore, even though the current solution permits to spread relevant information to other mobile robots so as to influence their local behaviour, an ideal implementation would publish the detected humans as cost-map obstacles allowing for a more aware and efficient global plan computation.

Chapter 7

Human-Robot Perception in Industrial Environments

An overview and the analysis of the existing methodologies for human-robot perception in industrial environments are presented in this chapter, summarizing the main contributions presented in [11]. There are several methodologies and sensors combination to deal with human perception in shared working places and they depend on the application requirements and the robot type used in a specific environment. A further instance of this was presented in Chapter 6, in which a camera-laser sensor data fusion algorithm was presented, where the calibration method was employed to merge the data from those sensors and then processed using artificial intelligence algorithms to classify the obstacles when the mobile robot is navigating in an environment shared with human operators.

The structure of the chapter is organized as follows: Section 7.1 introduces some background information related to the research work. Then, Section 7.2 investigates how the main types of robotic systems perceive the human presence and how this is handled using different sensors; a brief overview of human and environment perception in the industrial context is also provided. Section 7.3 illustrates a proof of concept, developed to investigate possible future improvements for collaborative robotic applications, thanks to enhanced capabilities of human perception and interaction. Last but not least, Section 7.4 draws conclusions and sketches future trends.

7.1 Background

The perception features of robots will gain ever-greater importance in the next smart factories. The robot has been gaining an increasingly important role within factories and warehouses for decades, recently witnessing a boost in its use as a support to human workers, as a team member or as a flexible part of manufacturing processes. Autonomous and collaborative robots will be increasingly involved in operations requiring a shared working space with human actors. Most of the operations will have to be done by avoiding obstacles, collaboratively working with human beings, as well as autonomously locating and identifying the parts to be worked or moved.

This perspective of a collaborative environment between humans and robots in production settings goes beyond the concept of Cyber-Physical Production System (CPPS) [156, 157]. In CPPSs, a smart production plant is a Cyber-Physical System (CPS) integrating cyber aspects such as computation, communication, control, and networking technologies into the underlying physical system. A CPS can quickly react and adapt to market changes by negotiating production resources as in [158], or using some intelligent reasoning tools as suggested in [159], but humans are generally considered to be intruders into the automated tasks. CPSs should be able to reprogram their activities reacting to the presence of humans or other mobile systems, but generally, they do not interact or collaborate actively with them. This leads to conceive the next future evolution of the CPSs towards Cyber-Physical Human Systems (CPHSs) [160, 161], where the control, communication and automation technologies, physical plants and humans must pursue a common goal. The latter fact opens up new challenges with respect to the conventional interpretation of CPSs, where humans were very often considered to be independent passive entities that operate, use or consume the CPS resources. This also motivates the research for new solutions for developing trustworthy, safe, and efficient Human-Robot (HR) perception to achieve an enhanced HR Interaction (HRI) in collaborative work environments, thus allowing the development of CPHSs.

In the context of CPHS, the adoption of HR teaming is still hindered by the lack of clear guidelines for safety, interfaces and design methods [162]; the HR Perception (HRP) step and its requirements are then fundamental to successfully implement the paradigm of CPHSs as the core of the Factory of the Future (FoF). The digitalization of the whole manufacturing system requires managing plenty of heterogeneous sensors able to share and fuse the information provided by other

sensors, as well as increasing capabilities not limited to detection. Indeed, a high technological level is needed by the sensors, which have not only to read the data and reduce the noise effects, but also to process them (edge computing) to enable predictive maintenance operations [163]. In such a scenario, the availability of sensors for the HRP becomes a key issue for managing the operations of HRI in the FoF.

The choice of the type of perceptive system to be used is highly related to the following requirements: (i) the task to be fulfilled, (ii) the level of autonomy to be guaranteed, and (iii) the kind of HRI that must be established. It is worth mentioning that there exist three types of HRI in the industrial scenario [164]:

- HR Coexistence: In this scenario, humans and robots share the same working space, but perform tasks with different aims. Here, the human is perceived as a generic obstacle to be avoided, and the robot's actions are limited to collision avoidance only.
- HR Cooperation: In this context, humans and robots perform different tasks but with the same objectives that should be fulfilled simultaneously in terms of time and space. The collision avoidance algorithm deployed in the robot includes human detection techniques, so the human obstacle is distinguished from a generic object.
- HR Collaboration (HRC): Within this framework, direct interaction is established between the human operator and the robot while executing complex operations. This can be achieved either by coordinated physical contact or by contactless actions, such as speech or intention recognition.

In this chapter, several sensors and perception techniques adopted for HRI applications are analyzed, specifically for robot guidance and collision avoidance for all the main types of robotic systems commonly used in industry, such as fixed-base manipulators, collaborative robots (cobots), mobile robots and mobile manipulators. The analysis investigates how these robotic systems perceive the presence of human operators and how they react during cooperative and collaborative applications.

Various applications that strongly rely on HRP to achieve HRC are reviewed with a particular focus on the handled type of data, so as to motivate the need to fuse the information that comes from different sensors to guarantee an efficient and

safe HRI, as well as the specific requirements of the perception tasks, for instance, the perception range, the safety issues and the environmental influences. Particular attention is devoted to vision and distance sensors, which are the most used for human perception in different types of robotic systems. Monocular RGB (Red Green Blue), stereo, RGB-D (Red Green Blue-Depth), and more recent event-based cameras stand out among the vision sensors, whereas the most used distance sensors are based on light scan technology, such as the LIDAR. The aim of this chapter is to provide useful information to accomplish several robotic tasks in HR collaborative industrial environments. Furthermore, a quite complete overview of the different solutions proposed in the literature and of the modalities with which they have been applied to different types of robotic systems are presented. The carried-out analysis presents detailed and aggregated information about the various types of sensors adopted to handle the presence of human operators in several industrial scenarios, as well as about the sensors and algorithms combinations that seem to offer the best performance.

Moreover, this chapter is completed by the introduction of a proof of concept for a possible collaborative robotic application based on enhanced capabilities of human perception and interaction. In particular, it proposes a collaborative extension of a framework for autonomous mobile robots in an industrial space shared with human operators. In the basic version of this architecture, the agents can move to guarantee the safety of the human operators encountered during the motion, but without any type of collaboration with them. In the envisaged extension, the agents act as collaborative mobile robots, able to recognize trained operators and perform collaborative operations, directly requested by the operators through a pre-defined sequence of movements, properly interpreted by the robots.

7.2 Robotic Systems and Human-Robot Perception

In this section, several types of robotic systems commonly used in industry, namely, fixed-base manipulators, cobots, mobile robots and mobile manipulators, are considered with the aim of illustrating how they perceive and react to the presence of human operators or static obstacles in the industrial workplaces, for cooperative and collaborative applications.

7.2.1 Fixed-base manipulators and cobots

HRC in the context of fixed-base robots is a topic of great interest in recent research. In particular, those applications where humans and robots coexist in the same working environment can be divided into two categories:

- The robot is required to be fully aware of the human presence and the environment.
- The robot has sufficient capabilities to safely manage the shared workplaces, so as to guarantee injury avoidance toward humans during the robot's motion.

In the first scenario, different types of sensors are used to track humans or obstacles in the manipulator workspace able to scan and provide a complete 3D model of the environment. In such a way, it is possible to monitor the distance between the robot and any object in the workspace, whereas a high-level controller can re-plan the robot's trajectory to avoid collisions or stop the system, if necessary. In the second case, collisions are generally detected by estimating the dynamic properties of the robot, together with the information coming from the proprioceptive sensors already included in industrial robots. The robot's motion is then re-planned and controlled to limit the contact forces so that eventual non-intentional collisions with humans or obstacles are not critical.

This section investigates the improvements achieved in recent HR applications mainly using exteroceptive sensors, without excluding the available proprioceptive-based methods to estimate the robot contact forces through the knowledge of the robot dynamics. Without the use of exteroceptive sensors, the robot has no perception of the 3D external environment, and as a consequence, the robot is unaware of the presence of humans and obstacles. In order to keep a safe HRI, the interaction contact forces are kept limited. In other words, by employing proprioceptive sensors only, collisions are not avoided, but they do not represent a threat to humans. Currently, commercial cobots mainly adopt methodologies that limit contact forces and avoid the use of vision sensors. Cobots are able to work closely with human workers thanks to their smooth surfaces and operating velocities that are adequate for collaborative operations. Nevertheless, cobots are usually not able to perceive the presence of humans but can safely interact with them only by contact.

According to ISO/TS 15066:2016 [165, 166], some solutions have been recently proposed to also make traditional manipulators able to establish some kind of collaboration with humans. The method proposed in [167] limits the force for a traditional industrial manipulator and detects collisions without the use of external sensors. In particular, it implements time-invariant dynamic models and supervised feed-forward input-delay neural networks on signal processing to estimate the current signals required for a given robot motion. The predicted current signals are then compared with the ones that are effectively absorbed by the motor, which are continuously measured by the robot controller. In this context, a collision is then detected when the current required by the manipulator is greater than the predicted one. An approach proposed in [168], avoids the use of external force sensors, which generally are not present in standard manipulators, and exploits the dynamic model of the robot in both dynamic and quasi-static modes to detect the external forces.

To overcome the existing limitations in HR applications, the use of exteroceptive sensors such as vision sensors can be a valid solution, even if there are still problems related to accuracy and repetitiveness. Indeed, the performance of vision sensors is highly affected by environmental conditions, e.g., exposure, brightness, reflectiveness, etc. Nonetheless, such sensors are the most suitable to enhance the robot's environmental awareness, so as to avoid obstacles and re-plan better trajectories. For this reason, vision sensors are generally used to check the workspace, for detecting different objects and enabling humans' safety.

The most deployed vision systems for HR collaborative applications are stereo cameras, RGB-D cameras, proximity sensors or laser scanners. These kinds of sensors allow obstacle tracking data and their usability can be extended to many applications, e.g., estimating human intentions, creating models of the 3D environment, calculating distances between the robot and the obstacle, integrating data coming from the virtual and real world to test an application in simulation, etc. Hereafter, the current state of the art is reviewed, with a main focus on the three fundamental aspects of the application development:

- **Sensor type**: the sensor output depends on the sensor technology. Hence, the algorithms used to filter and process the information depend on the data type.
- **Methodology to detect obstacles in the scene**: the chosen methodology depends on the sensor type but also on how is positioned. The sensor can be

installed somewhere in the environment to monitor the entire scene or can be mounted on the robot arm. In the first case, it is required to distinguish humans and obstacles from the manipulator, otherwise, the robotic system can identify itself as an obstacle. In the second case, the the sensor position is not fixed and must be estimated to recreate the 3D scene.

- **Anti-collision policy**: once the obstacle is detected on the robot path, and the risk of a possible collision is determined, the robot can be stopped by providing some warning at first (e.g., sounding an alarm) or its trajectory can be automatically re-planned to avoid the obstacle.

There are several works in the literature that propose the use of a Kinect RGB-D sensor, which provides RGB and depth space images to reconstruct the 3D environment. In [169], the Kinect sensor is used to generate 3D point cloud data and to study the collision prediction of a dual-arm robot (Baxter). To detect obstacles in the scene and prevent self-collision avoidance, the authors proposed a self-identification method based on the over-segmentation approach using the forward kinematic model of the robot. To improve the processing speed, a region of interest is determined based on the skeleton of the robot, then a collision prediction algorithm estimates the collision parameters in real-time for trajectory re-planning.

Flacco et al. [170] presented a fast method to calculate the distance between several points and moving obstacles (e.g., between robot joints and a human) in depth space with multiple depth cameras (Kinect). The robot kinematics is used to identify the point cloud data representing the robot itself to eliminate it from the scene. The distance is used to generate repulsive vectors that control the robot while executing a motion task, thus achieving a collision avoidance application. Also, in [171], the Kinect sensor was used to add data coming from real obstacles in a virtual scene, where the robot is modelled. This approach aims at testing re-planning algorithms and HR interaction in safe conditions, simulating possible scenarios where humans and robots must collaborate. However, in all these works the Kinect sensor shows its limits in terms of accuracy and reliability. In [172], a method was proposed to improve the accuracy of the Kinect sensor merging real and virtual world information; in particular, some accuracy problems are overcome using a skeletal tracking approach. A highly detailed avatar is created to represent human behaviour in the 3D scene, consisting of thousands of polygons. Then, the Kinect sensor is used as an input device for the skeletal tracking and positioning of the user. Nevertheless,

there are different types of low-cost RGB-D cameras; useful information regarding the choice among the most used in research can be found in [173], where sensor performance is compared in an agriculture application.

The use of a simple RGB camera to detect obstacles was proposed in [174], in a case study in which an industrial manipulator is used. The robotic system is provided with smart sensing capabilities, such as vision and adaptive reasoning, for real-time collision avoidance and online path planning in dynamically changing environments. The machine vision module, composed of low-cost RGB cameras, uses a colour detection approach based on the hue saturation value space to make the robot aware of environmental changes. This approach allows the detection and localization of a randomly moving obstacle; the path correction to avoid collision is then determined by exploiting an adaptive path planning module along with a dedicated robot control module. It must be underlined that using only a standard RGB camera, the obstacles detection can be performed in 2D assuming a constant height along the third direction. This solution may be valid for manipulators employed for simple pick-and-place tasks and it can be executed in a fast-working cycle.

A different solution, which integrates sensors used for virtual world interaction, is proposed in the field of robotics surgery, where any possible collision between the robot and the medical staff is considered to be critical [175], but some of its characteristics could be exploited in different contexts, such as the manufacturing one, for applications requiring a strict HR collaboration. The HTC VIVE PRO controllers are used as an Internet of Things technology to measure the distance between surgeons and the robot. When the distances between humans and the robot, measured through the smart controllers, become critical, a virtual force is applied to the manipulator to move the robot elbow in a spare workspace. This avoids the direct hands-on contact of the surgical robot arm by applying the virtual force to move the swivel angle of the KUKA iiwa. Due to the kinematic redundancy of the manipulator, a swivel motion with the robot elbow can be performed without moving the robot tool pose avoiding compromising the surgical intervention. In [176], the same authors previously investigated the cartesian compliance strategy that involves online trajectory planning to avoid violation of some defined constraints.

A novel sensor proposed in [177], which consists of skins with proximity sensors mounted on the robot's outer shell, provides an interesting solution to occlusion-free and low-latency perception. The collision avoidance algorithms, which make

extensive use of these properties for fast-reacting motions, have not yet been fully investigated in this work. A collision avoidance algorithm for proximity sensing skins is proposed as a first solution by formulating a quadratic optimization problem. The authors point out that compared with common repulsive force methods, the algorithm confines the approach velocity to obstacles and keeps motions pointing away from obstacles unrestricted.

It is worth noting that a good HRC requires good HR interfaces and the possibility for the human operator to easily establish some kind of communication with the collaborative robot [178]. A proper use of adequate sensors is fundamental to this aim. Cameras can be employed, but better results can be achieved by integrating also specific sensors such as the Leap Motion, which can be used to recognize coded gestures of the operator as input commands to the robotic systems (e.g., as for the teleoperated robotic arm in [179]), but also to enhance the perception capabilities provided by cameras, as in [180]. Here, a multi-source heterogeneous vision perception framework is proposed to acquire information about the human workers in various conditions and in the working environment during HRC tasks in manufacturing. The proposed system includes RGB-D cameras (i.e., Kinect sensors), located around the working area to produce 3D point cloud data, and Leap Motion sensors on the workbench to track the worker's hands. In this way, a wide and clear perception is achieved of both the working area and the worker.

In [181], a system composed by five Inertial Measurement Unit (IMU) sensors is used to recognize human gestures. The IMU sensors are distributed in the upper part of the operator's body, along with an ultra-wide band positioning system. The latter activates the collaborative mode when the human operator is in close proximity to the robot. Static and dynamic gestures used to command the robot are processed and classified by an Artificial Neural Network (ANN). A similar work related to gestures in the industrial context is presented in [182], in which IMUs and a stereophotogrammetric system are used to track and analyze the human upper body motions, in particular when he/she picks and places several objects at different heights. The gestures sequences are collected in a database, and can be used to optimize the robot trajectories and guarantee the safety of the human operator.

A sensor data fusion algorithm is proposed in [183] to estimate and predict the human operator occupancy within the robot workspace. The algorithm merges the information coming from two different depth sensors, a Microsoft Kinect and an

ASUS Xtion, defining a set of swept volumes that represents the space occupied by the human. In this way, the motion of the robot can be re-planned to be compliant with the safety constraints, thus avoiding any collision with the human operator.

More insights into hand gesture recognition by means of the Leap Motion and other solutions for HR interaction are provided in the following subsections.

7.2.2 Mobile Robots

Mobile robotics is gaining ever-increasing importance within the industrial context. Indeed, Industrial Mobile Robots (IMRs) represent essential elements of the present and future production line and logistics workspaces (Figure 7.1). Specifically, Autonomous Mobile Robots (AMRs) allow the improvement of flexibility of the working setup, since they get rid of the path constraints of classical Automated Guided Vehicles (AGVs). Spatial and temporal flexibility, when considering production plants, can improve productivity and reduce overall downtime, for example, when the production sequence configuration must be changed. Indeed, flexibility requirements (dictated by recent market demands for custom products) inevitably affect the current and future production line design [14].

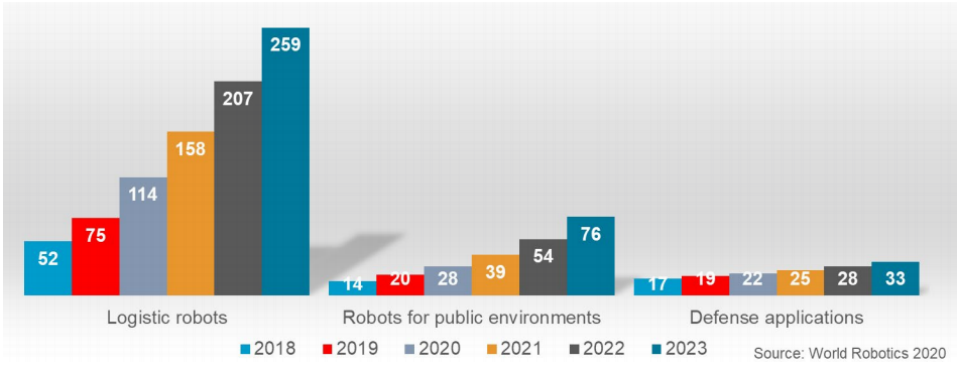


Fig. 7.1 Service robots for professional use. Top 3 applications unit sales 2018 and 2019, potential development 2020–2023 (thousands of units) [1].

It is clear that the optimization of processes has a pivotal role for the overall efficiency (e.g., productivity, energy consumption) of a working setup, as discussed in Section 7.1 with reference to the well-established CPS concept. IMR perception of its surroundings thus acquires relevance for achieving optimal integration with other CPS elements, going beyond the basic role in the localization of the platform during

navigation. In particular, autonomous navigation of AMRs introduced a further need for effective HRP approaches. Indeed, for what concerns cooperative operations, effectiveness may be undermined by the perception that human operators have of moving autonomous agents. Conversely, the mobile base task execution could be slowed down by the ill-managed perception of humans. As a matter of fact, the predefinition of AGV motion paths guarantees predictability in opposition to the AMRs motions, which are often hard for a human operator to interpret. The perception systems of traditional AGVs [184] have undergone heavy changes [185, 186] to achieve navigation autonomy and advanced perception of the environment and humans in industrial scenarios [132], favoring the investigation of real-time approaches [187]. Moreover, due to the gradual and now extensive use of fixed-base collaborative robots along the production line, the implementation of safe collaborative operations using IMRs has been attracting a lot of interest. Industrial mobile platforms then need to elevate their perception level from a merely informative approach to a semantic interpretation of the robot surroundings.

Despite the advanced perception of humans seems to be an emerging topic within the industrial context, it has already been widely explored and adopted in other fields, from assistive service robotics to agricultural ones, where robotics plays a significant role in the process chain. In fact, it is interesting and relevant to report some of the approaches to HRP developed in these fields, since it is not unlikely that the FoF will implement similar or comparable approaches on intelligent IMRs. In [188], a non-intrusive solution to robot-aware navigation is presented, which lets the user preferences determine the robot behaviour in a domestic workspace sectioned in virtual areas. In [189], the human and the mobile robot share a common task, since the robot is teleoperated by an operator, whose visible 360-degree scene is enriched by interactive elements drawing the attention to information-rich areas; a 360-degree camera is exploited, and its frames are processed using the You Only Look Once (YOLO) Convolutional Neural Network (CNN)-based framework [190]. In this case, goal achievement is common, and the perception of the human operator and the robot somehow enhances each other.

Also, in [191] teleoperation is implemented, using a hybrid shared control scheme for HRC. The operator sends commands to a remote mobile robot using an electromyography (EMG) signal sensor to reflect muscle activation; the human partner is provided with a haptic device, which receives force feedback to inform about the existence of an obstacle. The work presented in [192] aims at highlighting challeng-

ing natural interactions between a mobile robot and a group of human participants sharing a workspace in a controlled laboratory environment, demonstrating that humans follow less jerky and irregular paths when navigating around one autonomous navigation condition than around a teleoperated robot. The experiments are performed on autonomous mobile robots using optimal reciprocal collision avoidance, social momentum and teleoperation as navigation strategies. In [193], an approach named RObot Perceptual Adaptation (ROPA) is proposed. This algorithm learns a dynamical fusion of multi-sensory perception data, capable of adapting to continuous short-term and long-term environmental changes; a special focus is set on human detection, based upon different types of features extracted from colour and depth sensors placed on the mobile robot, with the aim of achieving long-term human teammate following. A structured light camera is used for colour-depth data and a digital luminosity sensor for luminosity data. Similarly, in [194], the authors introduced a representation learning approach that learns a scalable long-term representation model, for scene matching. The features of multiple scene templates are learned and used to select, in an adaptable way, the most characteristic subset of templates to build the representation model for the current surrounding environment. The latter procedure is performed with the aim of implementing long-term delivery of information in collaborative HRP applications, taking advantage of Augmented Reality (AR). Furthermore, what seems clear from works reviewed within the agricultural field, concerning collaborative applications and relative perception between humans and robots, is the focus on safety without leaving out the comfort of the interaction [195, 196]. The work presented in [197] proposes a planning model based on RNNs (Recurrent Neural Networks) and image quality assessment, to improve mobile robot motion in the context of crowds. Acquired images are pre-processed by exploiting OpenCV (Open Computer Vision) calibration tools and then the background noise is filtered out using the designed RNN-based visual quality evaluation. Additionally, concerning the assistance service robotics context, the bidirectional meaning of perception is particularly evident, since the robot should be perceived by users as naturally as possible, and the robot itself must have capabilities of intention recognition to be actually of some utility to the human counterpart, e.g., in Sit-To-Stand assistance [198]. Moreover, the SMOOTH robot project, presented in [199], provides an example of adaptive sensory fusion computed via a single multi-sensory neuron model with learning, to boost perception of human capabilities of a welfare robot. The robot is equipped with a front safety laser scanner and

two cameras, one front and one back facing. Finally, the survey presented in [200] highlights the importance of data fusion to enhance the perception capability of mobile robots. The reviewed works consider data coming from multiple sensors (e.g., LIDAR, stereo/depth and RGB monocular cameras) to obtain the best data for the tasks at hand, which in this case are autonomous navigation tasks such as mapping, obstacle detection and avoidance or localization.

Given these example approaches, it is easy to envision how they could greatly impact the emerging HRP research in the industrial context. Many algorithms are being developed with the aim of being ideally applicable in any context involving humans and robots. The need for a unified framework to enable Social-Aware Navigation (SAN) is stressed in [201], where the authors propose a novel approach for an autonomously sensed interaction context that can compute and execute human-friendly trajectories. They consider several contexts and implement an intent recognition feature at the local planning layer.

For what concerns the industrial logistics context, the authors of [202] propose a range finder-based SAN system to implement collaborative assembly lines with a special emphasis on human-to-robot comfort, considering the theory of proxemics. A cost function is assigned both to assembly stations and operators to affect the cost map for the mobile robot navigation. In [203], a human-aware navigation framework is proposed, to work within logistics warehouses. The simulated mobile robot is equipped with a laser scanner and an RGB-D camera to detect a person and estimate the pose to consider it as a special type of obstacle and avoid it accordingly. The proposed strategy is made up of 2-steps: (i) the use of information related to the depth for clustering and identifying 3D boxes that are likely to enclose human obstacles, then (ii) the computation of a confidence index for human presence based on the RGB data. Instead, the approaches proposed in [204] aim at demonstrating the integration of AR as an enabler for enhanced perception-based interactions along assembly Manufacturing Execution Systems (MES). The authors propose an application involving mixed reality smartglasses for AR implementation for collaboration with a cobot, and a path visualization application for humans working with AGVs, using an AR computing platform. Another work proposes a solution to HRI using (i) gesture control and eye tracking technologies for the robot to interpret human intentions, and (ii) a pocket beamer to make robot information interpretable by the human operator [205]. Finally, in [206] the authors propose an HR skill transfer system: a mobile robot is instructed to follow a trajectory previously demonstrated by

a human teacher wearing a motion capturing device, an IMU in this case. A Kinect sensor is used for recording the trajectory data, used to model a nonlinear system called a Dynamic Motion Primitive. Then, exploiting multi-modal sensor fusion, the pose and velocity of the human teacher undergo a correction process and a novel nonlinear model predictive control method is proposed for motion control.

7.2.3 Mobile Manipulators

Manipulators have been employed in many applications, increasing the efficiency in the industrial production line. However, these are usually located in fixed positions along the line, which is a limitation for some applications that need to cover large working spaces, as in the automotive or aerospace industry. To overcome this problem, it is possible to rely on mobile manipulation. A manipulator attached to a mobile platform improves the flexibility for many tasks, since the redundancy offered by a mobile manipulator allows the planning of human-like motions while avoiding singularity configurations. Due to the mobility advantages, it is also used for intralogistics and service robotics applications [207, 208].

Most of the mobile manipulators available on the market consist of a combination of a collaborative lightweight manipulator and a mobile platform. The mobile platform in these cases may be collaborative or not. It is worth highlighting that currently there are no safety standards specific to these hybrid systems so, in order to be compliant with collaborative operations and safe constraints with mobile manipulators, a combination of two or more standards should be considered, e.g., ISO/TS 15066 [165] and/or ISO 10218-1 [209] for manipulators, and ISO 3691-4 [210] for mobile robots.

Since the collaborative manipulator itself was designed for collaborative applications, it can react to the physical contact of the human operator with no harm. However, to allow the robot to perceive better its environment, and therefore improve the decision-making process for the motion planning needed for a specific task, other sensors may be integrated to the robot. The way the robot may sense its surroundings and the way it reacts to the human actions strongly depend on the application. In fact, there are applications in which vision sensors are widely used to emulate the decision-making procedure based on the human vision.

For example, the mobile manipulator proposed in [211] is designed for HRC tasks, in which object detection and manipulation are considered to be critical skills. According to the authors, using an RGB-D camera is more robust than using stereo vision cameras, since the latter ones only rely on image features. The images and videos coming from RGB-D cameras are also useful for either (i) configuring the motion constraints based on the human presence, differentiating human-type obstacles from the generic ones, or (ii) predicting the human activity, so the robot can react accordingly to the operator action [212]. A sensor system that provides reliable 2½D data for monitoring the working space is presented in [213]. The system elaborates data coming from three pairs of grayscale stereo vision cameras and a Time-of-Flight camera that monitors the motion of the human operator collaborating with the manipulator. The area monitored by the sensor system corresponds to the safety zone, in which specific actions of the robot are enabled when the hand of the human operator is close to the manipulator tool.

The mobile manipulator proposed in [214] uses two sensors that perceive the environment: an RFID sensor that lets the robot know where the objects are in the space and an RGB-D camera that identifies tags with unique IDs, which contain semantic information and properties of the world entities. The paper did not specify if the robot is working with a human or not, but the interesting fact is that the robot can learn from experience, and each time it must perform an action, the motion planning comes from experiential knowledge and the geometric reasoning for doing such task.

To give more information to the robot regarding human intentions or actions, gestures and speech are commonly used for controlling a robot. Nevertheless, hand gestures are preferred over speech, since the industrial environment is often noisy, and verbal communication is difficult [215]. The gesture recognition is performed by analyzing two features from an RGB-D camera: a convolutional representation from deep learning and a contour-based hand feature. This permits the robot to recognize the hand gestures of the human and execute specific commands. Moreover, the same authors proposed alternative methods for human tracking [216], such as applying multi-sensor integration (for instance, mounting low costs laser range finders and camera systems at specific poses) and using laser readings and training the tracking system according to human body patterns. The authors in [217] suggest that a 3D sensing system is important for human detection and for understanding the overall behaviour. In that regard, a redundant sensory system, such as a combination of

2D laser scanners and sensors that reconstruct the environment in 3D using stereo vision, may ensure safety and be compliant with the ISO 10218-1 and ISO/TS 15066 regulations, which are related to safety for collaborative robots. Nevertheless, those standards involve collaborative manipulators, so the safety concerning the mobile platform should be also considered, as discussed in [218] that analyzes the possible hazards of mobile robotic systems in industry and proposes some countermeasures for those risks. Therefore, the use of sensor fusion or artificial intelligence-based methods are suggested, since they increase the coverage of the information from different sensors and overcome safety problems.

A framework referred to as ConcHRC [219], which represents an extended version of the previous FlexHRC framework [220], allows the human operator to interact with several robots simultaneously for carrying out specific tasks. The architecture is composed of three layers: perception, representation and action. In particular, the perception layer elaborates the information related to human activities and object locations in the robot workspace. The overall scene is measured through motion capture sensors, the objects to be manipulated are detected using an RGB-D camera, while the data related to the operator action comes from the inertial sensor of a smartwatch.

A teleoperated mobile manipulator proposed in [221] is controlled according to the posture for the operator's hand. The tracking of the operator's hand is achieved by employing a Leap Motion sensor, in which a Kalman filter is used for the position estimation while the orientation is computed by a particle filter. A similar contactless hand gesture recognition system is presented in [222] for safe HRI. This multi-modal sensor interface uses proximity and gesture sensors, and it can identify real-time hand gestures to control the robot platform. An ANN is used for the recognition of hand gestures.

Other approaches, such as the one presented in [223], can work along with a human through an admittance interface, allowing conjoined action. If the human is not in close proximity, the mobile manipulator can perform its routine work autonomously. In particular, the admittance interface is a mechanical connection from the robot hand to the human wrist and transmit the interaction forces of the human to the robot to perform conjoined movements. When the human needs assistance, it is possible to "call" the robot using the armband that recognizes the gestures of the human operator.

7.2.4 A brief overview of HRP in industry

Presently, most of the sensors used for robotic systems to perceive the environment and the human operators are of vision type. In particular, in the field of human collaboration with manipulators, the most used sensor is the RGB-D camera. Indeed, the use of new types of sensors may require a huge effort to define new algorithms and exploit their characteristics. Moreover, the already developed obstacle detection algorithms would need to be rethought to work with different data types. An interesting new vision sensor is proposed in [224], as Dynamic and Active-pixel VIsion Sensor (DAVIS). This novel sensor seems to have great potential for high-speed robotics and computer vision applications and incorporates a conventional global-shutter camera with event-based sensors in the same pixel array, allowing the combination of their benefits as well: low latency, high temporal resolution, and very high dynamic range. However, for the moment more algorithms should be required to fully exploit the sensor characteristics and cope with its unconventional output, which consists of a stream of asynchronous brightness changes (called "events") and synchronous grayscale frames. In those applications in which the employment of vision sensors is not sufficient or accurate, other kinds of sensors are used instead.

For what concerns IMRs, applications involving human perception mainly exploit laser range finders, to perceive the environment and also humans included. Usually, those sensors are combined with a vision sensor to perform data fusion. The massive use of laser range finders for human-perception goals is expected and justified, as it is a sensor typically present on IMRs both for obvious navigation requirements and for industrial safety guidelines (safety-rated scanners).

In the same way, mobile manipulators exploit predominantly laser sensors for navigation, while vision sensors are mainly used for the manipulator to perceive the human operator, in particular, to have some visual guidance and be able to imitate the movements of the human. Most of the vision sensors used in mobile manipulators are of RGB-D type, since they give accurate information related to the image and depth of the detected object. An alternative way for the robot to perceive human actions is based on the use of an inertial sensor, attached to the human wrist to detect motions, and let the robot to predict and react according to the human movements.

Figure 7.2 gives a visual overview of the most relevant sensors used for HRP depending on the robot type, according to the authors' research. Furthermore, a

summarizing overview of the relevant sensors and methodologies obtained from the described state-of-the-art analysis is available in [11], specifically the figures that aggregate the sources based on the employed sensors types, also carrying information about the used robot type. It is worth noting that by splitting sensors according to their presence on robots or on human operators, it is clear how (based on what has been analyzed) the sensory equipment for HRP are currently mainly positioned on the robot counterpart. Moreover, Table 7.1 aims to enrich the overview presentation and ease the reader consultation, focusing on algorithms to implement HRP.

Fig. 7.2 Relevant sensors for HRP within the industrial context. Vision sensors are highlighted in blue, safety laser scanners rays in red, and wearable sensors in orange.

Although it is true that the provided material is a useful tool for getting a taste of the trending sensors and algorithms in HRP, it should be considered that it is limited to the authors' research and, for this reason, it may not be exhaustive.

Human–robot perception seems to be inevitably linked to robot-human perception: each piece of information on the human worker's behaviour is perceived by a robot, interpreted and transformed into action but, at the same time, human reaction to the presence of a robot is affected by the perception the operator has of the robot itself. An optimal reciprocal perception is however not easy to implement, given the lack of a common ground for cognitive skills among humans and robots, which affects the interaction. The robot not only has to detect the human presence but also to understand the context of collaboration, with the aim of effectively assisting human collaborators to improve the productivity of the overall collaborative system while maintaining safety.

Table 7.1 Most relevant algorithms for HRP applications, grouped according to how the perception is implemented.

Collision Avoidance
[167] Collision prediction using time-invariant models and neural networks on signal processing.
[169] Collision prediction based on over-segmentation using forward kinematic model.
[170] Collision avoidance through generation of repulsive vectors.
[171] Collision avoidance and re-planning algorithms.
[172] Collision avoidance exploiting skeletal tracking and positioning of the user.
[174] Collision avoidance using color detection and allows online path planning.
[175] Collision avoidance through virtual forces applied on the manipulator.
[177] The algorithm imposes velocity limitations only when the motion is in proximity of obstacles.
Aware Navigation
[188] The robot travels in virtual areas defined a-priori by users.
[192] Social momentum, teleoperation and optimal reciprocal collision avoidance are used as navigation strategies.
[197] The planning model is based on RNNs and image quality assessment, to improve mobile robot motion in the context of crowds.
[201] An autonomously sensed interaction context that can compute and execute human-friendly trajectories.
[202] Robot navigation takes into consideration the theory of proxemics to assign values to a cost map.
[203] A confidence index is assigned to each detected human obstacle, enclosed in a 3D box, to avoid it accordingly.

(To be continued)

Environment Representation
[193] The algorithm adapts to continuous short-term and long-term environment changes with focus on human detection, through feature extraction.
[194] Scene matching through a representation learning approach that learns a scalable long-term representation model.
[214] Object localization and tags recognition allow the robot to gather semantic information about the environment.
Recognition of Objects and Behavior
[199] The robot assistance is improved using adaptive sensory fusion.
[179] Teleoperation using coded gestures recognition as input commands.
[180] Simultaneous perception of the working area and operator's hands.
[205] Gesture control and eye tracking technologies are used by the robot to interpret human intentions.
[206] The human motion here is registered and used for skill transfer purposes.
[213] The motion of the collaborating human operator is monitored to enable specific robot actions
[215] Gesture recognition is performed considering a convolutional representation from deep learning and a contour-based hand feature.
[216] Human tracking is implemented and trained according to human body patterns.
[217] Human detection and behavior recognition is implemented exploiting redundancy of sources to reconstruct the environment.
[219] The information related to the human activities and object locations in the robot workspace are used for the approach.
[221] The operator's hand pose is estimated using a Kalman filter and a particle filter.
[222] Real-time hand gesture recognition is implemented using a ANN.

(To be continued)

[181] The gestures used to command the robot are processed and classified by an ANN.
[182] The motion of the human operator's upper body is tracked, with a focus on objects manipulation.
[183] Sensor data fusion algorithm for prediction and estimation of the human occupancy within the robot working area.
Conjoined Action
[189] The 360-degree scene is enriched by interactive elements to improve the teleoperated navigation.
[191] Teleoperated navigation is implemented through a hybrid shared control scheme.
[204] Enhanced perception-based interactions using AR for collaborative operations.
[223] Interaction forces of the human are transmitted from the admittance interface to the robot to perform conjoined movements.

To achieve a comparable level of cognition among humans and robots, multi-modal sensor fusion is the preferred solution for the robot's perception system, either when considering environment perception or human perception, which are obviously interlinked. Data fusion is a key module for autonomous systems to implement perception. Multi-modal data is analyzed at a raw level for fusion processing and then interpreted at a higher level to identify relevant features. Through multi-modal sensor fusion, the the sensor performance can be enhanced by exploiting data fusion. The latter can potentially bring out interesting information which, if single-source data only were considered, would have not emerged. This allows the implementation of a more informed perception of the environment and the humans within it.

Furthermore, what emerged is that safety is one of the factors leading to the choice of algorithms and sensors: their combination must aim at satisfying safety conditions suggested by standards. Moreover, along with safety, also an appropriate interface plays a significant role when developing HRC tasks.

It is clear that where sensor accuracy is lacking, algorithmic complexity aims for compensation, to achieve an overall reliable interaction. As can be easily inferred, the balance between the accuracy of the sensor and algorithm computational effort strongly depends on the available resources and on the application requirements.

7.3 A proof of concept for future applications of perception technologies: Collaborative Sen3Bot

This section describes a POC of a collaborative behaviour implementable upon the Sen3Bot mobile agent, the main element of the Sen3Bot Net [14, 9]. The Sen3Bot is an AMR able to pursue an assigned task within a space shared with human operators. Beyond the standard tasks of AMRs, the Sen3Bots are given the main role of serving as *meta-sensors*: they themselves represent a distributed network of sensors supporting a fleet of traditional industrial AGVs, informing it about the human presence in areas at risk of hazardous situations. In fact, the Sen3Bot has human detection and avoidance capabilities, tested on a real demonstrator [5], allowing for cooperation. First, to lay the groundwork for a more collaborative approach, the Sen3Bot behaviour could be improved by incorporating in the human avoidance algorithm relevant factors coming from the Proxemics Theory, for instance, speed adjustment and direction of approach.

It is undeniable that safety is the main design requirement for IMR sharing the workspace with human workers [225]. Please note that for the proposed POC, the working area subdivision according to a critical level will be considered (as described for the Sen3Bot Net). The definition of such areas was mainly inspired by ANSI safety standard guidelines for driverless vehicles [226], whose corresponding standard in Europe is the ISO 3691-4:2020 *Industrial trucks—Safety requirements and verification—Part 4: Driverless industrial trucks and their systems*,which specifies safety requirements and verification means for driverless industrial trucks, including AGVs and AMRs [210].

Nevertheless, what emerged is that safety standards struggle to keep pace with the fast evolution of collaborative/cooperative AMRs. Current guidelines limit the flexibility that would be potentially achievable with new AMRs. For instance, the fact that AMRs paths do not need to be pre-defined allows the removal of the limitations

given by the physical installation constraints imposed by many traditional AGVs; however, standards suggest that AMRs paths should be marked, which hinders a fast reconfiguration.

To overcome this limitation, an online supervisory planning algorithm for mobile robots was presented in [6]. Given a static map, the mobile agent can follow a virtual safe path in an industrial-like scenario and the trajectory is re-planned when a human operator is in close proximity to the robot in motion.

Even though in this cooperative scenario the human operator is safe during the robot's motion, it is worth observing that the inflation radius assigned to the identified human can possibly be so large that the robot may have problems navigating in narrow spaces. An improvement to this system could be to identify human operators to distinguish trained operators from the general staff and, in the first case, reduce the safety radius surrounding the operator. Further details will be explained hereafter, illustrating a general idea to improve the system and provide collaborative capabilities to a Sen3Bot through the implementation of safe interactions, exploiting and improving HRP capabilities intended as the interpretation of data at a behavioural level.

To illustrate the idea, the following assumptions will be considered:

1. In the light of the envisioned collaborative extension:
    - If the area has the highest criticality level (area of type 1), the AMR must work in cooperative mode, implying that conservative avoidance of humans is implemented.
    - If the area has medium criticality (area of type 2), the AMR can switch between two modes, cooperation or collaboration with human operators.
2. The working space taken into account is an area with a criticality level equal to 2, i.e., a sub-critical area, corresponding to a zone that includes cobots workstations and manual stations where human operators are likely to be present, but expected to be mostly static.
3. In such a sub-critical area, the human operators are assumed to be mainly trained ones, i.e., they are aware that the area is shared with AMRs and know how to interact with them. Operators of this type are identified by a tag, e.g., a QR code, on the front and the back of a wearable leg band.

4. According to the Sen3bot Net rules, if a critical area of type 2 is foreseen to be crossed by an AGV, two Sen3Bots are sent to the scene if the human operator is moving within the environment.

Collaborative extensions of the Sen3Bot Net can then be developed in the scenario described hereafter, under the above-listed assumptions. Two Sen3Bots monitor the scene in a sub-critical area: such a redundancy ensures the operator's safety by taking into account its dynamic behaviour, and understanding if he/she needs some assistance from the mobile agents. In principle, both AMRs can act as collaborative mobile robots. Nevertheless, once on the scene, only one enters a *wait4col* state, i.e., an idle state where the AMR waits for a triggering command from the operator and signals its state through a visual indicator, e.g., a led signalling the current AMR active mode. As the operator is recognized as trained, proximity rules can be less conservative: the robot can reduce the inflation radius around the detected human, since the latter is supposed to be aware of the former's behavior.

If the operator needs assistance from one of the Sen3Bots, he/she will have to perform a pre-defined sequence of movements:

- The operator approaches the AMR in *wait4col* idle mode.
- The operator stops at a fixed distance *Df* in front of the mobile robot letting it read for a given time *Tf* the front leg band tag, which contains relevant information (such as the operator ID).
- If the operator turns around, letting the robot read the back tag for a time *Tb*, then the collaborative mode of the Sen3Bot is activated.

Feasible values for *Df* are in the range of 0.5 m–1 m, while *Tf* and *Tb* can be chosen between 2 and 3 seconds. Please note that the above sequence can trigger several collaborative applications, e.g., Follow-Me, assistance with materials or tools. Furthermore, when the operator executes such a sequence for the second time in front of the same robot, the collaborative mode of the mobile agent is deactivated. In this case, a new task can be re-assigned to the Sen3Bot, for example, monitoring a different area. Figure 7.3 shows the described behavior.

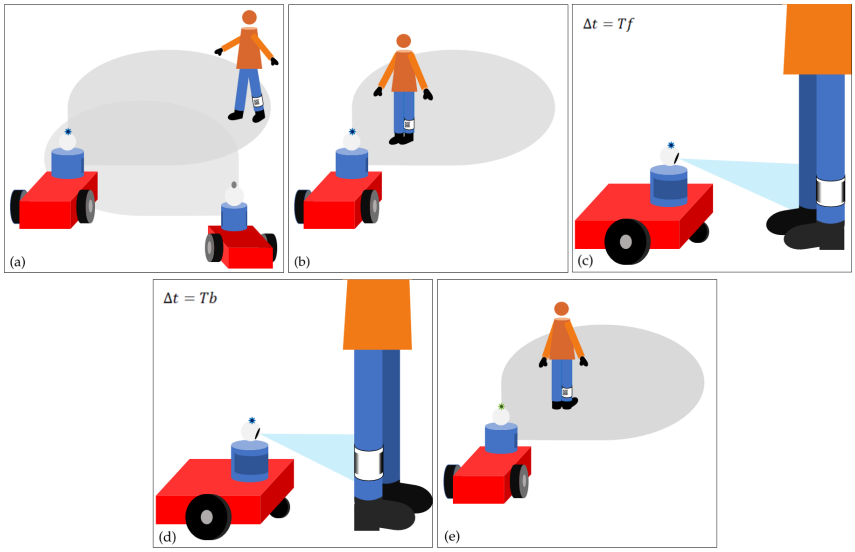


Fig. 7.3 Workflow for activating the collaborative mode of the Sen3bot. The grey area represents the intersection of the vision and laser sensors FOV. (**a**) A human operator within the monitored area of type 2 needs assistance from one of the Sen3Bots. (**b**) Given the led blue colour, the operator identifies the Sen3Bot ready for collaboration. (**c**) The human operator stops at a distance *Df* allowing the robot to scan its front QR code. (**d**) The human operator turns around allowing the robot to scan its back QR code. (**e**) The green indicator light indicates that the mobile robot entered the collaborative mode.

It is worth noting that the Front-Back sequence intends to emulate the human interaction that is likely to take place when two persons are conversing. In this way, the mobile agent can understand this common human intention demonstration, allowing a narrowing of the gap between the different cognitive skills between humans and robots. Please note that the time and distance parameters considered in the sequence take into account a suitable tolerance. A schematic representation of the Sen3Bot modes characterizing the proposed collaborative approach is given in Figure 7.4. In particular, the HUMAN REQUEST flag is set by default to 0, since the robot starts with the cooperative mode and becomes 1 when the conditions of the first Front-Back sequence are valid, allowing the robot to switch to the collaborative mode. The flag is reset to 0 when the human operator performs a valid Front-Back sequence for the second time to the same robot.

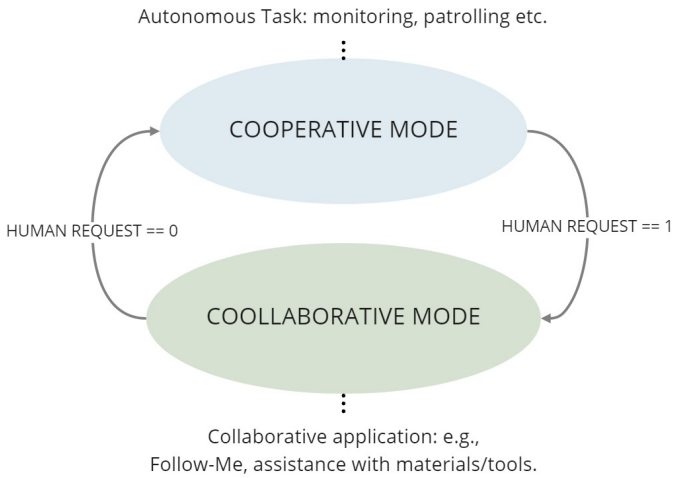


Fig. 7.4 Modes switching schema for a Sen3Bot monitoring an area of type 2, enabled to wait for collaborative task triggering, i.e., with $wait4col == 1$.

The proposed idea could speed up the time required for human operators to be trained to use mobile agents within the industrial workspace, since the interaction with them recalls a human-like behaviour. Even a non-robotic expert could employ the mobile agents, enabled by a user experience similar to the manual guidance of cobots, which streamlines the robot programming and therefore the production line setup. The idea of wearing a leg band with a tag represents a low-cost solution for identifying operators, tracking their activities and thus ensuring safety within the industrial environment. In fact, a tag such as a QR code may contain all the needed information for the robot and the overall system. However, it is not possible to state how robust the solution is, at least, not until a real implementation in a real industrial scenario is tested. Furthermore, running an algorithm that identifies the human operator, tracks his/her location within the map and at the same time filters and processes relevant information coming from the QR code, may need a high-performance system for computing the perception algorithm.

Finally, taking into consideration the current implementation of the Sen3Bot agent, the such envisioned collaborative module could be implemented by taking advantage of tools for tag recognition such as Zbar, for which a ROS (Robot Operating System) wrapper node is available [227]. Also, other tools are present as Open-Source material, mainly based on the OpenCV library, whose ROS compatibility is well-established as well. Please note that this additional collaboration module

would not imply the need for further sensors, since the already available IP camera video stream would allow the application of the aforementioned vision tools.

7.4 Discussion

Human intention recognition is a trending topic within the HRI research field, especially for industrial environments. Human–robot perception and the prediction of possible humans' unsafe conditions will be a fundamental enabler for anticipating human operators' behaviour and needs, to implement proactive collaboration among humans and robots. In advanced manufacturing plants, many applications require HRC operations, where humans and robots perform joint tasks or share the same environment. These robots should then be able to adapt their motion to the human presence and, if required, accomplish cooperative tasks. On the other side, the shared workspace between robots and humans may decrease productivity, if the robot is not aware of the human position and intention. Future smart production processes will have to cope with the need to guarantee the satisfying of production KPIs (Key Performance Indicators), and the new robotic systems integrated into such processes will have to be compliant with these requirements, too [228]; therefore, not only boosting HRP will help to satisfy safety requirements, but it will also contribute to hit ideal overall KPIs. An efficient collaboration with humans will be a fundamental element for various production processes, which are still only partially automated. The manual execution of some operations within the manufacturing process, as well as the presence itself of human operators in the environment shared with robots, will have to become an added value for the quality of the results, and not a potential cause of low efficiency. This goal can be achieved only through a proper choice of sensors and techniques, suitable for each particular kind of robotic system and application. This work has surveyed the main sensors and techniques, currently available to perceive and react to the presence of human operators in industrial environments, with reference to the various types of robotic systems commonly used in industry. On the basis of the carried-out analysis, some general considerations can be drawn:

- Vision sensors are fundamental to handle the human presence for any kind of robotic system, and in particular the most used one is the RGB-D camera.

- The combination of different kinds of sensors, possibly located on the robot and/or the human operator, can allow new types of collaboration and applications.
- Laser sensors are often used for human-perception purposes in combination with vision sensors in the case of mobile agents and manipulators, since they are typically present and used for navigation.
- The use of new, non-standard sensors is still limited, mainly due to the critical management of their somehow unconventional outputs.
- Most of the methods involved in HRP are enabled by the recognition of objects or human behaviour, especially taking advantage of artificial intelligence algorithms.
- New HRC applications can be envisaged also with sensors more commonly available, thanks to an innovative use of the information provided by them, as in the first presented POC, or through a coded collaborative HR behaviour, as in the second POC.

It must be finally underlined that several even smarter industrial HRC applications can be envisioned, provided that an efficient multi-modal sensor fusion can be guaranteed, possibly also including those sensors and methods that are currently mostly adopted in other contexts, such as assistive service robotics, agriculture and robotic surgery.

Chapter 8

A smart meta-sensor framework for human-robot perception in industrial environments

In this chapter, some applications and scenarios employing the sensor data fusion algorithm are presented. The main focus is to highlight the results obtained in [9] and [10]. In particular, the former has been briefly described in the previous chapter, in Section 7.3. Nevertheless, the aim of this chapter is to provide more detailed documentation related to the research work as well as some proof of concepts leveraging the experimental results already validated in a laboratory environment. It is worth mentioning that, as a co-author of both research works presented in this chapter, the main contributions are related to experimental setting up and testing.

The chapter is divided into the following sections: firstly, Section 8.1 introduces some related works and existing methodologies for sensor data fusion as well as how to improve the production when including the human in the loop. Section 8.2 provides an overview of smart meta-sensors employing AMRs while Section 8.3 presents a framework that allows AMRs to learn about human behaviours. Finally, the research work is discussed in 8.4.

8.1 Background

In a shared working space where mobile robots and human operators coexist, safety is a fundamental factor to be taken into account and so, hard constraints imposed by international standards must be satisfied. This may be difficult for setting up the AGVs. As a matter of fact, industrial manufacturing systems that use only AGVs may need to evaluate a trade-off between an ad-hoc smarter solution and a more traditional one having a wider application even if less efficient. On one hand, this becomes necessary due to the fact that traditional AGVs have technical limitations and, on the other hand, their system upgrade is expensive [229]. On the contrary, AMRs have better equipment, which allows them to perform reactive tasks, such as collision avoidance of dynamic obstacles and relatively intelligent path planning. The work presented in [230] supports the fact that the manufacturing performance in terms of productivity, flexibility and costs can be improved by introducing the AMR in industrial production systems. They presented an analytical model that shows the performance of AMRs in material handling production lines, highlighting also the fact that the flexibility obtained with the support of AMR can be achieved without the re-design of the production lines.

A distributed multi-agent system should efficiently divide and perform tasks. Nevertheless, the procedure adopted for this kind of system is generally focused on the algorithms for a particular category, e.g., motion planning for delivery operations, but without considering the human as a variable for the planning process. Even so, the Highway Code implementation proposed in [231] focuses on how to handle safety between robots and human operators. For example, simulations are highly recommended, due to the fact that experimental testing in real-world environments would be dangerous and time-consuming. Thereby, it is possible to build a list of risk assessments according to the available ISO standards for Human-Robot Collaboration (HRC) operations [232]. Alternatively, there are several approaches that may ensure safety within the industrial environment. An example of this includes installing fixed sensors around the working space, for monitoring and supervising the operations of the mobile agents and the human workers. Such sensors create virtual barriers that may deactivate or slow down the robot if human presence is detected nearby. In [233], specific areas are classified by colours, indicating the operations executed by the robots and the relative potential hazardousness for the

human operator. Notwithstanding, the fact of having a fixed network of sensors prevents flexibility since it depends strongly on the environmental infrastructure.

Cyber-Physical Systems (CPSs) are fledging technologies that integrate functionalities for connecting real-world operations with computing and communication infrastructures through a well-defined network [234]. In this context, sensor systems that exchange relative information from the real world with other agents enable to build cooperative and/or collaborative planning in the manufacturing process and hence, guarantee safety within the working space. For what concern CPSs, there are many researchers that have designed multi-agent system approaches to measure and share the relative position of each robot, as well as all the obstacles within the environment.

The combination of information coming from different sources can be achieved by employing sensor data fusion methods. The integration of data from different sensors allows to address the limitations of individual sensors, e.g., limited field of view or the information type. For instance, the authors in [235] propose a control method that keeps the visibility among robots when they are equipped with limited field-of-view sensors, e.g., LIDAR, cameras and optical sensors. The idea is to maintain multiple lines of sight formed by the robots while they are moving. The visibility is modelled using graphs and the edges of the line-of-sight of each robot within the sensory network.

Moreover, a sensor data fusion algorithm for cooperative trail-following tasks is proposed in [236]. Each robot can regularly share visual information with other robots, so the decision-making mechanism depends on its local view and the exchanged data from others. In particular, the visual information coming from both ground and aerial robots is combined and then tested in the real-world environment, successfully dealing with the "limited view" problem, which is usually found in single-robot systems. Likewise in [237], the use of an air-ground robot combined with ground robots is proposed, thanks to the fact that the Unmanned Aerial Vehicle (UAV) allows an easy global view. By aligning the data from the UAV and the ground robots' camera frames it is possible to estimate the global pose of each ground robot. However, the combination of ground and aerial robots cannot be easily employed due to the fact that UAVs are not suitable for any indoor environment.

Most of the mobile agents' cooperative localization algorithms have been designed with the aim of introducing new robots with better functionalities to replace

the older ones. On the other hand, many worldwide industries are still working with non-collaborative robots and a total machinery substitution would require a huge investment [238]. Unlike larger firms, a complete refurbishment for Small, Medium and Micro Businesses (SMMEs) for becoming smart factories may be a big issue, due to the high cost and limited resources [63].

By taking advantage of concepts related to CPS, it is possible to integrate intelligent agents, such as the AMRs, with the existing elements within the industrial infrastructure without the need for a total technology renewal, while still benefiting from Industry 4.0 or even Industry 5.0 solutions. In addition, since the future scenario envisages robots and human operators working very closely in the same environment, there is the need for a shared acceptance of the robots as part of the process and feedback from human workers must be taken into account. In fact, the probabilistic behaviour of an AMR leads to a sceptical attitude from workers, since they are not able to predict the unexpected motions or reactions of the mobile robot [58].

Furthermore, in the context of Industry 5.0, which is human-centred, the overall productivity can be increased when the robot learns about the human worker behaviour beforehand so it can provide support to human activities. For instance, a mobile robot could learn about the trajectory of the human operator when performing different tasks.

A human trajectory tracking algorithm mixing human-oriented Global Nearest Neighbour (GNN) data association and Kalman filter-based human tracking is proposed in [239]. Vision-based approaches to detect humans and objects are widely used; however, they do not provide accurate range information. For this reason, the authors of [239] combined the information of a 2D lidar and a RGB-D-based YOLO (You Only Look Once) system to correct missed information while tracking the human motion.

A dataset containing human motion trajectory and eye gaze data called THÖR (Tracking Human motion in the ÖRebro university) is presented in [240]. The data of humans moving in a room are collected mainly through a motion capture system running at 100 Hz, moreover, the overall dataset is enriched with information from a 3D lidar, eye gaze detectors and a RGB-D camera. The recorded trajectories are available in 2D maps, which are often used for training and motion prediction models of human motion. In [241], human body pose and gaze are analysed to obtain an accurate prediction of the human's intentions. A Recurrent Neural Network (RNN) is

used to predict sequences of multiple and variable length actions. Gaze and skeleton dataset is collected using the Optitrack motion capture and Pupil Labs binocular eye gaze tracking systems, while the multiple action sequence comes from a CAD120 RGB-D motion dataset.

A Multiple Predictor System (MPS) for human motion prediction is proposed in [242]. Depending on the context, it automatically switches between three individual classifiers: velocity-based position projection, time series classification and sequence prediction. In order to enhance the robot's ability to adapt its behaviour in environments shared with humans, the MPS is added in the path planning algorithm. In particular, the human's head 2D coordinates are used as features for the predictors [243].

In [244], a human motion prediction algorithm is proposed using a Hidden Markov Model (HMM). The HMM learns a set of movements executed by a human operator in an assembly task and then generates motion transition and observation probability matrices. In this way, it is possible to predict the motion of the human operator and perform assistive motion planning in Human-Robot Collaborative applications. Similarly, HMM is proposed in [245] to recognise human activities based on the principle object affordances, i.e., the relationship between the activity and a particular object/tool.

AlexNet, a Deep Convolutional Neural Network (DCNN) is modified employing transfer learning-enabled algorithm in [246], to enhance the robot's capability to learn human's actions. In particular, human actions can be divided in: (i) generic body motions, e.g., grasping or holding a tool and (ii) specific movements related to a context, e.g., actions performed while using a tool. The training procedure involves two separated deep neural networks, that analyse the human motion and identify the tools associated to the tasks.

A multisensor framework exploiting online transfer learning techniques for human tracking is presented in [136]. The performance for all possible combinations of 3D lidar, 2D lidar and RGB-D cameras are evaluated, and in particular, the solution that combines 2D lidar and RGB-D camera achieved the best results in terms of performance and precision to learn people's movements in the environment. In fact, the sensor's choice may enhance the robot's perception of the human [11], and therefore improve its learning curve about human intentions.

8.2 Sen3Bot: Smart meta-sensor for human-robot shared environments

This section aims at providing behaviour details on the architecture introduced in [14]. Additionally, the meta-sensor AMR module is here taken into account as a fully functional module, whose desired features have been implemented and tested in [5].

First of all, it is necessary to define what is considered a meta-sensor. A meta-sensor is an AMR equipped with a heterogeneous selection of sensors that becomes a sensor itself, with the specific capability of facilitating industrial scenarios monitoring, so as to support traditional or semi-autonomous AGVs. The meta-sensor AMRs must not be considered as an evolution of traditional AGVs, but as AGVs enhancers to enable smart factories' benefits. In order to reach such a goal, the relative localization information of each AMR along with the data about its surroundings are merged and shared with all the mobile agents. Special focus will be devoted to the dynamic detection of human operators. To put it differently, when one or more AMRs detect the presence of human workers in a specific area, the relative position of the latter ones will be updated on the shared map. The described capability is shown in Figure 8.1.

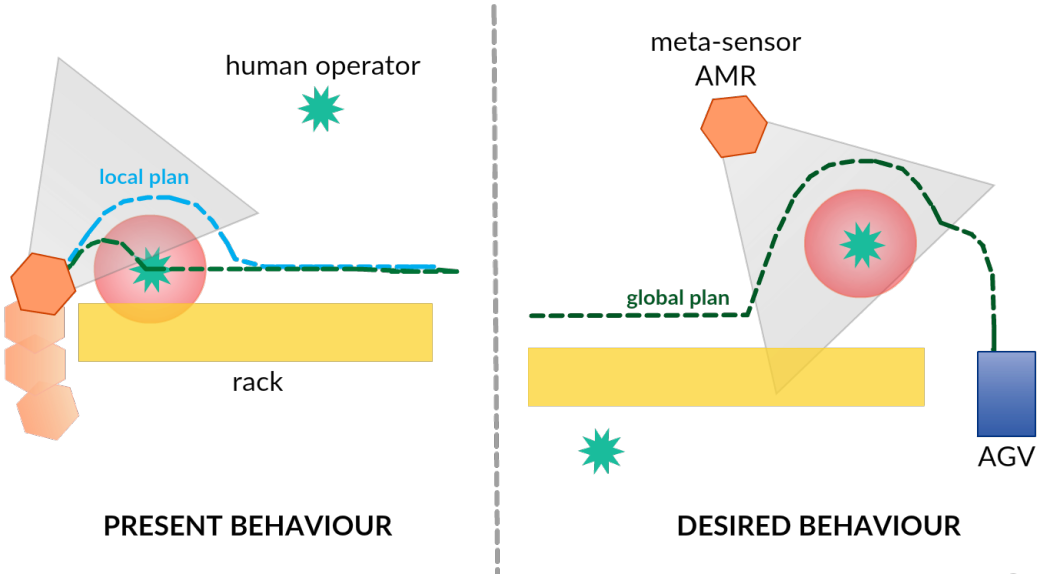


Fig. 8.1 Global and local human avoidance rule schematic representation.

The interface in charge of coordinating the AGVs will process all the gathered information from the meta-sensors and send the proper control commands to the AGVs depending on the following working conditions:

- The dangerousness of the area, and may depend on the infrastructure (shared or not with human operators).
- The activities of the operator.

In this way, it is possible to ensure safety for the overall system, as well as to be compliant with collaborative tasks between human operators and robots. In particular, a framework that improves the coordination and decision-making process of AGVs with the measurements acquired from the meta-sensors is presented here.

Throughout this chapter, the term Sen3Bot (Sentry roBot) will be used to refer to the meta-sensor AMR. The perception system of the Sen3Bot component is able to detect and identify humans, whose design is already illustrated in Chapter 6. This functionality is implemented by leveraging the real-time object detection system YOLO [59], in which a convolutional neural network is used to process the video stream of a low-cost IP camera placed on the mobile robot. This visual data is combined with the corresponding distance value through a camera-laser data sensor fusion algorithm, allowing to correctly place the identified humans within the plant map and to impose a more conservative behaviour specifically around them. The conservative behaviour related to the detected humans is to assign a larger inflation radius to the human. The Sensors Synergy Center (SSC) component is responsible for dealing with the coordination of the sensor data, such as performing sensor-fusion and map traffic updates, receiving the current poses of the AGVs by interfacing with the existing AGV Coordination Center (AGV CC), to take decisions for the Sen3Bots task allocation and execution based on the AGVs currently pursued tasks. Lastly, the AGV Coordination Center Interface (AGV CCI) allows to process and transform the relevant data gathered by the SSC, through the Sen3Bots, into proper commands that the AGV CC will use to suitably adjust the AGVs motion.

The S3B Net (Sen3Bot Network) is a network of autonomous and interacting hybrid agents, namely, intelligent robots that can autonomously perform actions on the basis of a planning algorithm while being able to sense and act even when there is an environmental change. Moreover, each meta-sensor AMR localizes itself

implementing the Adaptive Monte Carlo Localization (AMCL) algorithm, provided by the ROS Navigation Stack. Based on a recognized design workflow pattern [247], the system is described according to the following characterizing blocks: task decomposition, coalition formation, task allocation and execution.

Task decomposition

The main role of the S3B Net is to guarantee a safe motion for each already functional AGV within the manufacturing system. This task can be performed by taking advantage of the added value provided by the so-called *meta-sensor fusion*, in other words, the integrated information gathered from the involved monitoring Sen3Bots. It should be noticed that, apart from its main task, each Sen3Bot can be exploited for automatized tasks, e.g., transporting tools or materials to human operators depending on the availability. In fact, when the mobile robot is not busy with its sentry role, the Sen3Bot can take on traditional AMR tasks.

Coalition formation

The coalition formation in the S3B Net is set a-priori and it depends on the critical level of the different areas of the factory visited by the AGVs during their tasks. To better understand what is meant here by critical level, some references to standard definitions are provided hereafter to explain better the concept.

Taking into account the co-existence of both AGVs and meta-sensor AMRs in the industrial scenario, the Sen3Bot monitored space must cover as much as possible the non-restricted area that the supported AGV is going to cross, so as to provide a real-time awareness of the environmental dynamical changes *before* the AGV even reaches the location.

Additionally, since today's smart factories and the ones of the very near future have enhanced manual stations and cobot workstations fully integrated within the automated lines [248], there are specific zones where the presence of human operators is highly probable and hence, it is possible to consider them to be areas of interest.

According to [9], the main areas of interest are critical (area type 1.1 and 1.2) and sub-critical ones (area type 2), where there is limited visibility for the approaching AGV and/or there are human operators moving around. In fact, as soon as an AGV

has its path assigned and in the case that it crosses a specific area of interest, different approaches for the a-priori coalition formation can be defined depending on the level of criticality of the first crossed area. For example, if an AGV enters a critical area (1.1 or 1.2), two Sen3Bots are assigned to monitor it. On the other hand, when a sub-critical area is scheduled to be crossed, only one Sen3Bot is sent to the scene, to scout the area. Furthermore, the final number of Sen3Bots needed to monitor the scene, until the AGV will overcome the area, also depends on the new real-time data measured by the meta-sensor AMRs once they reach their respective monitoring poses, e.g., the detected human operators' speeds and directions. These additional data are used to decide whether the number of Sen3Bots sent to the scene is feasible for that scenario. It should be noted that areas of type 1.1 are the most critical ones since making up for the lack of visibility is crucial to avoid undesired collisions. In that case, if the crossed area is of type 1.1, two Sen3Bots are required at all times, until the AGV leaves the area. Less strict policies can be adopted when considering types 1.2 and 2 areas.

The number of detected humans and their behaviour within a specific area influences the final number of employed Sen3Bots. In fact, if the detected humans are quasi-static or moving very slowly, the information to be sent to the AGV does not require redundancy. However, in the case that the humans behave as dynamic obstacles, this situation necessitates robust measurements to be shared, so redundant data related to the human operators' location, speed and orientation is preferred.

In order to define how a *static* and a *dynamic* behaviour are distinguished in the Sen3Bot detection, it is worth recalling how human obstacles are represented in the shared map. As described in [5], human obstacles are enclosed in *virtual cages*, namely, they are provided with a larger safety radius value with respect to other obstacles. In addition, a virtual obstacle is published in correspondence with each detected human. The extension of this virtual obstacle depends on the human bounding box extension, which is detected at the image processing phase. This sort of virtual cage around the human operator remains integral to the operator's movements, dynamically adjusting to the detected person bounding box. Furthermore, the human obstacle is conservatively enclosed since the safety distance is maintained based on the left and right edges of the vision derived from the bounding box, thus guaranteeing that all mobile platforms are aware of the obstacle enlargement. This behaviour is the one depicted in Figure 6.6 in Section 6.4.

Therefore, for the sake of simplicity, the human behaviour is classified depending on the following conditions: if the centre of the human obstacle moves more than 1 m from the first detected position, in either direction, then the behaviour is considered to be dynamic. Otherwise, if the previous condition does not meet, the behaviour will be considered to be static. It should be noted that each Sen3Bot (if more than one is monitoring the scene) will provide such data from a different point of view, thereby allowing to capture more complete information about human motion. Note that this threshold value may be modified according to the area features and user requirements.

In addition, depending on the area criticality level and how the detected human operators behave, a *priority index p* is assigned to every task pursued by a Sen3Bot. In such a way, it is possible to classify tasks based on their priority and enable enhanced flexibility in the case that a Sen3Bot is required for a higher-priority task. A further instance of this is supposing that a Sen3Bot is pursuing a classical AMR task, for instance, transporting material, and the system requires a Sen3Bot to assist the passage of an AGV in a critical zone of type 1.1. In this case, if the mentioned Sen3Bot is eligible for being assigned this task, it will interrupt the pursued low-priority task and move to the critical area, postponing its initial task. The maximum priority is given by $p = 0$ and increased when the priority of the task decreases. Further details related to coalition formation decision process can be referred to [9].

Task allocation and execution

Before going to the description of how task allocation is handled, it is worthwhile to provide a general assumption about the initial pose of the Sen3Bots. It is assumed that the power charging stations for the Sen3Bots are positioned strategically, namely, located in such a way that the Sen3Bots monitor the most critical areas in the factory, while they are charging or while at *home*, which is in close proximity to the station. This allows the S3B Net to be responsive when areas requiring the most attention are involved. Also, it permits getting rid of the limitations inevitably introduced by fixed sensors, enabling flexibility while ensuring safety. Note that if the Sen3Bot is not assigned any task, it goes back to its *home* pose and $p = 2$ is set. meaning that it has low priority. Again, the priority $p = 0$ is set when the Sen3Bot is heading to its recharging station due to the detected low battery.

The priority value p and the distance of each Sen3Bot from the interest area are taken into account to identify the eligible Sen3Bots for standing sentry in the area.

To facilitate a faster S3B Net response, the list of all Sen3Bots is sorted depending on how distant they are from the scene and then selected only if the task they have been assigned is of priority 1 or 2, which corresponds to medium or low priority respectively. In the case the Sen3Bot has been assigned a $p = 1$ task, it is selected but can be replaced if a more distant Sen3Bot with $p = 2$ is available. The selection process iterates until the number N of required Sen3Bots is reached. Further details about the task allocation process are represented in the scheme available in [5].

Note that the total number of Sen3Bots available in the plant is strongly dependent on the number of areas of interest and on the client's requirements. In fact, a good trade-off could be reached when it is considered that minimum coverage of all critical areas should be guaranteed. Additionally, the simulations reported in this work take into account the case in which a central system processes and schedules the task allocation for a relatively small number of agents. The overall system complexity increases depending on the number of mobile agents, so a distributed control strategy could be preferable in some cases. Provided that the application scenario needs a large number of mobile agents, the proposed architecture can be deployed as different distributed modules, each one acting in a particular area. Nevertheless, in order to maintain synchronized the whole system, which includes other mechatronic systems within the smart factory, for instance, cobots and mobile manipulators, these modules still have to communicate with a centralized supervisor that assigns tasks at a high level. This way, the performance of the entire manufacturing system can be enhanced through sequences of minimal corrective actions, as in [249], established by integrating the performance indicators of the production process with the capabilities and working functionalities of the robotic systems and agents.

8.2.1 Simulation: some test case scenarios

In this section, with the aim of demonstrating the S3B Net behaviour, some simulation scenarios are reported. The video featuring the considered demo cases can be found online at [250]. The simulations have been run using the Kinetic Kame distribution of ROS [36], with the corresponding compatible version of the Gazebo Simulator [251]. It must be noted that version 7 of Gazebo does not allow to build animated person models (actors) in the simulated environment. Even so, this does not hinder the simulation's aim of demonstrating the proposed framework behaviour. The created Gazebo *world* represents a portion of a plant accessible to human oper-

ators, i.e., a non-restricted area. Rows of racks and a workstation are included, to provide the minimum conditions for a demonstrative simulation.

Figures 8.2, 8.3 and 8.4 provide the top view for each simulated scenario. In particular, on the left, there is the simulated industrial workspace, and on the right, there is the *rviz* visualization, where laser data of all mobile robots along with the cost maps are visible.

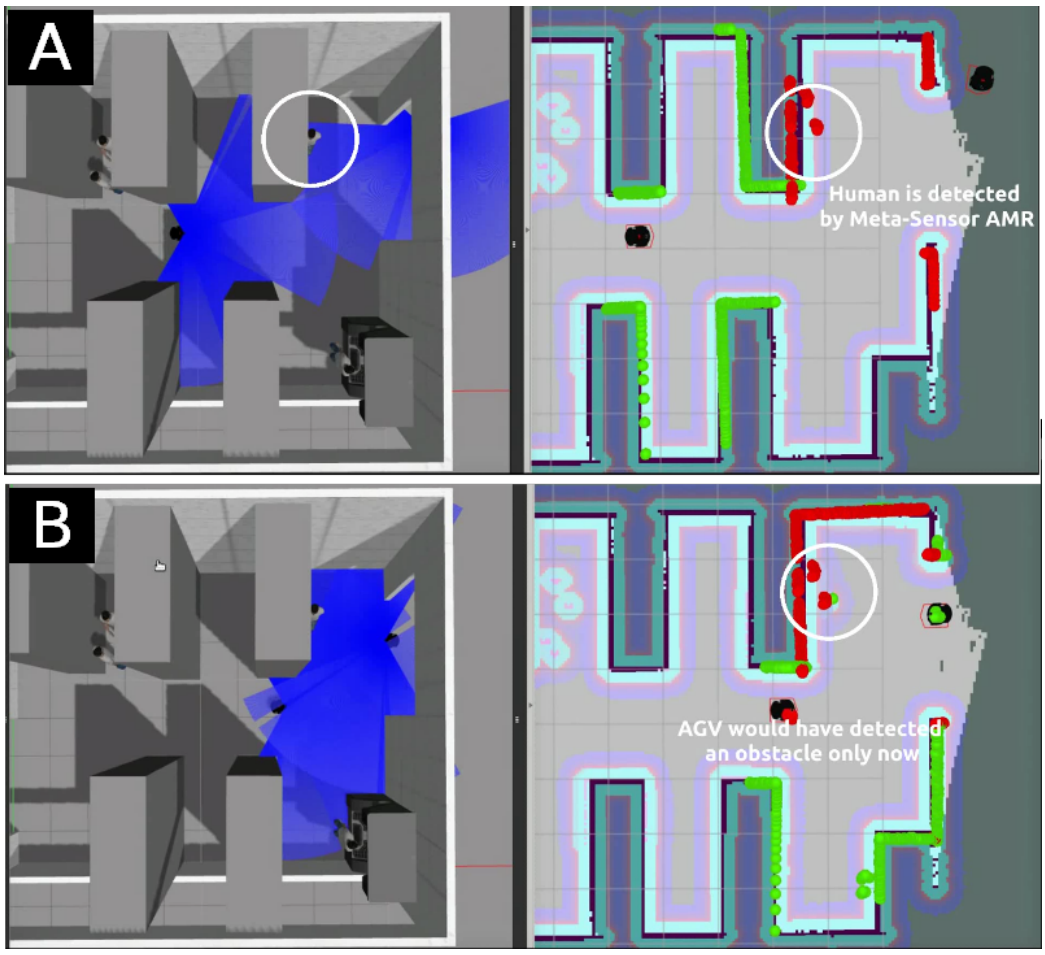


Fig. 8.2 Sen3Bot Net simulation: meta-sensor concept demonstration.

The first testing scenario setup is shown in Figure 8.2. The left mobile platform poses a traditional AGV (green laser points), while the right one simulates a meta-sensor agent (red laser points). This test aims at demonstrating the core role of the meta-sensor AMR concept, which integrates the information of the environment with the AGV knowledge about the path it is going to travel along, before reaching the area of interest. In particular, as seen in Figure 8.2A, the AGV is informed

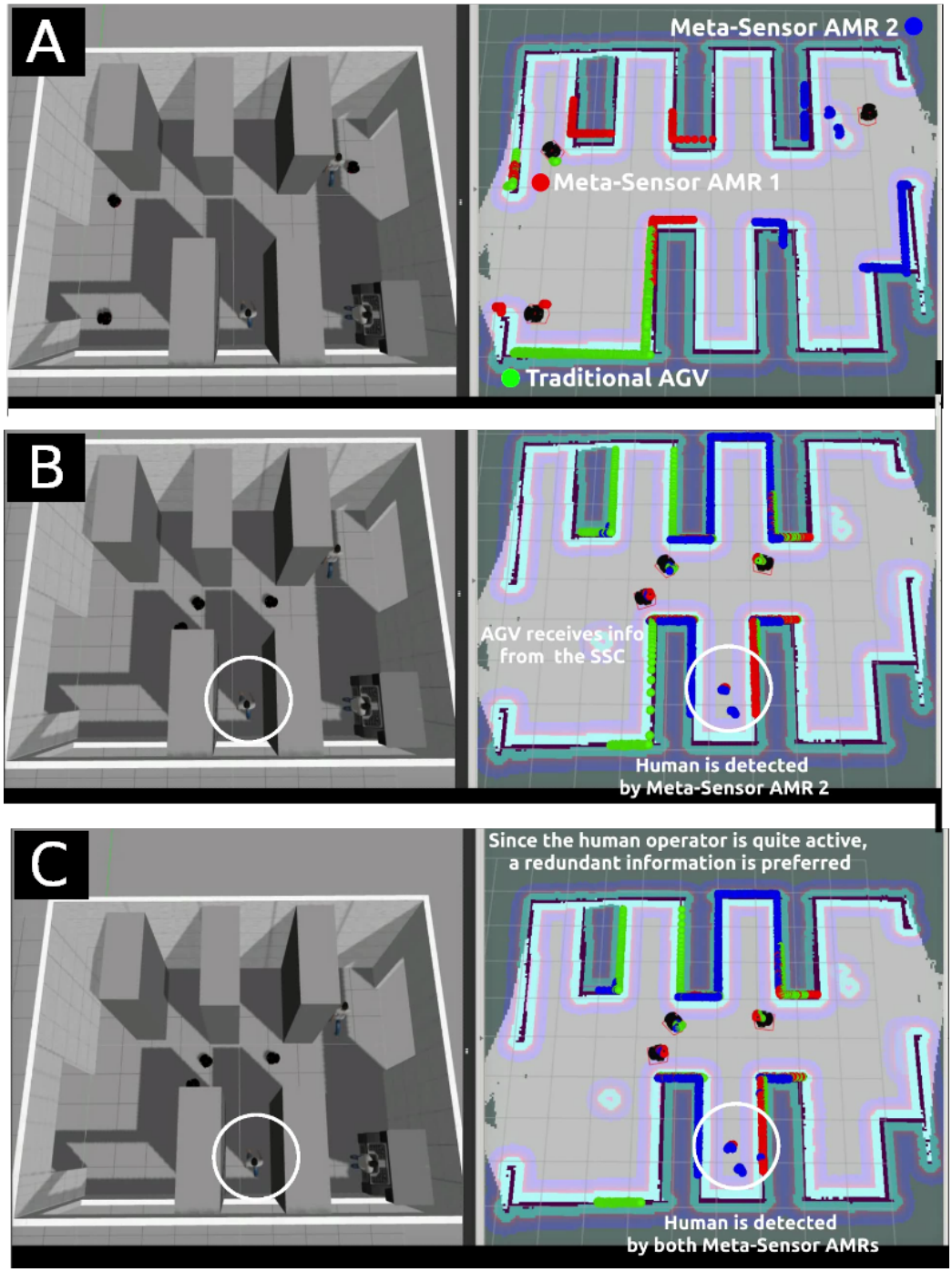


Fig. 8.3 Sen3Bot Net simulation: critical area 1.1 and dynamic human operator.

of the human presence even if it is out of its scope, thanks to the SSC processed data coming from the S3B Net. In this scenario, if the AGV relies only on the data measured from its onboard obstacle avoidance sensors, the human operator would be

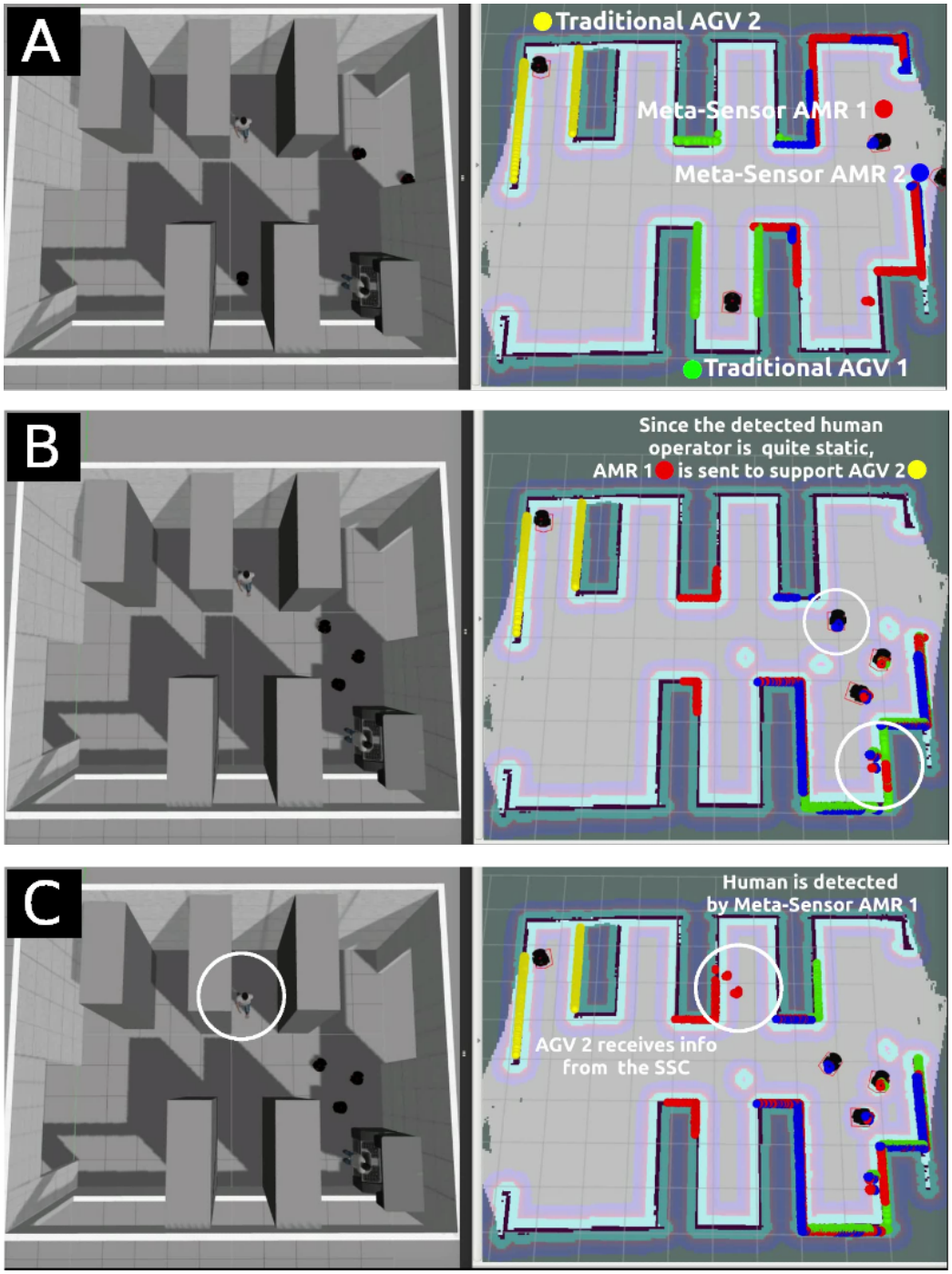


Fig. 8.4 Sen3Bot Net simulation: critical area 1.2 and static human operator.

detected too late, as can be seen in Figure 8.2B. The Sen3Bots meta-sensor fusion enables smart and safer behaviour. Note that in order to simplify the scene for the sake of clarity, the simulation does not consider the coalition formation rules.

The second scenario is depicted in Figure 8.3. It illustrates the case where a critical area of type 1.1 (see Section 8.2 for area types reference) has to be crossed by the AGV. In this case, two Sen3Bots are sent to the scene and inform the AGV about the presence of one human operator. It can be seen that in Figure 8.3A the different mobile robots involved in the simulation are identified. The AGV CC gathers this information from the SSC and accordingly modifies the AGV motion as it reaches the scene, as shown in Figure 8.3B. Then, since the behaviour of the human is perceived as dynamic, two Sen3Bots are instructed to both stand sentry in the area, as can be seen in Figure 8.3C.

Lastly, Figure 8.4 shows the third simulated scenario. In particular, it represents the situation in which an AGV has to cross a workstation area, usually classified as an area of type 2, which is located in a way that visibility is hindered by racks and thus considered a critical area 1.2. Keep in mind that Figure 8.4A indicates the role of each mobile robot role in the simulation. Two Sen3Bots are preliminarily sent to the scene. Nevertheless, during the AGV passage, the human obstacles are detected to be quasi-static, allowing for the system to set one of the meta-sensor AMRs to a priority $p = 2$, as shown in Figure 8.4B. Then, as the Sen3Bot is set available for pursuing other tasks, a monitoring task is requested in a very near area and meta-sensor AMR 1 is selected and sent to the requested sentry pose (Figure 8.4C).

8.3 Smart AMR learning from a human

This section briefly recalls the definitions and main concepts developed in [252], where a human-in-the-loop (HITL) data-driven framework has been introduced. The main focus of the outlined framework is to leverage information related to human actions to build a learning model for operation and task recognition, to make a mobile collaborative robotic platform or manipulator aware of the ongoing process. As a consequence, it enables anticipatory behaviour for improved collaboration along a flexible production line.

The idea is to emulate how a person usually perceives its surroundings: the decision-making anticipating an action is performed based on an approximated observation of the surrounding environment, in favour of efficiency and resource saving, in other words, when humans look around, they do not usually catch every

single information coming their way before taking a decision. The goal of the presented solution is to demonstrate that the path traversed by a human operator is a piece of sufficient information to identify the performed operation, to possibly anticipate human behaviour in the described restricted context scenario.

In order to record a set of poses occupied on a map by a human operator, the current solution takes advantage of the Sen3Bot meta-sensor framework [5], [9], a smart AMR whose role is to monitor the environment and safely cooperate with humans. Within the data-driven framework the project is brought towards a collaborative evolution, the Sen3Cobot. In particular, the *human operator modelling* and *data collection* functions are resolved considering solely the human positions, identified by a computer vision state-of-the-art object detection algorithm.

In the proposed solution, this set of positions is taken track of by plotting it as a path. The path data is preferred over the trajectory data since it allows extending the operation recognition to different operators, which of course take different total completion times for each operation. Hence, the chosen solution takes into consideration that the digital representation of path data, when plotted on a 2D map, is simply a matrix. By dropping the time information and given the duality of images as matrices, the spatiotemporal data recognition problem is translated into an image classification problem. This interpretation of data has been revealed to be crucial, as it allowed to add to the pool of possible methods to solve the problem a whole range of well-known and well-documented architectures, libraries and tools to implement deep learning models, along with a huge community often providing those tools as open source material.

Note that in the proposed solution the Sen3Cobot stack is improved with the understanding of the executed operation, i.e., implementing the framework *robotic system awareness* function. Indeed, the AMR currently implements passive HITL behaviour, as it monitors the area to gather position information from the detected human, and interprets it as an operation to be recognized. However, in the context of the overall data-driven framework, this passive step for operation recognition is fundamental for the decision-making before the action of the mobile cobot: based on the confidence of the classification, the robot will be given different trajectories to follow. To this end, the robot will need to act according to the probabilities associated with each class of operations. This means the system will iteratively check (while gathering new data) if the guess has changed and send a different reference to the

mobile cobot accordingly. With the aim of dealing with the data scarcity problem, the solution takes data augmentation through simple transformations as a first step toward model improvement. For what concerns data complexity, choosing image datasets rather than video ones allows in some way to have less noisy data, since the image contains only the map and the detected relevant information, namely, the human operator path.

Software tools

First of all, it is worth recalling that the data gathering provided by the Sen3Bot is ROS 1-based, featuring a vision module exploiting YOLO, containerized using Docker [253].

The development and testing of self-contained applications can be done in a lightweight and clean environment. Leveraging GPU within containers, local resources are exploited for running Neural Network (NN) based algorithms, which are approximately hundreds of times faster than a regular CPU. This also avoids lag problems derived from using Cloud available GPUs and security issues.

For what concerns the operation recognition/image classification problem, the `fastai` deep learning library, specifically its second version `fastai v2` [254], has been chosen. This library provides low, mid and high-level APIs to intuitively create deep learning models, either from scratch exploiting the Python libraries it is built on (PyTorch, NumPy, PIL, pandas and others), or allowing the use of architectures available in the literature and techniques made available following best-practices to get the most out of the available hardware. The authors also made available an interactive book written with Jupyter [255], an open-source project providing notebooks. A notebook is an interactive programming environment – whose cells' code can be modified and run straightaway – capable of working with different language backends, and kernels, so as to use several different programming languages. For this first solution implementation, notebooks represented a simple playground for code development and testing. The solution is built upon [256], which provides a docker image with pre-installed `fastai v2` libraries and notebooks, which has been modified to be adapted to solve the considered problem. This enabled the containerization of the recognition feature. Given the solution description, Figure 8.5 summarizes the overall structure and tool choices.

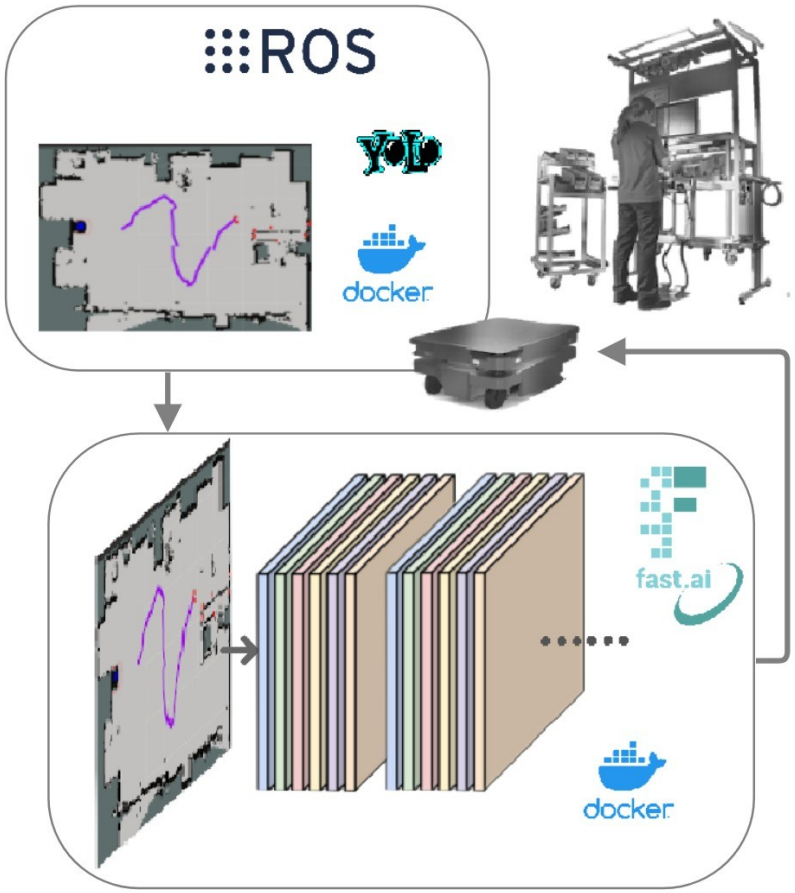


Fig. 8.5 The proposed solution aims at recognizing the human worker operations by tracking the human position on a 2D map. This output serves as the input to the *robotic system control* function.

In order to develop a baseline model for our image classification problem, some assumptions are made: (i) a single human operator is moving within the monitored area, without additional dynamic obstacles and (ii) the number of operation classes is limited to two. Note that we refer to *classes* of operations since this enables the possibility of fine-tuning pre-trained models using new operations, which may be variants of the main class. Taking in to account that an operation can be considered as a set of tasks performed at workstations (providing the needed machinery/cobot to bring on the necessary task), having main classes of operations with slightly different variants is a plausible situation. Additionally, the number of operations on a shopfloor is indeed usually limited to the available equipment and setup for the manufacturing of a certain product.

System implementation

Hereafter, the solution development and implementation details are illustrated.

- Positions collection: In the Sen3Bot stack, the positions associated with the detected human operator are published as virtual obstacles in the navigation local costmap of the AMR, so as to enable safe avoidance of the human obstacle. Within this work context, the published ROS topic provides a source of position messages to be plotted graphically on the RViz visualization tool.

- Path plotting: Filtering of messages was performed, since all points falling within the detected human obstacle bounding box are published as virtual obstacles. Such points have been filtered out and, among the points covering the bounding box width, only the nearest one has been kept at each sampling instant.

  Each human 2D position is then collected by a ROS node that pushes it in a type `Path` ROS message which is then published on an ad-hoc ROS topic. For instance, the resulting path is shown in Figure 8.6. Note that the smoothness of the path might be affected by noise and the ROS node spin rate.

- Data collection: Even though simulation has been taken into consideration, a collection of real data has been preferred. This is due to the fact that training a learner on purely synthetic data will unavoidably affect its capabilities of performing classification on real data samples, to the extent that it might not be able to recognize any operation at all.

  In order to speed up the data collection, the operation executions have been video recorded and periodic screenshots of the `rviz` map visualization have been generated.

- Dataset creation: Disjoint subsets of such collected samples have been used for training, validation, and testing of the algorithm, respectively. The motion samples have been gathered from two different people, so as to improve generalization capabilities of the learning algorithm. Data samples have been saved in the dataset `/train` and `/valid` folders. Note that training and validation sets include only images representing completed operations. This is because the desired behaviour is having the architecture to fit its parameters based on representative samples of each class of operations. On the other hand,

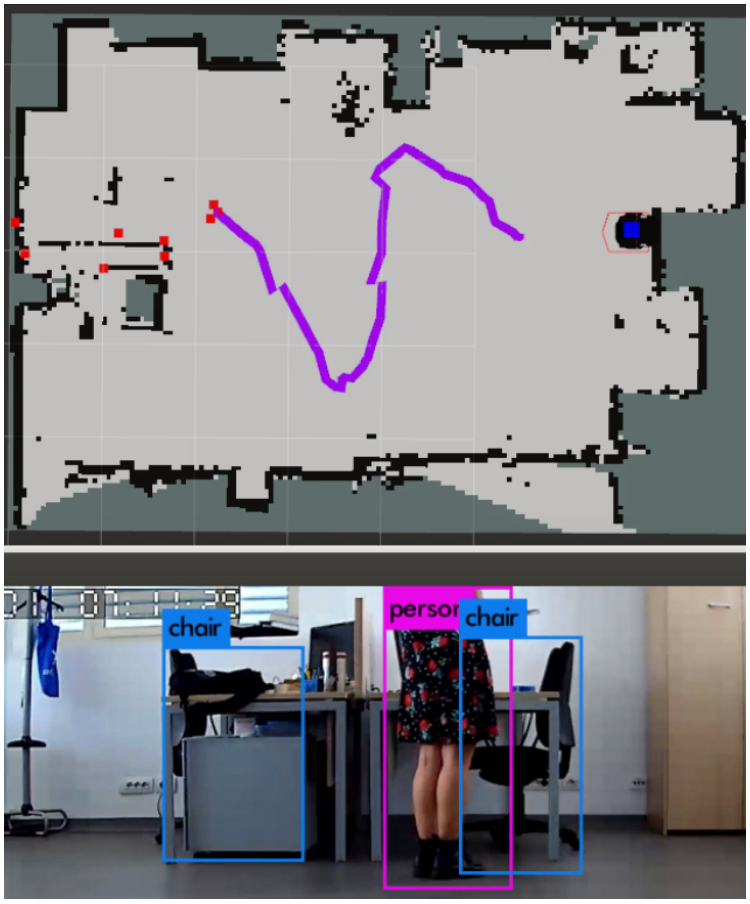
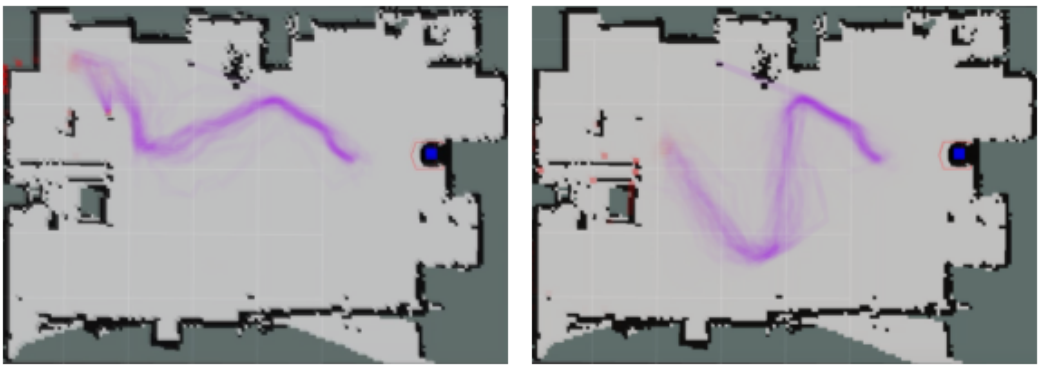


Fig. 8.6 After data filtering, the detected human path is published, and its content plotted on the 2D map by passing the type `Path` topic to an RViz Display.

to test the model capabilities to recognize an operation from the very beginning of its execution, the testing procedure is performed on samples of ongoing operation executions. In this way, it is possible to observe the model capability to improve its confidence as the operation/path goes towards completion.

To give a compact representation of the collected dataset content, Figure 8.7 shows the mean values for each training set for each operation. The mean of all the image tensors corresponding to a certain class of operations is obtained by taking the mean along dimension 0 of the stacked rank-4 tensor. Notice that the manipulated tensor is rank 4, since the processed data are RGB images.

- Architecture: For the plotted path recognition (considered here as an image classification problem), a Deep ResNet (Residual Network) architecture has been used. ResNets address the degradation of training accuracy (vanishing



(a) Class `A` operations mean value. (b) Class `B` operations mean value.

Fig. 8.7 Mean values images for each of the considered classes of operations, computed among the samples in each corresponding collected dataset.

gradient) problem, namely, the degradation of training accuracy, associated with network depth [257].

In particular, a `ResNet18` has been selected as learning architecture with the cross-entropy as the loss function, since it is the most common loss function employed for binary classification problems. This function will be minimized by the stochastic gradient descent procedure during weight stepping. The learning rate has been set to 0.002, as suggested by `lr_find()`, a `fastai` function that plots the loss against learning rate values and outputs a suggested value, corresponding to the point where the gradient is the steepest.

- Data augmentation: A total of 300 image samples per class of operations have been collected for testing the preliminary recognition capabilities. Despite the fact that overfitting issues may arise due to data scarcity, it has been decided to collect a low number of samples for the sake of achieving a low complexity setup.

  As a pre-processing step, all samples within the dataset have been resized to reduce their dimension, since size reduction does not seem to affect the model performance. Then, a set of transformations have been selected, i.e., small rotations and warping, and lighting editing. This set of transformations are defined along with their relative application probabilities, i.e., the probability with which the transformation will be applied to random batch elements during training. The batch size have been set to 64 and the training algorithm will take

care of shuffling in a random way across the training data set when choosing candidates for each mini-batch.

Furthermore, as a callback for every tweak of the training loop, the MixUp method [258] has been applied. Specifically, MixUp generates new data during the learning procedure through convex combination of random pairs of images and associated labels. A random sample of generated batch elements can be seen in Figure 8.8.

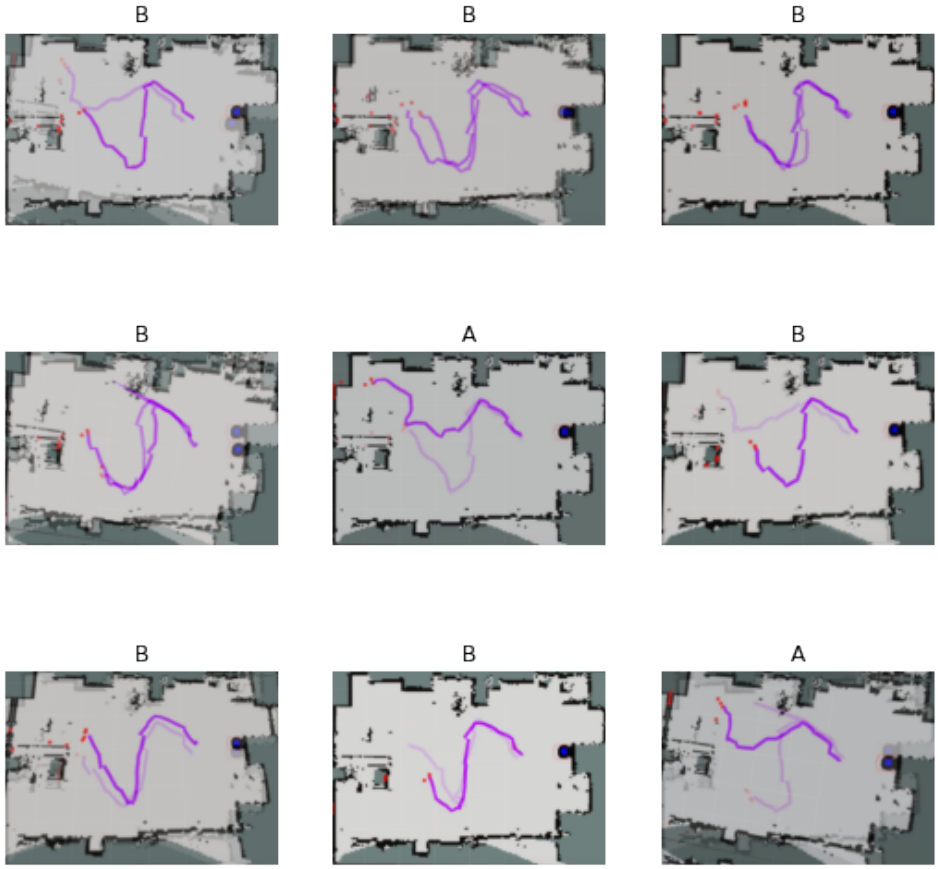


Fig. 8.8 Sample of batch elements generated by MixUp algorithm. As can be seen, shuffled samples are also affected by the randomly applied transformations for data augmentation.

- Model: Once the described hyperparameters have been definitively set, a one-cycle training policy have been performed [259], which is a commonly used method for training `fastai` models from scratch, i.e., without transfer learning. As expected, most times the accuracy saturated to 1 during the first couple of epochs, suggesting overfitting issues. Nonetheless, the main aim of

the developed work is to test the obtained model on images representing the sequence of sub-paths corresponding to an operation. Therefore, the number of epochs for training has been limited to 1. An accuracy of about 0.93 has been achieved, with a training time of 4 s, running on a PC equipped with a 4GB GDDR6 NVIDIA GeForce GTX 1650 GPU. The trained model has been saved as a baseline model for the considered problem. Figure 8.9 shows a subset of the top losses peaked during training.

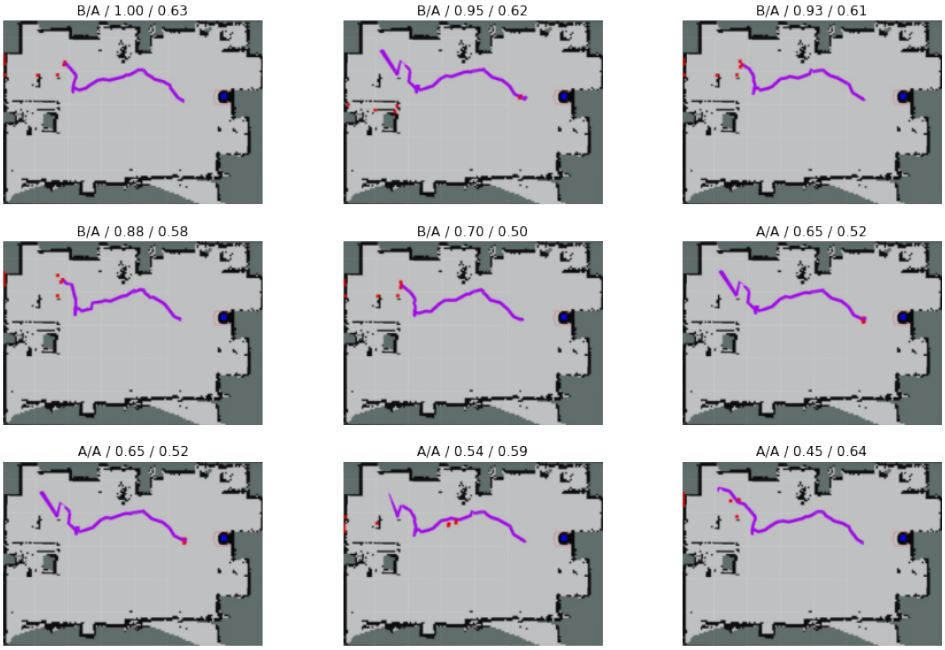


Fig. 8.9 Subset of samples that generated top losses. The title of each image shows: Predicted class / Actual class / Loss / Probability of actual class.

8.3.1 Experimental testing

In order to demonstrate the feasibility of the proposed solution for operation recognition to enhance collaborative applications, the obtained baseline model should be able to distinguish different classes of operations. In addition, it should ideally improve its guess confidence as it is provided with a sequence of images representing the progression of an operation execution. In fact, within the data-drive framework, according to the output of the proposed solution, a specific reference curve will be fed to the mobile robot control system. In particular, a reference path should be

generated from the weighted combination of candidate reference paths, where the weights are proportional to the associated operation class probabilities.

The baseline model has been first feed with progressive screenshots from a class `A` operation execution. Then, the same has been performed for a class `B` operation. Figure 8.10 reports the obtained testing results for both the class `A` operation and the class `B` operation testing samples.

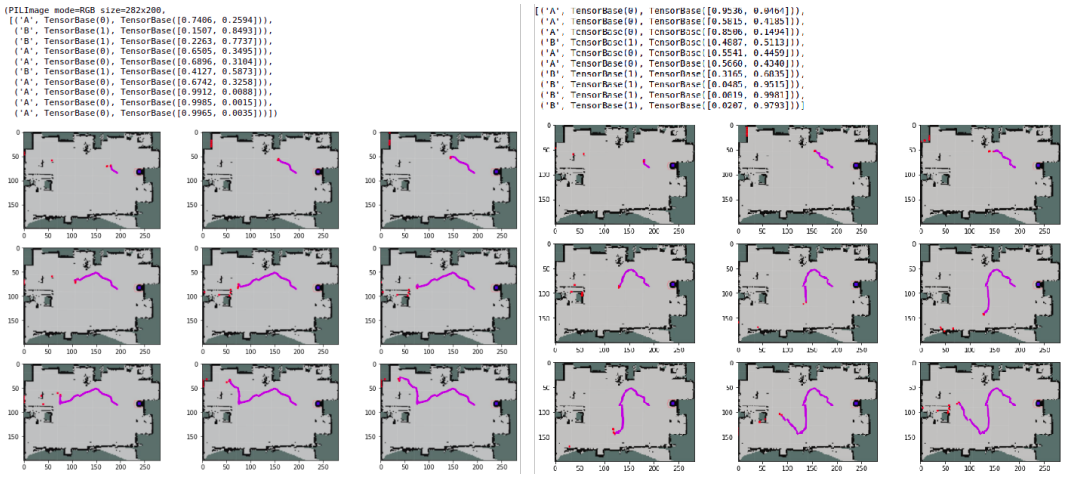


Fig. 8.10 On the left: prediction labels and probabilities, along with the set of testing sample images, corresponding to an operation `A` execution. On the right: prediction labels and probabilities, along with the input set of testing sample images, corresponding to an operation `B` execution.

As can be observed, for both sequences the model initially struggles to correctly recognise the ongoing operation. This is expected, since the first part of each representative path is identical for both classes of operations, as the first visited workstation is the same. Nevertheless, in the case of operation `A` recognition the classification results start with a couple of switching predictions, 74% `A` at step 1, passing through 59% `B` at step 6, and reaching 67% `A` at step 7, which lead to a prediction above 99% from step 8. The recognition of operation `B` generated similar results, starting from high probabilities associated to the wrong class (95% probability for `A` at step 1) and progressively switching to increasing probabilities for class `B`, eventually reaching above 95%, starting from step 8.

8.4 Discussion

This chapter presented two smart meta-sensor frameworks exploiting AMRs in an environment where humans and robots coexist. The first one, corresponding to the Sen3Bots, aims at enhancing the sensory system of the overall manufacturing plant, especially the perception system of existing mobile platforms working actively in the working station, e.g., AGVs and AMRs. The second one, the meta-sensor monitors and learn from human actions, particularly the path executed by the human operator during different phases of the operation, so as to generate the trajectories for the Sen3Cobot, supporting collaborative applications with human operators.

In particular, the Sen3Bot's architecture enables the upgrade of obsolete pre-existent systems integrating them with intelligent AMRs, allowing for deployment in a larger number of facilities and preventing huge renewal costs. This is in contrast with the ongoing trend of entirely substituting the existing mobile robot setup with a new network of intelligent AMRs. Safety is virtually guaranteed by using the AMRs as a supplement to the AGVs' sensor equipment, with the goal of enhancing the environment perception capability. Using ROS as a development tool promotes code-reuse through the *ROS Namespace* feature. In other words, the code running on each Sen3Bot is the same and threads belonging to a single robot are identified through a unique prefix. The latter approach permits to foster scalability.

Moreover, a framework that learns 2D human motion allows performing operation recognition in the context of human-robot collaborative applications. The solution exploits deep learning state-of-the-art libraries and architectures to obtain a model able to recognize the operation related to the motion of the monitored human operator. To mainly demonstrate the solution feasibility with the tackled problem, a small dataset has been prepared for training purposes.

Chapter 9

Framework for safe and intuitive human-robot interaction for assistant robotics

In this chapter, a framework for safe and intuitive human-robot interaction for assistant robotics is proposed. In particular, the aim of the work is to provide an overview of [13], which proposes an architecture that merges different aspects of human-robot collaborative applications in order to enable safe assistive applications. The proposed framework can be considered for general purposes and an example of its usability in a manufacturing context is provided, as well as the current development of the functional blocks that compose the framework.

The chapter is structured as follows: firstly in Section 9.1, a background of related works is presented. Then in Section 9.2, the proposed framework within a human-centred manufacturing scenario is unfolded, while Section 9.3 illustrates the robot features divided into main functional blocks. Last but not least, Section 9.4 discusses the proposed framework as well as sketches future works.

9.1 Background

The concept of Industry 4.0 has evolved since its very first definition, alongside the technologies it involved and the market pull behind it. The quest for new solutions

and novel research directions suggests that a new industrial revolution is taking place to comply with these new requirements: Industry 5.0. Its most distinctive feature is human-centricity, e.g., the role of the human workforce in the industrial automated smart processes. The human operator will increasingly have a leading role in mass or product customization, due to the fact that the cognitive skills that intelligent machines still lack some of the human's unique capabilities, for instance, creativity and critical thinking for problem-solving [260].

These new human-centric oriented solutions are envisioned to present autonomous collaborative robots as human assistants, able to improve their supporting role thanks to Artificial Intelligence (AI) based technologies. In this context, the cobot takes care of the operations requiring less cognitive skills along the supply chain management and covers repetitive and routine monitoring tasks. By executing background but fundamental work, the cobots allow humans to focus on tasks requiring complex reasoning and decision-making skills. In such a way, the human together with the AI-enabled machine become a symbiotic system allowing for intelligence augmentation [261]. In order to support human activity, the robot needs to be able to correctly perceive its surroundings, recognise humans and be aware of the dynamic changes in the environment. With this aim, Industry 4.0 solutions already featured complex and heterogeneous sensor systems [11]. On top of that, the current cobot evolution or Industry 5.0 cobot, is responsible of the following tasks:

- Lighten the human workload taking up repetitive and demanding work.
- Enhance its own perception and collaborative capabilities to enable a proactive Human–Robot Collaboration (HRC) paradigm.

Proactive HRC can be achieved by implementing bi-directional empathy and holistic understanding during the collaborative execution, spatio-temporal cooperation prediction, estimating the interaction among all elements involved (human, robot and workpiece - if any), and teamwork self-organization learning as a result of converging knowledge [262]. Sensor data fusion algorithms are used for information extrapolation for a complete interpretation of the collaborative robot's surroundings, as a foundation for implementing prediction of human operator intentions and bi-directional multimodal communication to improve the interaction. To do so, the authors of [263] suggest that much of the relevant information can be retrieved from vision sources, relying on computer vision-based cognition of objects, humans,

and environment, as well as exploiting visual reasoning to bridge the gap between scene understanding and proactive decision-making. Nevertheless, they highlight how vision data might not be enough to deal with the complexity of understanding the object affordance properties, in other words, the difficulty of interpreting the interactive properties of objects for manipulation, and the need for an unambiguous comprehension of gestures for proper interaction. Thus, ambiguities must be avoided in order to achieve effective interaction between humans and robots in collaborative industrial applications. This can be dealt with by possibly integrating further relevant pieces of information, such as natural language during bi-directional communication.

To implement a robotic assistant for smart manufacturing applications, developing object/tool recognition and grasping capabilities is key. In [264], the authors propose a mobile manipulator able to autonomously navigate while detecting humans and objects, so as to automate small and medium-sized enterprises' production. A point-voxel region-based convolutional neural network is used to robustly detect objects, allowing to get rid of 3D point cloud data uncertainty issues, while a 2D camera is employed to calibrate the collaborative robot's relative position to workstations. In [265], an event-based robotic grasping framework for known and unknown objects in a cluttered scene is presented. In particular, the model-based solution consists of 3D scene reconstruction, object clustering according to Euclidean distance, position-based visual servoing, and grasp planning of objects whose shape is a-priori known. The mentioned works are part of a substantial body of literature that exploits AI to implement object recognition and manipulation.

During the proactive collaboration, the assistant collaborative robot and the human worker may interact in different ways. One research direction for human-robot interfaces for collaborative assembly exploits Augmented Reality (AR) to provide the user with information coming from the robotic assistant. If correctly accepted by the user, AR can improve assembly cycle time performances [266]. In [267], a virtual representation of the cobot is rendered over the real robot allowing for direct visual feedback of the set of waypoints to be executed, previously instructed through gaze and speech. The operator then can let the robot execute the set path, through a speech command. Natural language and gesture interpretation is being featured in a large part of proposed solutions. In [268], the operator is supported by a natural language-enabled virtual assistant, which translates the operator's requests, informs him/her about the robot status, and supports new operators during the training process.

The involved collaborators can have variable decision-making power, depending on the executed operation and context. For example, the work presented in [269] uses a mobile manipulator as a flexible solution for tending operations on computer-numerical-controlled and additive manufacturing machines. Given the cost of such equipment, they prefer a semi-automated execution modality, where low-level decisions and grasping planning are automatically computed by the robot, but high-level decisions and pre-grasping poses are supervised by the human operator. This way, the operator can perform risk-informed decisions to accept, refine, or decline system-generated plans.

It is worth pointing out that most of the work provided by recent research directions for human-robot collaborative tasks, offer approaches involving AI-enabled mobile manipulators since they embody the main capabilities demanded of a robotic assistant, such as flexibility, autonomous mobility and improved dexterity. Despite that, some solutions opt for supervised or semi-supervised collaboration, decision-making data provided by the human operator during this kind of application could be collected with the aim of providing a dataset to the AI-enabled cobot. In such a way, a cobot would be able to perform high-level decisions on its own if it is trained on human-generated past collected information or experiences.

In fact, the cobot can learn from monitoring data, either to learn and emulate the human operators' motion, or to acquire it for later use, with the aim of action recognition and prediction. In [270], a dynamic movement primitives model is employed to simultaneously learn the motions of a human operator's body and hand, to be fed as reference trajectories for the mobile manipulator base and end-effector, respectively. Then an unscented model predictive control strategy is used for solving the trajectory tracking control problem for a mobile manipulator with uncertain parameters and disturbances.

To the best of the author's knowledge, current solutions solve separately human action prediction, object recognition, object affordance manipulation, and safe motion control, however, there is still a gap in what concerns the safe human-robot interaction for assistive applications that involve object affordance. The research work presented here aims to propose a framework for mobile manipulators assisting human workers. In the context of mass customization, the human participates actively in the production line since some unique human skills are required, such as creativity, problem solver mindset, promptness at analysing, and decision-making.

The role of the robotic system in this case is to give a sort of assistance to the human, which can enhance the performance of the overall production time.

9.2 Framework proposal

The proposed framework will be mainly focused on mobile manipulators since they provide more flexibility and have additional mobility features with respect to fixed base manipulators, which could come in handy for object manipulation to support human operator activities. In order to develop this framework, the robotic system should have at least the following functionalities:

(a) An AI-based perception system for recognition and learning of human actions, objects (tools or instruments), work context, and working spaces, whose data come from the perception system mounted on the robot and the sensors used to monitor the workstation. Figure 9.1 illustrates the schema of the proposed AI-based system.

- The idea is that the robot should observe and learn from the human while he/she is performing some tasks, for instance, assembly, disassembly or inspection. The sequence of tasks performed by the human can be predicted depending on the working context. An example of this is the scenario of an assembly/disassembly task, where the human worker may use a type of screwdriver and then use another screwdriver or wrench. Once the human activities are separately classified, the robot should be able to predict the next human's actions as well as the next tool that the person might require. so as to deliver the right tool as soon as the human completes the current task, and then take the tool that the human is no longer using.

(b) A control system involving the following blocks:

- A safe trajectory generator, which depends on the tool that the robot is holding.
- A safe trajectory tracking system.

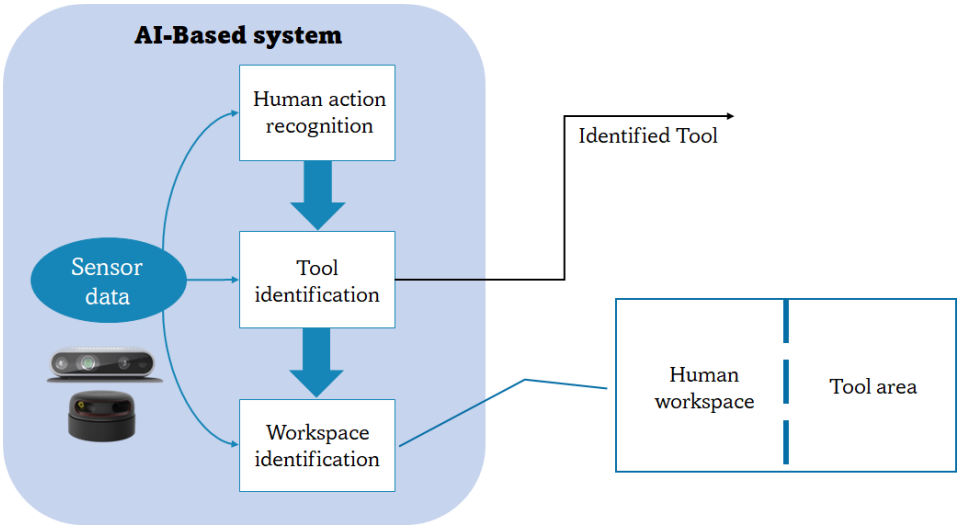


Fig. 9.1 AI-based system for human action recognition.

- A safe trajectory replanner/corrector for those cases in which the robot needs to perform a trajectory correction or replanning its trajectory while avoiding in a safe manner the human operator, in order to avoid unintentional harm to him/her.
- A mechanism for safe object grasping/releasing.

The scheme is shown in Figure 9.2.

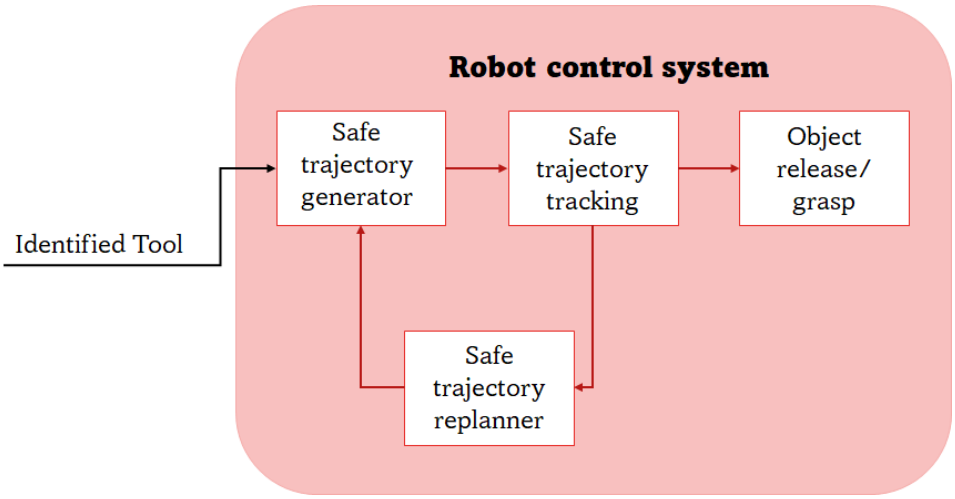


Fig. 9.2 Control block diagram for tool manipulation.

9.3 Framework feasibility analysis

In this section, the functional blocks that are considered fundamental for the development of the framework are described, while reviewing available current solutions. In this preliminary stage of the framework development, some assumptions are made for the sake of simplicity and to show correctly its workflow. For instance, the tasks performed by the human worker are limited to a sequence of short actions, and they are assumed to require a small number of tools.

9.3.1 Sensor data processing

The sensory system functional block is in charge of collecting data coming from different sources. Several sensors can be combined to provide a robust measurement of human action from different perspectives. A further instance of this is installing fixed visual sensors monitoring the human working space from above, and the data is cross-checked with the local information perceived by the robot's sensory system. Depending on the application requirements and complexity, a stereo camera could be used to monitor the working space, so as to be able to provide useful information related to human actions since the video stream could be fed to an AI-enabled computer to process and classified the different aspects of the working space while the robot.

The data collected by the sensory system will be used for recognizing human actions, identifying the objects or tools in the working station as well as monitoring separately the working spaces of the human and the robot.

9.3.2 Human action recognition

The functional block dedicated to recognize human activities has the aim to process and analyse the data coming from the sensors monitoring the person while performing some tasks, such as the ones equipped onboard on the robot or installed near the working space. Collecting and studying human action from different perspectives may allow the creation of a large database for a certain application, as well as enable a robust prediction of human motions.

Generally, a human tends to repeat some actions while performing assembly or disassembly tasks. These kinds of actions can be used during the AI-enabled robot training process in order to recognize and predict human actions. As an example, during an assembly task where the human opens a device/product, inspects it, modifies it, and then seals it up, it is likely that the human may need different tools for the task completion. Given the same working context and operations, the sequence in which the human will use the next tool can be predicted by the robot.

Hidden Markov Model (HMM) algorithms are popular for human motion prediction since the hidden state transitions can be used when there are uncertainties due to weak motion recognition. In [244], HMM is used to predict a sequence of human actions by generating motion and observation probability matrices. Moreover, the principle of object affordance is included in the HMM model proposed by [245] in order to give some context to the human actions.

9.3.3 Tool identification

The tools can be classified and labelled a-priori by tool type, based on their utility and also on how they are usually grasped/held/used by the human operator. Identifying the tool may allow the robot to grasp it correctly and deliver it in a safe manner. Moreover, being aware of which tools are used allows the robot to put them away where all the other tools are kept. The tool identification process should also involve an affordance detection [271], that localizes, classifies and labels the affordance of the detected objects. Through this feature, it is possible to identify the graspable and non-graspable parts of a tool, e.g., the handle and the blade of a knife. There are several algorithms that are able to identify and detect the affordance of different objects. For instance, the authors in [272] presented *AffContext*, an object-agnostic affordance recognition neural network that analyses and predicts affordances of object parts in an image. In particular, *AffContext* is able to recognize the object elements even novel ones that were not present in the training dataset.

9.3.4 Workspace organization

In a complex assembly task process, the human worker may need a set of different tools, however, it is better to have the working space as clear as possible, while

keeping the tools that might be used repeatedly close and those that are dangerous/heavy/seldom used in a dedicated toolbox/space, which can be accessed in case of necessity.

In order to define clear workspaces, which can be easily classified and comprehended by the robotic AI system, the working space (Figure 9.3) can be divided into the following areas:

- Human workspace. It is the area where the human worker can perform the tasks normally.
- Tool area. The tool area can be divided into two subregions:
  - An area in which the tools are released by the robot. The tools released in this part of the workspace are predicted by the AI system depending on the task performed by the human operator.
  - An area in which the robot collects the tools that the human is not using. The tools are retrieved by the robot and put back in place in the tool deposit.

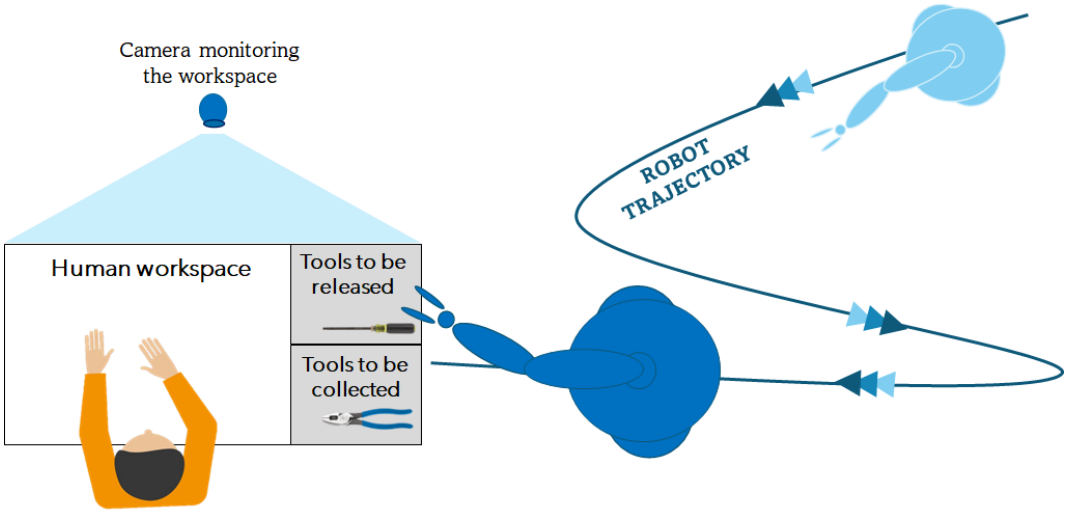


Fig. 9.3 Envisioned working space organization.

9.3.5 Mobile manipulator control system

Since a collaborative application is performed in a shared working space, safety is a relevant issue and the generated motions must take into account not only what is being executed but also how, to avoid hazardous situations. Inspiration can derive from tool-path smoothing methods, which aim at reducing rough tool motions that directly influence the dynamic performance of its motion control. For example, the tool path smoothing and interpolation algorithm proposed in [273] based on the finite impulse response filter, demonstrated to generate a path that has better tracking performance in the motion control process. This type of smoothing approach could be applied for improving tracking of generated paths according to recognized human actions, to ensure a more reliable behaviour.

Once the tool is identified, either after predicting the next tool the human operator may need, or when the unneeded tool must be put away, in order to ensure safe interaction between the robot and the human, a safe trajectory generator should be used for each tool type, e.g., screwdrivers, hammers, pliers. This is because each of these tools can cause harm if they are not handled correctly, in particular, while being held by a moving robot. The control system needs to consider the object affordance information as well as the motion of the human.

As the action is recognized, the cobot should be able to act according to this new information. To do so, control and manipulation approaches can be borrowed from non-collaborative frameworks. For example, in [274], the authors propose a methodology to perform sensor-based manipulation planning to automatically compute collision-free feasible trajectories. Then, a controller is associated with each trajectory segment generated by the manipulation planning algorithm. Given a list of tasks and relative sensor information, each controller computes a control variable corresponding to a sequence of tasks, which minimizes tasks' errors that take into account tasks' priorities. Moreover, [275] provides a high-accuracy method for estimating a workpiece pose for workpiece exchange, which consists in compensating the potentially poor positioning of the mobile base with a marker-based alignment of the manipulator. Exploiting end-effector camera images of fixed tags applied on workstations, the manipulator iteratively adjusts the camera to a previously recorded camera pose, so as to correctly compute the workpiece pose.

In this preliminary stage, the trajectory tracking will consider only the kinematic model of the mobile manipulator, while the safety issue will be dealt with using the onboard sensors, so as to avoid any contact or unintended harm to the human. In this case, the human can be initially modelled as a virtual obstacle with an enlarged inflation radius [5]. Future developments should consider the dynamic model of the robot in case of force/torque interactions; for instance, an impedance control architecture for collaborative material handling could be compliant with the safety measures in this framework [276].

9.4 Discussion

A framework that enables a safe and intuitive human-robot interaction for assistive tasks is sketched in this chapter. In particular, several functional blocks were identified and investigated in order to evaluate their feasibility and the current state-of-the-art. Current solutions work separately on their own, however, to the best of the author's knowledge, there is not yet an existing solution that combines all the functional blocks in a human-centred manufacturing environment, particularly for a trajectory planner that considers both the object affordance and safety towards the human operator.

The next steps will involve the development of each functional block, taking into account the data complexity and compatibility coming from each block to be fed to the control system, in particular for safe trajectory tracking and tool grasping.

Chapter 10

PoinTap system: a human-robot interface to enable remotely controlled tasks

In this chapter, a PoinTap system is proposed. In particular, it is a human-robot interface presented in [12] that enables remotely controlled tasks, mainly focused on manipulators. The research work presented here aims at demonstrating an alternative solution to user interfaces to safely control an industrial manipulator using open-source resources while keeping low-cost and safe. The idea is to develop a simulated touchscreen device to command a robot for pick-and-place applications.

The outline of this chapter is structured as follows: Section 10.2 illustrates the main conceptual elements of the proposed PoinTap interface. Then, with the aim of describing the functionalities of each block, the PoinTap system implementation and testing applied to a specific case study are unfolded in Sections 10.3 and 10.4, respectively. Finally, Section 10.5 draws some conclusions and identifies some open issues.

10.1 Background

The application of industrial manipulators has been growing quickly during the last few years. Typically, they are programmed to execute automated repetitive tasks

that may be dangerous or risky to humans, for instance, operations that are carried out at high speed or with heavy objects involved. Depending on the application requirements, there are many scenarios in which a robotic arm can be employed. For example, it can be installed on a fixed workstation to perform repetitive tasks, to work cooperatively with a human operator or to be remotely controlled if the environment is dangerous for human beings. In the latter case, a Human-Robot interface is required to properly monitor and control the robot's actions. A typical device to manually control and supervise industrial manipulators is the Teach Pendant, whose interface is similar to a joystick. Nevertheless, this kind of user interface is not very intuitive, and it may require a lot of practice in some applications [277].

In [278], a cyber-physical system is implemented to remotely control a robot in hazardous manufacturing environments. In particular, the human worker is able to control a physical robot located in a remote working station either using a virtual model of the system or guiding a collaborative manipulator in a human-robot workstation. In the second scenario, a wearable display device is used to give visual feedback of the teleoperated robot to the human operator.

A robot can also be teleoperated using methods that exploit typical human-to-human communication interfaces, e.g., speech or gestures. Industrial environments are typically very noisy, thus making it more difficult to operate the robot using vocal commands, so gestures are usually preferred over speech [215]. Additionally, gestures can be easily executed with only one hand, by articulating a sequence of fingers and hand movements and hence, they are also easy to understand. A common gesture used in human-human interaction is to point at a part, a location or indicate a position. This is as simple as it is useful to codify real-world assembly tasks [279].

In the literature, there are other interfaces available for teleoperation that may be adopted to ensure a quick response, which improves usability. Indeed, in [280], it was seen that people perceive more accurately and prefer a system they are more comfortable with, such as a touchscreen. Currently, most of the human-robot interfaces include a touch screen, becoming easy and intuitive to handle [281]. It is worth observing that in an industrial environment, human operators may wear gloves for safety reasons, so the Teach Pendants with a touch screen interface may not be convenient, since the screen size is relatively limited. An alternative solution to such a problem could be that of enlarging the touchscreen device. However, this approach could be not convenient, as it cannot be adopted if the operator wears gloves and

large touch screen devices are usually expensive. The authors in [282] proposed an infrared-matrix sensory system, which is used to emulate a touchscreen interface that recognizes the user's actions and therefore, controls the manipulator remotely. Four infrared matrices are attached to the corners of a display, in such a way that the infrared transmitters and receivers can detect correctly the (x, y) coordinates of the fingers touching the screen.

The quality of the interaction can be enhanced when there exists a confirmation feedback within the human-robot communication, as presented in [283]. It was demonstrated that the selection accuracy while performing a task is higher than in absence of any feedback. Similarly, gestures can be used in conjunction with visual feedback markers. Nonetheless, this method often needs expensive devices since it generally works with Augmented Reality (AR) techniques.

In [284], an AR system is included in the communication loop between humans and robots with the goal of enhancing human-robot interaction. In this framework, human operators wear Microsoft HoloLens glasses to receive real-time information related to the tasks, use the air tap gesture to instruct the robots and provide feedback to the main control system. In a similar way, in [285], AR is used to upgrade the robot teleoperation experience, providing a virtual foresight of the real robot trajectories and final position.

In this work, it will be considered a scenario where a production environment does not allow the presence of human operators due to safety reasons but, it requires the robotic system to be controlled onsite. In such cases, the sensors used for monitoring the activity of the robot and receiving instructions from the human operator are crucial for achieving efficient communication between humans and robots for teleoperated tasks [11].

In recent years, a considerable state-of-the-art has grown up around the theme of hybrid make-to-stock/make-to-order manufacturing strategies [286]. In particular, make-to-order manufacturing involves high-level decisions aiming at the customization of final products, given the market demand that sees an increasing trend towards customized goods production, and consequent evolution of the production line [14]. When some functional features are available on the robot's system, such as low-level motion planning, a sensory system to supervise the workspace, status updates and the possibility to receive tasks inputs from the operator, it is possible to control the robot with high-level tasks [287]. It should be noted that an interface that allows to control

the robot with high-level tasks enables the reduction of the human operator workload, as demonstrated in [288, 289], due to the fact the worker does not intervene in the motion of each joint, but instead the robot is able to automatically compute the shortest trajectories for carrying out the operations.

This research work aims at presenting an alternative way to remotely control a robotic arm to perform assembly tasks. The proposed system emulates a touch screen behaviour exploiting a hand-tracking sensor along with a liquid-crystal display (LCD). The latter is also employed to provide visual feedback to the operator. A monocular camera streams the workspace that contains objects to be identified and grasped.

10.2 PoinTap Interface – Concept

The proposed interface aims at providing an innovative combination of resources and devices to enable human-controlled actions on an industrial manipulator. In particular, the PoinTap live feedback allows to upgrade the interaction experience of the human operator, while exploiting a simple LCD screen. The latter is enhanced to an emulated touchscreen, externally enabled using a hand-tracking device, inspired by the work proposed in [290]. The present work enriches the emulated touchscreen with the possibility of pointing to the screen while receiving live visual feedback. The whole system is imagined as a human-robot interface to perform conjoined actions within an industrial context.

The PoinTap interface system architecture is shown in Figure 10.1, which is represented in the form of blocks. In particular, the arrows highlight the data flow, clarifying the input information combinations for each output. As can be seen, the main system blocks serve as the interface between the human workspace and the robot workspace.

Given the analysed state of the art, the PoinTap interface encapsulates previous works' functionalities to develop the envisioned interface on an LCD screen. The screen shows the robot workspace scene, provided by a top-view camera, based on which the human operator interacts through a "point at part" gesture, fed to the *Hand Tracking* and *Touch Emulation* (TE) blocks. The latters process this information and emit an output corresponding to a feedback marker on the screen. The system

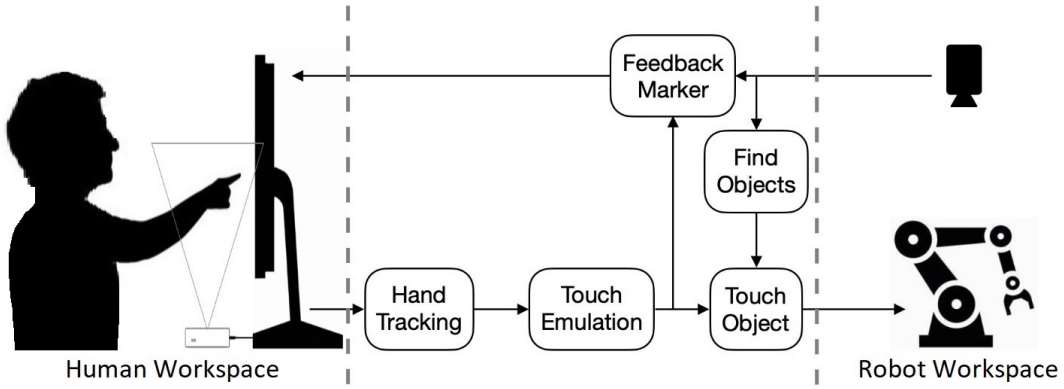


Fig. 10.1 Block diagram of the proposed system.

verifies that the pointed/tapped area on the streamed image is actually related to a recognized object and transforms its position into coordinates interpretable by the robot. Such a region is then identified as an area of interest for a desired task by the robot. In fact, the term PoinTap, introduced to denote the proposed solution, is just related to such a capability of translating the operator pointing/tapping gesture.

On one hand, Figure 10.1 helps to distinguish the information flow among input and output elements of the developed system, a top-down description comes in handy to deepen the role of every single block. On the other hand, Figure 10.2 schematically illustrates the PoinTap system as composed of several layers.

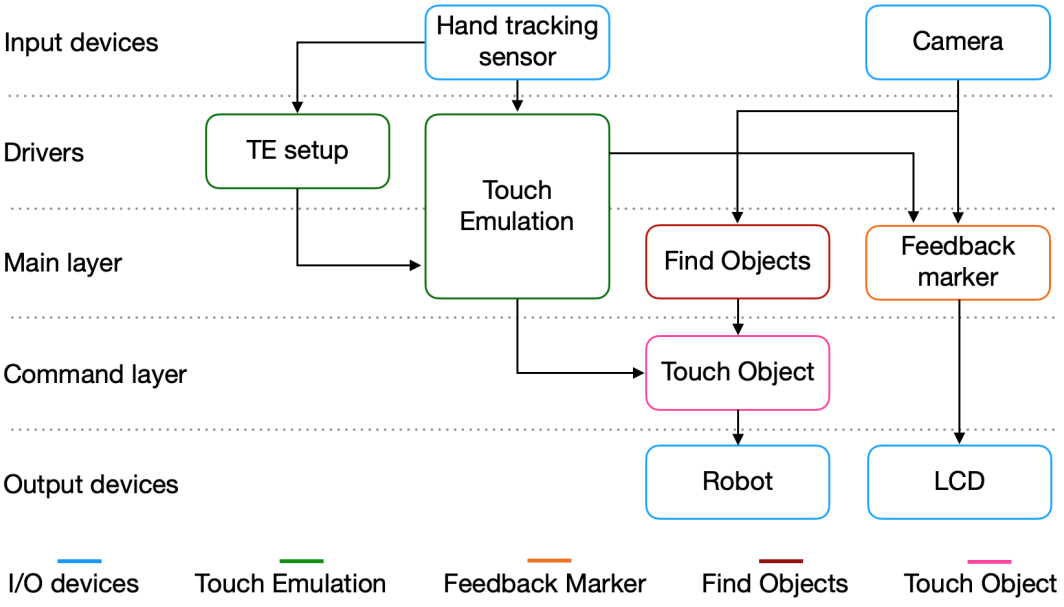


Fig. 10.2 Schematic representation of the proposed system and its division in layers.

The top and bottom layers of the scheme represent the input and output devices respectively, while the software is in the middle layers. Notice that data flows from the top to the bottom, namely, from the input devices to the output devices. The role and the actions of the various layers are illustrated afterwards, as well as the information flows into each of them:

- **Input devices:** a hand-tracking sensor is employed for recognizing human hands and the gestures, while a monocular camera positioned on top of the manipulator gathers visual data of the robot workspace.
- **Drivers:** the software tools in this layer are used as links between the input devices and the main layer. They perform a setting phase to process raw information to make it available to the command layer.
- **Main layer:** this layer represents the core of the system. IN particular, it is responsible for:
  - The visualisation and manipulation of data coming from the camera.
  - The implementation of the touch emulation block.
  - The detection and recognition of objects in the robot workspace.
- **Command layer:** this layer combines together information that comes from the user and the robot workspace. It takes the information provided by the *Touch Emulation* block and the one coming from the *Find Objects* block. The aim is to verify if the combination of the two leads to an intersection, namely, if a touched area actually corresponds to an identified object. If the latter condition is true, then the current layer sends the command to the final one.
- **Output devices:** the robot and the LCD screen are considered as output devices. The former executes a defined action while the latter shows images coming from the camera and the live feedback marker.

It is worth observing that most of the system components can be customized. This feature enables the adaptability of the generic system to both the available resources and the specific application. The PoinTap elements that can be adjusted to specific industrial application requirements are highlighted in Table 10.1.

Table 10.1 PoinTap Blocks and relative features.

Block	Features
Hand Tracking, Touch Emulation, Touch Emulation Setup	The hand-tracking sensor can be freely chosen. The development of the Touch Emulation block and its setup are not strictly sensor-dependent. In fact, the data required as input are the coordinates of the tip of the index, independently of the sensor that provides it.
Camera	The camera choice is independent of the other blocks. In particular, a custom choice can be made, however, it should be taken into account that the camera's technical specifications may affect the quality of the streamed image of the robot's working space.
Find Objects	The implementation of this block depends on the selected object detection system.
Touch Objects	This block should be used as it is. Depending on the software choices for the previous and next blocks, it may need some modifications in order to be compatible with the overall system.
Feedback Marker	This block strongly depends on the framework used to implement the PoinTap Interface.
Robot	Depending on the requirements specific to the application or depending on the available configuration, the manipulator can be changed.
LCD	The LCD screen specifications, such as the length, width and height, are free to be selected since they are saved as parameters. The only constraint is that the screen lies inside the field of view of the chosen hand-tracking sensor.

With the aim at demonstrating the PoinTap interface implementation, complete of the customizable blocks, a case study is considered in the next Section.

10.3 PoinTap Interface: Implementation

In order to provide a more detailed description of the PoinTap system development, a possible use case in the industrial context has been selected. The choice fell on one of

the most popular applications for an industrial manipulator, such as an assembly task achieved by performing a sequence of pick and place operations. In this scenario, the human worker is able to visualize the robot and a set of pieces on the screen. First, by using a “point at part" gesture, the user points at the piece of interest and, guided by the feedback mark, taps the area of interest on the screen. Then, this information is interpreted by PoinTap, which passes the data to the robot as corresponding to the first object used in the assembly task. This operation is performed one or more times to choose the other pieces needed for the completion of the assembly task. The main results obtained during the validation process, both in simulation and with a real robot, of a specific example of the envisioned industrial use case are described in Section 10.4.

10.3.1 Software and Hardware

As regards the implementation of the PoinTap system, the Robot Operating System (ROS) framework [36] has been exploited. Furthermore, as mentioned before, the development of the touch emulation block was inspired by the already available Layered Touch Panel [290], leading to the choice of the Leap Motion sensor [291] as a hand-tracking sensor. The image stream has been provided by a small and inexpensive C210 Webcam by Logitech [292] whose resolution is 640 x 480 pixels. The object detection task was entrusted to the `find_object_2d` [293], a ROS package which integrates the Find-Object application [294] provided by OpenCV [295] with the image stream coming from the webcam to implement SIFT, SURF, FAST, BRIEF and other feature detectors and descriptors to identify and recognise images from a pre-recorded database. The ROS visualization tool, *rviz*, was employed for the live feedback marker visualization, while the well-established integration with the Gazebo simulator [251] allowed for a smooth integration of the proposed interface with a simulated version of the chosen robot manipulator.

The implemented interface, even if imagined for the mentioned industrial use case, was tested on a research manipulator, given the possibility to validate it in a laboratory environment. For this purpose, a Niryo One manipulator [296] was utilized for the showcased example. Niryo One is a 6-axis desk robotic arm, mainly designed for educational and research purposes. It is Open Source and 3D printed, favouring its use as a low-cost option for technology prototyping and validation. Among the available end effectors for the manipulator, the “Gripper 1” was used.

It is worth highlighting that, to improve the system's performance, it was decided to distribute the system among several machines, leveraging thus one of the main features of ROS. The simulation software was run on a PC, the camera and vision nodes were executed on a Raspberry Pi board [44], and a further Raspberry Pi board ran the nodes in charge of interfacing and controlling the robotic arm, adapted to the case study example. A summarizing schema of the blocks developed for the PoinTap system is shown in Figure 10.3.

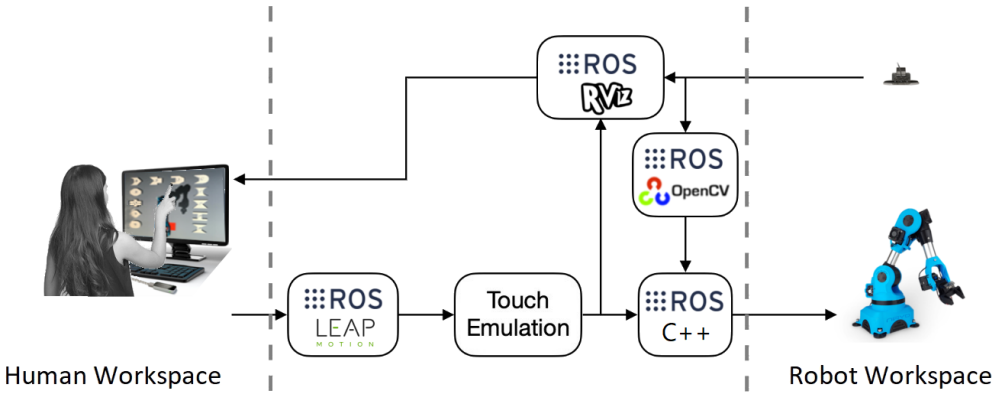


Fig. 10.3 Block diagram of the PoinTap system implementation.

Notice that blocks in the layers going from the *Input layer* to the *Command layer* (refer back to Figure 10.2) can be considered in common between the simulated version of the developed case study and its execution on the real robot. With the aim of providing a linear description of the implementation, the main PoinTap blocks (refer back to Figure 10.1) are illustrated in the next section, following the previously described layered organization.

10.3.2 Implementation

As reported in Figure 10.3, the human workspace is composed of the LCD screen and the Leap Motion sensor, both placed on a flat surface. In particular, the LCD screen serves both as an input and an output device. The positioning of the Leap Motion was considered to be perpendicular to the plane where the LCD screen lied, as it is quite common to have the described configuration on a working desk. Observe that, beyond its use as an output device to display images coming from the camera and the feedback marker, the LCD screen was used also as a reference to define the touch and virtual panels. Moreover, it is worth mentioning that the system implementation does

not depend on the chosen screen's technical specifications, with the only constraint that the latter remains within the Leap Motion field of view.

Concerning the robot working space, the robot was positioned at the centre of the world frame while the camera was placed above it, in such a way as to visualise the whole workspace. In addition, the image plane axes were set to match the plane where the robotic arm and pieces are placed, so as to have a direct correspondence of coordinates.

Touch Emulation

The implementation of this block was led by the identification of the following issues in gesture recognition:

- Understanding a pointing finger gesture may require the operator to achieve a high level of precision, resulting in an unnatural gesture execution. Achieving a precise gesture can be impractical for the user, particularly when the task is composed of a repetition of pick and place operations.
- Timing associated with the pointing gesture is relevant. In fact, a trade-off must be identified to enable the recognition procedure in an acceptable time while allowing the user to execute the supervised assembly task.
- It is complex to identify a "base point". The connection of the latter with the "tip point", corresponding to the tip of the index, defines the pointing line that intersects the pointed plane. In this case, as the user's head cannot be detected by the hand-tracking sensor, it could not have been used as the base point. Nevertheless, if the base of the finger were taken as the base point, the resulting pointing line would mismatch what is potentially expected by the operator, making the interaction unnatural.

The Touch Emulation system was designed to be independent of the above problems and to ease the interaction for the human operator by enriching the system output with a feedback marker. The Touch Emulation system proposed in this work provides a 1:1 scale between the scene image and the LCD screen, which corresponds to the touching plane. Taking into account the concepts related to the Layered Touch Panel [290], the real-time feedback was implemented by adding a virtual panel to the LCD. A virtual screen representation in *rviz* is depicted in Figure 10.4.

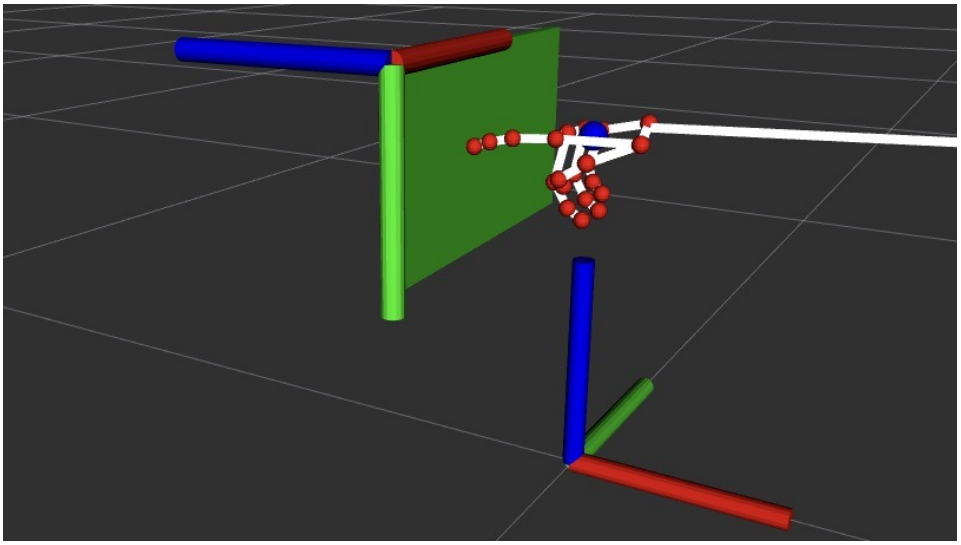


Fig. 10.4 Virtual representation of the screen in *rviz*.

Hereafter, an overview of the blocks composing the Touch Emulation system is provided.

(a) **Touch Emulation setup block:** First, a calibration procedure is necessary in order to use the Touch Emulation block. This procedure is required to be performed just once as soon as the system is set up. It consists of the screen size definition using the Leap Motion. The user touches with his/her index each screen edge clockwise, starting from the top left corner of the screen, and stores its coordinates by pressing the spacebar or the enter key. The edges' positions in space are processed to obtain the screen parameters: dimensions, position and orientation with respect to the world frame. Then, those parameters are saved on a `.yaml` file, which loads them on the ROS Parameter server at each system start. Note that the screen orientation, stored as the normal vector entering the screen, is provided by the assumption that the screen plane is perpendicular to the plane on which the hand-tracking sensor is placed.

(b) **Touch Emulation block:** After the completion of the calibration procedure, the Touch Emulation block is ready to work. Inspired by the Layered Touch Panel, in the Touch Emulation block two virtual planes parallel to the LCD were defined, at a distance equal to the *touching distance* and the *hovering distance*, respectively. The first virtual panel is used to capture a touching

event, and the second is for the hovering event. A schematic representation of the considered virtual panels is presented in Figure 10.5.

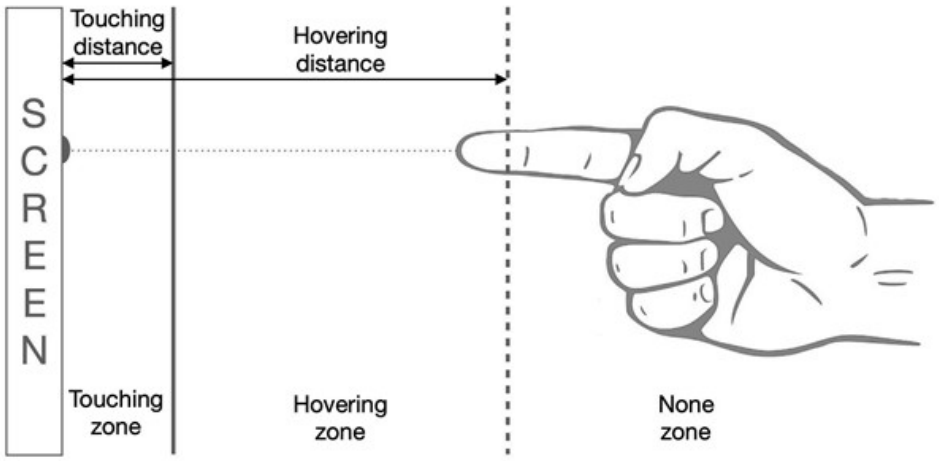


Fig. 10.5 Graphical representation of the two virtual panels.

The hovering and touching distances are received as input, along with the Leap Motion data and other parameters, such as the image dimension expressed in pixels and the pointer radius of the marker visualised on the screen.

At the *Drivers* layer, the screen parameters are used to create a new coordinate system to be associated with the screen and to make available its visualization. It was decided to publish the information about the pose (position and orientation) of the screen at a very low rate since it was assumed that the screen and the hand-tracking sensor do not move with respect to the world frame.

Subsequently within the *Main layer*, the position of the tip of the index is transformed from the hand-tracking sensor's reference frame to the screen's coordinate system. Given this information, the following conditions are checked:

- If the distance from the screen is less than a defined hovering distance, a hovering event is triggered and a message containing the position of the tip of the index is published on the dedicated ROS topic, otherwise nothing happens.
- If the tip of the index is within the hovering zone, another condition needs to be checked: if the distance from the screen is smaller than the touching distance, then the touching event is generated and a message containing the information about the index tip is published.

It should be noted that it is easier to verify these two conditions while working in the screen frame, due to the fact that only the values on the z axis are compared.

Feedback Marker

As mentioned before, feedback to the pointing gesture is usually implemented using expensive devices for AR. In this work, live feedback was integrated using low-cost devices. In fact, the LCD is used to:

- Provide a visualization of the real working space and emulate a touchscreen in conjunction with the Leap Motion hand-tracking sensor
- Display a feedback marker according to the fingertip position

The Feedback Marker block receives as input the image frames coming from the camera and the hovering position sent by the Touch Emulation block. When an image is available, the block simply displays it or, if it receives a message containing the hovering position from the Touch Emulation block, a virtual marker is included to the image.

The marker is portrayed by a black circle having a radius equal to a previously defined variable and it represents the point that the finger is going to touch. The position of the tip of the index is scaled to the screen range and converted into pixels so as to have a direct correspondence. It is worth pointing out that in this case, the position of the marker is simply the normal projection of the fingertip on the screen, which is different from previous studies in which it was given by the intersection of the plane and the pointing direction. This decision was made to give the user the freedom to touch the screen not only using the fingertip but also using the finger pad, which would lead to project the virtual marker far from the fingertip if the pointing direction was used.

Find Objects

The Find Object block gets the image stream provided by the camera and a set of images to be recognised in the working space. In particular, at setup time, using the Qt-based GUI provided by the `find_object_2d` package, the user can load

some objects that need to be recognized and detected in the form of locally stored images or take photos of the current workspace, as in this case. Notice that images of the *simulated* objects were used for recognition, both in the simulated case and the experimental testing (further details related to the experimental results are available in Section 10.4). As an object is recognised, a coloured polygon is drawn around it and a unique numerical ID is assigned to it. Furthermore, the block generates as an output a message containing the positions of the recognised objects with respect to the whole image, as soon as the application recognises a variation in the number of recognised objects.

Touch Object

The Touch Object block receives as input the information generated by the Find Objects and the Touch Emulation blocks. Firstly, when the message containing the positions of the identified objects is published by the Find Object block, the Touch Object block stores the information in an internal variable. Then, as a new message containing the touch position is made available by the Touch Emulation block, the Touch Object block checks if the position of the tip of the index corresponds to the recognised object and if the latter condition is true, it generates as output a message containing the ID of the object that has been touched. This information is then read by the robot and used for task execution. For the sake of simplicity, the condition to be verified is considered to be satisfied if the index's tip position is inside the rectangle inscribed in the polygon.

10.4 PoinTap Interface – A Case Study

In this section, the results obtained during the simulation and experimental validation are reported. It should be noted that, given the use of the ROS framework, the described PoinTap development was valid for both the simulated scenario and the experimental one. Generally, the ROS topics and node mechanisms allow for a smooth transition from the simulated case study to the real one by simply substituting simulation nodes with the ones needed to interface with the real robot. As previously anticipated, the envisioned use case sees the human operator using the PoinTap system in order to give high-level commands to the manipulator in an assembly task.

The idea is that pieces are chosen following some criteria that the manipulator is not able to evaluate, for instance, those choices that are influenced by customized product requirements.

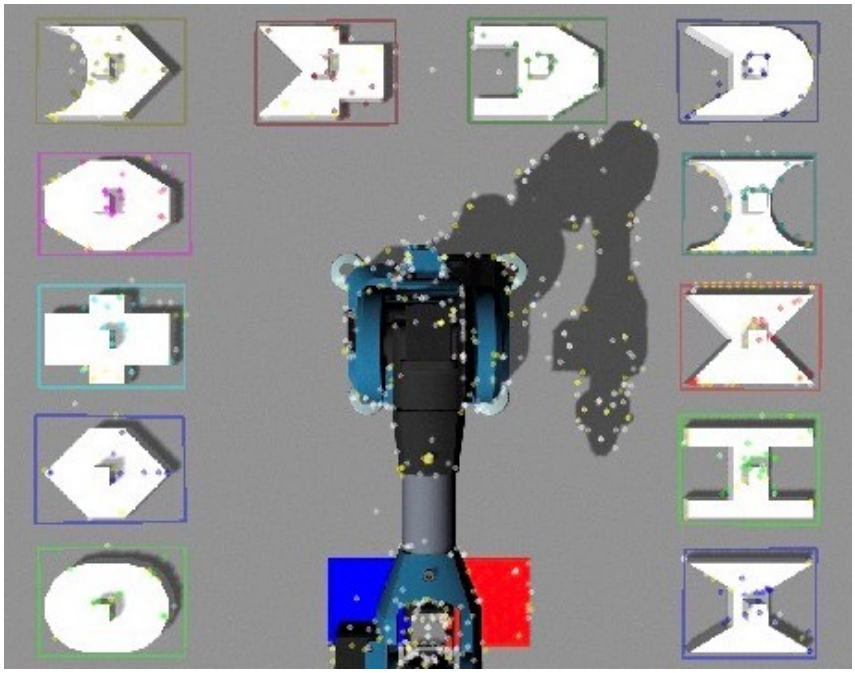
The human workspace was set up with an LCD screen positioned at a fixed pose with respect to the Leap Motion device. Regarding the robot workspace, twelve pieces were placed around the robot following a horseshoe shape. Specifically, in order to perform the experimental validation, the pieces used within the simulated environment (in Gazebo) were reproduced as accurately as possible using cardboard. As can be seen in Figure 10.6, displayed objects were enclosed in coloured polygons, indicating they were correctly recognized. Moreover, Figure 10.6a reports the twelve pieces in simulation, while the corresponding configuration in the real-world scenario is shown in Figure 10.6b.

The pieces were numbered from 1 to 12 starting from the bottom-left corner and then moving clockwise, as well as the ID related to each piece. As the human operator "point-tapped" the first object on the touch-emulated screen, the feedback marker appeared as shown in Figure 10.7.

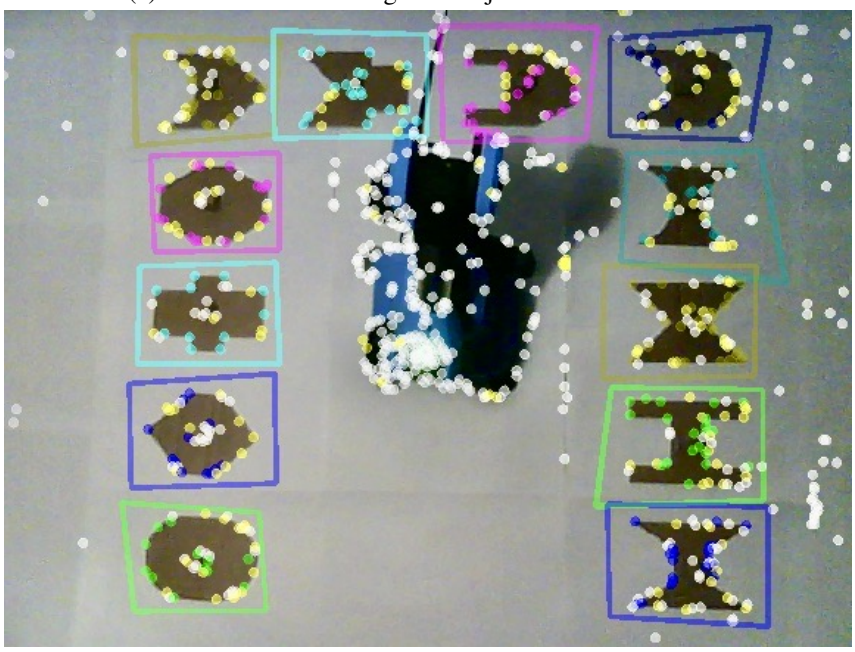
Taking into consideration the structure presented in Figure 10.3, it is worth highlighting that the simulated and real tests share the human workspace configuration, the devices within it, and the core PoinTap system including the robotic arm control and pick-and-place planning nodes. In fact, the simulation only involved the robot workspace and the devices within it, which are the robotic arm and the camera.

The PoinTap system was first tested within the simulated environment. To perform the assembly task, two pieces were selected by pointing to the screen and then, the simulated robot picked and placed them in a dedicated space for the assembly to happen. As soon as the first object (piece ID = 8, top-right corner) was point-tapped by the operator and identified by the system, the simulated Niryo One received the command and performed the first pick-and-place operation, then waited for the second object to be touched (piece ID = 5, top-left corner). As the second piece was detected and selected by touching it, the robot grasped and placed it next to the previous piece, as shown in Figure 10.8.

Subsequently, in order to experimentally validate the algorithm, two different pieces were point-tapped by the operator to command the real robot, so as to execute the operation. Maintaining the same pieces' configuration as in the simulation, the first object to be grasped was piece ID = 5, followed by piece ID = 10, as can be



(a) Visualisation of recognised objects in the simulation.



(b) Visualisation of recognised objects in the real world.

Fig. 10.6 Visualisation of the object recognition using the `find_object_2d` GUI.

seen in Figure 10.9. The experimental validation results have been recorded and they are available at [297].

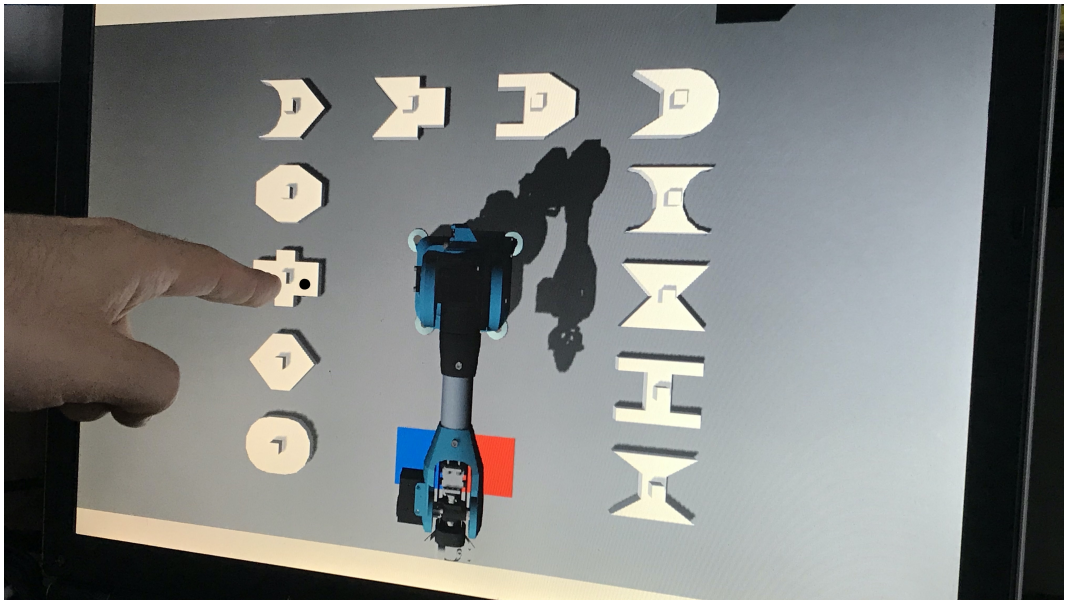


Fig. 10.7 Representation of immediate feedback (black circle) used in this system, for the simulated case study.

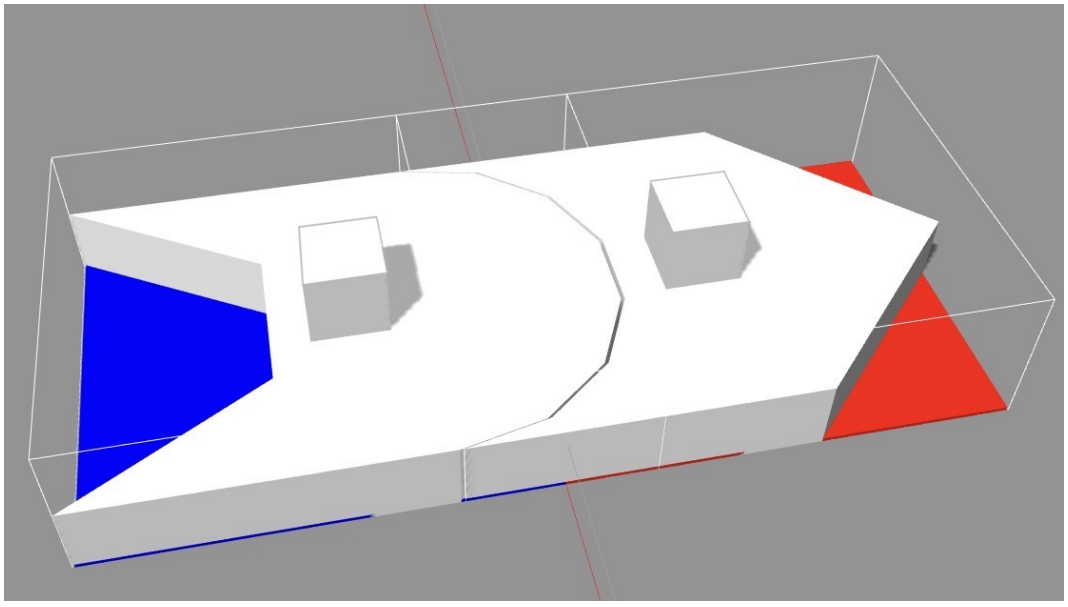
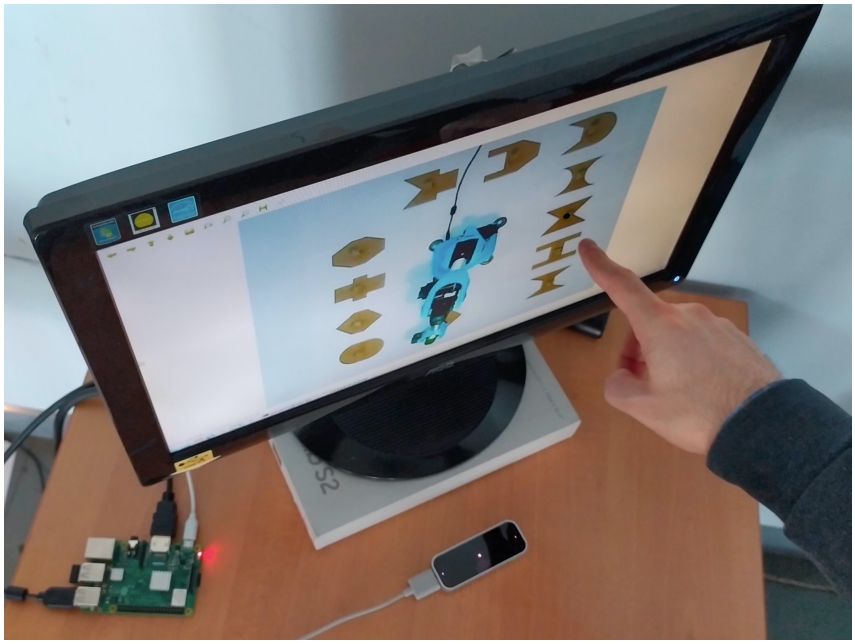


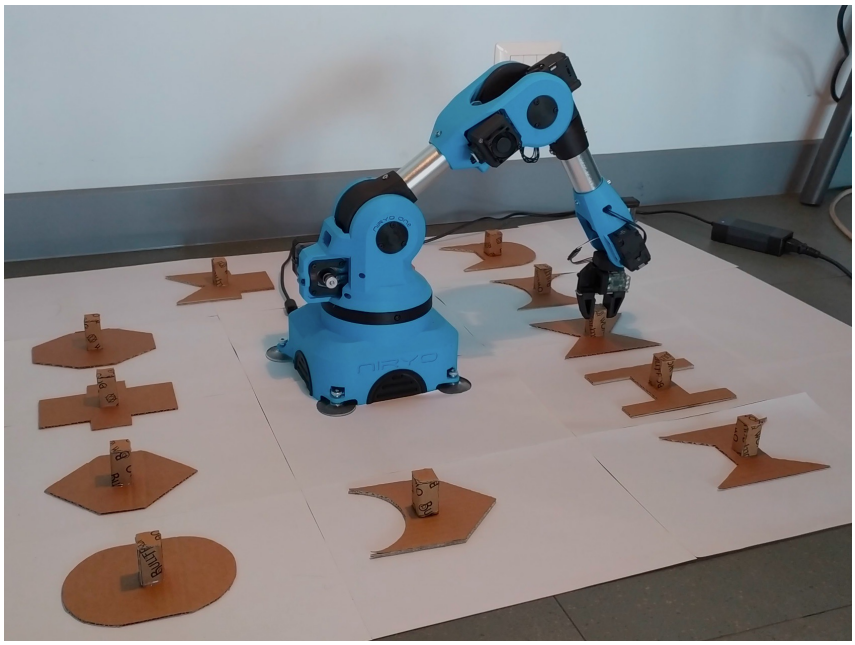
Fig. 10.8 Representation of the completed assembly task in the Gazebo world.

10.5 Discussion

This chapter presented an interface to remotely control a robotic arm, specifically for supervised pick-and-place operations within an industrial assembly task.



(a) The human operator point-taps the second piece for assembly.



(b) After the touched object is identified, the robot picks and places it, to complete the assembly.

Fig. 10.9 Pick-and-place procedure in the real-world scenario.

The PoinTap system can improve flexibility when is needed a re-organization of the assembly task. Interestingly, the sequence of objects to be assembled can be intuitively selected by point-tapping the LCD screen, without the need for trained

operators. In addition, as pointed out, human operators usually wear gloves in an industrial environment due to safety reasons, so they are not able to interact with typical touchscreen devices and the proposed approach may overcome this limitation. Furthermore, the proposed system has different functional blocks that can be customized, so they can work independently from each other, and consequently be substituted with other blocks whose functions are similar. Also, due to the fact that pieces are recognized through software, their shape is not relevant, as it is sufficient to train and develop the object detection system.

It is worth pointing out that, since the touch panel distance from the screen is tunable, the PoinTap system can be useful to avoid touching the screen so as to improve hygienic conditions. This is a topic of interest when considering health emergency situations that limit the contact of shared devices.

Nevertheless, the system presents some minor issues to be subsequently addressed. First of all, hand-tracking sensors have a limited field of view so, depending on the screen size, two or more sensors may be needed in order to cover the necessary area for a correct Touch Emulation implementation. Moreover, the requirement for an additional sensor would lead to further tuning in order to perform data fusion. Furthermore, since a monocular camera was used to monitor the scene, it is not possible for the robot to be aware of the true position of each piece, and therefore, if some pieces are initially positioned in locations that differ from the expected ones, the robot may not be able to grasp them.

The PoinTap system could be enhanced by installing the robot working space with additional vision sensors, with the aim of retrieving also the objects' positions. This would enable to correctly plan the needed trajectory to reach the object even if its position is slightly different from the expected one. Such an upgraded vision system could also improve the robot's accuracy and precision when pieces are released. Besides, depending on the LCD screen size, different positions for the hand-tracking sensor could be considered in order to cover the largest possible area and boost hand-tracking capabilities. Finally, to customize the assembly task, the range of input commands could be extended, for example, enabling the rotation of pieces or adding the possibility to cancel, pause or abort an operation.

Chapter 11

Conclusions

In this PhD dissertation, various research works related to applications of robotic systems in human-robot shared environments have been presented. In particular, the proposed works have been developed to provide a safe interaction between humans and robots in an industrial context, so as to improve productivity and flexibility while supporting human operators, as envisaged in Industry 5.0. The robotic systems that have been considered in this research work are those that are usually employed in smart manufacturing systems, such as mobile robots, manipulators and mobile manipulators.

Firstly, path planning algorithms to ensure safe navigation in environments shared with human operators have been investigated. Due to the fact that most of the traditional path planners do not take into account the human within the path optimization process, various path planning algorithms and architectures have been developed in this thesis. The supervisory global path planner is proposed as a top planning level (above the global and local ones) that generates a geometric curve that can be considered safe to be traversed. This planning hierarchy, upgraded later as an online supervisory global planner, has the aim to compute a deterministic virtual route for the robot, considered safe since it can be easily recognized by human operators. Moreover, whenever an obstacle or a human intersects the robot's trajectory, the three-level planning architecture prevents any collision with the human obstacle in a conservative way. These planning algorithms have been tested and validated experimentally on a laboratory demonstrator within a laboratory environment emulating an industrial working space.

Additionally, local planning algorithms have been studied to improve the robot's reaction against moving obstacles. In fact, a dynamic path planner developed in ROS2 has been presented, leveraging the costmap layer method which enlarges the inflated area around the moving obstacle with a Gaussian shape, whose parameters are proportional to the obstacle speed and orientation. Since the proposed approach has been only tested in a simulated environment, a future development envisages the implementation in a real mobile platform whose perception system is upgraded using vision sensors so as to be able to distinguish various obstacle types.

The path planning algorithm used to design the supervised global planner has been extended to the multi-agent case. In particular, it has been designed for patrolling in formation while avoiding collision with other agents and static obstacles. Currently, it implements a centralized architecture since it controls a small number of mobile robots simultaneously, however, it could be extended to a decentralized architecture as the number of agents increases. Moreover, in order to improve the obstacle avoidance capabilities, it could be integrated with a dynamic path planner along with an online update mechanism to maintain the route as the reference curve.

Path planning capabilities can be improved, especially when the robot navigates in an environment where the presence of human operators is expected if the robot's perception system is able to distinguish between a general obstacle and a human. Depending on the application requirements, there are several methods and sensor combinations that enhance human perception in an industrial environment. In fact, the most used algorithms and sensors have been investigated and analysed.

Furthermore, a sensor data fusion algorithm has been proposed, in which a camera-laser calibration procedure has been performed along with an object recognition system. The resulting framework has been employed for the development of meta-sensors, where AMRs are able to sense and share relevant data with other mobile robots, so as to optimize the perception and awareness of other agents. Such meta-sensors could also be employed to monitor and record human activities while performing different operations. To do so, the robot should be able to learn the path usually executed by the human worker and assimilate human-like navigation in an industrial environment, to support human activities.

The algorithms proposed here in this thesis can be considered as low-level functions, that can be integrated into an organized framework to enable a safe human-robot interaction. In particular, such a framework has been proposed to

enable safe robot assistive operations to support the human operator. Future work could be implementing the developed navigation algorithms in a mobile manipulator, and training the object manipulation algorithm based on the object affordance. Furthermore, if the robot already has low-level functionalities on its own, it is possible to give high-level commands to control the robot in a more intuitive way, so as to enhance productivity even for those novice human operators that did not receive enough work training.

Since the recognition procedure employed in most of the developed algorithms relies on AI-based techniques, a further improvement is to enrich the database with a higher amount of data samples. In this way, the system is able to generalize better the model and avoid overfitting issues. However, there is a trade-off between the data complexity and the hardware components that can handle huge amounts of data, so as to optimize the overall performance.

References

[1] IFR - International Federation of Robotics. IFR World Robotics 2020 Service Robots Report Presentation. Available online: https://ifr.org/downloads/press2018/Presentation_WR_2020.pdf (accessed January 2021).

[2] Xun Xu, Yuqian Lu, Birgit Vogel-Heuser, and Lihui Wang. Industry 4.0 and industry 5.0—inception, conception and perception. *Journal of Manufacturing Systems*, 61:530–535, 2021.

[3] Niloofar Jafari, Mohammad Azarian, and Hao Yu. Moving from industry 4.0 to industry 5.0: What are the implications for smart logistics? *Logistics*, 6(2):26, 2022.

[4] Marina Indri, Corrado Possieri, Fiorella Sibona, Pangcheng David Cen Cheng, and Vinh Duong Hoang. Supervised global path planning for mobile robots with obstacle avoidance. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 601–608. IEEE, 2019.

[5] Marina Indri, Fiorella Sibona, and Pangcheng David Cen Cheng. Sensor data fusion for smart amrs in human-shared industrial workspaces. In *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 738–743. IEEE, 2019.

[6] Marina Indri, Fiorella Sibona, Pangcheng David Cen Cheng, and Corrado Possieri. Online supervised global path planning for amrs with human-obstacle avoidance. In *2020 25th IEEE international conference on emerging technologies and factory automation (ETFA)*, volume 1, pages 1473–1479. IEEE, 2020.

[7] Pangcheng David Cen Cheng, Marina Indri, Fiorella Sibona, Matteo De Rose, and Gianluca Prato. Dynamic path planning of a mobile robot adopting a costmap layer approach in ros2. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2022.

[8] Pangcheng David Cen Cheng, Marina Indri, Corrado Possieri, Mario Sassano, and Fiorella Sibona. Path planning in formation and collision avoidance for multi-agent systems. *Nonlinear Analysis: Hybrid Systems*, 47:101293, 2023.

[9] Marina Indri, Fiorella Sibona, and Pangcheng David Cen Cheng. Sen3bot net: a meta-sensors network to enable smart factories implementation. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 719–726. IEEE, 2020.

[10] Fiorella Sibona, Pangcheng David Cen Cheng, and Marina Indri. How to improve human-robot collaborative applications through operation recognition based on human 2d motion. In *IECON 2022–48th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6. IEEE, 2022.

[11] Andrea Bonci, Pangcheng David Cen Cheng, Marina Indri, Giacomo Nabissi, and Fiorella Sibona. Human-robot perception in industrial environments: A survey. *Sensors*, 21(5):1571, 2021.

[12] Fiorella Sibona, Pangcheng David Cen Cheng, Marina Indri, and Danilo Di Prima. Pointap system: a human-robot interface to enable remotely controlled tasks. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 01–08. IEEE, 2021.

[13] Pangcheng David Cen Cheng, Fiorella Sibona, and Marina Indri. A framework for safe and intuitive human-robot interaction for assistant robotics. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4. IEEE, 2022.

[14] Marina Indri, Luca Lachello, Ivan Lazzero, Fiorella Sibona, and Stefano Trapani. Smart Sensors Applications for a New Paradigm of a Production Line. *Sensors*, 19(3, 650), 2019.

[15] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[16] Junfeng Yao, Chao Lin, Xiaobiao Xie, Andy JuAn Wang, and Chih-Cheng Hung. Path planning for virtual human motion using improved A* star algorithm. In *2010 Seventh international conference on information technology: new generations*, pages 1154–1158. IEEE, 2010.

[17] Mehmet Korkmaz and Akif Durdu. Comparison of optimal path planning algorithms. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*, pages 255–258. IEEE, 2018.

[18] Sven Koenig and Maxim Likhachev. Dˆ* Lite. *Aaai/iaai*, 15, 2002.

[19] Dave Ferguson, Maxim Likhachev, and Anthony Stentz. A guide to heuristic-based path planning. In *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)*, pages 9–18, 2005.

[20] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Practical search techniques in path planning for autonomous driving. *Ann Arbor*, 1001(48105):18–80, 2008.

[21] Saeid Sedighi, Duong-Van Nguyen, and Klaus-Dieter Kuhnert. Guided hybrid a-star path planning algorithm for valet parking applications. In *2019 5th international conference on control, automation and robotics (ICCAR)*, pages 570–575. IEEE, 2019.

[22] Lydia Kavraki and J-C Latombe. Randomized preprocessing of configuration for fast path planning. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 2138–2145. IEEE, 1994.

[23] S LAVALLE. Rapidly-exploring random trees: a new tool for path planning. *Research Report 9811*, 1998.

[24] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised*, volume 118. Springer, 2017.

[25] JJ Kuffner and SM LaValle. An efficient approach to path planning using balanced bidirectional RRT search. *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep*, 2005.

[26] Ehsan Taheri, Mohammad Hossein Ferdowsi, and Mohammad Danesh. Fuzzy Greedy RRT Path Planning Algorithm in a Complex Configuration Space. *International Journal of Control, Automation and Systems*, 16(6):3026–3035, 2018.

[27] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[28] Zaid Tahir, Ahmed H Qureshi, Yasar Ayaz, and Raheel Nawaz. Potentially guided bidirectionalized RRT* for fast optimal path planning in cluttered environments. *Robotics and Autonomous Systems*, 108:13–27, 2018.

[29] Jianping Tu and Simon X Yang. Genetic algorithm based path planning for a mobile robot. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 1, pages 1221–1226. IEEE, 2003.

[30] Byron Hernández and Eduardo Giraldo. A Review of Path Planning and Control for Autonomous Robots. In *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*, pages 1–6. IEEE, 2018.

[31] Gianluca Antonelli, Stefano Chiaverini, and Giuseppe Fusco. A fuzzy-logic-based approach for mobile robot path tracking. *IEEE Transactions on Fuzzy Systems*, 15(2):211–221, 2007.

[32] Elon Rimon and Daniel E Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, 1992.

[33] C Possieri and M Sassano. Patrolling and collision avoidance beyond classical navigation functions. In *European Control Conference (ECC)*, pages 1821–1826. IEEE, 2018.

[34] C Possieri and M Sassano. Motion Planning, Formation Control and Obstacle Avoidance for Multi-Agent Systems. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 879–884. IEEE, 2018.

[35] Corrado Possieri and Antonio Tornambè. On polynomial vector fields having a given affine variety as attractive and invariant set: application to robotics. *International Journal of Control*, 88(5):1001–1025, 2015.

[36] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[37] ROS Navigation Stack Official Documentation. Accessed: April 2019.

[38] MATLAB Robotics System Toolbox ROS Interface Documentation. Accessed: April 2019.

[39] ROS actionlib Official Documentation. Accessed: April 2019.

[40] Shama Baldi, Nina Maric, Rolf Dornberger, and Thomas Hanne. Pathfinding Optimization when Solving the Paparazzi Problem Comparing A* and Dijkstra's Algorithm. In *2018 6th International Symposium on Computational and Business Intelligence (ISCBI)*, pages 16–22. IEEE, 2018.

[41] Brian P Gerkey and Kurt Konolige. Planning and control in unstructured terrain. In *ICRA Workshop on Path Planning on Costmaps*, 2008.

[42] MobileRobots Inc. *Pioneer 3 Operations Manual*.

[43] SICK AG Waldkirch. *PLMS200/211/221/291 Laser Measurement Systems*.

[44] Raspberry Pi site. Accessed: April 2019.

[45] ROS gmapping Official Documentation. Accessed: April 2019.

[46] Giorgio Grisetti, Cyrill Stachniss, Wolfram Burgard, et al. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1):34, 2007.

[47] ROS rviz Official Documentation. Accessed: April 2019.

[48] Test Cases with the Pioneer3DX. Accessed: April 2019.

[49] Chaymaa Lamini, Said Benhlima, and Ali Elbekri. Genetic algorithm based approach for autonomous mobile robot path planning. *Procedia Computer Science*, 127(C):180–189, 2018.

[50] Hasan Mujtaba and Gajendra Singh. Safe Navigation of Mobile Robot Using A* Algorithm. *International Journal of Applied Engineering Research*, 12(16):5532–5539, 2017.

[51] Hong-Mei Zhang, Ming-Long Li, and Le Yang. Safe path planning of mobile robot based on improved A* algorithm in complex terrains. *Algorithms*, 11(4):44, 2018.

[52] BK Patle, Anish Pandey, DRK Parhi, A Jagadeesh, et al. A review: On path planning strategies for navigation of mobile robot. *Defence Technology*, 2019.

[53] Ulises Orozco-Rosas, Oscar Montiel, and Roberto Sepúlveda. Mobile robot path planning using membrane evolutionary artificial potential field. *Applied Soft Computing*, 77:236–251, 2019.

[54] Hang Li and Andrey V Savkin. An algorithm for safe navigation of mobile robots by a sensor network in dynamic cluttered industrial environments. *Robotics and Computer-Integrated Manufacturing*, 54:65–82, 2018.

[55] Matthias Lutz, Christian Verbeek, and Christian Schlegel. Towards a robot fleet for intra-logistic tasks: Combining free robot navigation with multi-robot coordination at bottlenecks. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4. IEEE, 2016.

[56] Maximilian Papa, David Kaselautzke, Kemajl Stuja, and Walter Wölfel. Different safety certifiable concepts for mobile robots in industrial environments. *Annals of DAAAM & Proceedings*, 29, 2018.

[57] Preview of R15.08 Industrial Mobile Robot Safety, by Michael Gerstenberger (Chair of the R15.08 Drafting Subcommittee). Accessed: May 2020.

[58] Alexandra Markis, Maximilian Papa, David Kaselautzke, Michael Rathmair, Vinzenz Sattinger, and Mathias Brandstötter. Safety of mobile robot systems in industrial applications. 05 2019.

[59] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *CoRR*, abs/1804.02767, 2018.

[60] Zhang Juan, Liu Wei, and Peng Xiamei. Research on technology transfer readiness level and its application in university technology innovation management. In *2010 International Conference on E-Business and E-Government*, pages 1904–1907. IEEE, 2010.

[61] Docker official website. Accessed: June 2020.

[62] MinSu Chae, HwaMin Lee, and Kiyeol Lee. A performance comparison of linux containers and virtual machines using Docker and KVM. *Cluster Computing*, 22(1):1765–1775, 2019.

[63] Lucas Gumbi and Hossana Twinomurinzi. SMME Readiness for Smart Manufacturing (4IR) Adoption: A Systematic Review. In *Conference on e-Business, e-Services and e-Society*, pages 41–54. Springer, 2020.

[64] Online supervised global planning algorithm for AMRs with human obstacle avoidance.

[65] Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. Integrated online trajectory planning and optimization in distinctive topologies. *Robotics and Autonomous Systems*, 88:142–153, 2017.

[66] Matteo De Rose. LiDAR-based Dynamic Path Planning of a mobile robot adopting a costmap layer approach in ROS2. Master's thesis, Politecnico di Torino, 2021.

[67] Francisco Rubio, Francisco Valero, and Carlos Llopis-Albert. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16(2):172988141983959, 2019.

[68] Steven M. LaValle. *Planning Algorithms*, chapter Introductory Material. Cambridge University Press, USA, 2006.

[69] Laurène Claussmann, Marc Revilloud, Dominique Gruyer, and Sébastien Glaser. A review of motion planning for highway autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 21(5):1826–1848, May 2020.

[70] Pablo Marin-Plaza, Ahmed Hussein, David Martin, and Arturo de la Escalera. Global and local path planning study in a ROS-based research platform for autonomous vehicles. *Journal of Advanced Transportation*, 2018, 2018.

[71] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The office marathon: Robust navigation in an indoor office environment. In *2010 IEEE International Conference on Robotics and Automation*, pages 300–307. IEEE, 2010.

[72] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[73] S Quinlan and O Khatib. Elastic bands: connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807 vol.2. IEEE Comput. Soc. Press, 1993.

[74] Christoph Rösmann, Wendelin Feiten, Thomas Wösch, Frank Hoffmann, and Torsten Bertram. Trajectory modification considering dynamic constraints of autonomous robots. In *ROBOTIK 2012; 7th German Conference on Robotics*, pages 1–6. VDE, 2012.

[75] B Cybulski, A Wegierska, and Grzegorz Granosik. Accuracy comparison of navigation local planners on ROS-based mobile robot. In *2019 12th International Workshop on Robot Motion and Control (RoMoCo)*, pages 104–111. IEEE, 2019.

[76] Isira Naotunna and Theeraphong Wongratanaphisan. Comparison of ROS Local Planners with Differential Drive Heavy Robotic System. In *2020 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pages 1–6. IEEE, 2020.

[77] Yuya Maruyama, Shinpei Kato, and Takuya Azumi. Exploring the performance of ROS2. In *2016 International Conference on Embedded Software (EMSOFT)*, pages 1–10, 2016.

[78] Why ROS 2? Accessed: April 2022.

[79] Steve Macenski, Francisco Martin, Ruffin White, and Jonatan Gines Clavero. The Marathon 2: A Navigation System. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2718–2725. IEEE, 2020.

[80] Navigation Dynamic Obstacle Integration. Student Program challenge launched in 2021 by the Nav2 community. Accessed: April 2022.

[81] Michele Colledanchise and Petter Ögren. *Behavior Trees in Robotics and AI: An introduction*. CRC Press, Jul 2018.

[82] David V Lu, Dave Hershberger, and William D Smart. Layered costmaps for context-sensitive navigation. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 709–715. IEEE, 2014.

[83] TEB local planner. Accessed: December 2022.

[84] DWB Controller. Accessed: December 2022.

[85] Ismail Hakki Savci, Abdurrahman Yilmaz, Sadettin Karaman, Hakan Ocakli, and Hakan Temeltas. Improving Navigation Stack of a ROS-Enabled Industrial Autonomous Mobile Robot (AMR) to be Incorporated in a Large-Scale Automotive Production. *The International Journal of Advanced Manufacturing Technology*, pages 1–22, 2022.

[86] Francisco Martín, Jonatan Ginés Clavero, Vicente Matellán, and Francisco J Rodríguez. Plansys2: A planning system framework for ROS2. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9742–9749. IEEE, 2021.

[87] Huixu Dong, Ching-Yen Weng, Chuangqiang Guo, Haoyong Yu, and I-Ming Chen. Real-Time Avoidance Strategy of Dynamic Obstacles via Half Model-Free Detection and Tracking With 2D Lidar for Mobile Robots. *IEEE/ASME transactions on mechatronics*, 26(4):2215–2225, 2021.

[88] Mateusz Przybyła. Detection and tracking of 2D geometric obstacles from LRF data. In *2017 11th International Workshop on Robot Motion and Control (RoMoCo)*, pages 135–141, 2017.

[89] Taketoshi Mori, Takahiro Sato, Hiroshi Noguchi, Masamichi Shimosaka, Rui Fukui, and Tomomasa Sato. Moving objects detection and classification based on trajectories of LRF scan data on a grid map. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2606–2611, 2010.

[90] Franz Albers, Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. *Online Trajectory Optimization and Navigation in Dynamic Environments in ROS*, pages 241–274. Springer, 01 2019.

[91] Alyssa Pierson, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Navigating congested environments with risk level sets. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5712–5719, 2018.

[92] Blob Detection Using OpenCV. Accessed: April 2022.

[93] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[94] DOL: A costmap-based Dynamic Path Planning approach in ROS2. Accessed: April 2022.

[95] J. Yu and S. M. LaValle. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics*, 32(5):1163–1177, 2016.

[96] B. Tang, K. Xiang, M. Pang, and Z. Zhanxia. Multi-robot path planning using an improved self-adaptive particle swarm optimization. *International Journal of Advanced Robotic Systems*, 17(5), 2020.

[97] J.Yu. Average case constant factor time and distance optimal multi-robot path planning in well-connected environments. *Autonomous Robots*, 44:469–483, 2020.

[98] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, 1992.

[99] M. C. De Gennaro and A. Jadbabaie. Formation control for a cooperative multiagent system using decentralized navigation functions. In *2006 American Control Conference*, Minneapolis, MN, USA, 2006.

[100] H. G. Tanner and A. Kumar. Towards decentralization of multi-robot navigation functions. In *2005 IEEE International Conference on Robotics and Automation*, pages 4132–4137, Barcelona, Spain, 2005.

[101] H. G. Tanner and A. Boddu. Multiagent navigation functions revisited. *IEEE Transactions on Robotics*, 28(6):1346–1359, 2012.

[102] H. Su, X. Wang, and Z. Lin. Flocking of multi-agents with a virtual leader. *IEEE Transactions on Automatic Control*, 54(2):293–307, 2009.

[103] C. Belta and V. Kumar. Abstraction and control for groups of robots. *IEEE Transactions on Robotics*, 20(5):865–875, 2004.

[104] M. I. El-Hawwary and M. Maggiore. Distributed circular formation stabilization for dynamic unicycles. *IEEE Transactions on Automatic Control*, 58(1):149–162, 2013.

[105] P. Tabuada, G. J. Pappas, and P. Lima. Motion feasibility of multi-agent formations. *IEEE Transactions on Robotics*, 21(3):387–392, 2005.

[106] N. E. Leonard and E. Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. In *40th IEEE Conference on Decision and Control*, volume 3, pages 2968–2973, Orlando, FL, USA, 2001.

[107] H. G. Tanner, G. J. Pappas, and V. Kumar. Leader-to-formation stability. *IEEE Transactions on Robotics and Automation*, 20(3):443–455, 2004.

[108] A. Loria, J. Dasdemir, and N. Alvarez Jarquin. Leader–follower formation and tracking control of mobile robots along straight paths. *IEEE Transactions on Control Systems Technology*, 24(2):727–732, 2016.

[109] S. He, M. Wang, S. Dai, and F. Luo. Leader–follower formation control of usvs with prescribed performance and collision avoidance. *IEEE Transactions on Industrial Informatics*, 15(1):572–581, 2019.

[110] S. Dai, S. He, X. Chen, and X. Jin. Adaptive leader–follower formation control of nonholonomic mobile robots with prescribed transient and steady-state performance. *IEEE Transactions on Industrial Informatics*, 16(6):3662–3671, 2020.

[111] Z. Peng, G. Wen, S. Yang, and A. Rahmani. Distributed consensus-based formation control for nonholonomic wheeled mobile robots using adaptive neural network. *Nonlinear Dynamics*, 86:605–622, 2016.

[112] C. Zong, Z. Ji, L. Tian, and Y. Zhang. Distributed multi-robot formation control based on bipartite consensus with time-varying delays. *IEEE Access*, 7:144790–144798, 2019.

[113] K. K. Oh, M. C. Park, and H. S. Ahn. A survey of multi-agent formation control. *Automatica*, 53:424–440, 2015.

[114] C Possieri and M Sassano. Patrolling and collision avoidance beyond classical navigation functions. In *European Control Conference*, pages 1821–1826, Limassol, Cyprus, 2018.

[115] Sean Wilson, Paul Glotfelter, Li Wang, Siddharth Mayya, Gennaro Notomista, Mark Mote, and Magnus Egerstedt. The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems. *IEEE Control Systems Magazine*, 40(1):26–44, 2020.

[116] R Goebel, Ricardo G Sanfelice, and Andrew R Teel. *Hybrid dynamical systems: modeling stability, and robustness*. Princeton University Press, Princeton, NJ, 2012.

[117] David A Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms*. Springer Science & Business Media, New York, 2015.

[118] David A Cox, John Little, and Donal O'Shea. *Using algebraic geometry*. Springer Science & Business Media, New York, 2006.

[119] Bruno Siciliano and Oussama Khatib. *Handbook of Robotics*. Springer Verlag, 2008.

[120] Francesco Martinelli, Corrado Possieri, and Antonio Tornambe. Motion planning for a unicycle-like mobile robot, using algebraic attractive curves. In *22nd Mediterranean Conference on Control and Automation*, pages 1335–1340, Palermo, Italy, 2014. IEEE.

[121] Thulasi Mylvaganam, Mario Sassano, and Alessandro Astolfi. A differential game approach to multi-agent collision avoidance. *IEEE Transactions on Automatic Control*, 62(8):4229–4235, 2017.

[122] Hassan K Khalil. *Nonlinear Systems*. Prentice-Hall, New Jersey, 2002.

[123] Thulasi Mylvaganam, Corrado Possieri, and Mario Sassano. Global stabilization of nonlinear systems via hybrid implementation of dynamic continuous-time local controllers. *Automatica*, 106:401–405, 2019.

[124] Alexandre Seuret and Christophe Prieur. Event-triggered sampling algorithms based on a Lyapunov function. In *50th IEEE Conference on Decision and Control*, pages 6128–6133, Orlando, FL, USA, 2011.

[125] Romain Postoyan, Paulo Tabuada, Dragan Nesic, and Adolfo Anta Martinez. A framework for the event-triggered stabilization of nonlinear systems. *IEEE Transactions on Automatic Control*, 60(4):982–996, 2015.

[126] Jun Chai, Pedro Casau, and Ricardo G Sanfelice. Analysis and design of event-triggered control algorithms using hybrid systems tools. In *56th IEEE Conference on Decision and Control*, pages 6057–6062, Melbourne, Australia, 2017.

[127] Corrado Possieri and Antonio Tornambe. On f-invariant and attractive affine varieties for continuous-time polynomial systems: the case of robot motion planning. In *53rd IEEE Conference on Decision and Control*, pages 3751–3756, Los Angeles, CA, USA, 2014.

[128] Experiment 1 Demo Video. Available online: https://youtu.be/Ziz63_-BmqE (accessed April 2021).

[129] Experiment 2 Demo Video. Available online: https://youtu.be/Bc43Eh8PTf0 (accessed April 2021).

[130] Experiment 3 Demo Video. Available online: https://youtu.be/Tt4CqQsyNi4 (accessed April 2021).

[131] Markus Bader, Andreas Richtsfeld, M Suchi, G Todoran, W Holl, and Markus Vincze. Balancing centralised control with vehicle autonomy in AGV systems for industrial acceptance. In *11th Int. Conf. on Autonomic and Autonom. Sys*, 2015.

[132] Jacobus Theunissen, Hang Xu, Ray Y Zhong, and Xun Xu. Smart AGV System for Manufacturing Shopfloor in the Context of Industry 4.0. In *2018 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pages 1–6. IEEE, 2018.

[133] Murat Köseoğlu, Orkan Murat Çelik, and Ömer Pektaş. Design of an autonomous mobile robot based on ROS. In *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, pages 1–5. IEEE, 2017.

[134] Harvey B Mitchell. *Multi-sensor data fusion: an introduction*. Springer Science & Business Media, 2007.

[135] Xian Wu, Jing Ren, Yujun Wu, and Jianwang Shao. Study on target tracking based on vision and radar sensor fusion. Technical report, SAE Technical Paper, 2018.

[136] Zhi Yan, Li Sun, Tom Duckctr, and Nicola Bellotto. Multisensor Online Transfer Learning for 3D LiDAR-based Human Detection with a Mobile Robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7635–7640. IEEE, 2018.

[137] Varuna De Silva, Jamie Roche, and Ahmet Kondoz. Robust fusion of LiDAR and wide-angle camera data for autonomous mobile robots. *Sensors*, 18(8, 2730), 2018.

[138] T. Linder, D. Griesser, N. Vaskevicius, and K. O. Arras. Towards Accurate 3D Person Detection and Localization from RGB-D in Cluttered Environments. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Workshop on Robotics for Logistics in Warehouses and Environments Shared with Humans*, 2018.

[139] Luciano Spinello and Roland Siegwart. Human detection using multimodal and multidimensional features. In *2008 IEEE International Conference on Robotics and Automation*, pages 3264–3269. IEEE, 2008.

[140] Angus Leigh, Joelle Pineau, Nicolas Olmedo, and Hong Zhang. Person tracking and following with 2D laser scanners. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 726–733. IEEE, 2015.

[141] Varuna De Silva, Jamie Roche, and Ahmet M. Kondoz. Fusion of LiDAR and Camera Sensor Data for Environment Sensing in Driverless Vehicles. *CoRR*, abs/1710.06230, 2017.

[142] Khalil Ahmad Yousef, Bassam Mohd, Khalid Al-Widyan, and Thaier Hayajneh. Extrinsic calibration of camera and 2D laser sensors without overlap. *Sensors*, 17(10, 2346), 2017.

[143] Qilong Zhang and Robert Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2301–2306. IEEE, 2004.

[144] Zhaozheng Hu, Yicheng Li, Na Li, and Bin Zhao. Extrinsic calibration of 2-D laser rangefinder and camera from single shot based on minimal solution. *IEEE Transactions on Instrumentation and Measurement*, 65(4):915–929, 2016.

[145] Juan Li, Xiang He, and Jia Li. 2D LiDAR and camera fusion in 3D modeling of indoor environment. In *2015 National Aerospace and Electronics Conference (NAECON)*, pages 379–383. IEEE, 2015.

[146] V. Digani, F. Caramaschi, L. Sabattini, C. Secchi, and C. Fantuzzi. Obstacle avoidance for industrial AGVs. In *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 227–232, Sept 2014.

[147] Nayan M. Kakoty, Mridusmita Mazumdar, and Durlav Sonowal. Mobile Robot Navigation in Unknown Dynamic Environment Inspired by Human Pedestrian Behavior. In Chhabi Rani Panigrahi, Arun K. Pujari, Sudip Misra, Bibudhendu Pati, and Kuan-Ching Li, editors, *Progress in Advanced Computing and Intelligent Engineering*, pages 441–451, Singapore, 2019. Springer Singapore.

[148] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.

[149] Janne Heikkila, Olli Silven, et al. A four-step camera calibration procedure with implicit image correction. In *cvpr*, volume 97, page 1106, 1997.

[150] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*, 2018.

[151] Gary Marcus. Deep Learning: A Critical Appraisal. *CoRR*, abs/1801.00631, 2018.

[152] Junwei Han, Dingwen Zhang, Gong Cheng, Nian Liu, and Dong Xu. Advanced deep-learning techniques for salient and category-specific object detection: a survey. *IEEE Signal Processing Magazine*, 35(1):84–100, 2018.

[153] ONVIF Official Site. Available online: https://www.onvif.org/.

[154] GStreamer official webite. Available online: https://gstreamer.freedesktop.org/.

[155] Human detection and avoidance with the Pioneer 3DX.

[156] X. Wu, V. Goepp, and A. Siadat. Cyber physical production systems: A review of design and implementation approaches. In *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1588–1592. IEEE, 2019.

[157] Andrea Bonci, Massimiliano Pirani, and Sauro Longhi. An embedded database technology perspective in cyber-physical production systems. *Procedia Manufacturing*, 11:830–837, 2017.

[158] Andrea Bonci, Massimiliano Pirani, Aldo Franco Dragoni, Alessandro Cucchiarelli, and Sauro Longhi. The relational model: In search for lean and mean cps technology. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pages 127–132. IEEE, 2017.

[159] Andrea Bonci, Massimiliano Pirani, Alessandro Cucchiarelli, Alessandro Carbonari, Berardo Naticchia, and Sauro Longhi. A review of recursive holarchies for viable systems in cpss. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pages 37–42. IEEE, 2018.

[160] Y. Yildiz. Cyberphysical human systems: An introduction to the special issue. *IEEE Control Systems Magazine*, 40(6):26–28, 2020.

[161] A. P. Dani, I. Salehi, G. Rotithor, D. Trombetta, and H. Ravichandar. Human-in-the-loop robot control for human-robot collaboration: Human intention estimation and safe trajectory tracking control for collaborative tasks. *IEEE Control Systems Magazine*, 40(6):29–56, 2020.

[162] Matteo Pantano, Daniel Regulin, Benjamin Lutz, and Dongheui Lee. A human-cyber-physical system approach to lean automation using an industrie 4.0 reference architecture. *Procedia Manufacturing*, 51:1082–1090, 2020.

[163] Andrea Bonci, Sauro Longhi, Giacomo Nabissi, and Federica Verdini. Predictive maintenance system using motor current signal analysis for industrial robot. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1453–1456. IEEE, 2019.

[164] Abdelfetah Hentout, Mustapha Aouache, Abderraouf Maoudj, and Isma Akli. Human–robot interaction in industrial collaborative robotics: a literature review of the decade 2008–2017. *Advanced Robotics*, 33(15-16):764–799, 2019.

[165] ISO ISO. Iso/ts 15066: 2016: Robots and robotic devices–collaborative robots. *Geneva, Switzerland: International Organization for Standardization*, 2016.

[166] Martin J Rosenstrauch and Jörg Krüger. Safe human-robot-collaboration-introduction and experiment using iso/ts 15066. In *2017 3rd International conference on control, automation and robotics (ICCAR)*, pages 740–744. IEEE, 2017.

[167] P Aivaliotis, S Aivaliotis, C Gkournelos, K Kokkalis, G Michalos, and S Makris. Power and force limiting on industrial robots for human-robot collaboration. *Robotics and Computer-Integrated Manufacturing*, 59:346–360, 2019.

[168] Bitao Yao, Zude Zhou, Lihui Wang, Wenjun Xu, and Quan Liu. Sensorless external force detection for industrial manipulators to facilitate physical human-robot interaction. *Journal of Mechanical Science and Technology*, 32(10):4909–4923, 2018.

[169] Xinyu Wang, Chenguang Yang, Zhaojie Ju, Hongbin Ma, and Mengyin Fu. Robot manipulator self-identification for surrounding obstacle detection. *Multimedia Tools and Applications*, 76(5):6495–6520, 2017.

[170] Fabrizio Flacco and Alessandro De Luca. Real-time computation of distance to dynamic obstacles with multiple depth sensors. *IEEE Robotics and Automation Letters*, 2(1):56–63, 2016.

[171] Thadeu Brito, Jose Lima, Pedro Costa, and Luis Piardi. Dynamic collision avoidance system for a manipulator based on rgb-d data. In *Iberian Robotics conference*, pages 643–654. Springer, 2017.

[172] Elias Matsas and George-Christopher Vosniakos. Design of a virtual reality training system for human–robot collaboration in manufacturing tasks. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 11(2):139–153, 2017.

[173] Adar Vit and Guy Shani. Comparing rgb-d sensors for close range outdoor agricultural phenotyping. *Sensors*, 18(12):4413, 2018.

[174] Jaime Zabalza, Zixiang Fei, Cuebong Wong, Yijun Yan, Carmelo Mineo, Erfu Yang, Tony Rodden, Jorn Mehnen, Quang-Cuong Pham, and Jinchang Ren. Smart sensing and adaptive reasoning for enabling industrial robots with interactive human-robot capabilities in dynamic environments—a case study. *Sensors*, 19(6):1354, 2019.

[175] Hang Su, Salih Ertug Ovur, Zhijun Li, Yingbai Hu, Jiehao Li, Alois Knoll, Giancarlo Ferrigno, and Elena De Momi. Internet of things (iot)-based collaborative control of a redundant manipulator for teleoperated minimally invasive surgeries. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9737–9742. IEEE, 2020.

[176] Hang Su, Juan Sandoval, Pierre Vieyres, Gérard Poisson, Giancarlo Ferrigno, and Elena De Momi. Safety-enhanced collaborative framework for teleoperated minimally invasive surgery using a 7-dof torque-controlled robot. *International Journal of Control, Automation and Systems*, 16(6):2915–2923, 2018.

[177] Yitao Ding and Ulrike Thomas. Collision avoidance with proximity servoing for redundant serial robot manipulators. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10249–10255. IEEE, 2020.

[178] Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 55:248 – 266, 2018.

[179] Maram Sakr, Waleed Uddin, and H. F. Machiel Van der Loos. Orthographic vision-based interface with motion-tracking system for robot arm teleoperation: A comparative study. In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, page 424–426, 2020.

[180] S. Yang, W. Xu, Z. Liu, Z. Zhou, and D. T. Pham. Multi-source vision perception for human-robot collaboration in manufacturing. In *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, 2018.

[181] Pedro Neto, Miguel Simão, Nuno Mendes, and Mohammad Safeea. Gesture-based human-robot interaction for human assistance in manufacturing. *The International Journal of Advanced Manufacturing Technology*, 101(1):119–135, 2019.

[182] Elisa Digo, Mattia Antonelli, Valerio Cornagliotto, Stefano Pastorelli, and Laura Gastaldi. Collection and analysis of human upper limbs motion features for collaborative robotic applications. *Robotics*, 9(2):33, 2020.

[183] Matteo Ragaglia, Andrea Maria Zanchettin, and Paolo Rocco. Trajectory generation algorithm for safe human-robot collaboration based on multiple depth sensor measurements. *Mechatronics*, 55:267–281, 2018.

[184] Liam Lynch, Thomas Newe, John Clifford, Joseph Coleman, Joseph Walsh, and Daniel Toal. Automated ground vehicle (agv) and sensor technologies-a review. In *2018 12th International Conference on Sensing Technology (ICST)*, pages 347–352. IEEE, 2018.

[185] Gabriel Fedorko, Stanislav Honus, and Roland Salai. Comparison of the traditional and autonomous agv systems. In *MATEC Web of Conferences*, volume 134, page 00013. EDP Sciences, 2017.

[186] Shengguo Zhou, Guanghe Cheng, Qinglong Meng, Haoyue Lin, Zhiwei Du, and Fucai Wang. Development of multi-sensor information fusion and agv navigation system. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 1, pages 2043–2046. IEEE, 2020.

[187] Emmanuel A Oyekanlu, Alexander C Smith, Windsor P Thomas, Grethel Mulroy, Dave Hitesh, Matthew Ramsey, David J Kuhn, Jason D Mcghinnis, Steven C Buonavita, Nickolus A Looper, et al. A review of recent advances in

automated guided vehicle technologies: Integration challenges and research areas for 5g-based smart manufacturing applications. *IEEE Access*, 8:202312–202353, 2020.

[188] Ying Zhang, Cui-Hua Zhang, and Xuyang Shao. User preference-aware navigation for mobile robot in domestic via defined virtual area. *Journal of Network and Computer Applications*, 173:102885, 2021.

[189] Kishan Chandan, Xiaohan Zhang, Jack Albertson, Xiaoyang Zhang, Yao Liu, and Shiqi Zhang. Guided 360-degree visual perception for mobile telepresence robots. In *The RSS-2020 Workshop on Closing the Academia to Real-World Gap in Service Robotics*, 2020.

[190] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[191] Jing Luo, Zhidong Lin, Yanan Li, and Chenguang Yang. A teleoperation framework for mobile robots based on shared control. *IEEE Robotics and Automation Letters*, 5(2):377–384, 2019.

[192] Christoforos Mavrogiannis, Alena M Hutchinson, John Macdonald, Patrícia Alves-Oliveira, and Ross A Knepper. Effects of distinct robot navigation strategies on human behavior in a crowded environment. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 421–430. IEEE, 2019.

[193] Sriram Siva and Hao Zhang. Robot perceptual adaptation to environment changes for long-term human teammate following. *The International Journal of Robotics Research*, page 0278364919896625, 2020.

[194] Fei Han, Sriram Siva, and Hao Zhang. Scalable representation learning for long-term augmented reality-based information delivery in collaborative human-robot perception. In *International Conference on Human-Computer Interaction*, pages 47–62. Springer, 2019.

[195] Lefteris Benos, Avital Bechar, and Dionysis Bochtis. Safety and ergonomics in human-robot interactive agricultural operations. *Biosystems Engineering*, 200:55–72, 2020.

[196] Juan P Vasconez, George A Kantor, and Fernando A Auat Cheein. Human–robot interaction in agriculture: A survey and current challenges. *Biosystems engineering*, 179:35–48, 2019.

[197] WZ Wang, RQ Wang, and GH Chen. Path planning model of mobile robots in the context of crowds. *arXiv preprint arXiv:2009.04625*, 2020.

[198] Jiawei Li, Lu Lu, Leidi Zhao, Cong Wang, and Junhui Li. An integrated approach for robotic sit-to-stand assistance: Control framework design and human intention recognition. *Control Engineering Practice*, 107:104680, 2021.

[199] William K Juel, Frederik Haarslev, Eduardo R Ramírez, Emanuela Marchetti, Kerstin Fischer, Danish Shaikh, Poramate Manoonpong, Christian Hauch, Leon Bodenhagen, and Norbert Krüger. Smooth robot: Design for a novel modular welfare robot. *Journal of Intelligent & Robotic Systems*, 98(1):19–37, 2020.

[200] Prasanna Kolar, Patrick Benavidez, and Mo Jamshidi. Survey of datafusion techniques for laser and vision based sensor integration for autonomous navigation. *Sensors*, 20(8):2180, 2020.

[201] Santosh Balajee Banisetty and David Feil-Seifer. Towards a unified planner for socially-aware navigation. *arXiv preprint arXiv:1810.00966*, 2018.

[202] Francisco Marques, Duarte Gonçalves, José Barata, and Pedro Santana. Human-aware navigation for autonomous mobile robots for intra-factory logistics. In *International Workshop on Symbiotic Interaction*, pages 79–85. Springer, 2017.

[203] Mourad A Kenk, M Hassaballah, and Jean-François Brethé. Human-aware robot navigation in logistics warehouses. In *ICINCO (2)*, pages 371–378, 2019.

[204] Andreea Blaga, Cristian Militaru, Ady-Daniel Mezei, and Levente Tamas. Augmented reality integration into mes for connected workers. *Robotics and Computer-Integrated Manufacturing*, 68:102057.

[205] Julia Berg, Albrecht Lottermoser, Christoph Richter, and Gunther Reinhart. Human-robot-interaction for mobile industrial robot teams. *Procedia CIRP*, 79:614–619, 2019.

[206] Dingping Chen, Jilin He, Guanyu Chen, Xiaopeng Yu, Miaolei He, Youwen Yang, Junsong Li, and Xuanyi Zhou. Human-robot skill transfer systems for mobile robot based on multi sensor fusion. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1354–1359. IEEE, 2020.

[207] Christof Röhrig and Daniel Heß. Omniman: A mobile assistive robot for intralogistics applications. *Engineering Letters*, 27(4), 2019.

[208] Christof Röhrig, Daniel Heß, C Röhrig, D Heß, C Röhrig, D Heß, J Bleja, U Grossmann, B Horster, A Roß, et al. Mobile manipulation for human-robot collaboration in intralogistics. In *IAENG Transactions on Engineering Sciences-Special Issue for the International Association of Engineers Conferences 2019*, volume 24, pages 459–466. World Scientific, 2019.

[209] ISO ISO. Iso 10218-1:2012-01, "robots and robotic devices - safety requirements for industrial robots - part 1: Robots (iso 10218-1:2011). *International Organization for Standardization, Standard DIN EN ISO*, 2012.

[210] ISO ISO. ISO 3691-4:2020 - 53.060-53-ICS Industrial trucks — Safety requirements and verification — Part 4: Driverless industrial trucks and their systems. Standard, International Organization for Standardization, Geneva, CH, 2020.

[211] Aaron Cofield, Zaid El-Shair, and Samir A Rawashdeh. A humanoid robot object perception approach using depth images. In *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, pages 437–442. IEEE, 2019.

[212] Przemyslaw A Lasota, Terrence Fong, Julie A Shah, et al. *A survey of methods for safe human-robot interaction*. Now Publishers, 2017.

[213] José Saenz, Christian Vogel, Felix Penzlin, and Norbert Elkmann. Safeguarding collaborative mobile manipulators-evaluation of the valeri workspace monitoring system. *Procedia Manufacturing*, 11:47–54, 2017.

[214] Mohammed Diab, Mihai Pomarlan, Daniel Beßler, Aliakbar Akbari, Jan Rosell, John Bateman, and Michael Beetz. Skillman—a skill-based robotic manipulation framework based on perception and reasoning. *Robotics and Autonomous Systems*, 134:103653, 2020.

[215] Gi Hyun Lim, Eurico Pedrosa, Filipe Amaral, Nuno Lau, Artur Pereira, Paulo Dias, José Luís Azevedo, Bernardo Cunha, and Luis Paulo Reis. Rich and robust human-robot interaction on gesture recognition for assembly tasks. In *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 159–164. IEEE, 2017.

[216] Gi Hyun Lim, Eurico Pedrosa, Filipe Amaral, Ricardo Dias, Artur Pereira, Nuno Lau, José Luís Azevedo, Bernardo Cunha, and Luis Paulo Reis. Human-robot collaboration and safety management for logistics and manipulation tasks. In *Iberian Robotics conference*, pages 15–27. Springer, 2017.

[217] Niki Kousi, George Michalos, Sotirios Aivaliotis, and Sotiris Makris. An outlook on future assembly systems introducing robotic mobile dual arm workers. *Procedia CIRP*, 72:33–38, 2018.

[218] Andreas Schlotzhauer, Lukas Kaiser, and Mathias Brandstötter. Safety of industrial applications with sensitive mobile manipulators–hazards and related safety measures. In *Austrian Robotics Workshop 2018*, page 43, 2018.

[219] Hossein Karami, Kourosh Darvish, and Fulvio Mastrogiovanni. A task allocation approach for human-robot collaboration in product defects inspection scenarios. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1127–1134. IEEE, 2020.

[220] Kourosh Darvish, Barbara Bruno, Enrico Simetti, Fulvio Mastrogiovanni, and Giuseppe Casalino. Interleaved online task planning, simulation, task allocation and motion control for flexible human-robot cooperation. In *2018*

27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pages 58–65. IEEE, 2018.

[221] Mingxuan Chen, Caibing Liu, and Guanglong Du. A human–robot interface for mobile manipulator. *Intelligent Service Robotics*, 11(3):269–278, 2018.

[222] Gorkem Anil Al, Pedro Estrela, and Uriel Martinez-Hernandez. Towards an intuitive human-robot interaction based on hand gesture recognition and proximity sensors. In *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 330–335. IEEE, 2020.

[223] Wansoo Kim, Pietro Balatti, Edoardo Lamon, and Arash Ajoudani. Mocaman: A mobile and reconfigurable collaborative robot assistant for conjoined human-robot actions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10191–10197. IEEE, 2020.

[224] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *The International Journal of Robotics Research*, 36(2):142–149, 2017.

[225] Angeliki Zacharaki, Ioannis Kostavelis, Antonios Gasteratos, and Ioannis Dokas. Safety bounds in human robot interaction: A survey. *Safety science*, 127:104667, 2020.

[226] ANSI/ITSDF B56.5-2019, Safety Standard for Driverless, Automatic Guided Industrial Vehicles and Automated Functions of Manned Industrial Vehicles (Revision of ANSI/ITSDF B56.5-2012). Standard, American National Standards Institute/Industrial Truck Standards Development Foundation, Geneva, CH, 2019.

[227] Zbar ROS Node Documentation page. Available online: http://wiki.ros.org/zbar_ros (accessed on January 2021).

[228] M. Indri, S. Trapani, A. Bonci, and M. Pirani. Integration of a production efficiency tool with a general robot task modeling approach. *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1:1273–1280, 2018.

[229] Guillaume Demesure, Damien Trentesaux, Michael Defoort, Abdelghani Bekrar, Hind Bril, Mohamed Djemaï, and André Thomas. Smartness versus embeddability: a tradeoff for the deployment of smart AGVs in industry. In *Service Orientation in Holonic and Multi-Agent Manufacturing*, pages 395–406. Springer, 2018.

[230] Giuseppe Fragapane, Dmitry Ivanov, Mirco Peron, Fabio Sgarbossa, and Jan Ola Strandhagen. Increasing flexibility and productivity in Industry 4.0 production networks with autonomous mobile robots and smart intralogistics. *Annals of Operations Research*, pages 1–19, 2020.

[231] A Liaqat, W Hutabarat, D Tiwari, L Tinkler, D Harra, B Morgan, A Taylor, T Lu, and A Tiwari. Autonomous mobile robots in manufacturing: Highway Code development, simulation, and testing. *The International Journal of Advanced Manufacturing Technology*, 104(9-12):4617–4628, 2019.

[232] Rafia Inam, Klaus Raizer, Alberto Hata, Ricardo Souza, Elena Forsman, Enyu Cao, and Shaolei Wang. Risk assessment for human-robot collaboration in an automated warehouse scenario. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 743–751. IEEE, 2018.

[233] Qiong Liu, Pengxiang Hua, Awais Sultan, Longzhang Shen, Egon Mueller, and Frank Boerner. Study of the integration of robot in cyber-physical production systems. In *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 367–370. IEEE, 2019.

[234] Yang Lu. Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6:1–10, 2017.

[235] Hasan A Poonawala and Mark W Spong. Cooperative visibility maintenance in se (3) for multi-robot-networks with limited field-of-view sensors. *Control Theory and Technology*, 15(4):246–257, 2017.

[236] Mingyang Geng, Shuqi Liu, and Zhaoxia Wu. Sensor fusion-based cooperative trail following for autonomous multi-robot system. *Sensors*, 19(4):823, 2019.

[237] Jianhua Zhang, Ruyu Liu, Kejie Yin, Zengyuan Wang, Mengping Gui, and Shengyong Chen. Intelligent collaborative localization among air-ground robots for industrial environment perception. *IEEE Transactions on Industrial Electronics*, 66(12):9673–9681, 2018.

[238] A Khalid, P Kirisci, Z Ghrairi, KD Thoben, and J Pannek. Towards implementing safety and security concepts for human-robot collaboration in the context of Industry 4.0. In *39th International MATADOR Conference on Advanced Manufacturing (Manchester, UK)*, pages 0–7, 2017.

[239] Hamed Bozorgi, Xuan Tung Truong, Hung Manh La, and Trung Dung Ngo. 2D laser and 3D camera data integration and filtering for human trajectory tracking. In *2021 IEEE/SICE International Symposium on System Integration (SII)*, pages 634–639. IEEE, 2021.

[240] Andrey Rudenko, Tomasz P Kucner, Chittaranjan S Swaminathan, Ravi T Chadalavada, Kai O Arras, and Achim J Lilienthal. Thör: Human-robot navigation data collection and accurate motion trajectories dataset. *IEEE Robotics and Automation Letters*, 5(2):676–682, 2020.

[241] Paul Schydlo, Mirko Rakovic, Lorenzo Jamone, and José Santos-Victor. Anticipation in human-robot cooperation: A recurrent neural network approach for multiple action sequences prediction. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5909–5914. IEEE, 2018.

[242] Przemyslaw A. Lasota and Julie A. Shah. A multiple-predictor approach to human motion prediction. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2300–2307, 2017.

[243] Vaibhav V Unhelkar, Przemyslaw A Lasota, Quirin Tyroller, Rares-Darius Buhai, Laurie Marceau, Barbara Deml, and Julie A Shah. Human-aware robotic assistant for collaborative assembly: Integrating human motion prediction with planning in time. *IEEE Robotics and Automation Letters*, 3(3):2394–2401, 2018.

[244] Hongyi Liu and Lihui Wang. Human motion prediction for human-robot collaboration. *Journal of Manufacturing Systems*, 44:287–294, 2017.

[245] Martijn Cramer, Jeroen Cramer, Karel Kellens, and Eric Demeester. Towards robust intention estimation based on object affordance enabling natural human-robot collaboration in assembly tasks. *Procedia CIRP*, 78:255–260, 2018.

[246] Peng Wang, Hongyi Liu, Lihui Wang, and Robert X Gao. Deep learning-based human motion recognition for predictive context-aware human-robot collaboration. *CIRP annals*, 67(1):17–20, 2018.

[247] Yara Rizk, Mariette Awad, and Edward W Tunstel. Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys (CSUR)*, 52(2):1–31, 2019.

[248] SPARC. Robotics 2020 Multi-Annual Roadmap. Accessed: May 2020.

[249] A. Bonci, M. Pirani, and S. Longhi. Robotics 4.0: Performance improvement made easy. In *22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2017)*, 2017.

[250] Meta-Sensor AMRs - DEMO. Accessed: May 2020.

[251] Gazebo Official site. Accessed: May 2020.

[252] Fiorella Sibona and Marina Indri. Data-driven framework to improve collaborative human-robot flexible manufacturing applications. In *IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6. IEEE, 2021.

[253] Dirk Merkel. Docker: lightweight Linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

[254] Jeremy Howard and Sylvain Gugger. Fastai: a layered API for deep learning. *Information*, 11(2):108, 2020.

[255] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter development team. Jupyter notebooks – a publishing

format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press, 2016.

[256] Amy Tabb. Docker fastai repository. Available online: https://github.com/amy-tabb/fastai-docker-example (accessed May 2022).

[257] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[258] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[259] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, page 1100612. International Society for Optics and Photonics, 2019.

[260] Praveen Kumar Reddy Maddikunta, Quoc-Viet Pham, B Prabadevi, Natarajan Deepa, Kapal Dev, Thippa Reddy Gadekallu, Rukhsana Ruby, and Madhusanka Liyanage. Industry 5.0: A survey on enabling technologies and potential applications. *Journal of Industrial Information Integration*, 26:100257, 2022.

[261] Lina Zhou, Souren Paul, Haluk Demirkan, Lingyao Yuan, Jim Spohrer, Michelle Zhou, and Julie Basu. Intelligence augmentation: Towards building human-machine symbiotic relationship. *AIS Transactions on Human-Computer Interaction*, 13(2):243–264, 2021.

[262] Shufei Li, Ruobing Wang, Pai Zheng, and Lihui Wang. Towards proactive human–robot collaboration: A foreseeable cognitive manufacturing paradigm. *Journal of Manufacturing Systems*, 60:547–552, 2021.

[263] Junming Fan, Pai Zheng, and Shufei Li. Vision-based holistic scene understanding towards proactive human–robot collaboration. *Robotics and Computer-Integrated Manufacturing*, 75:102304, 2022.

[264] Zhengxue Zhou, Leihui Li, Alexander Fürsterling, Hjalte Joshua Durocher, Jesper Mouridsen, and Xuping Zhang. Learning-based object detection and localization for a mobile robot manipulator in sme production. *Robotics and Computer-Integrated Manufacturing*, 73:102229, 2022.

[265] Xiaoqian Huang, Mohamad Halwani, Rajkumar Muthusamy, Abdulla Ayyad, Dewald Swart, Lakmal Seneviratne, Dongming Gan, and Yahya Zweiri. Real-time grasping strategies using event camera. *Journal of Intelligent Manufacturing*, pages 1–23, 2022.

[266] Florian Schuster, Uwe Sponholz, Bastian Engelmann, and Jan Schmitt. A user study on AR-assisted industrial assembly. In *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 135–140. IEEE, 2020.

[267] Wesley P Chan, Geoffrey Hanks, Maram Sakr, Tiger Zuo, HF Machiel Van der Loos, and Elizabeth Croft. An augmented reality human-robot physical collaboration interface design for shared, large-scale, labour-intensive manufacturing tasks. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11308–11313. IEEE, 2020.

[268] Chen Li, Andreas Kornmaaler Hansen, Dimitrios Chrysostomou, Simon Bøgh, and Ole Madsen. Bringing a natural language-enabled virtual assistant to industrial mobile robots for learning, training and assistance of manufacturing tasks. In *2022 IEEE/SICE International Symposium on System Integration (SII)*, pages 238–243. IEEE, 2022.

[269] Sarah Al-Hussaini, Shantanu Thakar, Hyojeong Kim, Pradeep Rajendran, Brual C Shah, Jeremy A Marvel, and Satyandra K Gupta. Human-supervised semi-autonomous mobile manipulators for safely and efficiently executing machine tending tasks. *arXiv preprint arXiv:2010.04899*, 2020.

[270] Dongdong Qin, Andong Liu, Jianming Xu, Wen-An Zhang, and Li Yu. Learning from human demonstrations for wheel mobile manipulator: An unscented model predictive control approach. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[271] Mohammed Hassanin, Salman Khan, and Murat Tahtali. Visual affordance and function understanding: A survey. *ACM Computing Surveys (CSUR)*, 54(3):1–35, 2021.

[272] Fu-Jen Chu, Ruinian Xu, Chao Tang, and Patricio A Vela. Recognizing object affordances to support scene reasoning for manipulation tasks. *arXiv preprint arXiv:1909.05770*, 2019.

[273] HongWei Sun, JiXiang Yang, DingWei Li, and Han Ding. An on-line tool path smoothing algorithm for 6R robot manipulator with geometric and dynamic constraints. *Science China Technological Sciences*, 64(9):1907–1919, 2021.

[274] Joseph Mirabel, Florent Lamiraux, Thuc Long Ha, Alexis Nicolin, Olivier Stasse, and Sébastien Boria. Performing manufacturing tasks with a mobile manipulator: from motion planning to sensor based motion control. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 159–164. IEEE, 2021.

[275] Yuta Oba, Kota Weaver, Anand Parwal, Hideki Nagasue, and Makoto Fujishima. High-accuracy pose estimation method for workpiece exchange automation by a mobile manipulator. *CIRP Annals*, 70(1):357–360, 2021.

[276] Michalis Logothetis, Charalampos P Bechlioulis, and Kostas J Kyriakopoulos. Decentralized impedance control of mobile robotic manipulators for collaborative object handling with a human operator. In *2021 29th Mediterranean Conference on Control and Automation (MED)*, pages 741–746. IEEE, 2021.

[277] Kamil Krot and Vitalii Kutia. Intuitive methods of industrial robot programming in advanced manufacturing systems. In *International Conference on Intelligent Systems in Production Engineering and Maintenance*, pages 205–214. Springer, 2018.

[278] Hongyi Liu and Lihui Wang. Remote human–robot collaboration: A cyber–physical system application for hazard manufacturing environment. *Journal of manufacturing systems*, 54:24–34, 2020.

[279] Brian Gleeson, Karon MacLean, Amir Haddadi, Elizabeth Croft, and Javier Alcazar. Gestures for industry intuitive human-robot communication from human observation. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 349–356. IEEE, 2013.

[280] Andreas Dünser, Martin Lochner, Ulrich Engelke, and David Rozado Fernandez. Visual and manual control for human-robot teleoperation. *IEEE computer graphics and applications*, 35(3):22–32, 2015.

[281] Julia Berg and Shuang Lu. Review of interfaces for industrial human-robot interaction. *Current Robotics Reports*, 1(2):27–34, 2020.

[282] Stephen Bier, Rui Li, and Weitian Wang. A full-dimensional robot teleoperation platform. In *2020 11th International Conference on Mechanical and Aerospace Engineering (ICMAE)*, pages 186–191. IEEE, 2020.

[283] Camilo Perez Quintero, Romeo Tatsambon, Mona Gridseth, and Martin Jägersand. Visual pointing gestures for bi-directional human robot interaction in a pick-and-place task. In *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 349–354. IEEE, 2015.

[284] Niki Kousi, Christos Stoubos, Christos Gkournelos, George Michalos, and Sotiris Makris. Enabling human robot interaction in flexible robotic assembly lines: An augmented reality based software suite. *Procedia CIRP*, 81:1429–1434, 2019.

[285] Michael E Walker, Hooman Hedayati, and Daniel Szafir. Robot teleoperation with augmented reality virtual surrogates. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 202–210. IEEE, 2019.

[286] Thyagaraj S Kuthambalayan and Samiran Bera. A review of the literature on mixed make-to-stock/make-to-order production systems: major findings and directions for future research. *International Journal of Services and Operations Management*, 37(3):372–406, 2020.

[287] Vivek Annem, Pradeep Rajendran, Shantanu Thakar, and Satyandra K Gupta. Towards remote teleoperation of a semi-autonomous mobile manipulator system in machine tending tasks. In *International Manufacturing Science*

and Engineering Conference, volume 58745, page V001T02A027. American Society of Mechanical Engineers, 2019.

[288] Todor Stoyanov, Robert Krug, Andrey Kiselev, Da Sun, and Amy Loutfi. Assisted telemanipulation: A stack-of-tasks approach to remote manipulator control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[289] Ashish Singh, Stela H Seo, Yasmeen Hashish, Masayuki Nakane, James E Young, and Andrea Bunt. An interface for remote robotic manipulator control that reduces task load and fatigue. In *2013 IEEE RO-MAN*, pages 738–743. IEEE, 2013.

[290] Yujin Tsukada and Takeshi Hoshino. Layered touch panel: the input device with two touch panel layers. In *CHI'02 Extended Abstracts on Human Factors in Computing Systems*, pages 584–585, 2002.

[291] Leap motion developer website. Available online: https://developer-archive.leapmotion.com/documentation/v2/cpp/index.html (accessed May 2021).

[292] Webcam c210 specifications. Available online: https://support.logi.com/hc/en-au/articles/360023462133-C210-Technical-Specifications (accessed May 2021).

[293] find-object ROS package. Available online: https://github.com/introlab/find-object (accessed May 2021).

[294] M. Labbé. Find-Object interface. Available online: http://introlab.github.io/find-object (accessed May 2021).

[295] Opencv website. Available online: https://opencv.org (accessed May 2021).

[296] NIRYO. Niryo one mechanical specifications, 2018.

[297] Experimental test video demo. Available online: https://youtu.be/7HvFw0sa3Lg.

[298] R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*. Springer, New York, 2009.

Appendix A

Proofs of theorems, lemmas and propositions of Chapter 5

A.1 Proof of Theorem 1

The proof follows the same lines of the proof of [123, Thm. 1] with minor adaptations to consider the stability of set $\mathscr{V} \times \mathscr{A}$ rather than of the origin and to consider the fact that k is constant during flows. Letting $\xi = \mathrm{col}(x,k)$ and

$$F(\xi) = \begin{bmatrix} f(\xi) \\ 0 \end{bmatrix}, \qquad G(\xi) = \begin{bmatrix} x \\ \operatorname{argmin}_{\kappa \in \Xi(\xi)} \Pi(\xi,\kappa) \end{bmatrix},$$

the dynamics of the hybrid system (5.10) can be rewritten as

$$\dot{\xi} = F(\xi), \qquad \xi \in \mathscr{C}, \tag{A.1a}$$

$$\xi^+ \in G(\xi), \qquad \xi \in \mathscr{D}. \tag{A.1b}$$

We firstly show that system (A.1) is well-posed. Since, by assumption, the functions ℓ and ρ are continuous, the sets $\mathscr{C}$ and $\mathscr{D}$ defined in (5.8) are closed. Furthermore, since f is continuous and hence locally bounded, the flow map $F(\xi)$ is outer semicontinuous and locally bounded for all $\xi \in \mathscr{C}$; see [298, Cor. 5.20]. On the other hand, under the hypotheses of the theorem, by [298, Ex. 5.22] the set-valued mapping $\xi \rightrightarrows \operatorname{argmin}_{\kappa \in \Xi(\xi)} \Pi(\xi,\kappa)$ is outer semicontinuous and locally bounded, thus implying that also $G(\xi)$ is outer semicontinuous and locally bounded for all $\xi \in \mathscr{D}$. Finally,

since $\Xi(\xi)$ is nonempty for all $\xi \in \mathscr{D}$ by Assumption 2, system (A.1) satisfies the "Hybrid Basic Conditions" stated in [116, Ass. 6.5] and hence it is well-posed by [116, Thm. 6.8].

Note that, by construction, if $k(0,0) \in \mathscr{A}$ then $k(t,j) \in \mathscr{A}$ for all $(t,j) \in \mathrm{dom}(k)$. Therefore, since $\mathscr{C} \cup \mathscr{D} = \mathbb{R}^n \times \mathscr{A}$ maximal solution of (A.1) starting in $\mathbb{R}^n \times \mathscr{A}$ are either complete or blow up in finite time. We can now prove the asymptotic stability of the set $\mathscr{V} := \mathscr{V} \times \mathscr{A}$ for system (A.1). To this end, consider the Lyapunov function $\bar{V}(\xi) = V(x)$ that satisfies

$$\bar{V}(\xi) \geqslant \alpha_1(\|\xi\|_{\bar{\mathscr{V}}}), \qquad \forall \xi \in \mathscr{C} \cup \mathscr{D} \cup G(\mathscr{D}), \tag{A.2a}$$

$$\bar{V}(\xi) \leqslant \alpha_2(\|\xi\|_{\bar{\mathscr{V}}}), \qquad \forall \xi \in \mathscr{C} \cup \mathscr{D} \cup G(\mathscr{D}), \tag{A.2b}$$

$$\langle \nabla \bar{V}(\xi), F(\xi) \rangle \leqslant -\mu \rho(\xi), \qquad \forall \xi \in \mathscr{C}, \tag{A.2c}$$

$$\bar{V}(g) - \bar{V}(\xi) = 0, \qquad \forall \xi \in \mathscr{D}, g \in G(\xi), \tag{A.2d}$$

where $\rho(\xi) = \rho(x)$ for all $\xi \in \mathscr{C}$, $\rho(\xi) \in \mathscr{P}\mathscr{D}(\bar{\mathscr{V}})$. By (A.2c) and (A.2d), such a function is monotonically non-increasing along solutions to system (A.1), thus implying that its sub-level sets, which are compact by (A.2a), are positively invariant with respect to system (A.1). Thus, solutions to (A.1) starting in $\mathbb{R}^n \times \mathscr{A}$ are complete and the set $\bar{\mathscr{V}}$ is stable for system (A.1).

It remains to prove uniform convergence of solutions to (A.1) starting in $\mathbb{R}^n \times \mathscr{A}$ to the set $\bar{\mathscr{V}}$, that is for any $r > 0$ and $\varepsilon > 0$ there is $T > 0$ such that each solution to system (A.1) starting in $r\mathscr{B} \times \mathscr{A}$ satisfies $\|\xi(t,j)\|_{\bar{\mathscr{V}}} \leqslant \varepsilon$ for all $(t,j) \in \mathrm{dom}(\xi)$ such that $t + j \geqslant T$. Given $r > \varepsilon > 0$, let $c_0 > 0$ be such that $(r\mathscr{B} \times \mathscr{A}) \subset \mathscr{S}_0 := \{\xi \in \mathbb{R}^n \times \mathscr{A} : V(\xi) \leqslant c_0\}$ and let $c_1 > 0$ be such that $\mathscr{S}_1 := \{\xi \in \mathbb{R}^n \times \mathscr{A} : V(\xi) \leqslant c_1\} \subset (\varepsilon\mathscr{B} \times \mathscr{A})$. Note that these two constants exist by (A.2a) and (A.2b), respectively. Thus, letting ρ be such that (A.2c) holds, define

$$\vartheta := \inf_{\xi \in \mathscr{S}_0 \setminus \mathscr{S}_1} \rho(\xi),$$

which is a strictly positive constant since $\overline{\mathscr{S}_0 \setminus \mathscr{S}_1}$ is a compact set and $\bar{\mathscr{V}} \cap (\overline{\mathscr{S}_0 \setminus \mathscr{S}_1}) = \emptyset$ due to the fact that $\bar{\mathscr{V}} \subset \mathscr{S}_1$. Furthermore, by considering that f is C^z for some sufficiently large $z \in \mathbb{N}$, there is a constant $\nu > 0$ such that

$$\langle \nabla(\ell(\xi) + \rho(\xi)), F(\xi) \rangle \leqslant \nu, \qquad \forall \xi \in \mathscr{S}_0 \setminus \mathscr{S}_1. \tag{A.3}$$

Therefore, define the constant

$$T := 1 + \frac{c_0(\vartheta(1-\mu)+\nu)}{\vartheta^2(1-\mu)\mu},$$

and assume by contradiction that there is a solution $\xi(t,j)$ to system (A.1) with $\xi(0,0) \in r\mathscr{B} \times \mathscr{A}$ that stays in $\mathscr{S}_0 \setminus \mathscr{S}_1$ for all $(t,j) \in \mathrm{dom}(\xi)$ such that $t+j \leqslant T$. Since $\xi^+ \in G(\xi)$, for all $h \in \mathbb{N}$, $h \geqslant 1$, such that $t_h + h \leqslant T$ it results that

$$\ell(\xi(t_h,h)) + \rho(\xi(t_h,h)) \leqslant 0. \tag{A.4}$$

Since $\rho(\xi) \geqslant \vartheta$ for all $\xi \in \mathscr{S}_0 \setminus \mathscr{S}_1$ and a jump occurs at hybrid time (t_{h+1},h) only if $\xi(t_{h+1},h) \in \mathscr{D}$, for all $h \in \mathbb{N}$, $h \geqslant 1$, such that $t_h + h \leqslant T$ it results that

$$\ell(\xi(t_{h+1},h)) + \rho(\xi(t_{h+1},h)) \geqslant (1-\mu)\rho(\xi(t_{h+1},h)) \geqslant (1-\mu)\vartheta. \tag{A.5}$$

Therefore, the conditions given in (A.3), (A.4), and (A.5) imply that there is a minimum dwell time $\tau = \frac{(1-\mu)\vartheta}{\nu}$ between two consecutive jumps of the solution $\xi(t,j)$. Such a solution is therefore non-Zeno and $(t,j) \in \mathrm{dom}(\xi)$ implies $j \leqslant \frac{t}{\tau} + 1$. Hence, following [116, Thm. 3.18], by (A.2c) and (A.2d), we have

$$\bar{V}(\xi(t,j)) \leqslant \bar{V}(\xi(0,0)) - \mu\vartheta t \leqslant c_0 - \mu\vartheta\frac{\tau}{\tau+1}(t+j-1),$$

thus leading to a contradiction since $\xi \in \mathscr{S}_0 \setminus \mathscr{S}_1$ if and only if $c_1 < V(\xi) \leqslant c_0$. Thus, the set $\mathscr{V}$ is globally uniformly asymptotically stable for system (A.1).

A.2 Proof of Proposition 1

Let the polynomials $p_{i,j} \in \mathbb{R}[x_i]$, $j = 1,\ldots,s_i$, $i = 1,\ldots,N$, be coerced into $\mathbb{R}[x]$ and consider the module $\mathscr{M}_i = \langle p_i, \ldots, p_{1,s_i} \rangle$ in $\mathbb{R}[x]$, $i = 1,\ldots,N$. By [118], one has that $\mathscr{V}_i = \boldsymbol{V}(\mathscr{M}_i)$, $i = 1,\ldots,N$, and $\mathscr{F} = \boldsymbol{V}(\langle q_1,\ldots,q_\omega \rangle)$. Therefore, by [117], one has that the affine variety $\mathscr{V}$ given in (5.14) is given by

$$\begin{aligned}\mathscr{V} &= \boldsymbol{V}(\langle q_1,\ldots,q_\omega \rangle + \sum_{i=1}^{N} \mathscr{M}_i) \\ &= \boldsymbol{V}(\langle p_{1,1},\ldots,p_{1,s_1},p_{2,1},\ldots,p_{2,s_2},\ldots,p_{N,s_N},q_1,\ldots,q_\omega \rangle).\end{aligned}$$

Thus, the statement follows by Algorithm 1 of [35].

A.3 Proof of Lemma 1

By Definition 3, a collision occurs if and only if there exists $t \in \mathbb{R}_{\geqslant 0}$ such that $x(t)$ belongs to the following variety

$$\left(\bigcup_{i=1}^{N}\bigcup_{w=1}^{W}\boldsymbol{V}(\varsigma_w(x_i))\right) \cup \left(\bigcup_{i=1}^{N}\bigcup_{j=i+1}^{N}\boldsymbol{V}(c_{i,j}(x))\right).$$

By [117], the affine variety above is given by $\boldsymbol{V}(\mathscr{M})$, where

$$\mathscr{M} = \left(\prod_{i=1}^{N}\prod_{w=1}^{W}\langle\varsigma_w(x_i)\rangle\right)\left(\prod_{i=1}^{N}\prod_{j=i+1}^{N}\langle c_{i,j}(x)\rangle\right).$$

Since each ideal in such a product is principal, then, by a trivial extension of [117, Ch. 4, §3, Prop. 6], the ideal $\mathscr{M}$ is principal and $\mathscr{M} = \langle b\rangle$. Therefore, a collision occurs at time $t \in \mathbb{R}_{\geqslant 0}$ if and only if $b(x(t)) = 0$.

A.4 Proof of Proposition 2

By computing the time derivative of $r(x)$ along the trajectories of system (5.21), one obtains that

$$\frac{d}{dt}r(x(t)) = \langle \nabla r(x(t)), -\eta(x(t))\beta(t)\rangle = -\eta^{\top}(x(t))\eta(x(t))\beta(t) \leqslant 0.$$

Therefore the function $r(x(t))$ is monotonically non-increasing in t. Hence, since $r(x(t)) \leqslant r(x_0) < \infty$ and $b(x(t)) = 0$ if and only if $r(x(t)) = \infty$, then there does not exist $t \in \mathbb{R}_{\geqslant 0}$ such that $b(x(t)) = 0$.

A.5 Proof of Theorem 2

Since $\lim_{x\to\partial\mathscr{R}}|\eta(x)| = \infty$ while $f_b(x)$ and $g(x)\zeta(x)$ are bounded, then $\gamma_1 b^2(x)f_b(x) + \gamma_1 b^2(x)g(x)\zeta(x) - \eta(x)\mu(x)p(x) \simeq -\eta(x)\mu(x)p(x)$ for all the points in the neighborhood of $\boldsymbol{V}(b) = \partial\mathscr{R}$. Thus, since $\mu(x)p(x) \geqslant 0$ for all $x \in \mathbb{R}^{2N}\setminus\mathscr{R}$, by the same

reasoning used to prove Proposition 2, the set $\mathbb{R}^{2N} \setminus \mathscr{R}$ is positively invariant with respect to system (5.22), *i.e.*, no collision occurs. Hence, define $V = p^\top p$, whose time derivative is given by

$$\begin{aligned}\dot{V} &= -p^\top \gamma_1 b^2 \Lambda p - p^\top \tfrac{\partial p}{\partial x} \eta \mu p \\ &= -p^\top (\gamma_1 b^2 \Lambda + \tfrac{\partial p}{\partial x} \eta \mu) p.\end{aligned}$$

Thus, the proof follows by classical Lyapunov arguments and by the fact that the set $\mathscr{Y}$ is positively invariant.

A.6 Proof of Theorem 3

Since the vector field $f(x,k)$ given in (5.23) and the functions $V(x)$ and $\rho(x)$ given in (5.24) satisfy Assumptions 1 and 2 in the set $\mathscr{Y}$, then the hypotheses of Corollary 1 are met, thus implying that the hybrid implementation (5.10) solves the patrolling in formation problem for all $x_0 \in \mathscr{Y}$.

It remains to prove that the path of motions of the hybrid implementation (5.10) are collision free. Following the same reasoning employed in Appendix A.5, note that $f(x,k) \simeq -\gamma_2 \eta(x) p^\top(x) p(x)$ for all the points in a neighborhood of $\partial\mathscr{R}$. Therefore, in such a neighborhood, one has that

$$\begin{aligned}\dot{r} &= -\gamma_2 \eta^\top \eta p^\top p \leqslant -\underline{\gamma}_2 \eta^\top \eta p^\top p, \\ r^+ &= r.\end{aligned}$$

where $\underline{\gamma}_2 = \min_{\mathscr{A}} \gamma_2 > 0$. Therefore, since there is a minimum dwell time between two consecutive jumps of the solution to the hybrid implementation (5.10) by the proof of Theorem 1, using the same reasoning used to prove Proposition 2, one concludes that the set $\mathbb{R}^{2N} \setminus \mathscr{R}$ is positively invariant with respect to system (5.10), *i.e.*, no collision occurs.