University of Modena and Reggio Emilia

DOCTORAL SCHOOL
INDUSTRIAL INNOVATION ENGINEERING

XXXII Period

Communication and Interaction for Mobile Robots: from Lab to Industry

Giuseppe Riggio

PhD Course Coordinator:
Prof. Franco Zambonelli

Supervisor:
Prof. Cristian Secchi

To my family

Abstract

Communicating and interacting with a robot are critical tasks in order to make the robot act as desired both in research and industrial scenarios. In this thesis, we developed different methods to interact with robots in many applications. Direct teleoperation allows the user to directly move the robot using some device to impose the motion of the robot. We studied a novel methodology for letting a user teleoperate a robotic system in a natural manner. The proposed approach does not require any specific device but relies on commonly used objects as a smartwatch. Using this approach, it is possible to interact with a mobile robot in an intuitive way by moving the user's forearm. Human-robot collaboration is fundamental when a cooperative tasks has to be done, as in the TIREBOT project. The TIREBOT project consists of the mechanical design and the collaborative control of TIREBOT, a robotic assistant helping tire workshop operators in the wheel replacement process. The operator can interact with the robot using either a gesture based interface or by teleoperation. The robot understands when the user wants to collaborate and when something else happens. In the first scenario, the user is allowed to get close to the robot. In the second case the robot has to move away from the user in order to keep the safety. Industrial applications as agricultural tasks can be labor intensive and can lead to a variety of injuries and illnesses, therefore it can be useful to let the robot do the hard work. In this field, we developed two different robotic systems. The first one is a low-cost autonomous system which can navigate through a vineyard while collecting grape pictures in order to provide a yield estimation. The second system is an apple harvesting robot. The robot can identify the apples in the scene, approach the apple to grasp and detach it from the tree. During the execution of the desired task, the human is not the only agent the robot has to interact with. It has to interact to other robots in order to achieve the desired goal. In multi-robot systems passive interconnections among agents are often exploited to achieve a desired and robustly stable cooperative behavior. We exploit the concept of energy tank for building a novel generalized interconnection that allows to impose any kind of dynamic coupling between two passive systems in a flexible way while preserving the passivity of the overall coupled system.

Sommario

Comunicare e interagire con un robot sono compiti fondamentali per far si che il robot agisca come desiderato sia in scenari di ricerca che industriali. In questa tesi, abbiamo sviluppato diversi metodi per interagire con i robot in molte applicazioni. La teleoperazione diretta consente all'utente di spostare direttamente il robot utilizzando un dispositivo per imporre il movimento del robot. Abbiamo studiato una nuova metodologia per consentire a un utente di teleoperare un sistema robotico in modo naturale. L'approccio proposto non richiede alcun dispositivo specifico ma si basa su oggetti di uso comune come uno smartwatch. Utilizzando questo approccio, è possibile interagire con un robot mobile in modo intuitivo spostando l'avambraccio dell'utente. La collaborazione uomo-robot è fondamentale quando si devono svolgere compiti cooperativi, come nel progetto TIREBOT. Il progetto TIREBOT consiste nella progettazione meccanica e nel controllo collaborativo di TIREBOT, un assistente robotico che aiuta gli operatori dell'officina nel processo di sostituzione delle ruote. L'operatore può interagire con il robot utilizzando un'interfaccia basata su gesti o tramite teleoperazione. Il robot capisce quando l'utente vuole collaborare e quando invece accade qualcosa di imprevisto. Nel primo scenario, l'utente è autorizzato ad avvicinarsi al robot. Nel secondo caso il robot deve allontanarsi dall'utente per mantenere la sicurezza. Le applicazioni industriali quali le attività agricole possono richiedere molto sforzo e causare una serie di lesioni e malattie, pertanto può essere utile lasciare che il robot faccia il duro lavoro. In questo campo, abbiamo sviluppato due diversi sistemi robotici. Il primo è un sistema autonomo a basso costo che può navigare attraverso un vigneto mentre raccoglie immagini di uva per fornire una stima della produzione. Il secondo sistema è un robot per la raccolta delle mele. Il robot può identificare le mele nella scena, avvicinarsi alla mela per afferrarla e staccarla dall'albero. Durante l'esecuzione dell'attività desiderata, l'essere umano non è l'unico agente con cui il robot deve interagire. Esso infatti deve interagire con altri robot per raggiungere l'obiettivo desiderato. Nei sistemi multi-robot le interconnessioni passive tra agenti vengono spesso sfruttate per ottenere un comportamento cooperativo desiderato stabile in modo robusto. Noi sfruttiamo il concetto di energy tank per costruire una nuova interconnessione generalizzata che consenta di imporre qualsiasi tipo di accoppiamento dinamico tra due sistemi passivi in modo flessibile preservando la passività dell'intero sistema accoppiato.

Acknowledgement

Listen to my story. This... may be our last chance.

(Tidus — FFX)

That's what Tidus said to his friends just before entering the Zanarkand Ruins, the end of their pilgrimage. Later, it will be clear that it's not the end of their journey, but it is the beginning of a new story where their fate is in their hands.

We are now in a similar situation. This journey is about to end and we are around the campfire to rest. It was a long journey, full of happy and sad moments. Sometimes I had to follow unexpected paths, but every single moment gave me something I could never have in any other way. I feel like I'm a better person than I was at the beginning.

I would like to thank all the people which helped me to reach this point. First, my family. Thank you for believing in me and for supporting me all these years. You were my strength along the way. I would like to thank all my friends. Your advices and suggestions helped me to overcome the obstacles during the journey. The time we spent together is priceless. It doesn't matter how far we are, our hearts always beat as one. Thanks to Cristian and the ARSControl group, it was a pleasure to work with you.

Finally, I would like to thank you, on the other side of the screen. Let me call you Raspberry. Maybe we will never meet and you will never read these words. But your help was essential to go on my way and reach the end. You gave me the best moments I had.

That's the end of this journey. It's time to begin a new story, as Tidus and his friends. I don't know what will happen but it's all on me.

This is my story. It'll go the way I want it... or I'll end it here!

(Tidus — FFX)

Contents

Chapter 1

Introduction

In the last decades, robots have been used in many different fields, from industrial applications to social environments. Robots replaced laborers who had to fulfil non-ergonomic tasks such as carrying heavy loads or repetitive operations which can lead to several injuries and illnesses. In many industrial applications, the use of robots allowed to reach high levels of productivity, precision and efficiency. In a social scenario, robots are used in human care for helping people which need assistance with many tasks, in education and entertainment.

This growing use of robots in everyday life made the researchers ask how to communicate and interact with them. Should robots replace people everywhere and work fully autonomously? Should robots only assist human in non-ergonomic tasks? Is it possible to fulfil some task exploiting both the advantages of robots and humans? Those questions led to different ways to use robots:

- teleoperation systems which directly move the robot as a tool;
- cooperation with a robot as an assistant;
- fully autonomous robots.

Teleoperation literally means "operate at a distance" and refers to the control of a robot by a human operator without any physical interaction between them. The most basic teleoperation system consists of two parts: the *master*, which is a device used by the human operator to give commands to the system, and the *slave*, which is the robot that behaves accordingly to the operator commands. The master and the slave devices can be located far or close to each other depending on the application. Teleoperation can be useful in many scenarios where the environment is dangerous or not suitable for a person as hazardous environments, or in those critical applications which require high precision such as surgery. There are three main control architecture in teleoperation:

Direct Teleoperation: The user commands are used to directly control the motion of the robot without any autonomous behavior;

Shared Control: The user commands control the motion of the robot which has some degree of autonomy to assist the user;

Supervisory Control: The user commands are used as high level goals for the robot, which has to be reach them in an autonomous way.

In Chap. 2 we studied a novel methodology to control a robot through direct teleoperation and supervisory control. The idea is to create a teleoperation system which uses as master device a commonly used object as a smartwatch. In this way there is no need for additional dedicated devices which may be less natural for implementing the desired commands or not portable. The developed system measures the accelerations and the angular velocities of the smartwatch in order to recognize the motion of the forearm of the user. These motions are elaborated to detect gestures which are used as high level commands for the robot or they are transformed into velocity input for direct teleoperation.

Collaborative robots are designed in such a way it is possible to have a safe physical interaction between the robot and the human operator. In this way it is possible the user to work alongside the robot in order to cooperate to reach a common goal. Human robot collaboration allows to combine the advantages of the robot, in terms of efficiency, workload and precision, and the human as flexibility and adaptability. In an industrial scenario, many tasks are still manually performed by the operators because they require his/her expertise and cognitive skills, but some of them can be executed by a robot. In particular, it is possible to make the robot accomplish those tasks which require to perform repetitive operations or to carry heavy loads. In Chap. 3 we developed TIREBOT (a TIRE workshop roBOTic assistant), a collaborative mobile robot capable of assisting the operator in a wheel management procedure. TIREBOT has to help the operator for carrying the wheels from the car to different machines. The robot has different working modalities:

Gesture Recognition mode: the user can communicate with the robot via many gestures in order to make it behave in the desired way, as following the operator or going to a specific machine;

Safe Cooperation mode: when the operator has to load the wheel on TIREBOT, the robot and the user have to be close to each other. In this case, the robot should let the operator get close but it has to be ready to go away from the operator to avoid collisions and guarantee the human safety;

Autonomous mode: when the robot has to move to a target position given by the user, it moves towards the goal avoiding the obstacles including the operator;

Teleoperation mode: through an haptic device it is possible to directly move TIREBOT.

Fully autonomous robot are suitable for many tasks which require high efficiency, speed and precision or repetitive and labour intensive tasks. Agriculture is full of those kind of tasks and many of them, e.g. harvesting, are still manually performed by people. Since the tasks can be very different from each other, the behaviour and the capabilities of an agriculture robot are task dependent. In Sec. 4.1 the task to fulfil is the yield estimation in a vineyard. In this case, the robot uses a laser scanner

to detect the vines and navigate through the vineyard while collecting pictures of the grapes using a RGB camera. The pictures are elaborated using machine learning techniques to estimate the quantity of grapes. In Sec. 4.2 we developed an apple harvester robot. The robot detects the apples to harvest using a RGB-D camera and a probabilistic segmentation of the scene in order to separate the fruits from the other elements like branches and leaves. The detected apples are the goals of the robot, which are sequenced and, for each one, the planner generates a collision free trajectory which leads the robot in front of the apple. The robot then moves towards the apple and grab it using the equipped gripper.

In order to fulfil a given task, it may be useful to have more than one robot cooperating to achieve the desired result. Multi-robot systems can perform tasks more efficiently than a single robot or can accomplish tasks not executable by a single one. Using a group of robots increase the flexibility of the overall system and the fault tolerance. A further improvement to the flexibility of the system is to consider heterogeneous robots with different skills, e.g. a wheeled mobile robot equipped with a manipulator and a flying robot with a camera which guides the first one. In a multi-robot system the agents are not independent from each other, but they are coupled in order to achieve the desired goal. It would be useful to freely change the coupling between the robots in order to implement a variable behaviour. Unfortunately, changing the coupling between the robots can lead to stability problems. In Chap. 5 we developed a tank based generalized interconnection which allows to couple the agents with any desired interconnection without losing the passivity, i.e. robust stability, of the overall system. We exploited the novel concept of *modulated multi-port tank* for reproducing the desired interconnection. The main idea is to use the energy stored in the tank for implementing the interconnection without loss of passivity even if the interconnection is not passive. If the stored energy is not enough, a variable damping is used to refill the tank.

1.1 Publications

- "TIREBOT: A collaborative robot for the tire workshop" Authors: Alessio Levratti, Giuseppe Riggio, Cesare Fantuzzi, Antonio De Vuono and Cristian Secchi. Robotics and Computer-Integrated Manufacturing 57, 129-137
- "On the Use of Energy Tanks for Multi-Robot Interconnection" Authors: Giuseppe Riggio, Cesare Fantuzzi, Cristian Secchi. International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 2018.
- "A Low-Cost Navigation Strategy for Yield Estimation in Vineyards" Authors: Giuseppe Riggio, Cesare Fantuzzi, Cristian Secchi. International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 2018.
- "Safe Navigation and Experimental Evaluation of a Novel Tire Workshop Assistant Robot" Authors: Alessio Levratti, Giuseppe Riggio, Antonio De Vuono, Cesare Fantuzzi and Cristian Secchi. International Conference on Robotics and Automation (ICRA2017), 2017.
- "A Natural Infrastructure-Less Human-Robot Interaction System" Authors: Valeria Villani, Lorenzo Sabattini, Giuseppe Riggio, Cristian Secchi, Marco Minelli and Cesare Fantuzzi. IEEE Robotics and Automation Letters.
- "Interacting With a Mobile Robot with a Natural Infrastructure-Less Interface" Authors: Valeria Villani, Lorenzo Sabattini, Giuseppe Riggio, Alessio Levratti, Cristian Secchi and Cesare Fantuzzi. Proc. IFAC 20th World Congress Int. Federation Autom. Control IFAC

Chapter 2

Human-Mobile Robot Interaction with an Infrastructure-Less Interface

2.1 Natural Infrastructure-Less Interface

2.1.1 Introduction

Recent advances in robot control and safety systems allowed the access, in the last few years, of robots in our daily life. Robotic systems have been applied to several fields, such as social assistance [1], surveillance [2], tour-guide [3] or floor cleaning [4]. These contexts are very different from traditional industrial applications, in which robots are typically utilized by expert users. In fact, in daily life applications, users are typically inexperienced, and do not have any specific education related to robot programming: the design of effective human-robot interaction (HRI) systems is then of paramount importance, for these applications.

In the last few years, the concept of natural user interfaces (NUIs) has been developed. The main idea is that of allowing a direct expression of mental concepts by intuitively mimicking real-world behavior. NUIs offer a natural and reality-based interaction by exploiting users' pre-existing knowledge and using actions that correspond to daily practices in the physical world [5]. To achieve this, NUIs allow users to directly manipulate objects and interact with robots rather than instruct them to do so by typing commands. Thus, they represent an evolutionary paradigm that overcomes the access bottleneck of classical interaction devices such as keyboards, mice and joysticks, by resorting to voice, gestures, touch and motion tracking [6, 7].

2.1.1.1 Contribution

Standing in this scenario, in this Section we propose a novel hands-free infrastructure-less natural interface for HRI, based on the recognition of the movements of user's forearm. We use the term *infrastructure-less* for indicating an interaction system that exploits an everyday device and does not need additional dedicated external sensors (such as, e.g., external cameras), which have a number of drawbacks. In fact they might be not portable, limit the physical area where interaction occurs and the

freedom of motion of the user, make the interaction less natural and require a set up phase, thus not being suited for instantaneous use while the user is performing everyday activities. In the application presented in this Section we consider a commercial smartwatch, but also common wristbands for activity tracking are suited. In general, any commercial multipurpose device equipped with accelerometers and gyroscopes, which can therefore measure movements of the forearm of the user, fits the proposed approach. Other than being a multipurpose device, the use of such an interaction means has the additional advantage of providing the user with freedom of movement and letting her/him be immersed in the environment where the robot moves, being able to track it with non-fragmented visibility [7].

Moreover, situation and environment awareness is increased through haptic feedback, provided by vibration with modulated frequency. In particular, vibration is used for two different purposes: for acknowledging the user's command (i.e. a vibration is provided if a gesture has been recognized), and for providing her/him with information on the status of the robot with respect to the environment (i.e. vibration modulated based on the distance from obstacles, or targets).

Considered these features, the proposed HRI approach takes the form of a *tangible* user interface (TUI) [7], in addition to being a NUI. The term TUIs encompasses a great variety of interaction systems relying on a coupling between physical objects and digital information, which is physically embodied in concrete form in the environment. Thus, TUIs provide direct mapping between the behavior of the robot and usage of such a robot, and between the behavior of control devices and resulting digital effects. The proposed HRI system relies on embodied interaction, tangible manipulation, physical representation of data and embeddedness in real space, which are the pillars of TUIs.

The interaction system proposed in this Section represents then a milestone towards interaction with robots for everybody, being (to the best of the authors' knowledge) the first example of a natural and tangible hands-free infrastructure-less HRI system.

2.1.1.2 Application scenario

In the application presented in this Section we consider the need to remotely control a quadrotor. However, the proposed approach is general and possibly holds valid also to control, among the others, mobile robots [8, 9] and industrial manipulators [10]. It can be scaled to multi-robot systems [11] and has been preliminarily applied to interact with industrial machines [12]. Depending on the application and the robot considered, the mapping between movement of the forearm and commands to the robot and between gestures and changes of the state machine has to be adapted. Moreover, in the case of nonholonomic robots state feedback linearization has to be considered.

Specifically, the proposed interaction system is advantageous in the following application scenarios. First, we consider those conditions when the user and the robot share the same working area, that is when the distance between the user and the robot is not intrinsic to the application, but would be introduced solely by the

human-robot interface. In this scenario, by using the proposed interaction system, the user can control the robot while standing in its proximity and following its movements. Second, we consider the case of remote control of a robot either when the user's view on the robot is partially occluded or in an uncluttered environment. In the first case, just as an example, we consider the inspection af an area inaccessible to the user due to closed road, debris obstructing the road, or architectural barriers in the presence of disabled users. In this scenario, the haptic feedback for obstacle avoidance compensates the partial occlusion of the field of view of the user. In the case of uncluttered environment, the smartwatch might be used to drive the robot for an aerial inspection of the area: consider for example the case of a quadrotor hosting an infrared camera that supports search and rescue activities in a hostile environment (like mountains, as in [11], or areas devastated by an earthquake) and provides a haptic feedback when someone is detected. In addition, we consider the necessity of inspecting the roof of a building under construction or maintenance, while the user is on the ground. In this scenario, a hands-free interaction is required due to the fact that typically users wear gloves, which impede interaction with input devices, such as touchscreen, mouse and keyboard, or might have greasy hands. In this kind of applications, hands-free control of a quadrotor represents a very effective solution that allows to reach the area of interest and acquire images (or measurements) remotely.

2.1.1.3 Related works

Gesture-based control represents one of the most frequently utilized paradigms for providing intuitive interaction with robotic systems. Despite being a quite new research field, several techniques have been proposed in the literature. The greatest majority of them relies on the use of vision systems [13, 9, 10, 14, 15]. Generally speaking, vision-based techniques require proper lighting conditions and camera angles [14], and gestures are recognized either by means of colored markers [14] or by directly detecting human palm [15]. The common assumption of all these methodologies consists in having controlled and uniform illumination condition. This represents a strong assumption, and often cannot be verified in real world applications, in particular in outdoor environment. Moreover, this kind of approach is not infrastructure-less since dedicated vision system instrumentation is required. This implies that, firstly, these interaction systems are not portable, since they are effective only when the user lies in the field of view of the sensors, that is in front of the camera [13]. Secondly, these systems cannot track the robot since the user can barely move and follow it. Thus, their field of application is greatly reduced: for instance, they are unsuited for any inspection application, where movement in large areas is typically required. This can be partially dealt with mounting the camera on the robot [8], and letting the user follow the robot in its movements. However, also in this case, the user must stand in front of the camera in order for the gestures to be recognized.

To overcome the limitations of vision-based HRI, physical interaction devices can be exploited for implementing non-vision-based methodologies. One of the most

popular techniques consists in recognizing gestures utilizing sensorized gloves [16]. An extensive survey on wearable systems for activity recognition is reported in [17], and classical methods for recognizing gestures, such as hidden Markov models, are also presented therein. Further, in [18] an overview of the most used approaches for recognizing gestures is reported.

An ad-hoc vibrotactile glove is proposed in [19] to guide the user on a semi-autonomous wheelchair by providing a vibration feedback that indicates obstacles or desired directions in the environment. While these devices provide a high measurement precision, they heavily limit the freedom of motion of the user, who is forced to wear uncomfortable ad hoc devices [14]. To overcome this, hands-free techniques have been developed that rely on accelerometer data. In particular, gesture recognition methods are proposed in [20, 21, 22] for recognizing the user's motion. In [23] the authors use a wristband and a headset to control a fleet of drones by means of gesture and speech recognition. The use of the headset increases the invasiveness of the approach and moreover the interaction with the robot is limited to a predefined set of primitives.

Non-vision-based gesture recognition can also be obtained utilizing touch screen technologies. Along these lines, [24] and [25] consider the use of multi-touch handheld technologies, such as tablets, to control robots. These interfaces require the use of both hands and, hence, are not practical in many circumstances. Moreover, they lack in spatial interaction [7] and environment skill [5], thus having limited learnability and intuitiviness.

2.1.2 Multi-modal control architecture for the interaction with a robot

2.1.2.1 Overview of the system

According to the interaction and control system considered in this work, the user wears the smartwatch or the activity tracker wristband: the motion of her/his forearm is then acquired by the device. In particular, characteristic measurements related to the motion are considered: three-dimensional acceleration (raw and with automatically compensated gravity) and three-dimensional angular velocity.

Data are then processed for implementing motion recognition: in particular, the motion of the user is analysed for recognizing gestures (used for imposing high-level commands to the robot), or for defining velocity commands (i.e. the robot velocity is controlled as a function of the user's motion).

Feedback is then provided to the user, to inform her/him about the current status of the system and the recognition of gestures. In particular, feedback is provided to the user in terms of modulated vibration frequency of the smartwatch, depending on the position and the status of the robot.

2.1.2.2 Control architecture for a quadrotor

In the following we describe a particular realization of the proposed interaction system, in which a smartwatch is exploited for letting a human operator interact

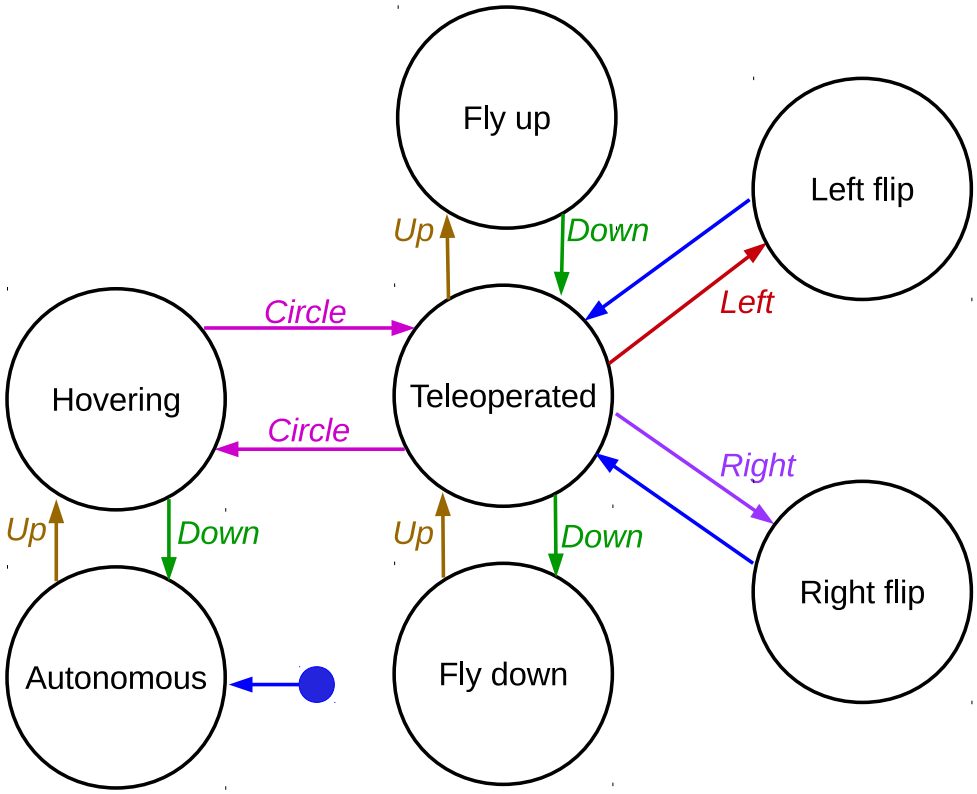


Figure 2.1: State machine of the control algorithm.

with a quadrotor unmanned aerial vehicle[1]. The overall control architecture utilized for controlling the quadrotor is described by the state machine diagram depicted in Fig. 2.1. Gestures are utilized for changing the state of the system. Even though the proposed architecture could be implemented with any set of gestures, in this application we consider the following five gestures, without loss of generality: 1) *Up*: sharp movement upwards, 2) *Down*: sharp movement downwards, 3) *Circle*: movement in a circular shape, 4) *Left*: sharp movement to the left, 5) *Right*: sharp movement to the right. As will be clarified hereafter, these gestures represent a natural choice, since they can be naturally used for implementing an intuitive coupling between user commands and effects on the robot behavior. In the following, Section 2.1.4 is devoted to the description of the algorithm used for gesture recognition.

The system is initialized in the *Autonomous* state. In this state there is no interaction between the user and the quadrotor. Hence, the quadrotor is controlled by means of the internal, preprogrammed, control algorithm. If no control algorithm is defined, as in the application presented in this work, then the quadrotor is stopped.

Using the *Up* gesture, the user can move to the *Hovering* state, where the quadrotor autonomously takes off, and reaches a stable hovering position. The *Down* ges-

[1]We will hereafter assume that the quadrotor is endowed with a low level control system, which enables it to perform desired trajectories. This can be obtained using standard quadrotor control techniques, such as [26].

(a) Angles of the smartwatch (b) Directions for teleoperating the quadrotor

Figure 2.2: Reference frames for the *Teleoperated* state.

ture takes the system back to the *Autonomous* state: if no control algorithm is defined, the quadrotor lands and stops.

From the *Hovering* state, it is possible to move to the *Teleoperated* state using the *Circle* gesture. In the *Teleoperated* state, the user can directly control the motion of the quadrotor. It is possible to go back to the *Hovering* state using again the *Circle* gesture.

From the *Teleoperated* state, it is possible to utilize the *Up* gesture to take the system to the *Fly up* state. In this state, the quadrotor increases its height, at a constant velocity. The gesture *Down* can then be utilized by the operator to take the system back to the *Teleoperated* state. In a similar manner, the gesture *Down* can be used to take the system to the *Fly down* state, in which the quadrotor decreases its height, at a constant velocity. The gesture *Up* can then be utilized by the operator to take the system back to the *Teleoperated* state.

From the *Teleoperated* state, it is possible to utilize the *Left* (*Right*) gesture for bringing the system to the *Left flip* (*Right flip*) state: in this condition, the quadrotor executes a flip maneuver in clockwise (counterclockwise) direction, with a preprogrammed control algorithm, and then automatically goes back to the *Teleoperated* state.

In the *Teleoperated* state, the pose of the smartwatch is translated into a velocity command for the quadrotor, and its vibration is exploited to provide the user with a feedback on the current performance of the quadrotor. In details, the angles of the smartwatch are directly translated into velocity control inputs for the quadrotor, in a natural and intuitive manner.

Referring to Figs. 2.2(a) and 2.2(b), the *Roll* angle is used to control the motion of the quadrotor along the *forward/backward* direction and the *Pitch* angle is used to control the motion of the quadrotor along the *left/right* direction. In particular, let ϑ_r, $\vartheta_p \in [-\pi/2, \pi/2]$ be the roll and pitch angle, respectively, and let v_x, $v_y \in \mathbb{R}$ be the velocity of the quadrotor along the forward/backward and left/right direction, with positive sign towards the forward and left direction, respectively. Then, the

velocity commands are computed as follows:

$$v_x = K_r\,\vartheta_r, \quad v_y = K_p\,\vartheta_p \tag{2.1}$$

where K_r, $K_p > 0$ are constants defined in such a way that the maximum angle that is achievable with the motion of the forearm corresponds to the maximum velocity of the quadrotor.

Conversely, the *Yaw* angle of the smartwatch is used to define a setpoint for the *Yaw* angle of the quadrotor. Namely, let $\vartheta_y \in [-\pi, \pi]$ be the yaw angle of the smartwatch, and let $\varphi \in [-\pi, \pi]$ be the yaw angle of the quadrotor. Then, the yaw rate of the quadrotor is controlled as follows:

$$\dot{\varphi} = K_y\,(\vartheta_y - \varphi) \tag{2.2}$$

where $K_y > 0$ is an arbitrarily defined constant.

2.1.3 Haptic feedback

Modulated vibration of the smartwatch is utilized for providing the user with a haptic feedback. In particular, vibration is utilized with two different purposes: for acknowledging the user's command, and for providing her/him with information on the status of the robot.

2.1.3.1 Command acknowledgement

Vibration is utilized for informing the user about recognition of gestures: namely, a single short vibration notifies the user when a gesture has been recognized by the control architecture. This provides an immediate feedback about the current status of the robot with reference to Fig. 2.1, and the possible actions, thus increasing user's situation awareness. Such feedback aims at solving, at least partially, the ephemerality of gestures [27]. Indeed, the ephemeral nature of gestures implies that if a gesture is not recognized or is misrecognized, the user has little information available to understand what happened [27]. This kind of sensory feedback tackles the first condition by informing the user whether a gesture has been recognized. In the current implementation, no information is provided about possible misrecognition: it could be provided by showing on the screen of the smartwatch which gesture has been recognized. However, it is worthwhile noting that, utilizing the gesture recognition algorithm that will be presented in Section 2.1.4, we experimentally verified that gesture misrecognition rate is quite low, which significantly reduces the need for this additional information.

2.1.3.2 Status of the robot

Vibration is also utilized for assisting the user in piloting the quadrotor. In particular, we considered the case where the quadrotor is moving in an environment which is populated by obstacles to be avoided, or targets to be reached. The user then

senses a vibration that is modulated based on the current altitude of the quadrotor, and its position with respect to the obstacles, or its distance from the target.

As regards altitude related vibration, it is mostly important indoor, where practical constraint on the maximum height of flight of the robot are to be considered. In particular, we introduce h_M and h_m, with $h_M > h_m > 0$, as the maximum and minimum allowed values for the height of the quadrotor, respectively. More specifically, let $h(t)$ be the height of the quadrotor at time t: then, we want to ensure that $h_m \leq h(t) \leq h_M, \forall t$. This requirement is due to the fact that our experiments are performed indoor; for outdoor scenarios it might be relaxed to $h(t) \geq h_m$. In the *Fly up* and *Fly down* states, the vibration frequency increases as the height of the quadrotor approaches h_m or h_M, in a proportional manner. Specifically, let $f_v^{(A)}(t)$ be the vibration frequency at time t corresponding to the feedback on altitude. In the *Fly up* state, it is defined as follows:

$$f_v^{(A)}(t) = \begin{cases} 0 & \text{if } h(t) < h_M - \delta \\ K_v \left(h(t) - h_M + \delta\right) & \text{otherwise} \end{cases} \tag{2.3}$$

where the threshold δ was defined as $\delta = 0.3(h_M - h_m)$, and $K_v = 0.5$ is a normalization constant, empirically defined to guarantee that the vibration frequency is sufficiently small at the discontinuity $h(t) < h_M - \delta$, thus attenuating the effect of the discontinuity and avoiding possible confusion and stress to the user due to a sudden strong vibration. In a similar way, the vibration frequency was defined as follows in the *Fly down* state:

$$f_v^{(A)}(t) = \begin{cases} 0 & \text{if } h(t) > h_m + \delta \\ K_v \left(-h(t) + h_M + \delta\right) & \text{otherwise} \end{cases} \tag{2.4}$$

Furthermore, the user is provided with a feedback related to the distance form either the target to reach or the obstacle to avoid, based on the application. Denoting by $f_v^{(T)}$ and $f_v^{(O)}$ the vibration frequency of these two haptic feedbacks, they are defined as

$$f_v^{(T)}(t) = K_v^{(T)} \frac{1}{d^{(T)}(t)}, \quad f_v^{(O)}(t) = K_v^{(O)} \frac{1}{d^{(O)}(t)} \tag{2.5}$$

where $K_v^{(T)}$ and $K_v^{(O)}$ are normalization constants set to 0.2 following the same empirical criterion as K_v, and $d^{(T)}(t)$ and $d^{(O)}(t)$ are the Euclidean distances between the instantaneous quadrotor position and the target and obstacle positions, respectively. Several techniques can be found in the literature for using the on-board cameras to detect the position of targets and obstacles in real time: however, this is out of the scope of the work, and will therefore not be addressed here. In Subsection 2.1.5.1, preliminary results of the validation of the haptic feedback for target proximity are presented.

2.1.4 Gesture recognition

To implement the control architecture introduced in Subsection 2.1.2.2, in the following we will consider gesture recognition by template matching [28, 29]. This

classical approach represents a standard algorithm for this kind of applications and, as will be shown in Subsection 2.1.4.3, exhibits satisfactory performance. However, we would like to point out that more advanced methods have been proposed in the literature, allowing higher recognition performance.

In the proposed system, gestures are recognized by template matching of signals recorded by accelerometers and gyroscopes in the smartwatch. Hence, an initial training step for template building is required. Then, when the system is activated, the signals acquired by the smartwatch are on-line compared to the templates and the decision whether a gesture has just occurred is taken by computing suited metrics of comparison.

2.1.4.1 Template building

A template is built for each signal recorded by the smartwatch and for each gesture. Let $\mathcal{S}$ be the set of signals that the smartwatch can acquire, namely angular displacements and linear accelerations (raw and with automatically compensated gravity), and let $\mathcal{G} = \{Up, Down, Circle, Left, Right\}$ be the set of gestures of interest. Moreover, let $N_{sig} = 9$ be the cardinality of $\mathcal{S}$, that is the number of signals acquired by the smartwatch, and let $N_{gest} = 5$ be the cardinality of $\mathcal{G}$. Hence, a total of $N_{sig} \cdot N_{gest}$ templates is required.

Training data used for the construction of templates consist of $N_{rep} = 60$ repetitions of each gesture. Epochs have been automatically extracted from the recorded signals, considering a window of fixed length $N = 35$. A preliminary alignment step, based on the normalized cross-correlation function [30], is required to correct the approximate selection of the beginning of each epoch. Given two real finite-duration sequences x, y of length N, with $x(n), y(n), n = 1, \ldots, N$, the n-th element of x, y, respectively, the corresponding cross-correlation is defined as

$$r_{xy}(l) = \sum_{n=i}^{N-|k|-1} x(n)y(n-l) \qquad l = 0, \pm 1, \pm 2, \ldots \tag{2.6}$$

where $i = l, k = 0$ for $l \geq 0$, and $i = 0, k = l$ for $l < 0$ and l is the time shift (or lag) parameter. The normalized cross-correlation sequence [30] is defined as

$$\rho_{xy}(l) = \frac{r_{xy}(l)}{\sqrt{r_{xx}(0)r_{yy}(0)}} \tag{2.7}$$

where $r_{xx}(0)$ and $r_{yy}(0)$ are the variance of sequences $x(n)$ and $y(n)$, respectively. It ranges from -1 to $+1$ and for $l = 0$ it corresponds to the correlation coefficient of $x(n)$ and $y(n)$, i.e., $\rho_{xy}(0) = \rho_{xy}$.

The cross-correlation sequence, and its normalized version, are a widely used tool to measure the similarity of two series as a function of the lag of one relative to the other. In this regard, they turn out to be useful to align the epochs within which gestures were performed. In particular, considering arbitrarily as a reference the first epoch, namely $x_0(n)$, of each data record, remaining epochs $x_i(n), i = 1, \ldots, N_{rep} - 1$

are shifted by a number l_i^* of lags such that the normalized cross-correlation sequence between the reference and the current epoch is maximized:

$$l_i^* = l \mid \rho_{x_0 x_i}(l_i^*) = \max_l \rho_{x_0 x_i}(l). \tag{2.8}$$

A threshold $\alpha = 0.7$ is set on the value of the maximum normalized cross-correlation sequence $\rho_{x_0 x_i}(l_i^*)$: only epochs with $\rho_{x_0 x_i}(l_i^*) > \alpha$ are involved in the definition of the template. Gestures not satisfying this constraint are discarded, since they are considered not representative of the correct execution of the gesture. Selected epochs are then averaged to build the template, denoted by $\widetilde{x}^{(sig,gest)}$.

2.1.4.2 Template matching

Templates are exploited to detect the occurrence of gestures in real-time, that is when interacting with the robot. Basically, each measured signal is compared to the corresponding templates of all possible gestures and a rule is established to decide which gesture is the most likely to have just occurred, if any. As a metric of comparison, we use the correlation coefficient [31]. Denoting by $x_{w,k} = x(k, \dots, k+N-1)$ the sliding window of signal sig starting at time index k, the correlation coefficient between such window and the template for the gesture $gest$ is given by

$$\rho_{\tilde{x}x_{w,k}}^{(sig,gest)} = \frac{\sigma_{\tilde{x}x_{w,k}}}{\sqrt{\sigma_{\tilde{x}\tilde{x}}^2 \sigma_{x_{w,k}x_{w,k}}^2}} \tag{2.9}$$

where $\sigma_{\tilde{x}x_{w,k}}$, $\sigma_{\tilde{x}\tilde{x}}^2$ and $\sigma_{x_{w,k}x_{w,k}}^2$ are the covariance and variance of the signals in subscript. Then, the correlation coefficients are averaged for each gesture, in order to obtain a coefficient, denoted by $\rho_{\tilde{x}x_{w,k}}^{(gest)}$, representing how much the movement in the current sliding window resembles each gesture. Thus, we have

$$\rho_{\tilde{x}x_{w,k}}^{(gest)} = \frac{1}{|\mathcal{S}^{*,gest}|} \sum_{sig^{*,gest} \in \mathcal{S}^{*,gest}} \rho_{\tilde{x}x_{w,k}}^{(sig^{*,gest},gest)} \tag{2.10}$$

where $|\mathcal{S}^{*,gest}|$ denotes the cardinality of the set $\mathcal{S}^{*,gest}$.

To exclude false positives (the algorithm detects a gesture, although no gestures have been performed), we set a threshold $\theta = 0.75$ on the correlation coefficient. The value for θ was coarsely set, without neither fine tuning nor optimization. Despite of this, as shown in the next Subsection, the accuracy of gesture recognition is satisfactory. Thus, the decision rule is as follows. If, for the current sliding window, all the coefficients are below the threshold, then we decide that no gesture has occurred. Otherwise, if we have $\rho_{\tilde{x}x_{w,k}}^{(gest)} > \theta$ for one gesture, then we claim that the corresponding gesture has occurred. If we have $\rho_{\tilde{x}x_{w,k}}^{(gest)} > \theta$ for more than one gesture, then we pick the gesture with the greatest $\rho_{\tilde{x}x_{w,k}}^{(gest)}$.

Finally, after the detection of a gesture, we establish a short latency time window of length 1 s in which we expect the subject not to perform any new gesture, waiting for the robot to be ready in the requested state (see the state machine diagram in Fig. 2.1). Such time interval starts with the detection of a gesture by the algorithm

described above. During the latency window the search for a new gesture by template matching is stopped: data acquired by the smartwatch are then not analyzed. Template matching is restarted at the end of such interval.

2.1.4.3 Validation of the gesture recognition module

To assess the performance of the algorithm for gesture recognition, we carried out three tests. First, a subject was asked to randomly perform $\mathcal{N}_1 = 100$ repetitions of the gestures in a steady state, that is sitting at a desk and holding the arm at rest between two consecutive gestures. Second, the same subject was asked to perform $\mathcal{N}_2 = 100$ repetitions of the gestures while moving, that is walking, jogging, (simulating) swimming, exercising, shaking the arm to touch his head, greeting someone, or looking at the watch.

In these experiments, the achieved accuracy is very high (100% and 95%, respectively), even when spurious movements were performed. However, this result might be biased by the facts that gestures are performed in a very controlled situation, and only one subject was involved.

To assess the inter-subject robustness of the gesture recognition, the third experiment involved 20 subjects, not participating in the training set, for a total of $\mathcal{N}_3 = 800$ repeated gestures. Preliminarily, each subject was briefly instructed about the system and performed few gestures to get confident. Then, she/he was asked to perform 40 repetitions of the gestures, while standing, as shown in the first part of the video[2]. Fig. 2.3 reports the performance of the algorithm in terms of confusion matrix[3]. Although no subject-tailored template building was carried out, results show that the implemented algorithm performs well with subjects not involved in the off-line training, achieving an overall accuracy higher than 86%. We would like to remark that this result, and in general any gesture misrecognition, do not affect the safety and attitude control of the robot. Indeed, in the considered system, gestures are only utilized for switching among different states in the state machine, whereas, in the *Teleoperated* state, the motion of the user directly defines the motion of the quadrotor as a function of the roll, pitch and yaw angles of the smartwatch.

2.1.5 Experimental validation

Experiments aimed at evaluating the usability of the proposed interaction mode with the quadrotor and assessing the effectiveness of the haptic feedback. Two experimental conditions were considered: first, we considered a simulated environment where we compared the interaction by means of the smartwatch and of a common joypad for videogaming and assessed the effectiveness of the haptic feedback; second, we considered the proposed interaction system in a real scenario and compared it

[2]https://drive.google.com/open?id=1Izy4aFjeBHsBvCddH_0SuWyxLqPCiW0n

[3]The confusion matrix is used in machine learning to report results achieved by a classification system [32]. It contains information about actual (rows) and predicted (columns) classifications and can be used to compute system's accuracy and misclassification rate. The last column (right) of the confusion matrix reports the occurrence of false negatives, i.e., performed gestures not detected by the algorithm.

		RECOGNIZED					
		↑	↓	↻	←	→	*none*
ACTUAL	↑	**78.9**	1.2	0	2.5	1.2	16.2
	↓	1.9	**84.4**	0	0	10.0	3.7
	↻	0.6	0	**90.1**	**3.1**	0	6.2
	←	0.6	0	0	**95.0**	0	4.4
	→	11.2	0	0	0	**85.0**	3.8

Figure 2.3: Confusion matrix of the gesture recognition. Values are in %.

with the official smartphone application that comes with the quadrotor. The joypad and the smartphone application were selected as standard reference interaction systems. Clearly, they require the use of hands, and, in the case of the joypad, are not infrastructure-less. Moreover, they provide a less natural mapping of user movements to quadrotor movements, as corroborated by the numerical results presented below and reported by subjects involved in the tests.

To experimentally validate the proposed HRI system, we utilized a Parrot AR.Drone 2.0 quadrotor and a Samsung Gear S smartwatch. Due to its limited computation capabilities, it was not possible to implement the gesture recognition and control architecture on-board the quadrotor. Thus, an external computer was utilized, and the architecture was implemented by means of ROS [33]. Wi-Fi was used for communicating with the quadrotor, which was equipped with a vision-based localization system as described in [34], and with the smartwatch.

2.1.5.1 Simulated environment

In this set of experiments, we simulated the quadrotor by using the robot simulator Gazebo[4] implemented in ROS [33]. The simulator was running on the same computer that was connected to the smartwatch and where gesture recognition was computed. Two experiments were carried out.

[4]http://gazebosim.org/

Twenty two subjects were involved in the tests (7 females and 15 males, age 31.2 y.o. ± 4.3). They are researchers from our team and students from our university who are not involved in this research topic at all. Some of them have some experience in the use of the joypad for videogaming, but they all have barely experience in the use of the smartwatch for controlling a quadrotor and are first-time users in this regard.

2.1.5.1.1 Usability assessment

First, the user was asked to drive the robot through a cluttered environment. In particular, the user had to move the robot in an area where five columns were placed on the ground, as shown in Fig. 2.4(a). The user was instructed to drive the robots from the initial position to the goal position, performing a slalom path, without touching any column. Each user was requested to perform the experiment with two different interaction systems: the smartwatch and a standard joypad. The same state machine of Fig. 2.1 was used for the joypad: some buttons were programmed to initiate actions like take-off, landing, hovering, flying up and flying down, as gestures do for the smartwatch. Then, the pose of two analog sticks was translated into a velocity command for the quadrotor, as with the pose of the smartwatch. To compare the two interaction means, we considered the total travel time. The results achieved with the smartwatch are significantly better[5] than the ones obtained with the joypad. On average, the total travel time in the case of the smartwatch was 35.8 s with standard deviation 12.6 s, whereas in the case fo the joypad the total travel time was 41.0 s with standard deviation 13.9 s.

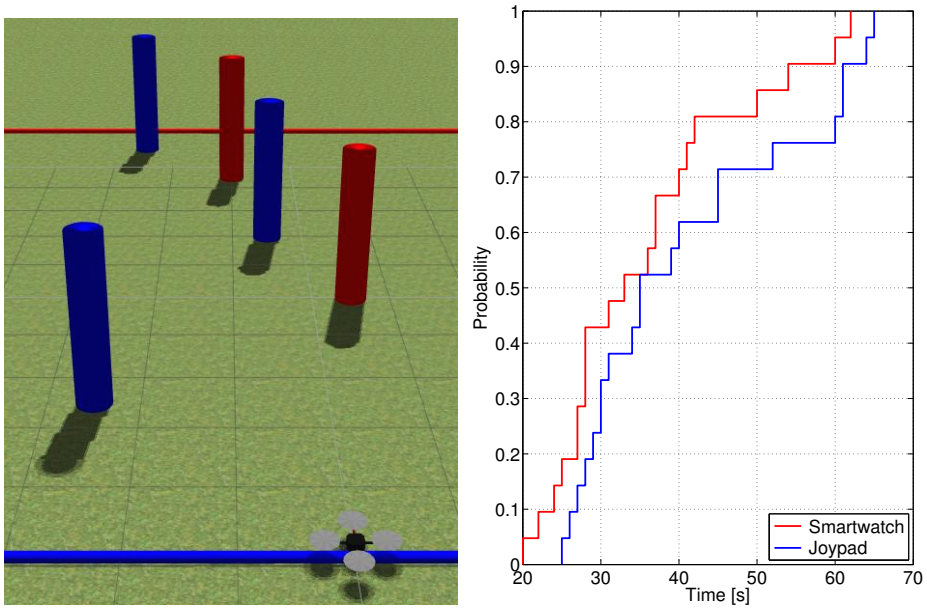
Moreover, the effectiveness of the use of the smartwatch has been also assessed in terms of the empirical distribution function [31] of travel time. The empirical distribution function provides a *complete* statistical description of the effectiveness, measured here as travel time, of the considered interaction approaches, as opposed to the mean, the variance, the median, or synthetic statistical tests, which give only partial information. Fig. 2.4(b) reports the empirical distribution functions of travel time for the two interaction devices. As the figure highlights, piloting the quadrotor with the smartwatch allows to complete the task in shorter time with higher probability than the case of the joypad.

2.1.5.1.2 Haptic feedback

Second, the user was requested to drive the robot around a column and then to visit a target location, identified by a red circle (diameter 40 cm) placed on the ground, as shown in Fig. 2.5(a). The goal of this test was to assess the effectiveness of the haptic feedback by measuring the distance from the target when the haptic feedback was or was not provided to the user. Experimental results have shown that the haptic feedback provides useful information to users[6]: indeed, in

[5] P-value $p = 0.1$.

[6] P-value $p = 0.02$.



(a) Experimental setup (b) Performance metrics

Figure 2.4: Assessment of the usability of the proposed interaction system in simulated environment. *Left*: experimental setup. *Right*: empirical distribution function of the travel times when piloting the robot with the smartwatch (red) and a joypad (blue).

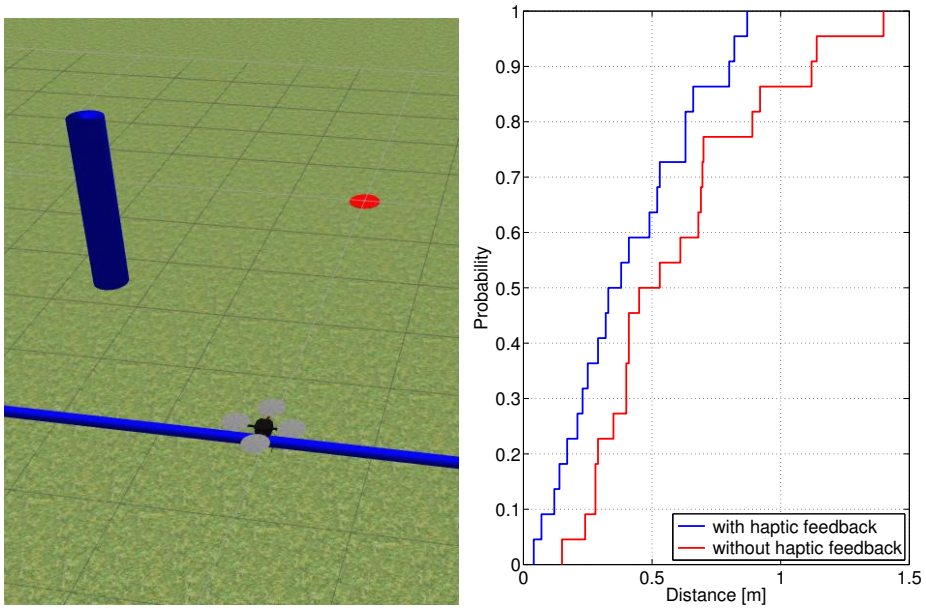
presence of the haptic feedback, on average, users landed at a mean distance from the target $\mu_{d(T)} = 0.41$ m with standard deviation $\sigma_{d(T)} = 0.25$ m, whereas when haptic feedback was not provided users landed at a mean distance from the target $\mu_{d(T)} = 0.59$ m with standard deviation $\sigma_{d(T)} = 0.33$ m. In terms of the empirical distribution function of measured distances, Fig. 2.5(b) highlights that the haptic feedback provides useful information to the user, since it allows to land at lower distance from the target with higher probability.

2.1.5.2 Real environment

The proposed HRI system was validated also by means of experiments involving users. The results of these experiments are shown in the second part of the video[7], where a user pilots the quadrotor by means of the smartwatch.

Specifically, we tested the usability of the proposed system, in comparison with the official smartphone application AR.FreeFlight to pilot the quadrotor. The AR.FreeFlight app implements the same degree of autonomy as the proposed system. Actions like take-off, landing, hovering, flip, flying up and flying down are internally implemented in the app and can be activated by the user by tapping a button in the interface, just as gestures do. The position of two analog sticks is

[7] https://drive.google.com/open?id=1Izy4aFjeBHsBvCddH_0SuWyxLqPCiW0n



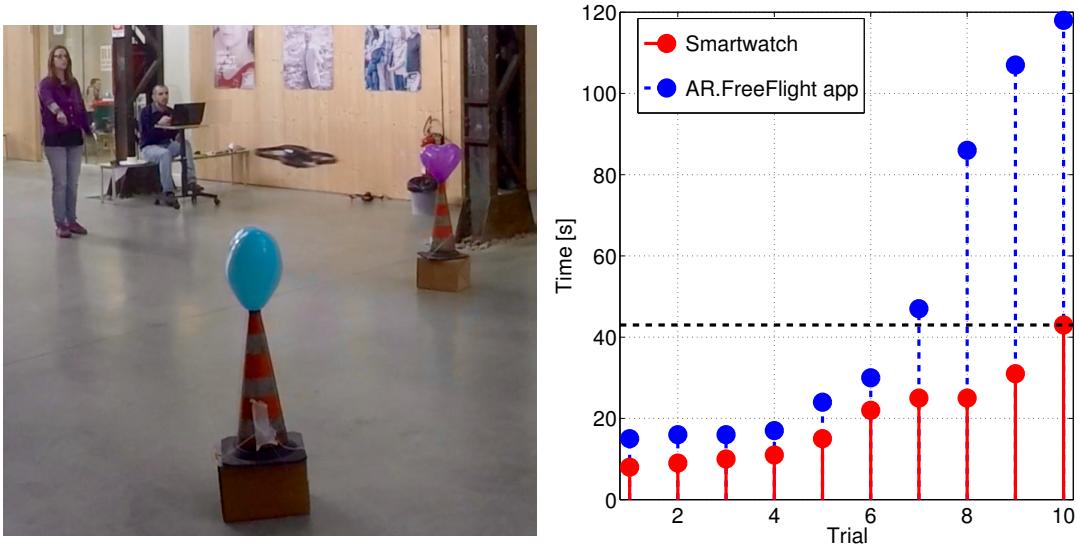
(a) Experimental setup (b) Performance metrics

Figure 2.5: Assessment of the effectiveness of the haptic feedback in simulated environment. *Left*: experimental setup. *Right*: empirical distribution function of the landing distances from the target with (blue) and without (red) haptic feedback.

translated into a velocity command for the quadrotor, as is done with the pose of the smartwatch in the *Teleoperated* state of Fig. 2.1.

The experiment consisted in asking the user to pop some balloons by piloting the quadrotor, equipped with four needles, to collide on them. A snapshot of the experiment is shown in Fig. 2.6(a), and a portion of the experiment is reported in the third part of the video[8]. As a metric to quantify the effectiveness of interaction, we considered the average time required to pop a balloon, starting from the quadrotor take-off. Two subjects were involved in the experiment and 10 balloons per method were popped. The application ran on a smartphone embedding a 5.2 in touchscreen. Piloting the quadrotor with the smartwatch proved to be more intuitive and both performers reported an increased cognitive burden when utilizing the smartphone application rather than the smartwatch. This is confirmed[9] by the fact that the use of the smartwatch resulted in an average time to pop a balloon of 19.9 s, whereas piloting with the application resulted in an average time of 47.6 s. In Fig. 2.6(b) we report the time to pop for both the piloting modalities, for each trial (red solid for the smartwatch, blue dashed for the application), sorted in ascending order. The figure shows that, in 4 trials over 10, the use of the application took a time greater than the greatest time (black dashed line) achieved with the smartwatch.

[8] https://drive.google.com/open?id=1Izy4aFjeBHsBvCddH_0SuWyxLqPCiW0n
[9] P-value $p = 0.04$.



(a) Experimental setup

(b) Performance metrics

Figure 2.6: Assessment of the usability of the proposed architecture in real environment *Left*: experimental setup. *Right*: time to pop a balloon when piloting the quadrotor with the smartwatch (red solid) and the official application (blue dashed).

2.1.6 Conclusions

In this Section we introduced a novel approach for hands-free natural human-robot interaction, based on the use of general purpose devices. In particular, the motion of the user's forearm was recognized by means of a smartwatch, and processed online for recognizing gestures, and for defining control inputs for a robot. Modulated vibration was exploited for providing the user with a haptic feedback, informing her/him about detected gestures and the current status of the robot. The usability of the proposed interaction system was experimentally evaluated, both in real and simulated environment, having users controlling the motion of a semi-autonomous quadrotor.

2.2 Human-Mobile Robot Interaction

2.2.1 Introduction

In this Chapter we propose a novel methodology for letting a user interact with a mobile robot in a *natural* manner. While traditional robotic systems are typically utilized in industrial contexts by specialized users, recent advances in robot control and safety systems are extending the field of application to diversified domains, including social assistance [1], surveillance [2], tour-guide [3] or floor cleaning [4]. Furthermore, in recent years costs related to the installation of robotic equipments have been dramatically decreasing, thus allowing also small industrial places to invest in such systems. As a consequence, robot users are significantly increasing, both in number and in variety, and inexperienced users have often the necessity of utilizing robotic systems.

A very important application in which mobile robots have been increasingly used in the last few years is goods transportation in industrial environments. The most well known examples are represented by Automated Guided Vehicles (AGVs) used for logistic operations [35]: these systems are fully autonomous, and the intervention of the user is limited to supervision operations. However, mobile robots can be also utilized as users' assistants, for goods transportation, in small-scale industrial scenarios. An example is represented by the system described in [8]: in this application, the mobile robot assists the operators of tire workshops, for transporting heavy loads (in this case, the wheels). This is a typical example in which, generally, operators do not have any specific robotic-related education: it is then mandatory to equip the robots with effective user interfaces, for the successful application of robotic systems in those contexts. In the last few years, the concept of natural user interfaces (NUIs) has been formalized. This paradigm entails utilizing direct expression of mental concepts by intuitively mimicking real-world behavior. NUIs provide a natural way, for the user, to interact with robotic systems, since the interaction is reality-based and exploits users' pre-existing knowledge, utilizing actions that correspond to daily practices in the physical world [5]. This is achieved, for instance, by letting users directly manipulate objects, thus spatially and physically interacting with robots, rather than programming them by typing commands on a keyboard (which is the traditional programming paradigm). Novel technologies are then utilized, such as voice, gestures, touch and motion tracking [6, 7].

In the literature, one of the most frequently utilized paradigms for providing intuitive interaction is gesture recognition. Gestures are typically recognized by means of colored markers [14], or directly detecting human palm [15]. Most of the techniques available in the literature exploit vision systems [13, 10, 14, 15]. One of the main drawback of such solutions is that, typically, vision systems require a careful control of the environmental conditions, in terms of proper lighting conditions and camera angles [14]. These conditions are often too restrictive, in real application scenarios, in particular in outdoor applications.

Moreover, it is worth noting that these interaction systems require a *dedicated infrastructure*: this implies that these systems are not portable, since they are ef-

fective only when the user lies in the field of view of the sensors, that is in front of the camera [13]. Furthermore, these systems cannot track the robot since the user can barely move and follow it.

The method proposed in [8] partially overcomes the latter problem: in this application, the camera is mounted on the robot, and the user is allowed to follow the robot in its movements. However, also in this case, the user must stand in front of the camera in order for the gestures to be recognized. While gestures are utilized for providing the robot with high level commands, they are not suitable for precisely controlling the robot's trajectory. For this purpose, in [8] the authors introduce the use of a remote control device (the Geomagic Touch) for the teleoperation of the robot. Although such a device provides the user with a haptic feedback, the main drawback of this interaction system is in the fact that it lacks embeddedness in real space [7], since the user is forced to sit at a desk, to teleoperate the robot. Instead, exploiting intuitive human spatial skills by letting the user move in real space while interacting and providing good spatial mappings between objects and the task are essential [36]. Thus, such interaction system appears to be mainly suited for teleoperated applications, where the distance between the user and the robot is intrinsic to the application, and is not introduced solely by the human-robot interface.

In Sec. 2.1 [37] we proposed a novel interaction system that overcomes the aforementioned main drawbacks of traditional systems. Specifically, we proposed a novel hands-free infrastructure-less natural interface for human-robot interaction, based on wrist-motion recognition. We utilized the term *infrastructure-less* for indicating the fact that the proposed interaction system does not require any additional dedicated equipments (such as a camera, for example), thus enforcing portability, without causing any limitation in the physical area where interaction occurs. The user has complete freedom of motion, which makes the interaction very natural and intuitive. By means of the proposed interaction system, the user has freedom of movement and is immersed in the environment where the robot moves, being able to track it with non-fragmented visibility [7]. This greatly improves the usability and effectiveness of interaction, in particular in those scenarios where the user shares the same working area as the robot, since she/he can control the robot standing in its proximity.

Specifically, the proposed interaction system is based on the use of an everyday device that allows recognition of the wrist and arm motion: in particular, without loss of generality, we utilized a commercial smartwatch, which could be easily replaced by a common wristband for activity tracking. In general, any commercial device equipped with accelerometers, gyroscopes and/or magnetometers, which can therefore measure movements of the wrist of the user, fits the proposed approach. In Sec. 2.1 we considered, as a target application, a quadrotor used for inspection operations. In this Chapter, we extend the proposed methodology to the case of a mobile robot, which can be used for heavy loads transportation in industrial and domestic applications. As a basis for comparison, we will consider the system recently presented in [8].

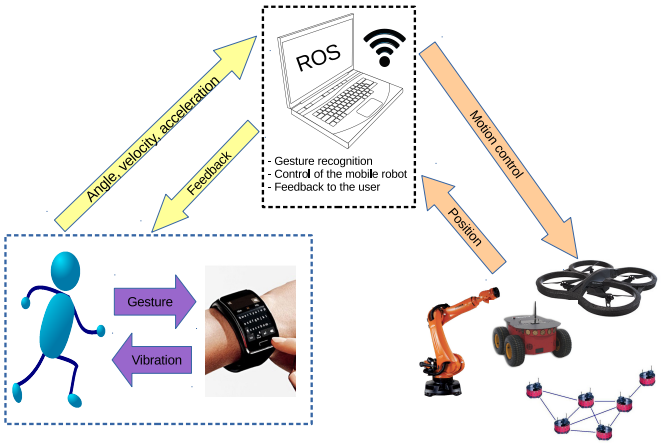


Figure 2.7: Overview of the proposed interaction and control system.

2.2.2 Multi-modal control architecture for the interaction with a mobile robot

2.2.2.1 Overview of the system

In this work we extend the interaction and control system first introduced in Sec. 2.1 [37]. The system is composed of different interconnected elements: the user wears the smartwatch or the activity tracker wristband, which is used to acquire the motion of her/his arm and wrist. Specifically, characteristic measurements related to the motion are considered, such as angular displacement, angular velocity and linear acceleration. A mobile robot is considered, and, in general, it is supposed to be equipped with a motion control and a localization system: since these problems have been widely addressed in the literature, they will not be considered in this work (see, for instance, [38] and references therein).

A data processing system is then implemented (on the smartwatch, on the robot on-board control system, or on a separate computation unit) for translating the user's motion into commands for the robot. Specifically, the motion of the user is analysed for recognizing gestures (used for imposing high-level commands to the robot), or for defining velocity commands (i.e. the robot's velocity is controlled as a function of the user's motion).

Feedback can be provided to the user, in terms of modulated vibration frequency of the smartwatch that informs her/him about the current status of the system.

2.2.2.2 Control architecture for a wheeled mobile robot

In the following we describe a specific realization of the proposed interaction system: in particular, we consider a smartwatch utilized for letting a human operator interact with a wheeled ground mobile robot.

We will hereafter consider a wheeled mobile robot, moving in a two-dimensional space. The configuration of the mobile robot is defined by three variables $[x, y, \theta]$. In

particular, $[x, y] \in \mathbb{R}^2$ represent the position of a representative point of the robot (e.g. its barycenter, the center of the wheel axis, etc.) with respect to a global reference frame, and $\theta \in [0, 2\pi)$ represents the orientation of the robot. In this work we will consider a mobile robot with differential drive kinematics, namely a mobile robot with two independently actuated wheels. This choice is motivated by the fact that this kind of robot is quite common in several applications, and its simple kinematic structure allows to keep the notation simple. Nevertheless, the proposed methodology can be trivially extended to consider more complex kinematic and dynamic models. For further details, the reader is referred to [38].

Define now $\omega_R, \omega_L \in \mathbb{R}$ as the angular velocities of the right and left wheels, respectively. Moreover, let $r > 0$ be the wheel radius, and $d > 0$ be the distance between the two wheels. Hence, it is possible to define the linear and angular velocities of the robot, v and ω, respectively, as follows:

$$v = \frac{r\left(\omega_R + \omega_L\right)}{2}, \quad \omega = \frac{r\left(\omega_R - \omega_L\right)}{d}. \tag{2.11}$$

Subsequently, the kinematic model of the differential drive robot can be written as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega. \tag{2.12}$$

The interaction and control architecture introduced in Subsection 2.2.2.1 requires the system to identify *gestures* performed by the user: these gestures are then utilized for changing the operational mode of the robot. A methodology for recognizing gestures from the arm and wrist motion was introduced in Sec. 2.1 and [12]: this method is based on template matching [28].This classical approach represents a standard algorithm for this kind of applications and was shown to exhibit good performance.

Specifically, data acquired by the accelerometer, gyroscope and magnetometer of the smartwatch are utilized. A template is built, off-line, for each desired gesture: subsequently, the user's motion is compared, at run time, with those templates, by using template matching. A decision whether a gesture has just occurred is then taken, on-line, by computing suited metrics of comparison. In particular, training data consisting of 60 repetitions of each wrist gesture are used to build templates. Signal epochs containing each repetition are manually selected and, then, aligned based on the normalized cross-correlation function [30]. Thus, templates are built by averaging aligned epochs. As a metric of comparison for template matching we use the correlation coefficient [31]. Further details and the experimental validation of this methodology for gesture recognition are in Sec. 2.1.

While this approach can be used, in principle, with any set of gestures, we defined templates, without loss of generality, for the following gestures:

1. *Up*: sharp movement upwards,

2. *Down*: sharp movement downwards,

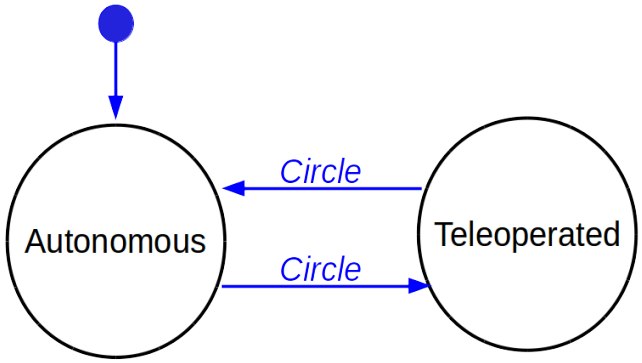


Figure 2.8: State machine of the control algorithm.

3. *Circle*: movement in a circular clockwise shape,

4. *Left*: sharp movement to the left,

5. *Right*: sharp movement to the right.

These gestures represent a common choice [22], and have been chosen since they represent movements that are easy to reproduce, without requiring previous training. At the same time, these movements are not likely to happen accidentally, as the user moves in her/his daily activities.

The high level control architecture utilized for controlling the mobile robot is described by the state machine diagram depicted in Fig. 2.8.

The system is initialized in the *Autonomous* state. In this state there is no interaction between the user and the mobile robot. Hence, the mobile robot is controlled by means of the internal, preprogrammed, control algorithm. If no control algorithm is defined, as in the application presented in this work, then the mobile robot is stopped.

Using the *Circle* gesture, the user can move the system to the *Teleoperated* state. In the *Teleoperated* state, the user can directly control the motion of the robot. It is possible to go back to the *Autonomous* state using again the *Circle* gesture. It is worthwhile noting that, among the gestures we are able to detect, the *Circle* is the only one exploited in the current implementation of the system. However, the state machine in Fig. 2.8 can be easily extended to include the other gestures, which could be utilized, as an example, to implement semi-autonomous behaviors such as tracking predefined trajectories, or following the user.

In the *Teleoperated* state, the pose of the smartwatch is translated into a velocity command for the robot: the angles of the smartwatch are directly translated into velocity control inputs for the robot, in a natural and intuitive manner. To this end, refer to Fig. 2.9 for the definition of the smartwatch angles, and consider the kinematic model of the robot introduced in (2.12). The *Roll* angle is used to control the linear velocity of the mobile robot. In particular, let $\vartheta_r \in [-\pi/2, \pi/2]$ be the

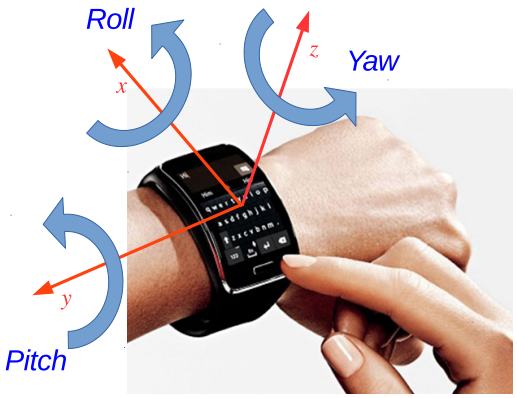


Figure 2.9: Angles of the smartwatch.

roll angle. Then, the linear velocity command is computed as follows:

$$v = K_r \, \vartheta_r \tag{2.13}$$

where $K_r > 0$ is a constant defined in such a way that the maximum angle that is achievable with the motion of the wrist corresponds to the maximum linear velocity of the mobile robot.

In a similar manner, the *Pitch* angle is used to control the angular velocity of the mobile robot. In particular, let $\vartheta_p \in [-\pi/2, \pi/2]$ be the pitch angle. Then, the angular velocity command is computed as follows:

$$\omega = K_p \, \vartheta_p \tag{2.14}$$

where the constant $K_p > 0$ is defined similarly to K_r in (2.13).

2.2.2.3 Feedback to the user

The vibration of the smartwatch is utilized for providing the user with a haptic feedback. In particular, in this application we utilize vibration to acknowledge the user's command, that is for informing the user about recognition of gestures. In particular, a single short vibration notifies the user when a gesture has been recognized by the control architecture. This is very useful for providing the user with an immediate feedback about the current status of the robot with reference to Fig. 2.8, and then on the possibility of teleoperating it. This significantly increases situation awareness.

It is worth noting that such a feedback aims at solving, at least partially, the ephemerality of gestures. As discussed in [27], the ephemeral nature of gestures implies that, if a gesture is not recognized or is misrecognized, the user has little information available to understand what happened. This kind of sensory feedback tackles the first condition, by informing the user whether a gesture has been recognized. Misrecognition has not been considered yet, in the current implementation: this can be solved utilizing visual information on the screen of the smartwatch, or speech synthesis methods. However, as shown in Sec. 2.1, we experimentally verified that gesture misrecognition rate is quite low, which significantly reduces the need for this additional information.

2.2.3 Experimental setup

To experimentally validate the proposed HRI system, we implemented the interaction and control system proposed so far. Specifically, the proposed system was implemented utilizing a Pioneer P3-AT mobile robot and a Samsung Gear S smartwatch. Due to the limited computation capabilities of the elaboration unit utilized for the control of the mobile robot, it was not possible to implement the gesture recognition and control architecture on-board the robot itself. Therefore, an external computer was utilized, and the overall architecture was implemented in ROS [33].

The experimental setup is then implemented as follows.

- The user wears the smartwatch on her/his right wrist: the motion of her/his forearm is then acquired by the smartwatch.
- Characteristic measurements related to the motion (namely accelerations and angular velocities) are then sent from the smartwatch to an external computer, via Wi-Fi communication.
- The computer is in charge of processing the received data, to perform gesture recognition and for computing the velocity commands, as defined in (2.13) and (2.14).
- Commands are then forwarded to the mobile robot via Wi-Fi communication.
- An acknowledgement is sent, via Wi-Fi, from the computer to the smartwatch, as soon as a gesture has been recognized. A short vibration is subsequently imposed to the smartwatch.

2.2.4 Evaluation results

Experiments aimed at evaluating the usability of the proposed interaction mode with the mobile robot. Additionally to the features of being hands-free and infrastructureless, the objective of this evaluation was to assess the real benefit of the embeddedness in real space enabled by the use of the smartwatch. For this purpose, the proposed interaction system was compared to a simple remote control system implementing unilateral teleoperation. For this purpose, we used the Geomagic Touch device: the user can move the end-effector of the device, and the motion is translated into a velocity command for the mobile robot.

Three different experiments were implemented, involving 13 users (in each experiment), for assessing the usability and the effectiveness of the proposed strategy. All the experiments were implemented both with the smartwatch and the remote control device. Details are given in the following; a video showing the experiments can be found at http://tinyurl.com/smartwatchVillani-IFAC2017.

2.2.4.1 Driving the robot through a cluttered environment

In this experiment, the user was requested to drive the robot through a cluttered environment. In particular, the user had to move the robot in an area of $3.3 \times 9.0\,[m^2]$,

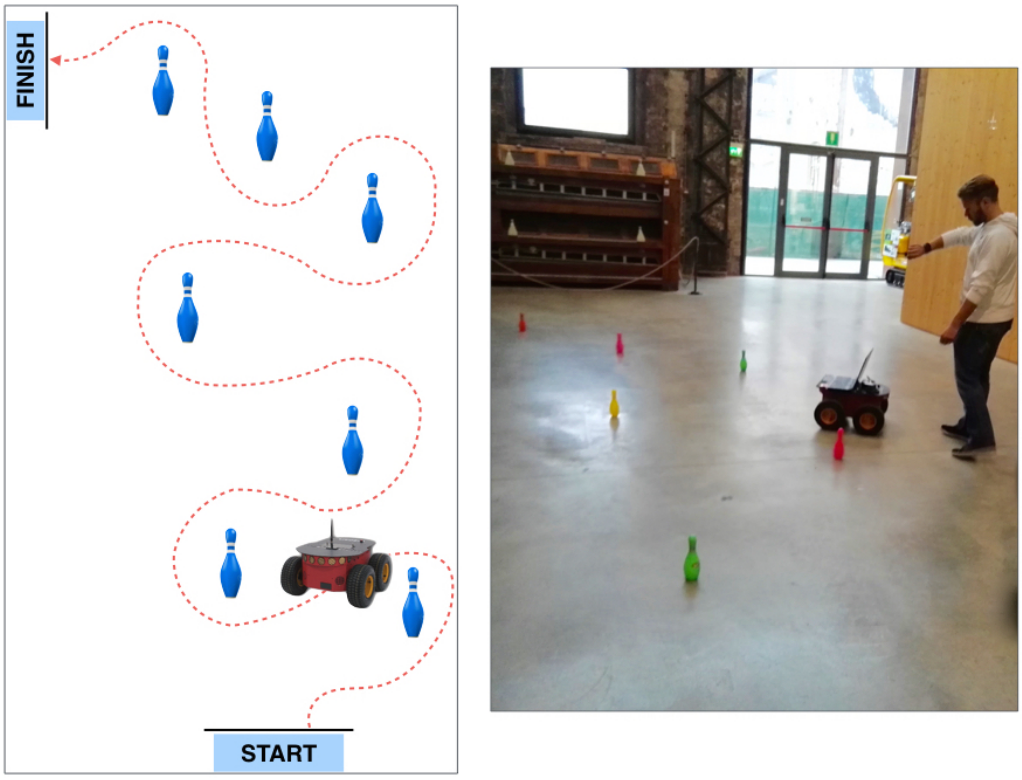


Figure 2.10: Driving through a cluttered environment: experimental setup.

where seven plastic pins were placed on the ground, as shown in Fig. 2.10. The user was instructed to drive the robots from the initial position to the goal position, performing a *slalom* path, without touching any pin. Each user was requested to perform the experiment with both the interaction systems.

For comparison purposes, we measured the total travel time, considering a penalty of $5\,[s]$ for each touched pin (if any). The results are summarized in Table 2.1. The results achieved with the smartwatch are significantly better than the ones obtained with the remote control device ($p < 10^{-3}$): while pins are typically avoided with both interaction systems, the smartwatch leads to a reduction of the execution time of approximately 33%.

2.2.4.2 Visiting a set of targets

In this experiment, the user was requested to drive the robot to visit a sequence of seven target locations, identified by plastic pins placed on the ground, as depicted in Fig. 2.11. Each target position was considered as visited as soon as the corresponding pin was knocked down. Each user was requested to perform the experiment with both the interaction systems. For comparison purposes, we measured the total time needed for knocking down all the pins.

The results are summarized in Table 2.2. The results achieved with the smartwatch are statistically significantly better than the ones obtained with the remote control device ($p = 0.002$): in fact, the smartwatch leads to a reduction of the

Table 2.1: Driving through a cluttered environment: results.

	Smartwatch			**Remote control**		
User	Travel time $[s]$	Pins	Total time $[s]$	Travel time $[s]$	Pins	Total time $[s]$
1	58	0	58	107	1	112
2	57	0	57	93	0	93
3	65	0	65	82	0	82
4	58	0	58	88	0	88
5	66	0	66	76	0	76
6	78	0	78	92	1	97
7	106	0	106	97	0	97
8	80	0	80	129	2	139
9	60	0	60	90	0	90
10	68	0	68	100	0	100
11	80	1	85	200	0	200
12	79	0	79	115	0	115
13	61	0	61	97	0	97
Mean	**70.46**		**70.85**	**105.08**		**106.62**

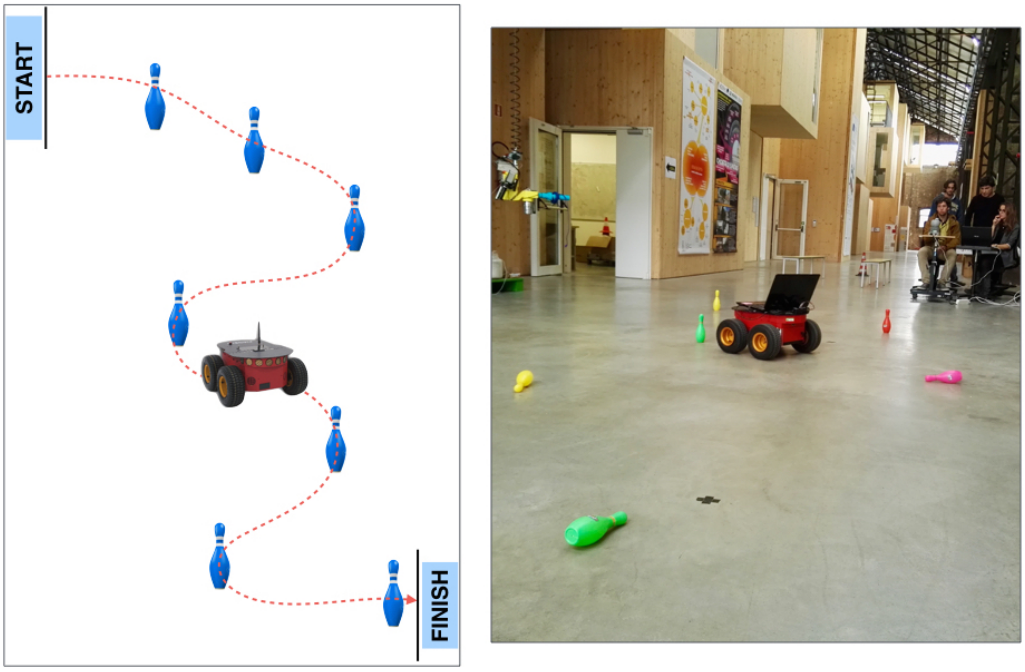


Figure 2.11: Visiting a set of targets: experimental setup.

Table 2.2: Visiting a set of targets: results.

	Smartwatch	**Remote control**
User	Total time $[s]$	Total time $[s]$
1	42	135
2	35	37
3	28	46
4	27	32
5	40	45
6	38	53
7	47	64
8	39	110
9	31	52
10	36	67
11	53	78
12	40	107
13	33	56
Mean	**37.62**	**67.85**

execution time of approximately 45%.

2.2.4.3 Exploring a building

In this experiment, the user was requested to drive the robot to perform a path that emulates the exploration of a building. The environment is depicted in Fig. 2.12. One of the advantages of the interaction system proposed in this work is in the fact that the user can move along with the robot, while exploring an environment. Conversely, utilizing the considered remote control device, the user is forced to sit at a desk: in this case, live streaming of images acquired from a camera installed on-board the mobile robot was provided to the user, to help her/him to drive the robot.

The results are summarized in Table 2.3. The results achieved with the smartwatch are statistically significantly better than the ones obtained with the other device ($p \simeq 10^{-7}$): in fact, the smartwatch leads to a reduction of the execution time of approximately 55%. The results of this experiment show that the use of the remote control device requires a big effort to the user to control the robot standing in a fixed position (i.e., sitting at the desk where the device is located) and not relying on direct view. On the contrary, being able to follow the robot and sharing the same physical space by means of the proposed interaction methodology increase the effectiveness of the interaction, since human spatiality is exploited [36].

2.2.5 Conclusions

In this Chapter we introduced a novel HRI approach that allows a hands-free natural interaction with a mobile robot. In particular, the motion of the user's wrist is

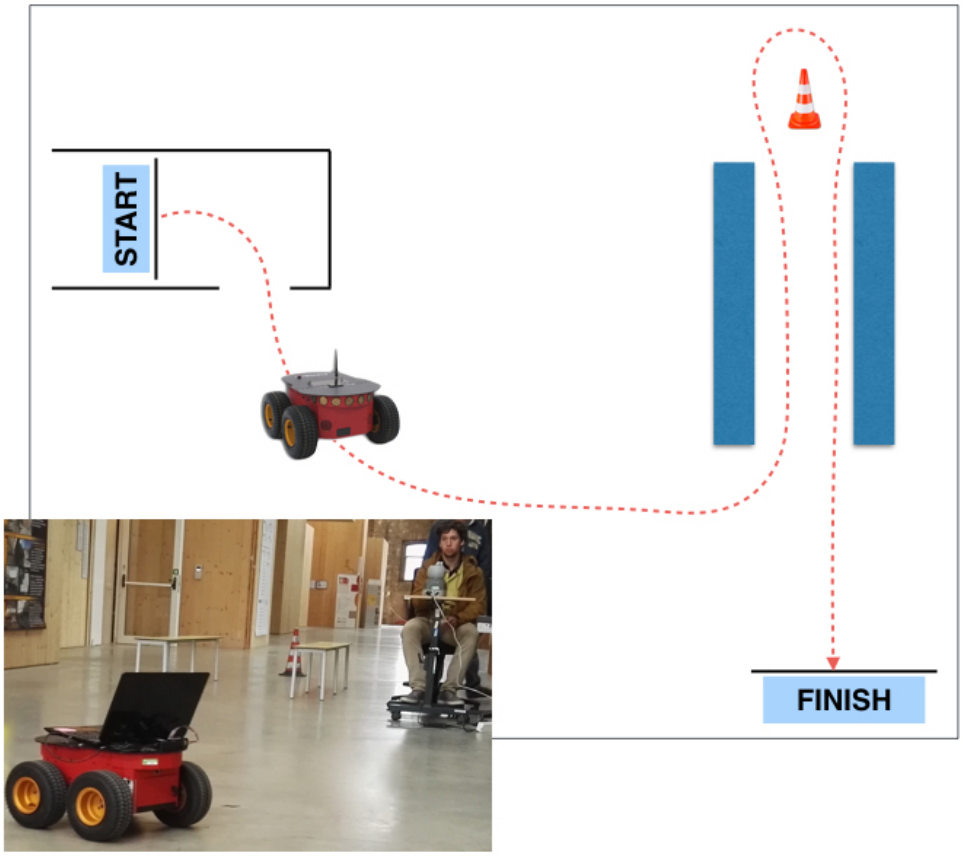


Figure 2.12: Exploring a building: experimental setup.

Table 2.3: Exploring a building: results.

	Smartwatch	**Remote control**
User	Total time $[s]$	Total time $[s]$
1	44	105
2	43	71
3	42	133
4	43	83
5	47	109
6	51	100
7	46	117
8	51	82
9	42	96
10	46	102
11	57	131
12	57	95
13	49	145
Mean	**47.54**	**105.31**

recognized by means of a general purpose device, such as a smartwatch, and exploited to recognize gestures and define control inputs for the robot. Haptic feedback is provided to the user by means of a modulated vibration that informs her/him about the recognition of gestures, thus increasing situation awareness. The usability of the proposed HRI approach based on the smartwatch was experimentally evaluated and compared to the use of a remote control device for the teleoperation of the robot. Usability was measured as the time required to follow three paths, having different goals and setups, by the two piloting modalities. The use of the smartwatch proved to be more intuitive and easy, allowing, in the 97% of the performed trials, to complete the tasks in much less than the time taken when using the haptic device.

Chapter 3

TIREBOT

3.1 Introduction

Collaborative robotics is getting more and more popular both in the research and in the industrial practice [39]. In the collaborative paradigm, robots do not replace humans but, rather, they help operators in accomplishing a common objective in a shared workspace. This idea has found many applications as, e.g., in cooperative surgery [40, 41, 42], in shared control of groups of robots [43, 44, 45] and in cooperative transportation [46]. Since humans and robots usually share the same working environment, one of the most important issues in human robot collaboration is safety, i.e. the robot must not represent a danger for the operator [47, 48]. Pushed by the need of safety metrics, also International Organization for Standardization (ISO) is moving towards the definition of a standard which regulates the Human/Machine interaction [49]. Several safe and collaborative robots are available in the market (e.g. Kuka iiwa[1] or Universal UR robot[2]) but, nevertheless, in order to optimize the collaboration for a given working process, task specific robots [50] have to be designed.

In the industrial setting, especially in SMEs (Small and Medium-sized Enterprises), many operations are still (partly) executed manually because they require the operator's expertise and capability of handling complex scenarios. Nevertheless, it frequently happens that such operations involve also the execution of simple activities (e.g. transportation of a load) that could be easily solved by a robot. Thus, it is often possible to split a manual operation into a set of low level tasks (e.g. load lifting, transportation) that can be assigned to a robot and into a set of high-level tasks (i.e. cognitive demanding activities) for which the presence of the human is necessary.

An example of such a kind of operations is wheel processing, carried on manually many times per day in tire-workshops. Each time a vehicle arrives, a wheel is extracted from the car and taken manually to the tire changer machine, used for changing the tire. Once a new tire has been mounted, the wheel is manually taken to a balancing machine, used for balancing the wheel, and then it is taken back to

[1] www.kuka.com

[2] www.universal.com

the vehicle (the process is summarized in Fig. 3.1). This procedure has to be done for each wheel of the vehicle, for many vehicles per day. In this process, the user has to transport and lift the wheel (that can weight up to 50 kg for cars) many times. While only the human can properly use the machines and mounting/dismounting the wheel on/from the vehicle, the wheel lifting and transportation can be delegated to a robot. In this case the robot and the operator will move in the same environment, the tire-workshop.

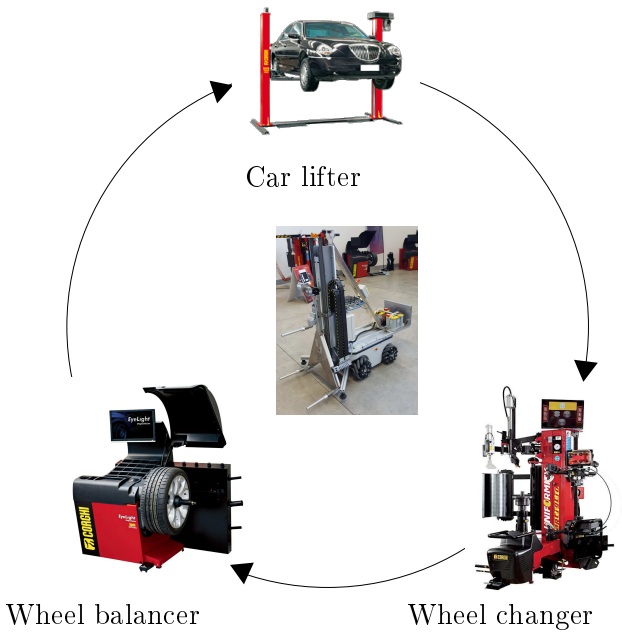


Figure 3.1: A scheme, summarizing the wheel processing

In this Chapter, we present TIREBOT (a TIRE workshop roBOTic assistant), a collaborative mobile robot capable of assisting the operator in a wheel management procedure. TIREBOT can lift and transport wheels and the operator can interact with the robot using either a gesture based interface or by teleoperation. TIREBOT is an omnidirectional robot since it requires to move in cluttered spaces and it has to quickly move aside in order to guarantee a safety distance from the human operator. Furthermore, the robot is endowed with a dedicated gripper used for firmly grabbing the wheels to lift and transport. TIREBOT needs to move in an environment populated by obstacles, either fixed (e.g. parts on the floor) or dynamic (e.g. other operators), and it has to cooperate very closely with the operator (e.g. when the operator takes the wheel transported by the robot). Purely reactive navigation techniques (see e.g. [51, 52]) would be suitable for moving TIREBOT in a cluttered environment but they would treat the operator as an obstacle and they would prevent the system from collaborating. On the other hand, collaborative reactive strategies like the danger field [53] would ensure a safe collaboration at the price of poor navigation performance. Thus, we propose a novel collaborative navigation strategy that combines potential based approaches with the danger field concept in order to obtain good (and computationally cheap) navigation per-

formance and a collaborative behavior. This makes TIREBOT a perfect helper for the tire-workshop operator since it can take care of heavy duty activities. The performance of the robot are validated on a real tire-workshop. Furthermore, since the ability of transporting heavy loads in a human populated environment is useful in many other scenarios, TIREBOT has been validated also in a different scenario: an electrical vehicle workshop. Here, TIREBOT was used as a smart forklift capable of moving heavy lead batteries from the storage room to the workshop's area where maintenance on electrical vehicles takes place.

Preliminary work on TIREBOT has been presented in [54, 55]. The contribution of this work is threefold:

- A complete description of the mechatronic design of TIREBOT and of its interfaces
- A collaborative navigation strategy blending standard artificial potential based navigation and the danger field concept
- An extensive experimental validation of TIREBOT, both in its native scenario, for testing its effectiveness and usability, and in a different scenario, for testing its potentials in other applications

3.2 Scenario and Requirements

In this section, the way TIREBOT needs to be used in the tire workshop is described and, consequently, the requirements for the mechatronic and control design of the robot are obtained.

Referring to Fig. 3.1, the operator can collaborate with TIREBOT as follows. Before starting the wheel process, the operator switches on TIREBOT, which is parked next to its control station. The robot recognizes the user and starts following him automatically avoiding possible obstacles, until the car lifter is reached. Then, the operator approaches to the wheel that has to be processed and starts to unscrew it from the vehicle. Once the wheel has been unscrewed the user steps aside and lets the wheel fall on the ground; then, the operator can push it on the lower forks of TIREBOT, that will lift the wheel and grab it by lowering the upper fork. When working with the operator, TIREBOT is controlled to actively ensure a safe behavior with respect to the user. The operator can, then, order the robot through gestures to take the wheel to the tire changer. While the robot is transporting the wheel, the operator can start unscrewing another wheel and when TIREBOT is back, it is ready to load the robot again for another mission. Once all the wheels have been transferred to the tire changer, the operator can go there, ordering TIREBOT to follow him, and start the maintenance on the wheel. In a similar way, the operator orders TIREBOT to take the wheels to the wheel balancer and then, after wheel balancing operation took place, back to the vehicle. TIREBOT is always wireless connected to the control station where the user can receive a video streaming from the robot and it can remotely control TIREBOT with the haptic device in order

to drive the robot to a different destination (e.g. because of a fault on one of the processing machines).

In order to achieve this scenario, TIREBOT will have to be:

R1 Agile and reactive, since it has to move in a cluttered space and next to the operator.

R2 Capable of grabbing and transporting a wheel

R3 Simple and intuitive, since it has to be commanded by the tire workshop operator while working

R4 Safe, since it shares the environment with humans

R5 Collaborative, since it has to cooperate with the operator

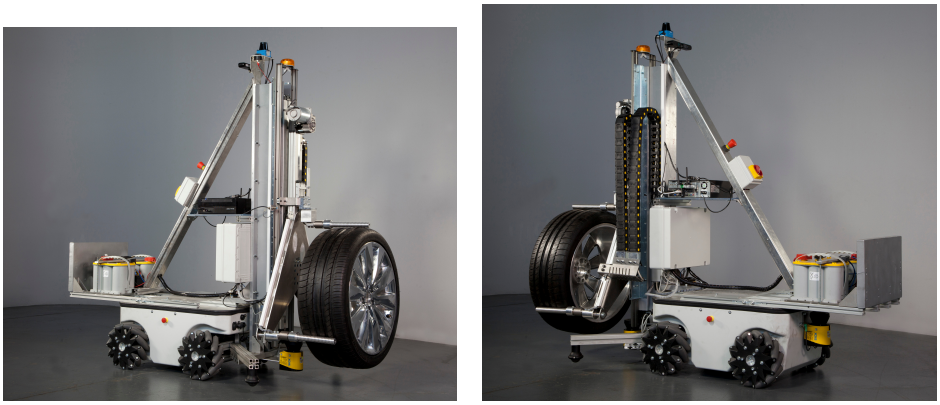
The robot is designed to operate in two modalities. When the robot has to move from a point of the working area towards a goal position, an Autonomous navigation modality is implemented, where the robot detects and avoid obstacles only according to their relative position. When the robot has to help the human co-worker, a Safe Cooperation mode is implemented. During cooperation with a human, the robot should work with a tire workshop operator that, in some phases of her/his job, must work close to the robot. For example when the operator, once the tire is removed from the car, has to load the tire on TIREBOT. In this case, the robot should assist the operator without hindering him, moving away if the user has to work in positions occupied by the robot and actuating the wheel lifter without harming the human operator. On the other hand, if for some reasons the operator, while working side by side to the robot, moves quickly towards the robot, TIREBOT has to move away to avoid collision with the co-worker that could cause injuries to her/him. Finally, the robot needs to be able to detect when to switch from the Free Navigation modality to the Safe Cooperation modality and vice-versa.

3.3 Mechatronics of TIREBOT

TIREBOT is a mobile collaborative robot helper explicitly designed for the job it has to help with. This high focus on the domain of application brings several advantages with respect to general purpose arms on a mobile base. First of all, the mechanical structure is simpler and, consequently, the control structure is less complex. Furthermore, the few tasks the robot has to assist the operator with are well defined and it is possible to design its behavior while preserving safety exploiting the simple control structure. The navigation strategy is simple because the robot is not assumed to move in a completely unknown location but all the available information is exploited (the layout of the workshop and the position of the goals, e.g. the position of the machineries). In summary, the specificity of TIREBOT allows to get rid of superfluous details that would make the control system unnecessarily more complex.

3.3.1 Mechanics

TIREBOT is represented in Fig. 3.2. In order to fullfil requirement R1, the Neobotix MPO-500, shown in Fig. 3.3(a), has been chosen as a mobile base for TIREBOT. The omnidirectional wheels of the base allow TIREBOT to move instantaneously in all directions, making it very agile and reactive and suitable for moving in the tire workshop, a very cluttered environment with parked vehicles and machines, where there is often no space for maneuvers.

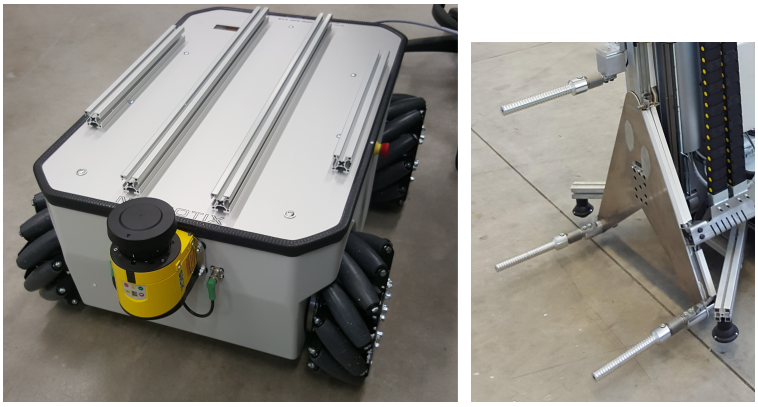


(a) (b)

Figure 3.2: The robot used during the experiments

TIREBOT has a frontal and a posterior SICK S300 laser scanners that are exploited for obstacles detection purposes. A further SICK TIM-310 laser scanner is fixed on the top of the robot: it is used to detect reflective markers, located in known positions, whose position is exploited for estimating the pose of the robot using an Extended Kalman Filter [56].

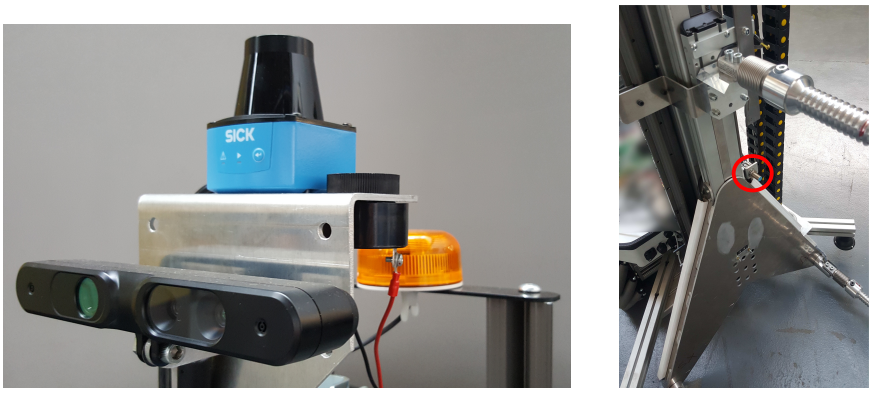
The robot is equipped with an ad-hoc built wheel gripper device, which makes the robot capable of loading cars and trucks tires and moving them in the tire workshop satisfying the requirement R2. The gripper is designed in order to grab wheels up to 50 kg. The wheel gripper can be moved by the lifting system shown in Fig. 3.2(a). The lifting system is composed by a linear guide, actuated by a brushless DC motor, which carries two forks (the lower ones) and a second and shorter linear guide, even this actuated independently by another brushless DC motor, with a third fork. The stroke of the linear guides' carts is limited by four limit switches fixed on the robot (Fig. 3.4(b)). The tire must be placed on the lower forks and held on by the operator until TIREBOT has grabbed it safely by pressing the upper fork on the upper part of the wheel. The forks have a particular notched shape, which lets them to grab firmly a tire. All the forks have a load cell that can sense the presence of the wheel and measure the wheel's weight (in the case of the lower forks) or if the tire has been grabbed properly (in the case of the upper cell). This lifting device is fixed to a steel girder enforced by a thick steel plate, which avoids the torsion of the lifting device due to the robot's movements and vibrations and to the load's inertia. The whole lifting structure lays on the rear side of the chassis of the Neobotix MPO-500 and on the two middle aluminum profiles. The structure is further strengthened



(a) (b)

Figure 3.3: The Neobotix MPO-500 (a) and a particular of the forks (b)

by adding an oblique aluminum profile, fixed both to the top of the lifting device and of the robot. We chose to place the wheel grabber on the rear of the robot for safety reason: if the forks (Figure 3.3(b)) were on the front of the robot they could possibly harm people during the normal robot's forward movements. Furthermore, on the rear side of the robot it was possible to exploit the Neobotix MPO-500 frame to support the whole structure of the wheel gripper.



(a) (b)

Figure 3.4: The equipment carried by TIREBOT (a) and the forks with the limit switches (b)

In order to avoid overturning when the robot is lifting a heavy load, two oblique aluminum profiles are fixed on the lower part of the robot. These profiles have also two spherical wheels to avoid creep and jamming if the robot flips when it is moving. Furthermore, on the front of the robot a thick steel plate is mounted. It weights 40 kg and, with the help of the batteries, (see Figure 3.2(b)), acts as a counterbalance. The robot weights 80 kg. Its length is 865 mm, its width is 692 mm and its height is 386 mm (frontal laser scanner included). It has a payload of 150 kg. It is already equipped with an internal computer with Ubuntu 14.04 (Trusty Tahr) and ROS (Robot Operating System [57]) Indigo Igloo.

TIREBOT is endowed with a RGB-D camera capable of detecting human users [58]. The gestures and the postures detected by the camera are translated into commands for the robot, allowing an intuitive and non-invasive communication between the operator and the robot and, therefore, satisfying R3. The camera is also used for tracking the user's position and to allow the robot following the operator. Further information on the robot and on the adopted Human/Robot interface can be found in [54].

3.3.2 Electronics

In order to avoid the Neobotix MPO-500 to be damaged by parasite currents the electronic circuitry of the wheel gripper and the sensor is physically separated by the one of the robot. Three batteries power the system: two 12 V batteries in series give power to all the devices but the load cells amplifier, which is powered with -12 V by the third battery and with 12 V by the motor board. A three way switch powers on all the devices, while a safety switch interrupts power only for the motors. Schematics are represented in Fig.3.5. The batteries power on the motor board and

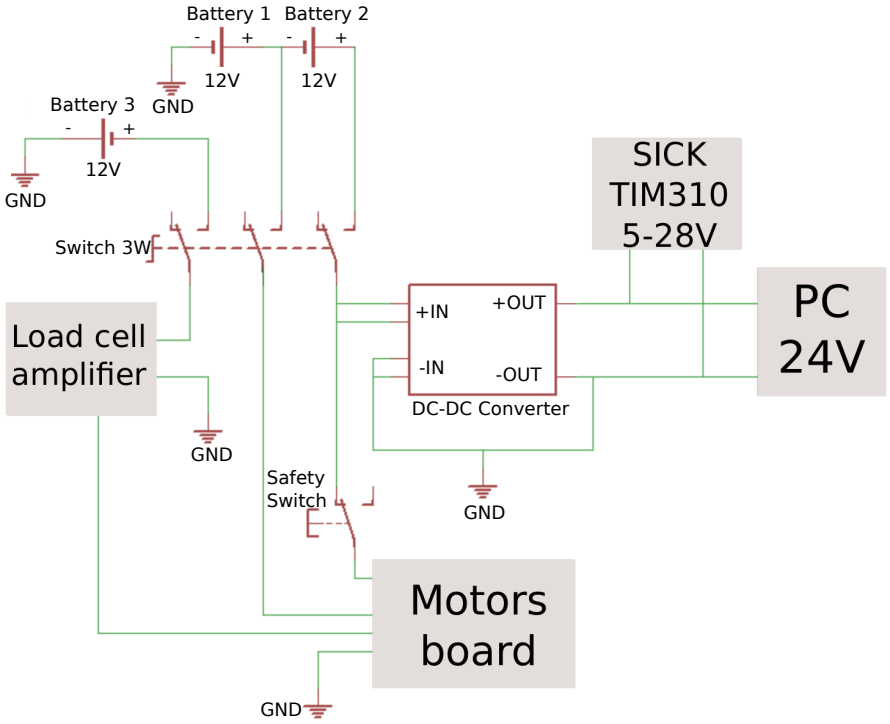


Figure 3.5: A sketch representing the power circuitry schematics

a DC/DC converter, which works as a current stabilizer that, in turn, powers on an industrial PC and the localization laser scanner. In order to avoid current laps, which could generate noise and disturbances for the sensors, all the devices have a common ground, which is connected to the robot's frame, that, in turn, discharges current to earth. The robot is equipped with other devices, which are not powered directly by the batteries but by the motor board. In particular, the robot is endowed with:

- Four limit switches placed on the linear guides; each guide has a lower and an upper limit switches that sense the linear guide's cart runs out, respectively, on the lower and upper limits (see Figure 3.4(b));

- Two warning devices: a buzzer that, when the robot is moving, sends acoustic signals to warn people who are in the robot's neighborhood of its movements and a flashing yellow light that, when the robot is moving, blinks warning people in the robot's surroundings (Figure 3.4(a)).

Furthermore, the robot is equipped with a RGB-D camera, the ASUS Xtion Pro Live, which is powered by the industrial PC through USB, and with the localization laser scanner SICK TIM-310 (Fig. 3.4(a)). The industrial PC is also connected to the motor board through a RS-422 serial communication cable.

The load cells of the two lower forks are connected in parallel, while the load cell on the upper fork is connected singularly to the amplifier. The amplifier is connected to the motor board's analogue input channel which also provides it the 12 V and the ground reference.

The motor board is powered by both 12 V and 24 V, actuates the 24 V DC brushless motors through two of its three H-bridges, provides an interface to the connected devices and powers them. The board also controls the safety flashing light and the buzzer. The motor board does not only provide power to the connected devices, but it also acts as an interface to them. The stroke of the motors is limited by the board's software thanks to the four limit switches.

In order to prevent damages to the board when the motor that moves the lower fork lowers a heavy load and starts to produce energy instead of consuming, a further device was added. This device is a comparator that closes a switch that reroute the energy produced by the braking motor on a resistance when the voltage on the H-bridge is greater than 30 V.

3.3.3 Software architecture

The software architecture of TIREBOT was developed by using UML specification. In particular, Sequence Diagrams and Finite State Machine charts were used to describe the behaviour that the robot should have.

TIREBOT can be used in three different working modalities: an autonomous mode, a teleoperation mode and the gesture interaction mode. The Finite State Machine (FSM) governing TIREBOT's behavior is represented in Fig. 3.6.

Once the robot is switched on, the user can choose a working modality through gestures.

In "Teleoperation Mode", the user can remotely interact with TIREBOT from a control station. The control station is made up of an haptic device, the Geomagic Touch, that allows to teleoperate the robot and to receive some force feedback to inform the user about the perceived obstacles, and by a laptop. The computer is connected to the robot's industrial PC through an ad-hoc wifi network, while the industrial PC communicates with the robot's internal PC through Ethernet. The whole system runs Ubuntu 14.04 (Trusty Tahr) and ROS Indigo Igloo. The laptop

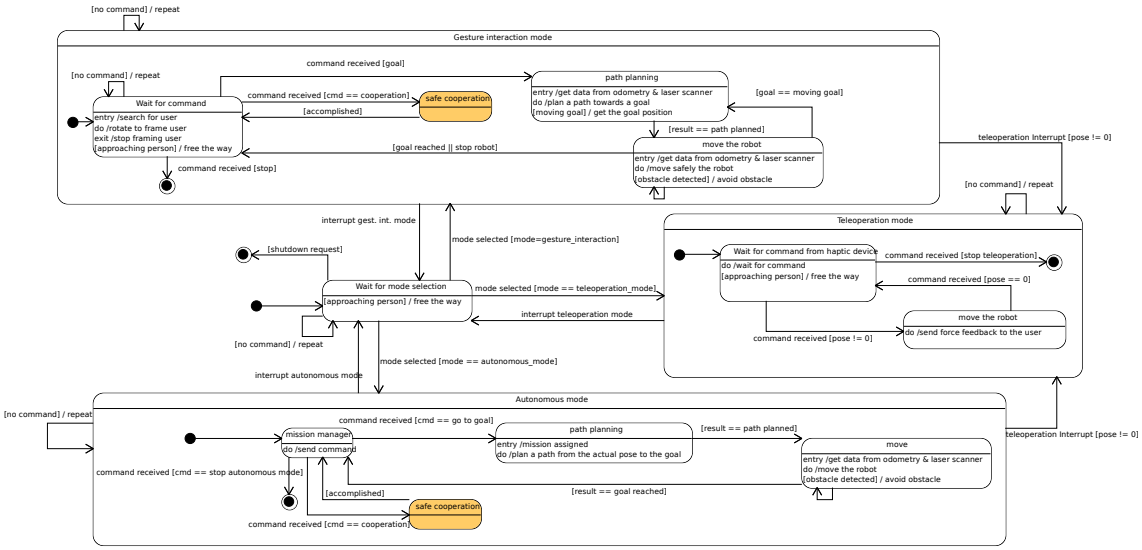


Figure 3.6: The TIREBOT's Finite State Machine

receives a visual stream from the RGB-D camera of TIREBOT and this allows the user to monitor the behavior of the robot. In order to improve safety, the user can always switch to this working mode from all the others modes, by simply moving the haptic interface.

In "Autonomous Mode", a commanded destination (e.g. to the balancing machine) is processed by the mission manager. Then, standard artificial potential fields navigation is exploited for moving the robot towards the goal while avoiding possible obstacles.

In "Safe Cooperation mode", the robot interacts in a safe but collaborative way with the operator. This working modality is described in detail in Sec. 3.4.

In "Gesture Recognition Mode" the user can send commands to the robot by moving his arms. Once the robot enters in this state, it starts rotating on itself in order to frame and recognize the user from whom will take orders with the RGB-D camera. Once the operator has been identified, the robot rotates on itself framing the user and waiting for a command recognizable by the robot. Then the user can choose to make the robot follow him, to send the robot to some predefined destinations (e.g. the wheel processing machines) or to actuate the wheel gripper.

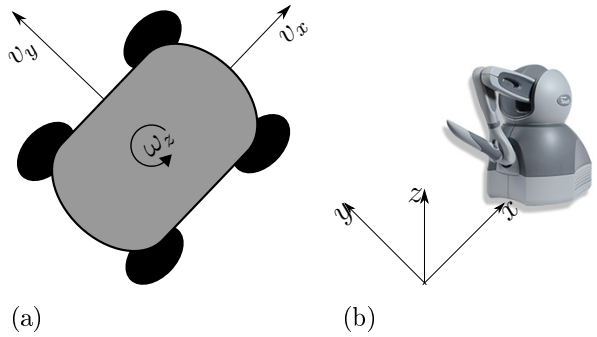
The operator can interact with the robot with a haptic device, the Geomagic Touch, which is connected to a laptop computer. These two devices constitute the robot's "control station". The laptop is connected to the robot's industrial PC through an ad-hoc wifi network, while the industrial PC communicates with the robot's internal PC through Ethernet. The whole system runs Ubuntu 14.04 (Trusty Tahr) and ROS Indigo Igloo.

3.4 Teleoperation and Navigation

In this section, we will describe into detail the teleoperation mode and the navigation strategy used by TIREBOT, i.e. a mix of an artificial potential based and of a safe cooperative navigation modalities.

3.4.1 Teleoperation

In teleoperation mode, the omnidirectional base of TIREBOT, sketched in Fig. 3.7(a) is teleoperated using the Geomagic Touch haptic device, represented in Fig. 3.7(b). The TIREBOT's base can be moved by applying three inputs: $v_x \in \mathbb{R}$, for moving forward, $v_y \in \mathbb{R}$, for moving laterally, and $\omega_z \in \mathbb{R}$ for rotating. The Geomagic Touch has six DOFs but only three of them are capable of providing a force feedback and they are used for controlling the motion of the base. The device has also a switch inside the inkwell that holds the pen, used to sense if a user is holding the pen.



(a) (b)

Figure 3.7: The reference frames for the robot (a) and for the teleoperation device (b)

The robot is teleoperated in rate mode [59], which means that the position registered on the haptic device is transformed into a speed which is then commanded to the robot, according to the following equation, where $k_v \in \mathbb{R}^+$ is a proportional constant:

$$\begin{aligned} v_x &= k_v \cdot p_x \\ v_y &= k_v \cdot p_y \\ \omega_z &= k_v \cdot p_z \end{aligned} \tag{3.1}$$

Here, p_x, p_y and p_z are the distances of the pen from a zero position which is in front of the device.

The user takes control of the robot's movements when the pen of the haptic device leaves its inkwell and the teleoperation commands has greater priority with respect to other sent commands (e.g. from the autonomous navigation).

The force feedback is computed using the data read by the frontal and rear laser scanners. Each ray i of the scanners reads the position of a point in the environment and returns in polar coordinates (d_i, α_i), where d_i is the distance of the scanner from the environment and α_i is the angular position of the environment with respect to the scanner frame. For each ray, a virtual form informing the user about the distance with the environment is produced and all the non zero forces are implemented on

the master haptic device. Formally, let $Q^* > 0$ be a predefined distance beyond which the user should not be informed about the presence of the environment. For each ray we design the following repulsive potential field:

$$U_{rep_i} = \begin{cases} \frac{1}{2} k_r \left(\frac{1}{d_i} - \frac{1}{Q^*} \right)^2 & d_i \leq Q^* \\ 0 & d_i > Q^* \end{cases} \tag{3.2}$$

and the corresponding repulsive force $F_i = -\nabla U_{rep_i}$. When $d_i > Q^*$, then U_{rep_i} is constant and, consequently $F_i = 0$ which means that if the environment is too far, no informative force is sent to the user. On the other hand, the smaller is the distance with the environment, the bigger is F_i, which means that the closer is the environment, the bigger is the magnitude of the force feedback provided to the user. Each force F_i is applied on the master device in the direction α_i of the corresponding ray.

The switch between autonomous and teleoperated modalities can lead to an abrupt feedback to the user, as reported in [60]. The force feedback we have experimentally experienced is rather smooth and therefore we have not implemented any compensation. Nevertheless, a smooth transition from autonomous to teleoperation mode can be ensured by implementing the technique proposed in [60].

3.4.2 Navigation

While navigating in the tire workshop, TIREBOT needs to wisely combine autonomous and safe cooperation mode. In the autonomous mode, the main goal of the robot is to reach a predefined goal while avoiding obstacles while, during the safe cooperation mode, the main goal is to stay close to the user for assistance purposes. Autonomous navigation will be achieved using the standard artificial potential method [51]. Safe cooperation will be achieved by exploiting the concept of danger field [61]. Finally, a condition for switching between the two navigation modalities will be provided.

More formally, TIREBOT is mounted on an omnidirectional base controlled in velocity and, therefore, its kinematic model is given by a simple integrator:

$$\dot{x} = u \tag{3.3}$$

where $x \in \mathbb{R}^2$ is the Cartesian position of the robot with respect to the inertial frame (i.e. achieved through the localization system). We do not consider the orientation because, thanks to the omnidirectionality of the robot, we will handle navigation and obstacle avoidance in the Cartesian space.

We will denote with $x_{o,i} \in \mathbb{R}^2$ the position of the i-th detected obstacle with respect to the inertial frame. Using the datat collected by the laser scanner, TIREBOT can compute the relative position and the relative velocity of the obstacle, $(x - x_{o,i})$ and $(\dot{x} - \dot{x}_{o,i})$.

The navigation is achieved by implementing a gradient descent, i.e. $u = -\nabla U_{TOT}$, where U_{TOT} is composed by three terms:

$$U_{TOT} = U_{ATT} + \eta\,(t)\, U_{REP} + [1 - \eta\,(t)]\, U_{DF} \tag{3.4}$$

The terms U_{ATT} and U_{REP} are the standard attractive and repulsive potential used in artificial potential navigation. Formally we have that

$$U_{ATT} = \frac{1}{2} k_{ATT} \|x - x_G\|^2 \tag{3.5}$$

where $x_G \in \mathbb{R}^2$ is the goal position and $k_{ATT} \in \mathbb{R}^+$ is the attractive gain. The gradient descent associated to this term, attracts the robot towards the minimum potential configuration, i.e. the goal configuration x_G. The repulsive potential U_{REP} is given by:

$$U_{REP} = \begin{cases} \frac{1}{2} k_{REP} \left(\frac{1}{\|x - x_{o,i}\|} - \frac{1}{Q^*} \right)^2 & if \ \|x - x_{o,i}\| \leq Q^* \\ 0 & otherwise \end{cases} \tag{3.6}$$

where $k_{REP} \in \mathbb{R}^+$ is the repulsive gain and $Q^* \in \mathbb{R}^+$ is a threshold parameter. The gradient descent associated to this term produces a behavior that moves the robot away from the obstacle, when the obstacle is close enough, or it does not influence the motion of the robot when the obstacle is far.

The term $U_{DF} = U_{SDF} + U_{KDF}$ is the Danger Field. It is composed by a static danger field U_{SDF}, that depends only on the distance between the robot and the obstacle, and by a kinetostatic danger field which also depends on the relative speed. Formally we have:

$$U_{SDF} = \begin{cases} \frac{1}{2} \frac{k_{SDF}}{\|x - x_{o,i}\|^2} & if \ \|x - x_{o,i}\| < Q_1 \\ & or \begin{cases} \|\dot{x} - \dot{x}_{o,i}\| \geq \bar{v} \\ \|x - x_{o,i}\| \leq Q_2 \\ cos\varphi > 0 \end{cases} \\ 0 & otherwise \end{cases} \tag{3.7}$$

and

$$U_{KDF} = \begin{cases} \frac{k_{KDF} \|\dot{x} - \dot{x}_{o,i}\| (1 + \cos\varphi)}{\|x - x_{o,i}\|^2} & if \ \|x - x_{o,i}\| < Q_1 \\ & or \begin{cases} \|\dot{x} - \dot{x}_{o,i}\| \geq \bar{v} \\ \|x - x_{o,i}\| \leq Q_2 \\ cos\varphi > 0 \end{cases} \\ 0 & otherwise \end{cases} \tag{3.8}$$

where $k_{SDF}, k_{KDF} \in \mathbb{R}^+$ are positive gains that can be used for tuning the effect of the danger field. Similarly to (3.6), the gradient of U_{SDF} implements a repulsive behavior if distance from the obstacle is lower than a certain threshold Q_1. The gradient of U_{KDF} produces a repulsive behavior if the robot and the obstacle are close enough, if their relative velocity is bigger than a certain threshold $\bar{v} \in \mathbb{R}^+$ and if the robot and the obstacle are approaching.

Term $\cos\varphi$ contains information about the direction the danger source is moving along and it is computed as:

$$\cos\varphi = \frac{(x - x_{o,i})^T (\dot{x} - \dot{x}_{o,i})}{\|x - x_{o,i}\| \, \|\dot{x} - \dot{x}_{o,i}\|} \tag{3.9}$$

The term $(1 + \cos\varphi)$ has a crucial importance: it nullifies the term U_{KDF} if the obstacle is moving away from the robot and modulates it, according the obstacle's direction, otherwise.

The gradient descent associated to the danger field produces a behavior that takes the robot away from the obstacles considering both the distance and the relative velocity between the robot and the obstacle.

In (3.4) the term $\eta \in \{0, 1\}$ is a signal used for selecting the navigation modality:

$$\eta(t) = \begin{cases} 1 \; if \;\; \|x - x_G\| \geq \delta \\ 0 \; otherwise \end{cases} \tag{3.10}$$

If the robot is close to its goal position (i.e. the distance of the robot from the vehicle or a wheel processing machines is lower than δ), then both the safe cooperation and the danger field are activated. Notice that when the cooperation mode is active, the robot is close to the goal destination and, therefore, the obstacle will be the operator. Otherwise, TIREBOT is navigating through the workshop and, therefore, the autonomous navigation mode is set. Thus, the robot is navigating using the standard artificial potential approach.

A collaborative and reactive safe navigation strategy is achieved by properly tuning the thresholds of the potential fields. In autonomous mode, the potential U_{ATT} is active and Q^* has to be big enough for ensuring that the robot stays sufficiently away from the obstacles. During the safe cooperation mode, it is necessary to choose $Q_2 > Q_1$. In this way, if the robot and the operator are far enough (i.e. $\|x - x_{o,i}\| > Q_2$), $\nabla U_{DF} = 0$ and the robot will not move away by the presence of the user. If the operator approaches TIREBOT too quickly (i.e. $\|\dot{x} - \dot{x}_{o,i}\| \geq \bar{v}$ and $\|x - x_{o,i}\| \leq Q_2$ and $cos\varphi > 0$), the gradient to U_{DF} pushes the robot away from the user in order to avoid dangerous situations. On the other hand, it is necessary that the user and the robot work very closely and, therefore, it is necessary that TIREBOT can approach the user without being pushed away. This actually can happen because if $\|\dot{x} - \dot{x}_{o,i}\| < \bar{v}$ the gradient of the danger field produces no effect on the behavior of the robot. This means that the robot can approach and cooperate with the user as long as it does not move too fast. Finally, if the user gets too close to the robot (i.e. $\|x - x_{o,i}\| \leq Q_1$) the danger field is active independently of the relative velocity and the gradient of U_{DF} produces a repulsive action on the robot to avoid damages. The threshold Q_1 has to be sufficiently small for avoiding to prevent the cooperation between the human and the robot. The threshold $Q_2 > Q_1$ needs to be chosen big enough to prevent the robot from hitting the operator while approaching at the maximum speed and for allowing a proper cooperation between the user and the robot.

The behavior of the robot can be summarized as follows (see also Fig. 3.8):

1. **Safe Area**: the obstacle is too far from the robot $\|x - x_{o,i}\| > Q_2$, the Danger Field is not active;
2. **Collaboration Area**: the obstacle is close enough to the robot $Q_1 < \|x - x_{o,i}\| < Q_2$, the Danger Field can be active based on its speed and on $\cos\varphi$;
3. **Forbidden Area**: the obstacle is too close to the robot $\|x - x_{o,i}\| < Q_1$, the Danger Field is always active.

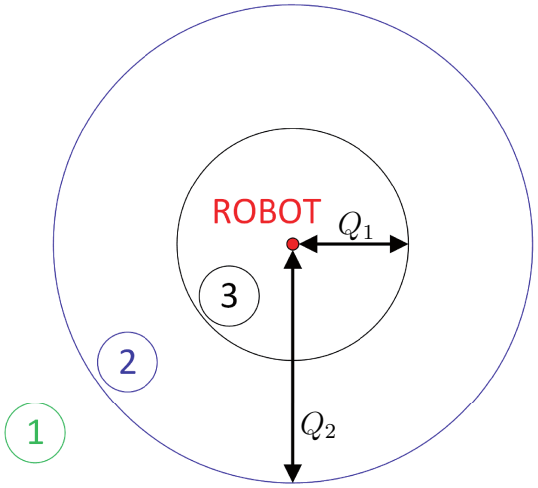


Figure 3.8: The areas involved into the "*Safe Cooperation*"

Note that the "*Safe Cooperation*" behavior, and consequently the "Danger Field", are switched off while the robot is teleoperated by the user. In this case, safety is achieved only by the capabilities of the user who receives a force feedback from the haptic device while teleoperating with it.

3.4.2.1 Navigation - Simulations and Experiments

In this section, the TIREBOT's navigation strategy is validated by means of simulations and experiments. Simulations are shown in the fist part of the video[3].

The robot (the black circle) stands still on the goal position (the small red circle) while a moving obstacle (the blue circle) approaches to it slowly (see Fig. 3.9). The speed limit for the activation of the Danger Field is higher then the speed of the obstacle, then the robot ignores the moving obstacle. Then the obstacle increases its speed and, when it approaches the robot, it is sensed as a danger by the robot that moves away. On the right side of the video it is shown the robot and the operator acting in an environment free from static obstacles, while on the left side of the video a non-moving obstacle (the green circle) was added. This was done with the intent to verify the behavior of the robot with the presence of both a moving and a still obstacle.

[3]https://youtu.be/dX4Hn2IS1eM

For the simulation, the following parameters were used: $k_{ATT} = 0.5$, $k_{SDF} = 0.3$, $k_{KDF} = 0.5$, $Q_1 = 0.5\ m$, $Q_2 = 2.0\ m$ and $\bar{v} = 0.5$.

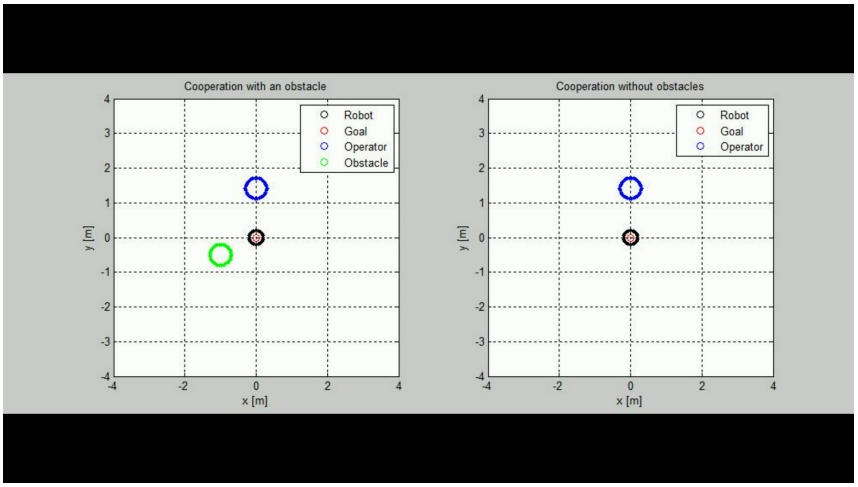


Figure 3.9: A snapshot from the simulation.

The video, then, shows intensities of the SDF, KDF and the overall DF evolving, in order to show how the intensities of the two components of the Danger Field and the overall DF change according to what's happening to the robot. The U_{SDF} diagram shows that the static component for the Danger Field changes with the distance of the robot from the obstacle. As the blue circle approaches the robot the U_{SDF} increases, but the robot stands still because the distance threshold for activating the Danger Field is set to a lower value. When the moving obstacle increases its speed, whose effect can be seen in the U_{KDF} diagram, the robot reacts by escaping from it avoiding the obstacle to get too close. The U_{DF} diagram shows how both the contributions are summed up.

Experiments have been performed in a workshop-like environment to make them as much realistic as possible and they are shown in the second part of the video. First it is shown the operator slowly approaching to the robot that stands still on its position letting the human user to get near. Then the operator runs towards the robot that steps aside. The video, then shows the operator walking around the robot that continuously moves in order to avoid collision with the human user. It can also be seen that while the user reduces his speed, the robot reacts slower, until the operator can touch the robot. The third experiment on the danger field's performance shows the robot avoiding both an approaching user and a fixed obstacle (a wheel); even if the operator tries to push the robot toward the wheel, the robot reacts by changing direction. In the last part of the danger field experiment, two users walk towards the robot. When the users approach the robot from opposite directions, TIREBOT avoid to collide both of them, changing also its direction, similarly to the case of the fixed obstacle.

During these experiments, the following parameters were set: $k_{ATT} = 0.5$, $k_{SDF} = 0.1$, $k_{KDF} = 0.6$, $Q_1 = 1.4\ m$, $Q_2 = 3.0\ m$ and $\bar{v} = 0.6$. The parameters setting must consider the dynamics of the robot (i.e. maximum acceleration

and speed) as well as its dimensions. In particular, for the Q_1 and Q_2 settings we referred as the center of the robot as the origin of its frame and considered also the dimension of the robot ($L \times W \times H = 1538 \times 944 \times 1666\ mm^3$).

3.5 Experiments

3.5.1 Tire Workshop Evaluation

The performance of the TIREBOT robot have been evaluated in a real tire workshop, where operators have cooperated with the robot for processing the wheels of a vehicle.

The test consisted in disassembling a wheel from a car, taking it to the wheel changer machine, and then taking it back to the vehicle where the wheel is then reassembled on the car. In order to evaluate the robot's performance with respect to the manually executed operation, each test consisted in an operator doing the sequence manually and then repeat it with the help of TIREBOT. Finally, the operators were asked to try also to maneuver the robot with the haptic interface.

Table 3.1 summarizes the performance evaluation. Columns' meaning is summarized as follows:

- **Test**: represents the experiment number;
- **Age**: age range of the operator who performed the test;
- **Experience in car servicing**: years the operator spent working in the car servicing field;
- **Required time**: indicates the time required by performing the task both manually and with the assistance of TIREBOT;
- **Effort reduction**: the effort reduction perceived by the operator while exploiting TIREBOT's help with respect to the manual operation;
- **Usability**: the evaluation of the usability of TIREBOT.

Some of the evaluated parameters, like robot's perceived usability and effort reduction, are neither objective nor countable. In particular, the perceived effort reduction required for accomplishing a particular task is subjective and it depends on many personal factors like health status, habit, musculature, age of the worker, etc. In order to evaluate these parameters, an assessment questionnaire was proposed to the operators who interacted with the robot and executed the tests.

Operators were asked to answer the following questions by giving a score in range $[1, 5]$:

- q_1: How much do you evaluate the difficulty in the use of TIREBOT? (1 = very easy, 5 = very difficult)

- q_2: How much has TIREBOT facilitated your job with respect to the current practice? (1 = for nothing, 5 = a lot)
- q_3: How comfortable do you evaluate TIREBOT's gesture interface? (1 = very uncomfortable, 5 = very comfortable)
- q_4: How comfortable do you evaluate TIREBOT's teleoperation and haptic interface? (1 = very uncomfortable, 5 = very comfortable)
- q_5: How difficult was to put the wheel on TIREBOT? (1 = very easy, 5 = very difficult)

We combined the results of the questionnaires for achieving two indicators: E, the perceived effort reduction with respect to the current manual practice, and U, the usability of TIREBOT. The indicators are defined as follows:

$$\begin{aligned} E &= \frac{[(5-q_1)+(q_2-1)+(5-q_5)]}{12} \\ U &= \frac{[(5-q_1)+(q_3-1)+(q_4-1)+(5-q_5)]}{16} \end{aligned} \tag{3.11}$$

Both E and U take value in $[0, 1]$; the bigger the value the better is the experience. The experimental data are reported in Tab. 3.1. Furthermore, the table reports the transportation time for taking the wheels from the vehicle to the machines and back, both manually and by using TIREBOT.

Test	Age	Experience in car servicing [*years*]	Time required [*s*]		Questions' score [1, 5]					Perceived Effort Reduction $E \in [0, 1]$	Perceived Usability $U \in [0, 1]$
			Manual	Assisted	q_1	q_2	q_3	q_4	q_5		
1	30-40	18	35	128	3	3	3	3	3	0.50	0.50
2	18-30	1	41	174	3	2	4	3	2	0.50	0.63
3	30-40	1.5	43	153	4	4	2	4	1	0.83	0.69
4	50+	45	43	132	3	4	2	1	4	0.50	0.25
5	40-50	22	24	139	3	3	4	3	2	0.58	0.63
6	30-40	21	25	117	2	3	4	3	2	0.50	0.56
Average			40.50	146.75	3.00	3.17	3.17	2.83	2.33	0.57	0.54

Table 3.1: Table reporting questionnaires' results and the performance evaluation of TIREBOT.

Several operators with different ages and experiences in the car servicing field have been involved in the experimental evaluation. Time required by TIREBOT to transport the wheel from the vehicle to the tire changer and on the way back to the vehicle is much bigger. Nevertheless, TIREBOT allows to pipeline the operations and while the robot is transporting the wheel the operator can start unscrewing the next wheel that is ready for transportation once TIREBOT is back. Furthermore, the transportation speed of TIREBOT can be easily improved by choosing a faster mobile base. In fact, the speed of the current mobile base is limited to 0.6 m/s, which is a very low value compared to the human velocity.

TIREBOT succeeded in significantly reducing the effort currently perceived by the operator. In fact, in average, the perceived effort is reduced of 57% and this means that the use of TIREBOT can lead to better working condition for the humans. The interface with TIREBOT has been evaluated as averagely usable and not very usable as we believed. Several operators pointed out that it is difficult to

use a gesture based interface and they would have preferred a vocal interface (which would not be so easy to implement since the tire workshop is quite noisy). Furthermore, only some operators (mostly videogamers) could easily use the teleoperation system that resulted uncomfortable to most older operators.

Thus, while TIREBOT has demonstrated its effectiveness in reducing the working effort, some further work needs to be done for improving the communication between the user and the robot in order to make TIREBOT usable by the average tire-workshop operator. From a mechatronic point of view, a faster mobile base needs to be adopted in order to augment the transportation speed.

3.5.2 Further experiments on TIREBOT

The realized robot prototype, TIREBOT, is not only bounded to work in a tire workshop. In fact, TIREBOT is a personal forklift that can easily help workers in different environments. Aware of this, we tested TIREBOT in an electric vehicles workshop of "Gruppo PRETTO s.r.l.", a small company located near Pisa, Italy, that performs maintenance on electrical vehicles. The robot was used to transport heavy (more than 35 kg) lead batteries used to power on electric garbage collectors vehicles. In order to make the robot capable of transporting batteries a steel shelf was mounted on the lower forks.

Once the user has switched on the robot, the experiment consisted in the following activities:

- The user gets recognized by the robot, which is standing still on its home position;
- The user orders the robot to go to the batteries depot;
- Once the robot has arrived, the user orders it to rotate in place, in order for him to be capable of loading the battery on the robot's loading platform;
- The robot rotates in place and the operator loads the battery on the robot's loading platform;
- The robot then, rotates again in place in order to frame the user, who orders him to take the battery near an under maintenance vehicle;
- Once the robot has reached the assigned goal, the user orders him to rotate in order to unload the battery from TIREBOT;
- Then, the user orders the robot to return to its home position and wait for other tasks.

This sequence of activities is also shown in the last part of the accompanying video.

The operators at Pretto enjoyed using TIREBOT and those who tested the robot found the gesture interface quite comfortable and easy to use. Furthermore, unlike the experiments in the tire workshop, the speed of TIREBOT has been deemed appropriate for the battery transportation task.

TIREBOT efficiently executed the battery transportation, by releasing the operator from such a tiring task and by allowing the personnel to work on more high-level tasks where their cognitive features are necessary. People who tested the teleoperation modality found it, in some cases, even easier to use than the gesture interface. Such an enthusiasm can be due to the young age of the operators that tested TIREBOT. This may now bias some experiments but it is a good hope for the future, when all the operators will have the same or even more enthusiasm for the introduction of robots in their working environment.

From a quantitative point of view, the perceived usability index has been evaluated at 53%, which is similar to the one obtained during the experiments in the tire workshop, and the perceived effort reduction has been evaluated at 63%. This means that the user interface of the TIREBOT system is evaluated as average also in the Pretto's setting and that, therefore, more work has to be done for improving it. On the other hand, the perceived effort reduction on the Pretto task is greater than the reduction perceived in the tire workshop. This is due to the fact that the task executed at Pretto consists mostly of transportation, which has been completely automated by TIREBOT, and only the lifting of the batteries was left.

3.6 Conclusions

In this Chapter a novel omnidirectional robot, capable of assisting operators in a tire workshop, was presented. This robot can work in many ways: autonomously, by recognizing the users and by receiving commands through his gestures, and with teleoperation through a haptic interface. Aspects of both mechanical and electronics design have been discussed in this manuscript, as well as the software architecture. The main novelty discussed in the work, is the aspect of the Human-Robot cooperation TIREBOT (a TIRE workshop roBOTic assistant) is capable of doing. The experimental evaluation has shown good results in reducing the effort perceived by the user during the wheel processing. The interaction between the operator and TIREBOT was evaluated averagely good.

Another key point discussed in the Chapter is the safe reactive collaborative navigation strategy for TIREBOT. The proposed method was tested both by simulation and by experiments in the real environment TIREBOT was designed for: the tire workshop. The presented Danger Field strategy proved to be very suitable for the cooperation of a mobile robot with a human co-worker that works close to the robot.

Acknowledgments

This work was done in collaboration with CORGHI, a company leader in the production of tire workshop tools and equipments (like tire changers and wheel balancers).

Fundings: This work was supported by the European Union in the scope of the European project ECHORD++(European Clearing House for Open Robotics Development Plus Plus) [grant numbers 601116].

Chapter 4

Agriculture Robotics

4.1 Grape yield estimation

4.1.1 Introduction

Precision agriculture [62] uses intensive data, information collection and processing to make more efficient use of farm inputs such as fertilisers, herbicides, seed, fuel by doing the right management practice at the right place and the right time. In particular, viticulture is successfully exploiting the latest technologies in order to improve the quality of the grape. In this context, a very important topic is accurate yield estimation since it leads to a better vineyard management and, consequently, to a better quality of the grapes and health of the vines [63], [64]. The knowledge of the vineyard yield conditions can be used to plan processes as fertigation and thinning or to help processors of juice and wine to anticipate the necessary tank space. Standard yield prediction techniques are labor intensive, expensive, spatially sparse and often destructive. Moreover, these methods require many years of data acquisition and experienced people for achieving a good estimation. Typical yield prediction techniques combine random samplings of the vineyard and knowledge of historical yields.
For these reasons, sensor-based systems have been proposed for accurate yield estimation. In [65] a system based on trellis tension monitors is presented to improve yield estimation, but it requires the installation of a permanent infrastructure. An approach based on multi-echo laser scanners is proposed in [66] while in [67] multispectral images are used. The most common systems that have been proposed are based on visible-light image processing. In [68], [69] color is used to detect grapes in the images, but this solution is not suitable for the white grape varieties or before the véraison, since the color of the berries is similar to the foliage. Grape shape and texture are exploited in [70] and [71], respectively. One of the most complete system has been proposed in [72]. In this work, color, shape and texture features, in combination with machine learning techniques, are used to detect the grapes. The provided results are collected over a large number of vines in multiple growing season, but artificial illumination and expensive hardware are used.
In order to collect the images to process, unmanned ground vehicles (UGVs) or

farmer-operated machinery can be equipped with standard RGB cameras. Autonomous path planning and navigation are very important challenges for moving a robot in open fields. Most of the systems proposed use a combination of GPS variant and other sensors. For instance, in [73] a real-time kinematic GPS (RTK-GPS) is used to achieve accurate guidance. This solution is quite expensive and unreliable in many kinds of vineyard, since it requires a simultaneous connection to multiple GPS satellites. Moreover, it requires a base station for proving correction data to the rover. In [74] a vision-based control system is proposed. An unsupervised classification technique and the Hough Transform are used to detect the central path between two rows. The weak point of vision-based approaches is that they are negatively affected by illumination conditions. In [75], a 3D laser scanner is used for row detecting and following in pergola structured orchards. 3D data are used to filter unwanted objects as weeds.
2D laser scanners represent a valid option for navigation. They are not affected by the RTK-GPS problems and they can be also exploited for obstacle avoidance and safety. In [76] an automatic guidance system for navigating between tree rows is presented. A laser scanner and the Hough Transform are used to detect the rows and control an agricultural tractor. A similar approach is proposed in [77]. The orchard rows are identified by an Extended Kalman Filter and the relative position between them and the vehicle is used to correct its pose.

In this Section, we aim to develop a low-cost autonomous system which is able to navigate through a vineyard while collecting pictures of the grapes in order to provide a yield estimation. The solution that we present uses the data provided by a laser scanner to identify the vines. Since vineyards have usually a regular and well defined structure, we exploit it to filter the points and reject false positive detection like tall grass or hanging twigs. An Extended Kalman Filter is used to detect the position of the row with respect to the robot. We use the distance of the robot from the row and their relative orientation to implement a pure pursuit controller, in order to follow the row at a distance defined a priori. During this part of navigation, the distance from the next vine in the row is computed. At pre-defined values of this distance, a picture is taken by a standard RGB camera mounted on the robot in a fixed position. When the laser scanner does not detect vines, the end of the row is reached and odometry is used to guide the robot to the next row, exploiting the knowledge of the vineyard structure. Starting from the pictures collected by the robot, we use a detection algorithm based on [78], [72] to detect the visible berries and provide a yield estimation. The proposed method was tested both in a simulated environment and in a real vineyard.

The main contributions of this work are the use of a low cost laser scanner to navigate through the vineyard in order to collect grape pictures and the use of this images for providing a yield estimation. The most similar work is [77], but the authors used a more expensive and performing laser scanner for the navigation and reflective landmarks at the end of each row and a pre-built map to localize the robot.

4.1.2 Problem statement

The goal of this work is to design a robotic system which is able to collect pictures of the grapes in a vineyard. The vineyards usually have a regular and defined structure, so it is possible to exploit it to navigate. It is possible to detect the rows formed by the vines and follow the the central path between them. The distances between two vines in the same row and between two consecutive rows may change from a vineyard to another one. Also, the vine training system that is used affects the grapes position and, consequently, the position of the camera has to be chosen wisely.
In our work, the structure of the vineyard and the training system are known (Fig. 4.1). In this training system, called Geneva Double Curtain (GDC), the grapes

Figure 4.1: The vineyard considered for our work.

grow below the foliage and they are not visible from the outside, therefore navigate in the central path between two rows with the camera pointed to the plants is not a suitable solution. For this reason, we chose to make the robot navigate closer to the rows, under the foliage, with the camera pointed to the other side. (Fig 4.2). While it navigates through the vineyard, the robot has to take enough pictures to see all the grapes on the vines but not too many, to avoid an excessive overlap. Starting from the photos, the number of visible berries is derived and a function is fitted between them and the harvest weight. This model can be used for yield estimation.

4.1.3 Navigation strategy

The main problem of collecting the pictures in the vineyard is to navigate through it in such a way the camera is pointed towards the grapes. As said in Sec. 4.1.2, the grapes are not visible from the outside of the foliage, due to the specific training system, so the robot has to navigate close to the rows formed by the vines. Since

Figure 4.2: Robot and camera displacement to collect the pictures.

the structure of the vineyard is regular and well defined, it is possible to exploit it for navigation purposes.
Consider a vineyard in which the vines form a grid-like structure. The distances between two vines in the same row d_{vine} and between two rows d_{rows} are known. The main idea of the navigation algorithm is to detect the row on the right side of the robot and to follow it at the target distance, until the end of the row is reached. At this point, the robot turns around to follow the other side of the same row or the next one, alternatively. While following the row, the robot takes the pictures of the grapes, based on its position with respect to the nearest vine.

4.1.3.1 Navigation modes

The navigation algorithm has different operating states, each of which defines the behaviour of the robot (Fig. 4.3). At the beginning of the row, the robot is in the

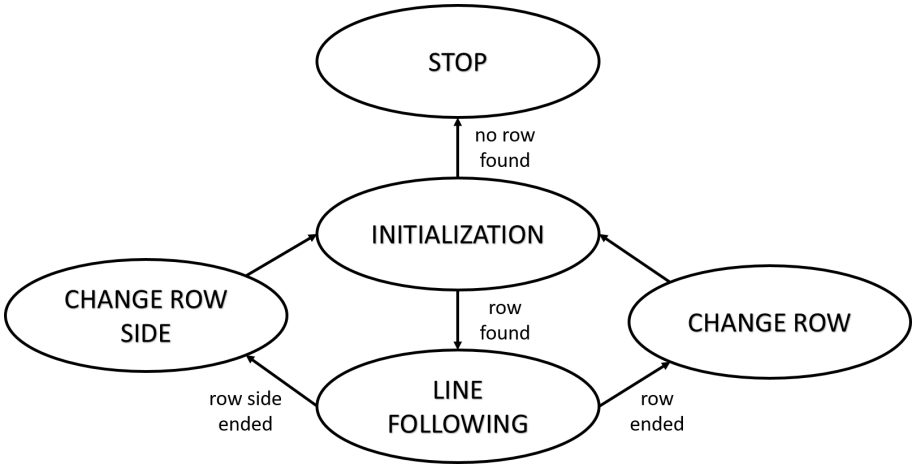


Figure 4.3: Operating states of the navigation algorithm.

initialization state. In this state, the robot identifies the row on its right side. This information is used in the *line following* state to initialize an Extended Kalman Filter (EKF). The prediction step uses the linear and the angular velocity provided by the odometry, while the update step exploits the data provided by the laser scanner.
Let d be the distance between the robot and the row and φ_{err} be the relative angle between them. A pure pursuit controller is used to keep the robot at the target distance d_{targ} from the row. The linear velocity is set to a constant value $v = v^*$, low enough to make focused pictures, and the angular velocity to $\omega = K_d d_{err} + K_\varphi \varphi_{err}$, where K_d and K_φ are positive gains, while $d_{err} = d_{targ} - d$. When there are no vines detected by the laser scanner, it means that the robot reached the end of the row and the algorithm switches to the states *change row side* or *change row* to turn back. In these states, the odometry is used to move the robot to the other side of the row and to change the row, respectively, and the *initialization* state starts again. It is possible to use odometry since the path to follow to change the row (or the row side) is fairly short, so the measure is still reliable. If there is no row to detect for the initialization, the navigation ends, since the end of the vineyard is reached. The algorithm is summarized in Alg. 1.

Algoritmo 1 Navigation Algorithm

```
1:  while forever do
2:    if no row is detected then
3:      Stop
4:    else
5:      Initialize the position of the row
6:    end if
7:    while the end of the row is not reached do
8:      Get v(k), ω(k)
9:      Update the position of the row
10:     compute d_err and φ_err
11:     Set v(k+1) = v*
12:     Set ω(k+1) = K_d d_err + K_φ φ_err
13:   end while
14:   Turn back
15: end while
```

During the *line following* state, the robot takes the pictures of the grapes. While moving, it computes the distance from the next vine in the row. The photos are taken corresponding to values of the distance defined a priori.

4.1.3.2 Line following

The main problem of the navigation strategy is the detection of the line to follow while the robot is moving. Let's consider the following unicycle kinematic model:

$$\begin{cases} x(k+1) = x(k) + v(k)Tcos(\theta(k) + \frac{\omega(k)T}{2}) \\ y(k+1) = y(k) + v(k)Tsin(\theta(k) + \frac{\omega(k)T}{2}) \\ \theta(k+1) = \theta(k) + \omega(k)T \end{cases} \tag{4.1}$$

where $q(k) = [x(k), y(k), \theta(k)]^T$ is the configuration of the robot at time k, T is the period and $v(k), \omega(k)$ are the input velocities. Since all the computations will be done in the local frame, it is convenient to reformlate Eq. 4.1 with respect to the robot frame at time k:

$$\begin{cases} x(k+1) = v(k)Tcos(\frac{\omega(k)T}{2}) \\ y(k+1) = v(k)Tsin(\frac{\omega(k)T}{2}) \\ \theta(k+1) = \omega(k)T \end{cases} \tag{4.2}$$

Using Eq. 4.2 it is possible to compute the homogeneous transformation matrix T_k^{k+1} that joins two consecutive poses of the robot.

Let $r(k) = [r_1(k), r_2(k), r_3(k)]^T$ be the line expressed with respect to the robot frame at time k, $P(k) = [x_P(k), y_P(k), 1]^T$ be the homogeneous coordinates of a point on the line. We have that at $k+1$, the same point can be expressed as:

$$P(k+1) = T_k^{k+1} P(k) \tag{4.3}$$

Considering the line equation with respect to the robot frames at time k and $k+1$, the folloing equation holds:

$$r(k)^T P(k) = r(k+1)^T P(k+1) \tag{4.4}$$

From Eq. 4.3 and 4.4:

$$r(k+1) = \underbrace{(T_k^{k+1})^{-T}}_{T_r} r(k) \tag{4.5}$$

The process model is given by:

$$\begin{cases} x(k+1) = v(k)Tcos(\frac{\omega(k)T}{2}) + w_x(k+1) \\ y(k+1) = v(k)Tsin(\frac{\omega(k)T}{2}) + w_y(k+1) \\ \theta(k+1) = \omega(k)T + w_\theta(k+1) \\ r(k+1) = T_r r(k) + w_r(k+1) \end{cases} \tag{4.6}$$

where $[w_x(k), w_y(k), w_\theta(k), w_r(k)]^T$ is a gaussian random vector that model the uncertainty of the model.
The observation model is given by:

$$y_m(k+1) = r(k+1) + v_{y_m}(k+1) \tag{4.7}$$

where v_{y_m} is a gaussian random vector that model the uncertainty of the measure. The measured line is found starting from the points provided by the laser scanner as described in the following. The vines are detected by grouping the points that are close together and the mean position of each group is taken as the vine position. Depending on the operating state of the robot, only some vines are considered:

- *initialization* state - only the vines on the right side of the robot within a distance equal to d_{row} are considered;
- *line following* state - only the vines close to the predicted position of the row (in the prediction step of the EKF) are considered.

In this way the number of vines to take in account is considerably reduced.
Although the laser scanner is placed high enough to avoid the weeds, it could still detect unwanted objects like hanging twigs or tall grass. Also, it could happen that the laser scanner is tilted due to the rough terrain. In this case, it also detects the weeds and the ground. Using all the detected points to fit the line can lead the robot to follow a wrong path during the navigation. For this reason, the following filtering algorithm is used. The first step consists of find all the segments that connect two points at a distance equal to d_{vine}. The points that cannot be connected are rejected. The segments that have an orientation different from the predicted line are discarded. The remaining ones are connected by the shared points to form longer segments. The longest one is chosen as the line to fit (Fig. 4.4).

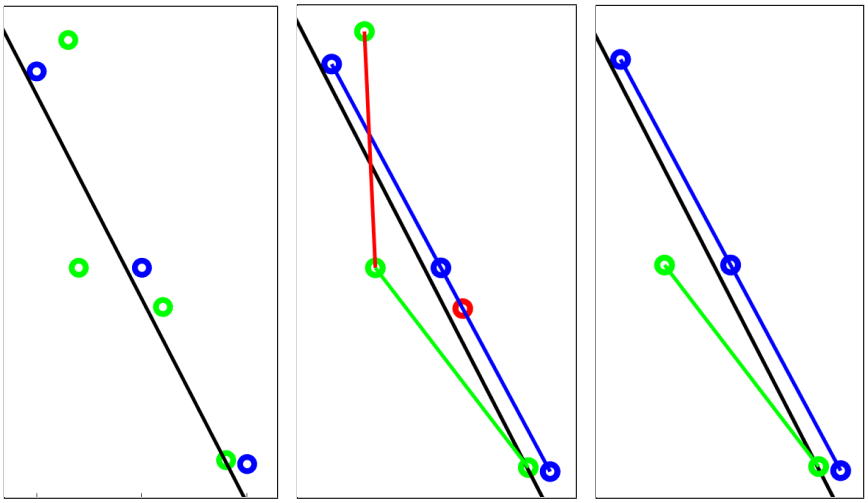


Figure 4.4: Example of the filtering algorithm: (a) A possible scenario detected by the laser. The blue points are the actual vines, the green ones are unwanted objects and the black line is the predicted position of the row. (b) The red point is rejected since it is not connected to another one. The red segment is discarded since its orientation is different from the predicted line. (c) The blue segment is the longest one, so it is chosen as the line.

In the unfortunate case in which there is not a unique longest line, the algorithm discards all the segments and the robot follows the predicted line.

4.1.4 Experimental results

4.1.4.1 Simulations

In order to prove the efficiency and robustness of the proposed algorithm, some simulations have been made, before experimenting in the real environment. Simulations have been made using ROS Indigo and Gazebo.
The virtual environment that was created is shown in Fig. 4.5. It consists of a plane ground and a set of cylinders representing the vines, to form the same grid-like structure of the actual vineyard. The starting position of the robot is at the beginning of the leftmost row. The position of each cylinder has a random bounded offset in

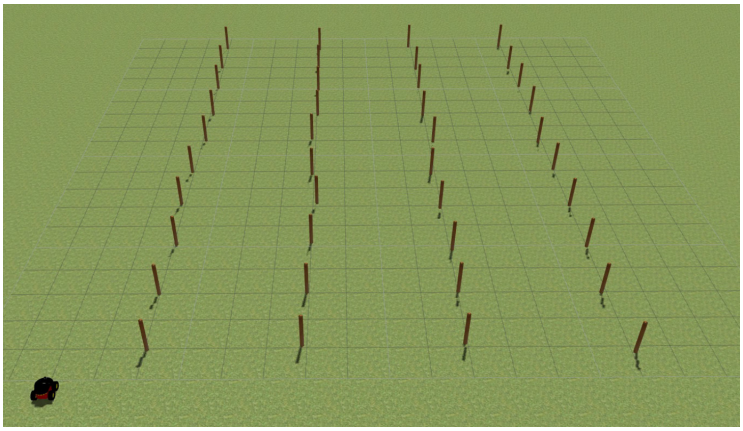
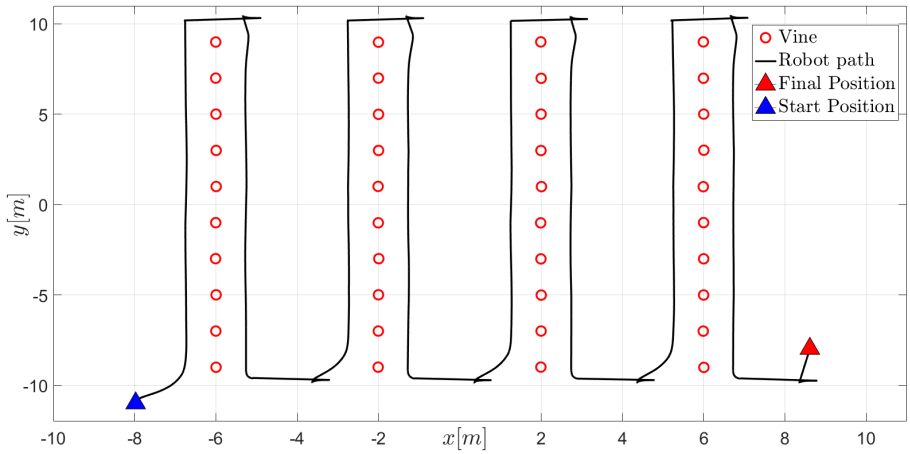


Figure 4.5: The virtual environment created using Gazebo.

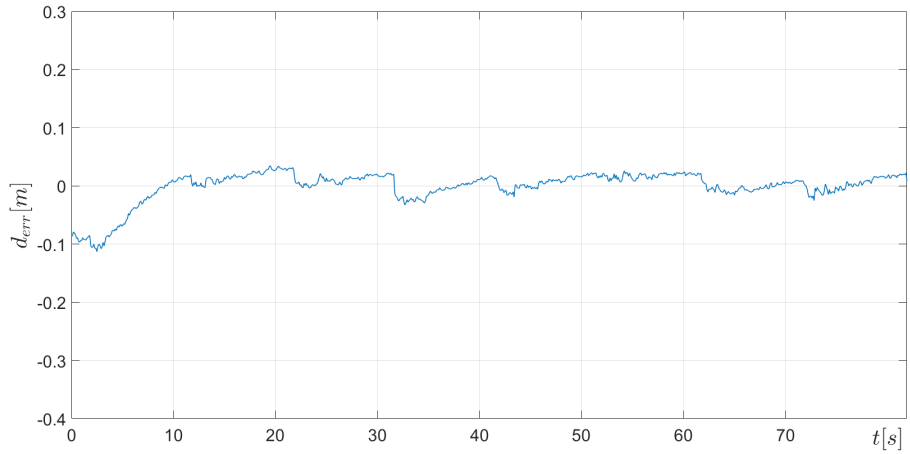
order to test the robustness of the algorithm with respect to the vines displacement. The value of the offset is conveniently chosen in order to be compared to the real environment.
Fig. 4.6 shows the robot path and the distance error d_{err} during the *line following* state. It is possible to see that the robot navigates through the virtual vineyard passing at the target distance from both sides of each row. The position offset of the vines does not affect the navigation algorithm.

4.1.4.2 Experiments

Experiments in a real vineyard have been made in order to test the robustness of the navigation algorithm with respect to the disturbances of the real environment, such as tall grass, hanging twigs and rough terrain. The robot used in the experiments is a Pioneer 3-AT equipped with a SICK S300 Expert laser scanner, a GoPro Hero 4 and a laptop (Fig. 4.2). The SICK laser scanner is not low cost, so its performances were limited by software to mimic less expensive devices. The tests evaluate all the steps of the navigation algorithm. The distance error d_{err} during the *line following*



(a)



(b)

Figure 4.6: Simulation results: (a) Robot navigation path in the virtual environment (b) Distance error during the *line following* state.

state is shown in Fig. 4.7. The mean value of the error is less the 0.05 m and its value is within 0.1 m more than 93% of the time. It is possible to see that the error has a few spikes with high values. They are due to wrong estimations of the row position that, nevertheless, are recovered in the next steps.

4.1.5 Yield estimation using collected data

As shown in Sec. 4.1.3, while the robot is navigating through the vineyard, it takes pictures of the grapes (Fig. 4.8). The photos are taken with a GoPro Hero 4 at 3000x2250 resolution that is mounted on the robot as shown in Fig. 4.2. The dataset consists of two varieties of grape (Ancellotta and Salamino), each one of eighteen plants. For each plant there are four photos, with a certain overlap. The number of visible berries is derived from the photos by a detection algorithm based on [78], [72]. It consists of four steps:

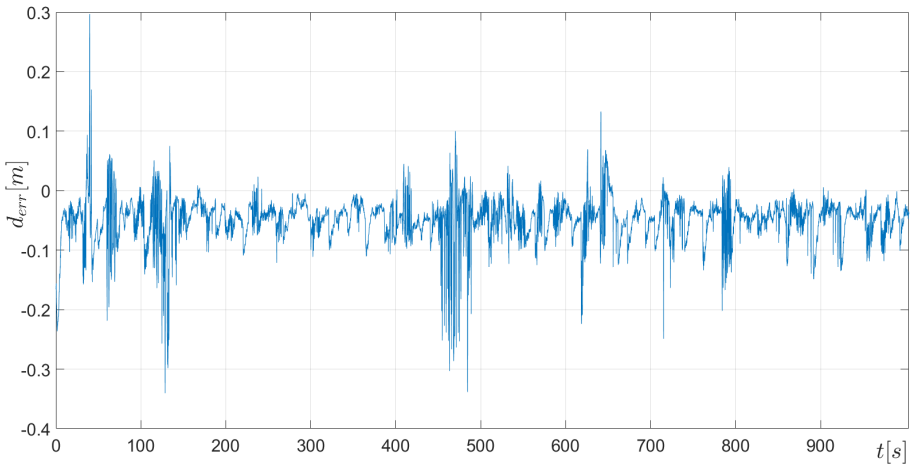


Figure 4.7: Distance error during the navigation.

Figure 4.8: Example image of Ancellotta grape variety.

1. detect potential berries by shape;
2. classify detected berries by texture features;
3. remove isolated berries;
4. group berries into clusters.

In the first step, the radial symmetry transform [79] is used to find points with high level of radial symmetry. They are potential centers for grape berries. For each detected point, a set of features (RGB channels, L*a*b color channels and SURF [80]) is extracted from the part of the image around the point. In order to classify the

points as grape or not grape, a *random forest classifier* [81] was constructed from a subset of the collected images. Since the grapes occur in clusters, the berries without a certain number of other berries in their neighborhood are removed, while the remaining ones are grouped into clusters based on their relative distance (Fig. 4.9).

(a) (b)

Figure 4.9: Example of the detection algorithm output: (a) Detected berries (b) Group near berries into clusters.

It is possible to see that the algorithm can not precisely identify the clusters if they are overlapped in the image, so only the number of detected berries is used for the estimation.
For each variety of grape, a linear function is fitted between the number of detected berries and the harvest weight. The points with the highest residual values are considered as outliers and they are not used for the fitting. Fig. 4.10 and Fig. 4.11 show the fitted functions for the two varieties.

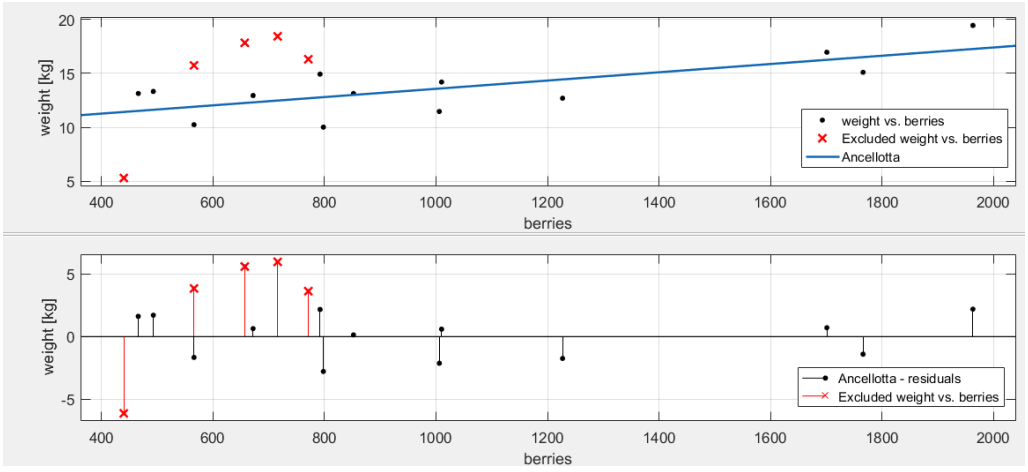


Figure 4.10: Fitted function for Ancellotta variety.

The coefficient of determination for Ancellotta variety is $R^2 = 0.55$ with a root-

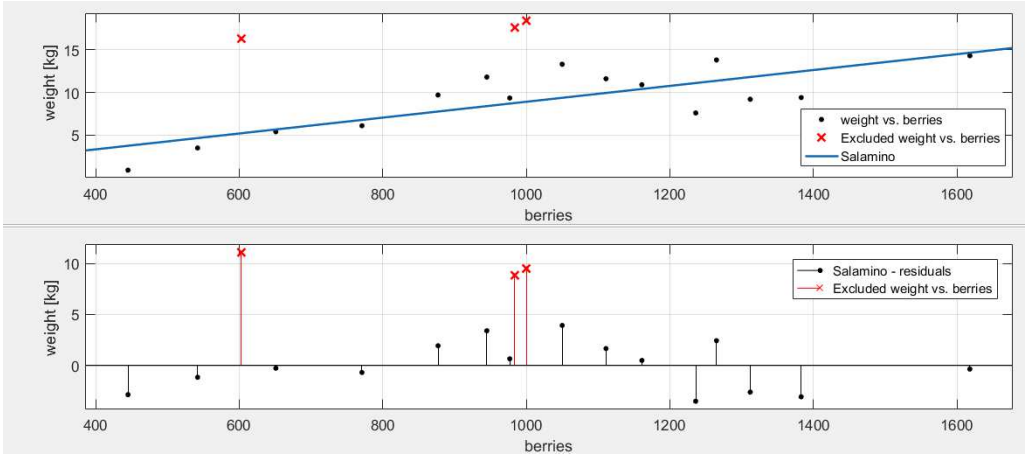


Figure 4.11: Fitted function for Salamino variety.

mean-squared error $RMSE = 1.81kg$, while for Salamino variety these parameters are $R^2 = 0.62$ and $RMSE = 2.46kg$. The values of R^2 of both the fittings are lower with respect to the ones found in similar works [78], [72] (≥ 0.74). The main reasons are:

- in this work, external illumination sources are not used, so the natural light affects the quality of the photos;
- the position of the camera on the robot is fixed and in some cases it may not capture all the grapes;
- grape occlusion is not considered.

Nevertheless, it is possible to see that the estimation results are promising, compared to the real weight of the grapes for each vine (Fig. 4.12 and Fig. 4.13).

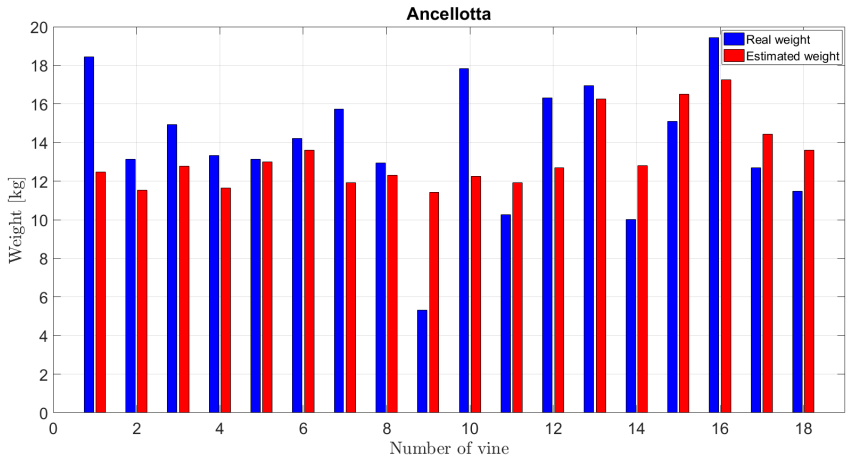


Figure 4.12: Comparison between real and estimated weight for Ancellotta variety.

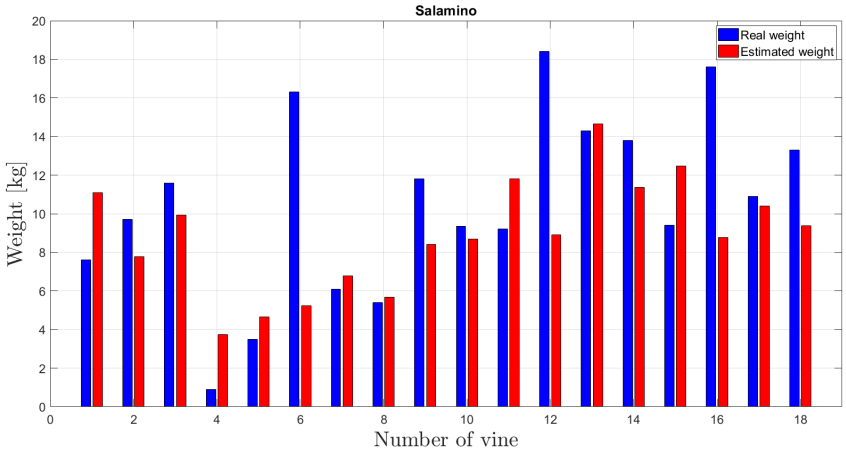


Figure 4.13: Comparison between real and estimated weight for Salamino variety.

4.1.6 Conclusions

In this Section we have proposed a low-cost autonomous system which is able to navigate through a vineyard while collecting pictures of the grapes in order to provide a yield estimation. Exploiting the knowledge of the vineyard and a low cost laser scanner, the robot can navigate following each row at a distance defined a priori, without using other sensors or artificial landmarks. During the navigation, the system takes pictures of the grapes by using a standard RGB camera. Also, the collected photos are used for providing a yield estimation.

The proposed system was successfully tested both in a simulated environment and in a real vineyard, in order to evaluate its robustness to disturbances such as tall grass, hanging twigs and rough terrain.

4.2 Apple picking

4.2.1 Introduction

In the last decade, the world has seen a major increase in apple production due to growing demand. In order to meet growing demand, it is necessary for the industry to innovate its process of producing apples.

One of the most labour intensive and resource expensive tasks in the apple production chain is harvesting [82]. To this day apples are primarily hand-harvested and carried to a storage container. A major consequence of hand-harvesting is that it can lead to a variety of injuries and illnesses [83], mainly due to repetitive operations such as excessive reaching, lifting and carrying heavy loads; often requiring to be done so in awkward postures. Another issue faced by apple farmers is the limited availability of workers, leading to unpicked fruit and hence underselling [84, 85, 86].

A highly beneficial solution to these challenges is to utilise robotic harvesters. Harvesting robots have been well studied and prototypes have been tested since the 1980s. There exists many examples of robots capable of harvesting crops such as cherry tomatoes [87], strawberries [88] and sweet peppers [89].

An apple harvesting robot is presented in [90] which yielded a harvest success rate of approximately 80%. The detection algorithm used colour for segmenting apples from leaves and the background. To minimize the effect of light and weather conditions, a canopy and all around curtain is used. They used a silicon gripper that assumes the shape of the apple when activated. In contrast, authors in [91] used HSI colour space to handle varying light conditions and extracted shape features from images to locate fruit. Image Based Visual Servoing (IBVS) is used for grasping with a five-DOF manipulator. The gripper used was a spoon-shaped gripper equipped with several sensors, including a vision sensor and a pressure sensor, and they achieved a harvesting success rate of 77%. A majority of the reported failures were due to occlusions.

More recently, authors in [92] use a Circular Hough Transformation and Blob Analysis to locate the apples and estimate their spatial position by a 3D camera. To aid with the image segmentation, a black curtain was used as a uniform background during the experiments. The manipulator was seven-DOF equipped with an under actuated gripper. The system successfully harvested 127 out of 150 apples with an overall success rate of 84% and average picking time of $6.0s$ per fruit. These are promising results, however, maintenance of the crop was required to remove occlusions and clustered apples.

A review of fifty harvesting robots [93] showed that only 6% of the authors reported task planning performance. Effectively sequencing tasks can have a significant influence on the overall execution time. Those who did consider task sequencing formulated the problem as a travelling salesman problem (TSP) and used euclidean workspace distance or height of the targets as cost metrics [89, 92, 94]. This can be a poor estimate of the actual trajectory costs since a small distance in euclidean workspace may not necessarily correspond to a small trajectory cost due to the non-linearity of manipulators and factors such as obstacles.

This issue is evidenced by the results in [95] where a grape vine pruning robot suffered its main performance bottleneck due to high execution times. While an accurate heuristic can be computed, obtaining this heuristic is non-trivial and often equivalent to solving the original problem [96].

In this Section we propose a an apple harvesting robot with a perception system that uses a Dirichlet mixture of Gaussian Process Implicit Surfaces (GPIS) for probabilistic segmentation of the scene into distinct pieces of fruit and non-fruit components [97, 98]. This method does not rely on colour, rather the segmentation algorithm operates on depth data and hence mitigates several issues that colour-based segmentation algorithms have in varying light conditions.

We use two different planners for approaching the located apples. The *Fast Reliable and Efficient Database Search Motion Planner* (FREDS-MP) [99] is used for sequencing the goals and for generating a collision free trajectory which leads the robot in front of the apple to grab. Visual Servoing [100] is then used to approach the detected apple. During the motion, the manipulator is kept far from the joints limit by exploiting its redundancy. The need of two planners is justified because it is known [101] that visual servoing techniques come up against difficulties when the initial and desired robot positions are distant so we use FREDS-MP to overcome these difficulties.

An ad-hoc gripper prototype was designed to grab the apples without bruising or surface damages. The proposed system was tested in a realistic environment consisting of an artificial apple trellis with the same characteristics as typical orchard trellises. The experiments showed that the system was capable to harvest the 88.75% of the apples.

The main contributions of this work are:

- The development of an end-to-end system capable of harvesting apples among clutter efficiently and reliably. Using depth data the detection algorithm is more robust in varying lighting conditions. Moreover, the capability of resolving clusters, occlusions and partially observed apples by exploiting the 3D geometry from multiple views alleviates the dependence on maintenance of the crop.

- A planning method that aids with the control regime, yielding higher success rates and lower computation and execution times than a baseline method.

4.2.2 Problem statement

The goal of this work is to design a robotic system that is able to harvest apples in an efficient and reliable way without damaging the fruit. The high-level functional requirements can be summarised as follows:

- Detect and localise apples, regardless of occlusions and clusters.

- Pick the apples in an efficient sequence.

- Grasp, detach and release the detected apples without damaging them or the trellis.

We consider a modern vertical trellis architecture as a working environment for the robot. As shown in Fig. 4.14, in this training system the apples are aligned approximately along a plane and hence more easily accessible compared to traditional tree canopies.

Figure 4.14: The considered apple training system (Photo: Apple and Pear Australia Ltd. https://apal.org.au/)

A single robot operates in a 3D Euclidean workspace, $W = \mathbb{R}^3$. The robot's end effector can be commanded to achieve any arbitrary 6D pose $T \in SE(3)$ given that a valid inverse kinematic (IK) solution exists and at least one of these solutions is collision free.

Obstacles in the workspace, O_{env} such as the robot and ground are known in advanced. The apple trellis is modelled as a 3D bounded volume $W_{trellis} \subset W$. It is treated as an obstacle, hence the obstacle region is $O = W_{trellis} \cup O_{env}$.

Let us define the configuration space, C, to be the set of all possible configurations of the robot. The robot is defined as a rigid body, $A \subset \mathbb{R}^3$, with a fixed base and robot arm configuration $q \in C$. The obstacle region is then: $C_{obs} = \{q \in C \mid A(q) \cap O \neq \emptyset\}$, where $A(q) \subset W$ is the space occupied by the robot and the sensor at configuration q. The free space region is then: $C_{free} = C \setminus C_{obs}$.

4.2.3 System overview

4.2.3.1 Hardware

The platform consists of a combination of off-the-shelf and custom made hardware. The manipulator is a seven-DOF Rethink Robotics Sawyer mounted on a custom made static base with adjustable height. A redundant DOF is advantageous for apple picking from a task and motion planning perspective and has shown higher success rates and lower execution and computation times [99]. Additionally, the Sawyer provides an easy to use trajectory execution interface and built in ROS support allowing for easy integration with the rest of the system.

The perception sensor used for this prototype is a Realsense SR300 which uses structured light to provide RGBD data. Any 3D sensor could be used however the SR300 was chosen out of convenience given its low cost, weight, size and power requirement. A drawback of the sensor is that it is unable to produce depth data outdoors when exposed to ambient sunlight. However, for proof of concept the experiments were carried out indoor.

The gripper used is an in-house custom made design that uses a single screw actuator to open and close its jaws around the apple. The actuation consists of two stages, first a linear stroke to push any obstructions away from the centre of the gripper and then finally a closing motion to secure the apple. The design intends that the jaws act as human fingers, encompassing the apple regardless of its position and angle of approach. Furthermore, contact between the apple and the gripper is reduced, hence mitigating the risk of bruising the apple.

4.2.3.2 Software

The harvesting software system can be broken into three sub-systems:

- The perception sub-system, which detects and localises the apples on the trellis
- The motion planner sub-system, which sequences the apples and moves the arm into a pre-approach configuration(see Sec. 4.2.4)
- The grasping sub-system, which moves the arm into a position to grasp, detach and release each apple (see Sec. 4.2.5)

The subsystem execution order is visualised in Fig. 4.15. Initially the manipulator actively perceives the environment for apples using an algorithm described in [102]. Once the apples are reconstructed, they are segmented and localised with the method described in [97]. The segmentation algorithm is based on a GPIS method which uses spherical priors to robustly resolve partially reconstructed apples in 3D.

Given the apple positions outputted by the perception system, a Kalman Filter is instantiated. The initial estimate of the apple locations is passed as input to FREDS-MP and a sequence of apples is determined. FREDS-MP attempts to plan to a position offset from the apple position, if it fails then the apple is removed from the list and the next apple is planned for. When a plan is successful a visual servoing routine is used to guide the gripper to the apple and grasp it. During

this routine, the GPIS algorithm continuously runs and the Kalman Filter updates the position estimates. If unsuccessful, a recovery procedure is initiated where the approach trajectory is played backwards to place the arm back in the state it was in before the servoing began. Alternatively, if the robot successfully reaches the apple it grasps it and then moves to a drop position. In this work the drop position is the same as the position that was planned to by FREDS-MP. The robot then simply drops the apple in place where a hypothetical catching device would secure the apple, such as a suspended net.

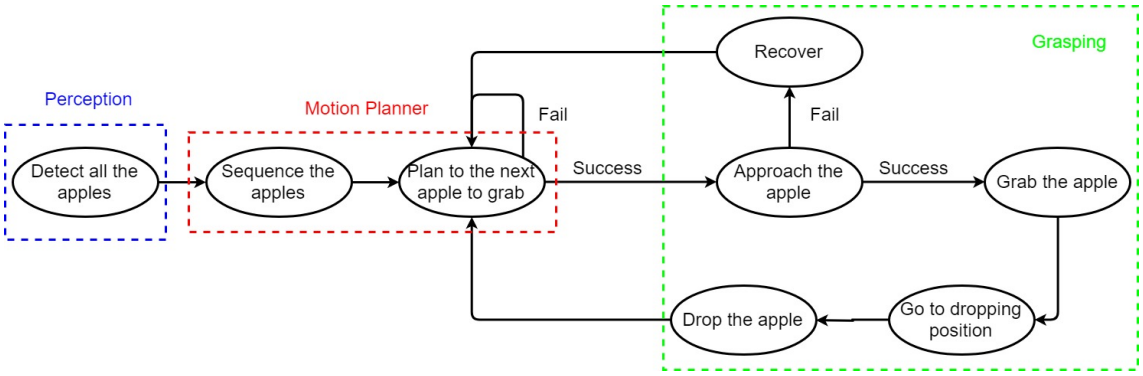


Figure 4.15: An illustration of the state machine of the harvesting algorithm.

4.2.4 Arm control and motion planning

Apple sequencing, plan generation and online execution prior to the grasping routine is computed using the FREDS-MP framework [99]. Planning consists of three phases, offline, task and online. The offline planner computes a database of trajectories which are then leveraged by the task planner to compute an efficient sequence of trajectories. The online planner adapts the offline trajectories and processes them for execution on the robot arm.

4.2.4.1 Offline Planner

In the offline phase, the environment is anticipated and modelled as a bounded union of basic shapes. This ensures trajectories are guaranteed to be collision free, however not time-continuous, in the real scenario. This is a reasonable assumption for the intended application given that the apple orchard's row spacing and height is known in advanced.

An offline planner discretises a sub-volume of the workspace into a graph of nodes, $v \in G$, where each node represents a 6D pose, and computes optimistic trajectories between every pair of nodes. Importantly, the assumption is that the manipulator will be operating repeatedly within or near this sub-volume. Given n goal poses, the worst case number of possible combinations of trajectories is $O(x^n)$, where x is the cardinality of the largest set of inverse kinematic (IK) solutions for a single pose.

To address this computational complexity, a single optimal IK solution $q^* \in Q(v)$ is assigned to each node $v \in G$. Using Dijkstra, the shortest path from a source node to all other nodes is computed. As each node v is expanded a single IK solution that minimises the Chebyshev, or maximum-coordinate-difference, distance L_∞ between v and its neighbour u is assigned to that neighbour, given the constrain that L_∞ is less than ϵ. This constraint encourages two adjacent configurations to be close in configuration space which assists with adapting the trajectories during the online phase.

If the constraint is violated no edge is created between v and u, i.e. cost $c(v, u) = \infty$. If the constraint is satisfied, $c(v, u)$ is assigned as the Euclidean distance, L_2 in configuration space between the neighbouring configurations, plus a hyperbolically increasing penalty term based on the configuration's distance from the joint limits of the manipulator:

$$c(v,u) = \|q(v) - q(u)\| + \frac{1}{n}\sum_{j=1:n} \tanh\left(2\left|\frac{q(u)_j - q_{min_j}}{q_{max_j} - q_{min_j}} - \frac{1}{2}\right|\right). \tag{4.8}$$

This cost provides a measure of how close in configuration space all the IK solutions are for a given graph. Additionally, it penalises configurations that are near the joint limits in order to mitigate low manipulability and joint limit failures when approaching the apple.

This process is repeated for all IK solutions for every possible starting node. If a node is unreachable from the source node of the graph it is considered to be disconnected. It is possible for this to happen due to the L_∞ constraint. The graph that minimises the total number of disconnected nodes is chosen. Given a path cost $\pi(v_0, u)$ from a source node, v_0 to another node $u \in G \mid u \neq v_0$, the optimisation problem can be written as:

$$\min |\{u \in G | \pi(v_0, u) = \infty \text{ and } u \neq v_0\}|. \tag{4.9}$$

To break ties where two or more sets of unique solutions yields the same number of disconnected nodes, the solution that minimises the sum of path costs over the graph is chosen:

$$\min \sum_{\forall u \in G \neq v_o} \pi(v_0, u) | \pi(v_0, u) \neq \infty. \tag{4.10}$$

When a minimum graph is found, the computed optimal configurations for that graph are assigned permanently and Dijkstra is run with every possible node as the source node. The result is an optimistic pre-computation of all possible paths between pairs of nodes. Paths to and from disconnected nodes are computed using an ensemble of motion planners [103, 104, 105]. These paths and their costs are stored into a database where they can be later queried by a pair of nodes.

4.2.4.2 Task Planner

The task planner takes as input a set of poses, $\hat{t}_{online}$ in continuous space that need to be reached by the arm prior to the approach state. Leveraging the pre-computed

database from the offline phase, the k closest poses in Euclidean workspace to an input pose $t \in \hat{t}_{online}$ are retrieved. All the IK solutions $Q(t)$ for the input pose are computed and the solution $q \in Q(t)$ with the lowest Euclidean distance in configuration space to any of the k closest poses' database configurations is kept, denoted as $q_{min} \in Q(t)$. The corresponding node from the database is also kept, denoted as $v^t_{min} \in G$. This is repeated for every pose $t \in \hat{t}_{online}$.

Then for every pair of input poses $(t_i, t_j) \in \hat{t}_{online}$ the corresponding path and cost is retrieved from the pre-computed database using $(v^{t_i}_{min}, v^{t_j}_{min})$. The Euclidean difference between the solutions $q_{min} \in Q(t)$ and the end points of the pre-computed path are added to the path costs. Then a weighted adjacency matrix is constructed using these modified costs. Since each pose has a single cost, the sequencing problem can be formulated as a TSP and any TSP solver can be used. Lastly, each pose's corresponding q_{min} is appended to the retrieved path.

4.2.4.3 Online Planner

For execution on the robot arm the modified database trajectory priors are adapted online via an ensemble of trajectory optimisers [103, 104]. This is necessary since the database trajectories are not guaranteed to be time-continuous safe. Rather, the trajectories act as effective priors since they are initially collision free and can be used as initial seeds for the trajectory optimisers, resulting in faster convergence and can help mitigate local minima issues. In the case that these optimisers fail, a global planner BIT* [105] is called as a last resort.

Finally, the online planner time parameterises the geometric trajectories produced by the motion planners and up-samples them. The Sawyer provides an easy to use velocity control interface which takes as input a list of time stamped waypoints. The internal controller then determines appropriate joint velocity commands via interpolation and sends them the arm so that the given trajectory is followed.

4.2.5 Apple grasping

4.2.5.1 Apple tracking

Let $p_G \in \mathbb{R}^3$ be the position of the centre of the gripper and $p_A \in \mathbb{R}^3$ the 3D position of the apple to grab given by the perception system. Let n_A be the normal vector to the trellis plane and n the orientation of the gripper. The velocity of the end effector to make the gripper reach the apple is given by:

$$v = k_v(p_A - p_G) \tag{4.11}$$

$$\omega = -k_\omega(n_A \times n)\cos^{-1}(n_A \cdot n) \tag{4.12}$$

where $k_v, k_\omega \in \mathbb{R}$ are positive gains, $\cdot$ is the dot product and $\times$ is the cross product. Given the jacobian J of the robot, it is possible to compute the joint velocities $\dot{q}$ of the robot such that the gripper velocity is $(v^T\ \omega^T)^T$ [106]:

$$\dot{q} = J^+ \begin{bmatrix} v \\ \omega \end{bmatrix} + (I - J^+ J)\dot{q}_0 \tag{4.13}$$

where $J^+ = J^T(JJ^T)^{-1}$ is the pseudo-inverse of J, I is the identity matrix and $\dot{q}_0$ is a vector of arbitrary joint velocities. The second term in Eq. (4.13) exploits the redundancy to generate internal motions that change the robot configuration without changing the end-effector pose. A common choice is

$$\dot{q}_0 = k_0 \left(\frac{\partial U(q)}{\partial q} \right)^T \tag{4.14}$$

where $U(q)$ is a function of the joint variables to locally maximize. This function is typically used for avoiding obstacles or for increasing manipulability during the motion to avoid singularities. In this work, we exploit the redundancy for keeping the joints away from the joint limits.

For this purpose, we designed the following repulsive potential functions for each joint i:

$$U_{u_i}(q) = \begin{cases} \frac{1}{2} k_{r_i} \left(\frac{1}{d_{u_i}} - \frac{1}{Q_{u_i}} \right)^2 & \text{if } d_{u_i} \leq Q_{u_i} \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

$$U_{l_i}(q) = \begin{cases} \frac{1}{2} k_{r_i} \left(\frac{1}{d_{l_i}} - \frac{1}{Q_{l_i}} \right)^2 & \text{if } d_{l_i} \leq Q_{l_i} \\ 0 & \text{otherwise} \end{cases} \tag{4.16}$$

where $k_{r_i} \in \mathbb{R}$ is a positive gain, d_{u_i}, d_{l_i} are the distances of the joint position from, respectively, the upper and the lower joint limit and $Q_{u_i}, Q_{l_i} \in \mathbb{R}$ are positive thresholds used to limit the repulsive effect in the neighbourhood of the joint limits. The global repulsive function is $U_{rep}(q) = [U_{u_1} + U_{l_1}, \ldots, U_{u_n} + U_{l_n}]^T$. It is possible to move away from joint limits by setting:

$$\dot{q}_0 = -\frac{\partial U_{rep}(q)}{\partial q} \tag{4.17}$$

Since the repulsive potential contribution is mapped into the null-space of the Jacobian, there may be still situations in which the robot reaches the joint limits. If it happens, the algorithm executes the recovery procedure.

4.2.5.2 Manipulability

During the approaching phase the algorithm checks if the robot is close to singularity by computing the manipulability [107]. There are many possible measures of the manipulability a robotic arm [108]. We used the reciprocal of the condition number [109]:

$$\kappa = \frac{\sigma_{min}}{\sigma_{max}} \tag{4.18}$$

where σ_{min} and σ_{max} are the minimum and the maximum are the singular values σ_i of J, with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$. If, during the motion, this value decrease under a threshold, the robot is reaching a singular configuration and the algorithm executes the recovery procedure.

We decided to not exploit the redundancy to maximize the manipulability during the motion using Eq. (4.14). The reason is that otherwise the elbow of the robot would move considerably, possibly resulting in self-collisions.

4.2.5.3 Recovery procedure

The recovery procedure is used whenever the algorithm detects that the robot is reaching a joint limit or a singular configuration. During the approaching phase, the trajectory followed by the robot is recorded as a sequence of joint positions. When the recovery procedure is used, the robot executes the recorded trajectory backwards. This action is necessary since in this way the planner doesn't have to plan starting from a singular or constrained configuration, decreasing the chances of planning failures.

4.2.5.4 Grasping and dropping

The grasping phase starts when

$$\|p_A - p_G\| \leq d^* \tag{4.19}$$

where $d^* \in \mathbb{R}$ is a positive threshold. Since the gripper is sensorless, in order to grab the apple the algorithm set a constant velocity ω_{grasp} to the motor for a predefined amount of time T_{grasp} . After this time, the dropping phase begins. Since we considered the starting position of the approaching phase as dropping position for each apple, in order to reach it the algorithm executes the recorded trajectory backwards, as in the recovery procedure. This choice simplifies the planning and make it trivial to avoid singularities and joint limits. The robot then sets $-\omega_{grasp}$ to the motor the same period T_{grasp} to release the apple.

Table 4.1: Simulation Results.

Targets	Sequencing Time (s)		Approach Time (s)		Joint Limit (%)		Low Manipulability (%)	
	Naive	FREDS-MP	Naive	FREDS-MP	Naive	FREDS-MP	Naive	FREDS-MP
5	0.006	1.26	11.87	11.97	22.0	1.0	8.0	2.0
10	0.008	1.81	11.95	11.98	13.5	0.5	4.5	1.5
15	0.01	2.87	11.96	12.01	13.33	0.67	9.0	3.67
20	0.31	4.03	11.94	12.00	18.75	0.0	7.25	2.25

4.2.6 Software simulations

4.2.6.1 Setup

The Sawyer arm and its environment are shown in Fig. 4.17(b). The arm is simulated in OpenRAVE, an Open Robotics Automation Virtual Environment for developing and testing motion planners [110]. The simulations are run on two Intel NUC mini PC's with i7 quad core processor, one dedicated to the perception sub-system and the other to the motion planning and grasping sub-systems.

For the simulation experiments, tests were separated into four groups with 5, 10, 15 and 20 targets. Targets were drawn randomly from a uniform distribution with ranges ([0.8, -0.35, 0.2], [0.9, 0.35, 0.7]) in metres relative to the robot arm base in

the forward, lateral and vertical direction respectively. Gaussian noise with means [0.1, 0.0, 0.0] and standard deviation of 0.05 was added to the targets to simulate uncertainty in the real scenario. Once noise is added, if a collision free IK solution to the target cannot be found, it is re-generated until one is found.

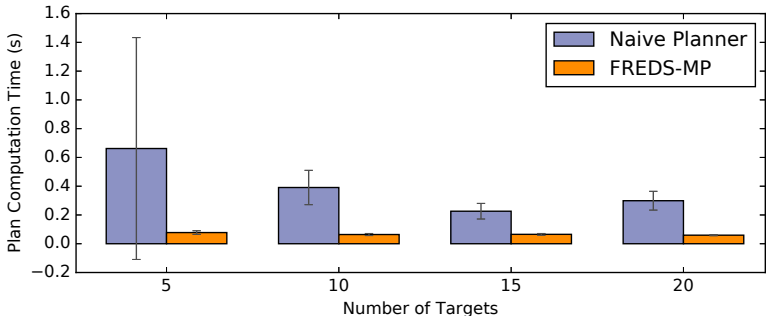
The rationale behind these workspace parameters is based on the observation that the motion planning and grasping are sensitive to the forward distance of the target relative to the base of the robot. If a target is too close then the arm cannot plan without self-collision, particularly in regions where there is not much distance between the obstacle and the robot; bearing in mind the arm needs a sufficient approach distance to update the target position. However, if the target is too far away then it may be unreachable or the visual servoing algorithm is more likely to reach a singular configuration or a joint limit. Further, if the target was below a certain height, approximately lower than 0.2m with respect to the base of the robot, the servoing motion often caused a self-collision. This is due to the absence of collision avoidance in the grasping routine.

To test the effectiveness of FREDS-MP it is compared against a baseline method we call a “Naive Planner". This method sequences the targets based on Euclidean distance in workspace and then greedily picks the IK solution for the next target with the minimum Euclidean distance to the current configuration. The same planners as FREDS-MP are used to generate a trajectory to the configuration, however for the trajectory optimisers a naive initial guess is used, in this case a straight path through configuration space; as is done in practice.

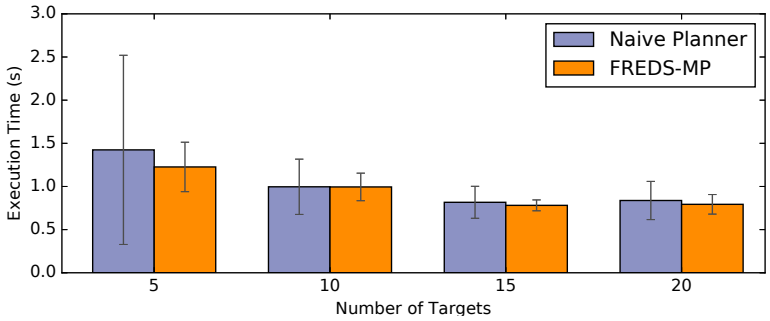
Each test group was run 20 times with new random targets and noise, yielding a total of 80 unique experiments and 1000 targets tested per planner method. For fair comparison, both methods use the same targets, noise and initial configuration. For every new scenario, the start configuration is randomly sampled from the set of IK solutions for a manually chosen “home" pose. Furthermore, for the hardware experiments the joint velocities of the manipulator are reduced to 30% of their maximum speed for safety purposes and to avoid damaging USB and motor cables. This velocity reduction was applied for the simulations.

4.2.6.2 Results

Several metrics are reported and used to compare the performance of the various components of the system. The average plan computation time in Fig. 4.16(a) refers to the time computing the trajectories that move the arm to the next target in the sequence before the grasping phase. The average plan execution time in Fig. 4.16(b) is the time taken by the robot arm to execute this plan. Approach time in Tab. 4.1 is the time taken by the grasping routine to move within reach of a target, as described in Eq. (4.19). Failure cases are additionally reported in Tab. 4.1 where “Joint Limit" and “Low Manipulability" refer to the events that trigger a recovery, as described in Sec. 4.2.5.



(a) Average Plan Computation Time with Std. Dev.



(b) Average Plan Execution Time with Std. Dev.

Figure 4.16: Simulation results for varying number of targets.

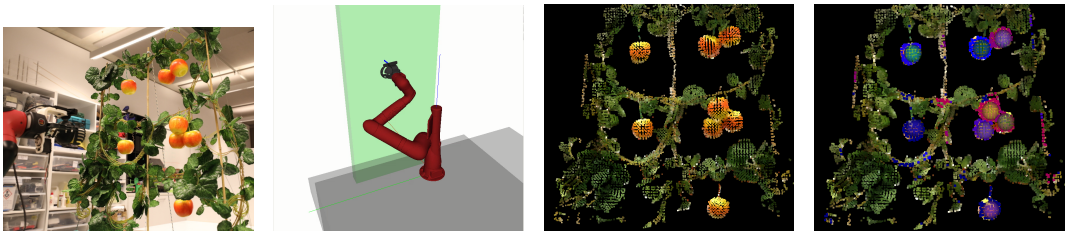
4.2.6.3 Discussion

Observing the results in Tab. 4.1 it is clearly evident that FREDS-MP is better capable at avoiding joint limit and low manipulability failures. In particular, the highest joint limit failure rate was only 1% where as the Naive Method's was 22%. Whilst there was less of a separation in the low manipulability failures, it still outperformed the Naive Planner consistently.

FREDS-MP's sequencing time is higher than the Naive Planner's, however given that it is executed once at the beginning, the gains in execution and computation time outweigh the loss in time. It should be noted that the majority of sequencing time for FREDS-MP is spent computing valid IK solutions for database matching, which could be easily parallelised. Where as the Naive Planner computes the IK solutions at plan time which contributes partly to the high computation times, however regardless it is clear that the computation times are significantly higher for the Naive Planner when removing this overhead. In Fig. 4.16(a) and 4.16(b) the variance of the Naive Planner is significantly larger than FREDS-MP which suggests that the latter is indeed producing accurate heuristics during the sequencing phase.

An interesting trend that emerged is that the computation and execution times tend to gradually decrease with higher number of targets. It is intuited that as more targets are sampled, the likelihood of them being closer together increases, causing the trajectories to be shorter. Lastly, it can be seen that the approach time remained consistent across all experiments, suggesting that for this metric neither method is favourable.

4.2.7 Experiments



(a) (b) (c) (d)

Figure 4.17: The artificial environment used for the experiments. (a) Real hardware, (b) simulated environment, (c) stitched point cloud and (d) segmented point cloud with GPIS entropy overlay (blue means low entropy and red means high entropy).

4.2.7.1 Experimental Setup

The experiments were conducted in a mock-up environment consisting of an artificial apple trellis with realistic appearance, however the stem attachment and weight of the apples are not realistic. The experimental setup can be viewed in Fig. 4.17(a).

Ten experiments were carried out with a real sawyer arm which attempted to harvest 8 fake apples off the trellis. For each experiment a unique apple configuration is manually arranged on the trellis. The aim was to provide a diverse range of scenarios such as clusters of apples with varying sizes and occlusions. The experimental process was run as described in Sec. 4.2.3.

Furthermore, the experimental setup is an effective test bed since it emulates sources of uncertainty that the proposed control and planning method should be able to deal with in the real world. This includes errors from multi-view fusion of point clouds and apples moving around as a consequence of other apples being detached.

4.2.7.2 Results

The results of the experiments are found in Tab. 4.2 and Tab. 4.3. The last column shows the success rate of the experiments. The proposed algorithm could successfully harvest 88.75% of all apples in the experiments.

As shown in Tab. 4.3, the main causes of failures are due to:

- *Misalignment*: the gripper is not aligned to the target apple, missing it or closing the jaws on it;
- *Gripper Obstruction*: the gripper jaws get caught onto the trellis structure and the gripper stops closing or opening the jaws;
- *Failed Detachment*: the gripper grasps the target apple, but it slips out of gripper jaws while detaching from the trellis;

Table 4.2: Experimental results using real robot arm and mock-up trellis.

Experiment	Plan Time (s)	Plan Duration (s)	Approach Time (s)	Success (%)
1	0.07	3.11	11.42	100.00
2	0.07	2.82	11.46	75.00
3	0.05	3.06	11.42	100.00
4	0.09	3.41	11.45	100.00
5	0.06	3.05	11.52	75.00
6	0.22	3.80	11.45	87.50
7	0.05	3.15	11.42	100.00
8	0.04	2.74	11.47	75.00
9	0.07	3.10	11.39	75.00
10	0.10	3.65	11.52	100.00
Total	0.08	3.19	11.45	88.75

Table 4.3: Classification of hardware experiment failures.

Experiment	Misaligned	Gripper Obstructed	Failed Detachment	Poor Prioritisation
1	0	0	0	0
2	1	0	0	1
3	0	0	0	0
4	0	0	0	0
5	0	0	1	1
6	0	0	0	1
7	0	0	0	0
8	1	1	0	0
9	0	0	1	1
10	0	0	0	0
Total	2	1	2	4

- *Poor Prioritisation*: the gripper hits an apple in front or near the target apple, pushing it out of the gripper's jaws or an apple is unintentionally detached whilst another target apple is grasped.

In Fig. 4.17(c) and 4.17(d) an example of the output of the perception algorithm during the experiments is shown. As can be seen, it correctly detects all the apples and is able to resolve clusters effectively.

4.2.7.3 Discussion

The proposed system achieved high success rates suggesting that it is robust to a diverse range of apple configurations. This success was aided by the robust perception and segmentation algorithm which correctly detected all the apples in the experiments. Moreover, there was no plan, joint limit or low manipulability failures.

The failures in Tab. 4.3 can be largely attributed to the gripper and its interaction with the artificial trellis. Although the environment is realistic, it is not fully representative, i.e. real apples are heavier and more rigidly attached. This may help prevent some failures resulting from poor prioritisation. Further, we believe that with a modest amount of gripper adjustments many of the failures could be mitigated.

It should be noted that plan durations are slightly higher than the simulated trajectories due to the imprecise nature of the manipulator's controller, particularly when near the target configuration. Commanding the arm to the last configuration for an extra padded amount of time is necessary for it to reach the desired pose.

4.2.8 Conclusions

In this Section we have presented an apple harvesting system capable of reliably harvesting apples among clutter. The planning method we proposed outperformed the baseline method in terms of computation time, execution time and success rate. Visual Servoing was used to approach the target apple robustly, exploiting the redundancy of the robot for avoiding the joint limits. The overall system was tested using an artificial trellis resulting in a harvesting success rate of 88.75%.

Chapter 5

Multi-Robot Interconnection

5.1 Introduction

Establishing a dynamic coupling between robots is very useful for several contexts. Interconnections which are equivalent to spring or spring-damper couplings have been widely used in bilateral teleoperation [111, 112, 113] and in multi-robot systems [114, 115, 116] Using nonlinear springs and dampers it is possible to achieve more complex behaviors as connectivity maintenance [117, 118] or formation control and synchronous operation in power network [119].

One of the main reasons behind the success of dynamic couplings in multi-robot systems is that, since the coupling is equivalent to a physical passive dynamics, if the robots to be coupled behave passively the overall coupled system will behave passively [59]. Passivity guarantees a robust stability and a stable interaction with any, possibly unknown, passive environment [59]. This allows to achieve, e.g., a robust cohesive behavior in multi-robot systems [116, 43] and a safe interaction with the environment [120].

Nevertheless, it is often useful to change the coupling to achieve different kind of behaviours [121, 122, 123]. Unfortunately, implementing a variable behavior may lead to a loss of passivity of the overall system and, consequently, to a loss of robust stability.

The concept of energy tank [60, 124] has been introduced for unconstraining passivity based control from a specific physical dynamics. The tank stores the energy that can be exploited for implementing any kind of behavior without violating the passivity constraint. When no more energy is available, non passive actions are forbidden. Energy tanks allow to achieve flexibility while preserving passivity and, consequently, robust stability. They have been successfully exploited in bilateral teleoperation [60, 124, 125], in multi-robot systems [116, 126], in force control [127, 128] and in human-robot collaboration [129]. So far, the tank and the local control action has been linked to a robot and not on the interconnection between two or more robots.

In this Chapter we aim at building a *tank based generalized interconnection*, namely a flexible and disembodied strategy that allows to passively implement any kind of interconnection. We will exploit the concept of energy tank for flexibly and

passively reproducing the desired interconnection. We will show that if the desired interconnection is passive, then it can always be passively implemented using the proposed approach. A variable damping will be exploited for refilling the tank when more energy is necessary for implementing a non passive interconnection. The unwanted dynamics introduced by the damping is the price to pay to make any desired interconnection implementable. The proposed interconnection element will be illustrated on a multi-robot system.

The contribution of this work is a novel passive interconnection, disembodied from any physical dynamics, that can be exploited for implementing any desired coupling in a flexible and passive, i.e. robustly stable way.

5.2 Background

In this section we will provide some background on energy tanks. For a more detailed treatment see, e.g., [60, 124].

Consider a passive system defined by:

$$\begin{cases} \dot{x} = f(x) + g(x)u \\ y = h(x) \end{cases} \tag{5.1}$$

where $x \in \mathbb{R}^n$ and $u, y \in \mathbb{R}^m$. Since the system is passive there exists a non negative energy function $H : \mathbb{R}^n \to \mathbb{R}$ such that $\dot{H} + P_d(t) = u^T y$. It means that the (generalized) power $u^T y$ introduced by the power port (u, y) is either stored as energy ($\dot{H}$) or dissipated ($P_d(t) > 0$). Finding a controller to interconnect to (5.1) for achieving a desired behavior while preserving a passive energy balance introduces several constraints and it is one of the main challenges of passivity based control [130].

Energy tanks [60, 124, 129] allow to make the control design more flexible. An energy tank is an energy storing element represented by

$$\begin{cases} \dot{x}_t = u_t \\ y_t = \frac{\partial T}{\partial x_t} \end{cases} \tag{5.2}$$

where $x_t, y_t, u_t \in \mathbb{R}$ are the state of the tank and the pair u_t, y_t represents the power port of the tank and $T = \frac{1}{2}x_t^2$ is the stored energy. The tank keeps track of the energy dissipated by a passive system and, therefore, the energy available in the tank can be exploited for implementing any control action without violating the passivity constraint. Considering (5.1), this can be done by setting

$$\begin{cases} u_t = \frac{1}{x_t} P_d + w^T y \\ u = w y_t \end{cases} \tag{5.3}$$

where $w = (w_1, \ldots, w_m)^T \in \mathbb{R}^m$. It can be shown that the coupling of (5.1) and (5.2) through (5.3) is passive for any, possibly time-varying, value of w in (5.3) [60]. Any desired input $u_d = (u_{d_1}, \ldots, u_{d_m})^T$ can be obtained exploiting the energy

available in the tank by setting $w_i = \frac{u_{d_i}}{x_t}$, for $i = 1, \ldots, n$. When the desired action is dissipative, the corresponding dissipated energy is stored in the tank. On the other hand, if u_d is a non passive action, the generated energy is extracted from the tank. If the energy available in the tank is not enough for implementing the desired input, some tank refilling or input adjusting strategies have to be activated (see e.g. [60, 124] for examples).

5.3 Problem statement

Consider a set of N passive dynamical systems represented by:

$$\begin{cases} \dot{x}_i = f_i(x_i) + g_i(x_i)(u_{I_i} + u_{E_i}) \\ y_i = h_i(x_i) \end{cases} \qquad i = 1, \ldots, N \tag{5.4}$$

where $x_i \in \mathbb{R}^n$ and $u_{I_i}, u_{E_i}, y_i \in \mathbb{R}^m$. The input u_{I_i} comes from the interconnection with the other systems and the input u_{E_i} is due to the interaction with the external world. For each system the following energy balance holds:

$$\dot{H}_i + P_{d_i}(t) = u_{I_i}^T y_i + u_{E_i}^T y_i, \quad i = 1, \ldots, N \tag{5.5}$$

where $H_i : \mathbb{R}^n \to \mathbb{R}$ and $P_{d_i} > 0$ represents the stored energy and the dissipated power respectively. Let $u_I = (u_{I_1}^T, \ldots, u_{I_N}^T)^T \in \mathbb{R}^{Nm}$, $u_E = (u_{E_1}^T, \ldots, u_{E_N}^T)^T \in \mathbb{R}^{Nm}$ $y = (y_1^T, \ldots, y_N^T)^T \in \mathbb{R}^{Nm}$ be the input and the output vectors of all the systems.

Let $\mathcal{I}$ be a desired time-varying dynamic interconnection among the N passive systems be represented by:

$$\mathcal{I} : \begin{cases} \dot{x}_{\mathcal{I}} = \phi(x_{\mathcal{I}}, y, t) \\ \mu_d = \gamma(x_{\mathcal{I}}, y, t) \end{cases} \tag{5.6}$$

where $\mu_d = (\mu_{d1}^T, \ldots, \mu_{dN}^T)^T \in \mathbb{R}^{Nm}$ is the desired input for implementing $\mathcal{I}$ and $x_{\mathcal{I}} \in \mathbb{R}^q$ represents the state of the interconnetion.

We aim at finding a tank based generalized interconnection that can reproduce in a passive way any $\mathcal{I}$. In particular, if $\mathcal{I}$ is passive, the generalized interconnection has to be able to reproduce it exactly and permanently. In case $\mathcal{I}$ is non passive, the tank based generalized interconnection has to reproduce it until passivity is violated. Then, a minor modification has to be made in order to implement it without violating the passivity constraint.

5.4 Tank Based Generalized Interconnection

In this section we define the concept of Tank based Generalized Interconnection and we show how it can implement any desired interconnection while preserving the passivity of the overall system.

5.4.1 The Modulated Multi-port Tank

In order to use a tank to interconnect multiple passive systems, it is necessary to make (5.2) a multi-port tank.

Let $(u_{t_i}, y_{t_i}) \in \mathbb{R} \times \mathbb{R}$, for $i = 1, \dots P$, be the P ports we would like to endow our tank with. A multi-port tank can be defined as:

$$\begin{cases} \dot{x}_t = \sum_{i=1}^{P} u_{t_i} \\ y_{t_1} = y_{t_2} = \cdots = y_{t_P} = \frac{\partial T}{\partial x_t} \end{cases} = \begin{cases} \dot{x}_t = 1_P^T u_T \\ y_T = 1_P \frac{\partial T}{\partial x_t} \end{cases} \tag{5.7}$$

where $u_T = (u_{t_1}, u_{t_2}, \dots, u_{t_P})^T \in \mathbb{R}^P$, $y_T = (y_{t_1}, y_{t_2}, \dots, y_{t_P})^T \in \mathbb{R}^P$ and 1_P denotes the P dimensional vector of ones. The function

$$T(x_t) = \frac{1}{2} x_t^2 \tag{5.8}$$

represents the energy stored in the tank.

Although the multi-port tank described by (5.7) can interact with the external world by means of P power ports, all the outputs y_{t_i} have the same value and this would prevent the use of the energy in the tank to implement different dynamic behaviors in each port.

In order to solve this problem, we introduce a modulation matrix $\alpha(t) = diag(\alpha_1(t), \dots, \alpha_P(t)) \in \mathbb{R}^{P \times P}$ that allows to modulate the output of each port. Thus, (5.7) becomes a *modulated multi-port tank* (MMT):

$$\begin{cases} \dot{x}_t = 1_P^T \alpha(t) u_T \\ y_T = \alpha(t) 1_P \frac{\partial T}{\partial x_t} \end{cases} \tag{5.9}$$

The MMT is passive with respect to the pair (u_T, y_T) independently of the time-varying modulation matrix $\alpha(t)$. In fact, from (5.8) and (5.9) we get:

$$\dot{T} = x_t \dot{x}_t = x_t 1_P^T \alpha(t) u_T = y_T^T u_T = \sum_{i=1}^{P} y_{t_i} u_{t_i} \tag{5.10}$$

Thus, the MMT stores the energy flowing through the P power ports and $\alpha(t)$ can be freely used to modulate the outputs of each port without violating passivity. In particular each output can be set to a desired value. If $y_d \in \mathbb{R}^P$ is the desired value for y_T, we can impose it by setting $\alpha_i = \frac{y_{d_i}}{x_t}$, for $i = 1, \dots, P$. As long as there is some energy in the tank (i.e. $x_t > 0$) no singularity occurs in the design of α.

5.4.2 Passive Interconnection

Consider the N systems to be interconnected represented by (5.4). In order to join the systems we build a MMT as represented in (5.9) with $P = Nm$. The systems are linked to the MMT by means of the following power preserving interconnection

$$\begin{cases} u_I(t) = y_T(t) \\ u_T(t) = -y(t) \end{cases} \tag{5.11}$$

From (5.9), (5.11) and (5.10) we can obtain:

$$\begin{cases} \dot{x}_t = -\frac{\sigma}{x_t} y^T u_I \\ u_I = \alpha(t) 1_{Nm} \frac{\partial T}{\partial x_t} \end{cases} \tag{5.12}$$

where $\alpha(t) = diag(\alpha_1(t), \ldots, \alpha_{Nm}(t))$ and $\sigma \in \{0, 1\}$ is used for bounding the amount of energy that can be stored in the tank. In particular, as discussed in [60], in order to avoid singularities in the implementation of the tank and in order to prevent the possibility of implementing practically unstable actions, the energy stored in the tank needs to be lower and upper bounded. If $0 < \underline{\varepsilon} < \bar{\varepsilon} < \infty$ denote the bounds, then

$$\sigma = \begin{cases} 0, & \text{if } T = \bar{\varepsilon} \quad \text{and} \quad y^T u_I \leq 0 \\ 1, & \text{otherwise} \end{cases} \tag{5.13}$$

which means that energy storage is prevented in case the upper bound is reached and the system is trying to convey more energy into the tank. Thus, considering (5.13), we have that

$$\dot{T} = -\sigma y^T u_I \tag{5.14}$$

Using (5.12), we can exploit the energy stored in the tank for implementing any desired input for the interconnected systems as long as some energy is available in the tank. Thus, it is important to understand how to reproduce the desired interconnection in order to prevent the tank from reaching its lower bound $\underline{\varepsilon}$.

Consider the generic time varying interconnection (5.6) and its effect on the tank according to (5.14). Let $\delta(t) = y^T(t)\mu_d(t)$ be the energy flow due to the implementation of the desired interconnection. Each interconnection can instantaneously behave in three different energetic ways, characterized by the value of $\delta(t)$. If $\delta(t) = 0$, the interconnection is behaving in a power preserving way and its implementation neither stores nor extracts energy from the tank. If $\delta(t) < 0$ the interconnection is behaving in a dissipative way and its implementation stores energy in the tank. Finally, if $\delta(t) > 0$ the interconnection is behaving in a generative way and its implementation extracts energy from the tank. In order to prevent the depletion of the tank, we add a variable damping factor [116] to the desired inputs, i.e. we set

$$u_{Id} = \mu_d - \beta(t) y \tag{5.15}$$

where u_{Id} is the desired value for u_I and $\beta(t) \geq 0$ is defined as:

$$\beta(t) = \begin{cases} \frac{\delta}{y^T y} & \text{if } T = \underline{\varepsilon} \quad \text{and} \quad \mu_d^T y = \delta > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.16}$$

If the implementation of the desired interconnection does not require energy from the tank or as long as some energy is available in the tank, $\beta(t) = 0$ and the

desired interconnection μ_d is reproduced. If the energy in the tank has reached its lower bound and the reproduction of μ_d still requires energy, it is necessary to refill the tank. This is done by activating the damping $\beta(t)$ whose role is to inject into the tank an amount of dissipated energy exactly equal to the one necessary for implementing the desired interconnection. In fact, the power exchanged through the interconnection is given by:

$$u_{Id}^T y = \delta - \beta(t) y^T y \tag{5.17}$$

If $T = \underline{\varepsilon}$ and $\delta > 0$, thanks to (5.16), we have that the power requested by the tank for implementing u_{Id} is zero. Of course the damping introduces an unwanted effect on the implemented interconnection and this is the price to pay for preserving passivity. Nevertheless, as long as some energy is available in the tank, generative interconnections are implemented with no spurious corrections. The damping is activated as a last resort when the tank is about to get empty and when some energy is stored again the damping will be deactivated. Finally, notice that $\beta(t)$ is always well posed. In fact, if $\delta > 0$ then at least one component of y is different from zero and, therefore, $y^T y > 0$.

In summary, using (5.6) and (5.15) for defining a desired input u_{I_d}, the tank will be never depleted and, therefore, using (5.12) it is possible to exploit the energy in the tank for reproducing the desired input by setting the elements of the tuning matrix as $\alpha_i = \frac{u_{Id_i}}{x_t}$ for $i = 1, \ldots, Nm$.

Proposition 1. *Consider N passive systems described by* (5.4) *and interconnected by* (5.12) *for implementing the time varying interconnection* (5.15)*. The overall interconnected system is passive with respect to the pair (u_E, y).*

Proof. Consider the following non negative storage function:

$$H = \sum_{i=1}^{N} H_i(x_i) + T(x_t) \tag{5.18}$$

Using (5.5) and (5.14) we have

$$\dot{H} \leq u_E^T y + (1 - \sigma) u_I^T y \tag{5.19}$$

According to (5.13), if $\underline{\varepsilon} < T(x_t) < \overline{\varepsilon}$ or if $T(x_t) = \overline{\varepsilon}$ and $y^T u_I < 0$ then $\sigma = 1$ and therefore, from (5.19) we have that $\dot{H} \leq u_E^T y$. In case $T(x_t) = \overline{\varepsilon}$ and $y^T u_I \leq 0$, we have that $\sigma = 0$ and, therefore, from (5.19) we have:

$$\dot{H} \leq u_E^T y + u_I^T y \leq u_E^T y \tag{5.20}$$

Thus, the overall system always behaves passively and this concludes the proof. □

5.5 Passive interconnections

In this section we show that if the desired interconnection is represented by a passive system, then the corrective damping reported in (5.16) will never be activated. This result is an extension of [118, Proposition 4] where only two systems were considered.

Consider a passive system to be used for interconnecting the N systems (5.4):

$$\begin{cases} \dot{x}_C = f_C(x_C) + g_C(x_C)u_C \\ y_C = h_C(x_C) \end{cases} \tag{5.21}$$

where $x_C \in \mathbb{R}^C$, $y_C, u_C \in \mathbb{R}^{Nm}$ Furthermore, the following passivity balance holds:

$$y_C^T u_C = \dot{H}_C + P_{d_c} \tag{5.22}$$

where $H_C : \mathbb{R}^C \to \mathbb{R}$ is a non negative energy function and $P_{d_c} \geq 0$ represents the power dissipated. Assume that during the dynamic behavior of the interconnected system $H_C(x_C) \leq \bar{\varepsilon}$. The assumption is usually satisfied in practice because of the actuation bounds of the actuators of the interconnected robots.

The passive interconnection can be implemented by joining the (5.21) to (5.4) by means of a power-preserving interconnection, e.g., w.l.o.g., the standard feedback interconnection

$$\begin{cases} u_I = y_C \\ u_C = -y \end{cases} \tag{5.23}$$

Using (5.11) and (5.12) it is possible to implement the passive interconnection (5.21), (5.23) by setting:

$$\begin{cases} u_I(t) = y_T(t) = y_C(t) \\ u_T(t) = -y(t) = u_C(t) \end{cases} \tag{5.24}$$

Thus, we have that the power crossing the port of the MMT is the same as the one that would cross the power port of (5.21) if the passive dynamic were directly implemented, i.e.

$$u_T^T y_T = u_C^T y_C \tag{5.25}$$

Thanks to this relation, we can prove the following result.

Proposition 2. *If $x_t(t_0)$ is chosen such that*

$$T(x_t(t_0)) = H_C(x_C(t_0)) + \epsilon \tag{5.26}$$

for some $\epsilon > \underline{\varepsilon} > 0$, then $T(x_t(t)) > \epsilon$ and $\beta(t) = 0 \ \forall t > t_0$.

Proof. From the passivity balance (5.22) we have that

$$H_C(x_C(t)) - H_C(x_C(t_0)) \leq \int_{t_0}^{t} y_C(\tau)^T u_C(\tau) d\tau \tag{5.27}$$

From (5.10) we have that:

$$T(x_t(t)) - T(x_t(t_0)) = \int_{t_0}^{t} y_T(\tau)^T u_T(\tau) d\tau \tag{5.28}$$

From (5.25), (5.27) and (5.28) it follows that

$$H_C(x_C(t)) - H_C(x_C(t_0)) \leq T(x_t(t)) - T(x_t(t_0))$$

Therefore, by initializing $T(x_t(t_0)) = H_C(x_C(t_0)) + \epsilon$, for some $\epsilon > 0$, we obtain

$$T(x_t(t)) > H_C(x_C(t)) + \epsilon > \epsilon, \quad \forall t > t_0$$

Since $\epsilon > \underline{\varepsilon}$, because of (5.16), $\beta(t) = 0 \ \forall t > t_0$. □

Thus, using the Tank Based Generalized Interconnection proposed in Sec. 5.4 it is possible to exactly replicate any passive interconnection for an infinite amount of time without the need of introducing spurious dissipations.

This result shows that the proposed interconnection is very suitable for transitioning among passive interconnections in a passive way, without the need of resorting to complex time-varying control analysis. In fact, if the systems are coupled by a passive interconnection (e.g. a set of springs), it is possible to change the dynamic behavior (e.g. increasing the stiffness of the spring, a non passive operation[60]) by exploiting the energy stored in the tank (possibly using the correction reported in (5.16)). Once (5.26) is satisfied for the desired new passive interconnection (e.g. stiffer springs), the system will always be able to precisely implement it using the energy in the tank.

5.6 The Multi-Robot Case

In this section we will apply the tank based generalized interconnection to a multi-robot system application. Consider a group of N robots that behave as second order systems (as in, e.g., [116]) described by:

$$\begin{cases} \dot{p}_i = F_i^a + F_i^e - B_i M_i^{-1} p_i \\ v_i = \frac{\partial \mathcal{K}_i}{\partial p_i} = M_i^{-1} p_i \end{cases} \quad i = 1, \ldots, N \tag{5.29}$$

where $p_i \in \mathbb{R}^3$ and $M_i \in \mathbb{R}^{3\times3}$ are the momentum and positive definite inertia matrix of agent i, respectively, $\mathcal{K}_i(p_i) = \frac{1}{2} p_i^T M_i^{-1} p_i$ is the kinetic energy stored by the agent during its motion, and $B_i \in \mathbb{R}^{3\times3}$ is a positive definite matrix representing a velocity damping term. The input $F_i^a \in \mathbb{R}^3$ represents the interconnection force between the agent i and the other agents. The force $F_i^e \in \mathbb{R}^3$ is an additional input that can be used for implementing other tasks. Finally, $v_i \in \mathbb{R}^3$ is the velocity of the agent. The system is passive with respect to the port $(v_i,\ F_i^a + F_i^e)$ and storage function $\mathcal{K}_i$ [116].

The total interconnection force applied to the agent i is given by:

$$F_i^a = \sum_{k=1}^{N} F_{ik}^a \tag{5.30}$$

Rather than interconnecting agents with a fixed physical dynamics, each pair of agents is interconnected with a MMT. Thus, each pair of agents shares and updates the following system:

$$\begin{cases} \dot{x}_{t_{ij}} = -1_6^T \alpha_{ij}^T \begin{bmatrix} v_i \\ v_j \end{bmatrix} \\ \begin{bmatrix} F_{ij}^a \\ F_{ji}^a \end{bmatrix} = \alpha_{ij} 1_6 \frac{\partial T_{ij}}{\partial x_{t_{ij}}} \end{cases} \tag{5.31}$$

where α_{ij} represents the modulation necessary for achieving the desired interconnection forces between the agents i and j:

$$\alpha_{ij} = \begin{bmatrix} \alpha_{ij} & 0 \\ 0 & \alpha_{ji} \end{bmatrix} = \frac{1}{x_{t_{ij}}} \begin{bmatrix} F_{ij}^{ad} & 0 \\ 0 & F_{ji}^{ad} \end{bmatrix} \in \mathbb{R}^{6\times 6} \tag{5.32}$$

From (5.31) it follows that

$$\dot{x}_{t_{ij}} = \underbrace{\begin{bmatrix} 0 \ \dots \ \bar{\alpha}_{ij}^T \ \dots \ \bar{\alpha}_{ji}^T \ \dots \ 0 \end{bmatrix}}_{A_{ij}^T} \begin{bmatrix} v_1 \\ \vdots \\ v_i \\ \vdots \\ v_j \\ \vdots \\ v_N \end{bmatrix} \tag{5.33}$$

where $\bar{\alpha}_\star = \alpha_\star * 1_6$. From (5.29) and (5.30) we can derive the overall system:

$$\begin{cases} \begin{bmatrix} \dot{p} \\ \dot{x}_t \end{bmatrix} = \begin{bmatrix} -B & A \\ -A^T & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial H}{\partial p} \\ \frac{\partial H}{\partial x_t} \end{bmatrix} + \begin{bmatrix} I_N \otimes I_3 \\ 0 \end{bmatrix} F^e \\ v = \begin{bmatrix} I_N \otimes I_3 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial H}{\partial p} \\ \frac{\partial H}{\partial x_t} \end{bmatrix} \end{cases} \tag{5.34}$$

where $p = (p_1^T \ \dots \ p_N^T)^T$, $x_t = (x_{12} \ \dots \ x_{N-1,N})^T$, $B = diag(B_1, \ \dots \ , B_N)$, $A = (A_{12} \ \dots \ A_{N-1,N})^T$, $F^e = (F_1^{eT} \ \dots \ F_N^{e\,T})^T$. The operator $\otimes$ denotes the *Kronecker product* among matrices. The system reported in (5.34) is in a port-Hamiltonian form [59] with a positive energy function

$$H = \sum_{i=1}^{N} \mathcal{K}_i(p_i) + \sum_{i=1}^{N-1} \sum_{j=1}^{N} T_{ij}(x_{t_{ij}}) \tag{5.35}$$

and, therefore, the overall system is passive [59] regardless of the time varying value of $A(t)$. This means that, it is possible to implement any desired behaviour to each agent while preserving the passivity of the overall system. Unlike, e.g., [116, 126], where the interconnection among the agents is implemented using a spring-damper interconnection and energy tanks where used for handling some temporarily active behaviors in sometimes a cumbersome way, using a generalized tank based interconnection the whole variability of the behavior is taken inside $A(t)$ which can be arbitrarily easily tuned as shown in (5.32) for getting the desired interconnection in a passive way.

5.7 Simulations and Experiments

5.7.1 Simulations

We considered a group of $N = 5$ quadrotors keeping a desired formation. The dynamics of each quadrotor can be represented by (5.29). The leader of the multi-robot

system is connected to a pre-defined point x_G through a spring-damper coupling:

$$F_l^e = K_{P_G}(x_G - x_l) - K_{D_G}\dot{x}_l \tag{5.36}$$

where K_{P_G}, K_{D_G} are positive gains and $x_l \in \mathbb{R}^3$ is the position of the leader. Each pair of agents is interconnected by (5.31), in which the desired interconnection force is:

$$F_{ij}^{ad} = \begin{cases} K_{P_{ij}}(x_{ji} - x_{ji}^d) + K_{D_{ij}}\dot{x}_{ji} & \text{if } d_{ij} < D \\ 0 \qquad otherwise \end{cases} \tag{5.37}$$

where $K_{P_{ij}}, K_{D_{ij}}$ are positive gains, $x_{ji} = x_j - x_i \in \mathbb{R}^3$ is the relative position of the agents, $x_{ij}^d \in \mathbb{R}^3$ is the desired relative position, d_{ij} is the distance between the agents and $D > 0$ is a threshold which represents the sensing range of each quadorotor. Moreover, each agent can detect obstacles and avoid them by a repulsive potential field:

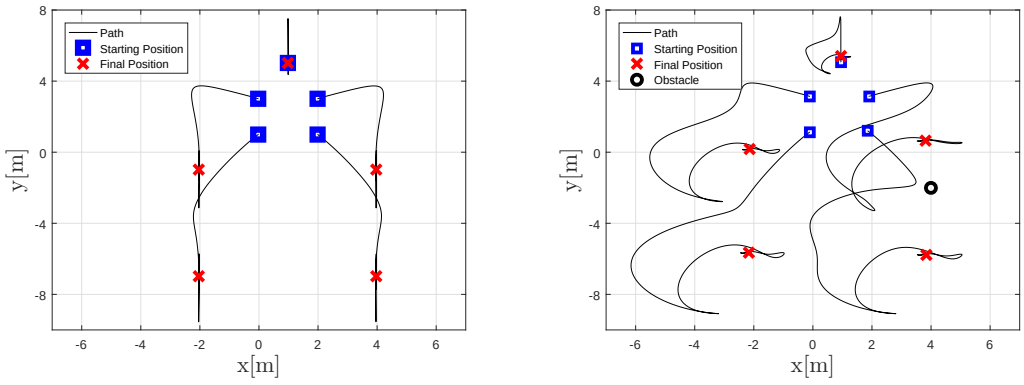
$$F_i^e = -\nabla U_{r_i} \tag{5.38}$$

where U_{r_i} represent the repulsive potential function described by:

$$U_{r_i} = \sum_{k=1}^{q} U_{r_{ik}}, \; U_{r_{ik}} = \begin{cases} \frac{1}{2}K_{r_{ik}}(\frac{1}{d_{ik}} - \frac{1}{Q})^2 & \text{if } d_{ik} <= Q \\ 0 \qquad otherwise \end{cases} \tag{5.39}$$

where $K_{r_{ik}}$ is a positive gain, d_{ik} is the distance between the agent i and the obstacle k and $Q > 0$ is the distance beyond which the agents disregards the obstacle.

We want to modify the formation of the group by changing the rest length of the springs between the agents, namely x_{ji}^d. This behaviour was simulated both in an empty virtual environment and in a virtual environment which has an obstacle. Fig. 5.1-5.2 show the paths followed by the agents when the rest length of all the springs is changed. Changing the rest length of the springs is a non passive,



(a) (b)

Figure 5.1: Change of formation by increasing the rest length of the springs. (a) Without the obstacle. (b) The presence of the obstacle changes the path and the final position of the UAVs

possibly destabilizing, operation [59], therefore some energy is extracted from the

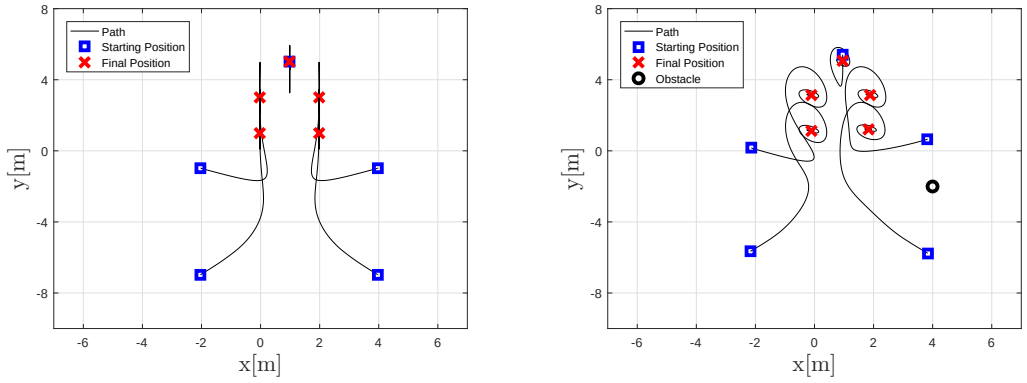


Figure 5.2: Change of formation by decreasing the rest length of the springs. (a) Without the obstacle. (b) The presence of the obstacle changes the path of the UAVs

corresponding tanks (Fig. 5.3). Thanks to the proposed interconnection, using the energy of the tank the variation of the rest length is implemented in a passive way and it leads to a passive and stable behavior of the overall multi-robot system. An animation of the simulations can be appreciated in the video[1].

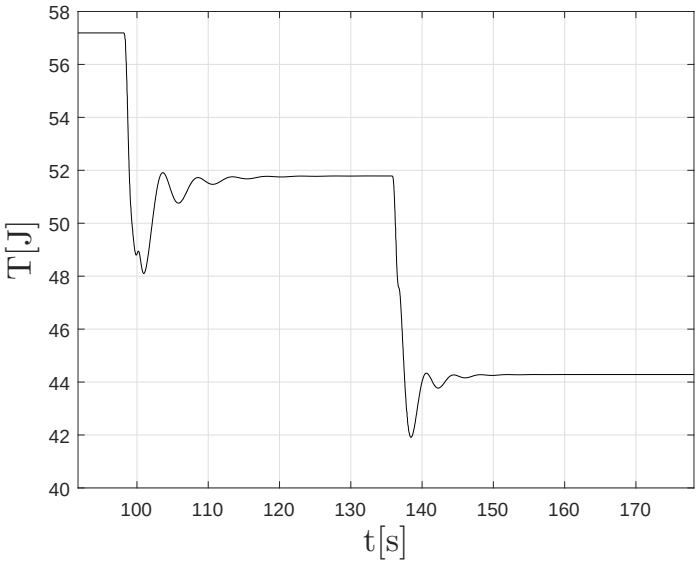


Figure 5.3: Energy stored in the tank between two agents during the simulation.

[1]https://drive.google.com/open?id=1hDSKTIHh-Lluz1rO3OD3gWDdsfKai8rN

5.7.2 Experiments

The experiments were made using $N = 2$ mobile robots which keep a desired relative position using a laser scanner for measuringing the relative position. Since the velocity of both the robot is low, it is possible to use a kinematic model:

$$\begin{cases} \dot{x}_i = u_{I_i} + u_{E_i} \\ y_i = x_i \end{cases} \tag{5.40}$$

where x_i is the position of the robot. The agents are interconnected with a spring:

$$u_{I_i} = K_{I_{ij}}(x_j - x_i - x_{ji}^d) \tag{5.41}$$

As for the simulation, we want to modify the relative position by changing the rest length of the spring. Fig. 5.4 shows the energy level of the tank during the experiment. Note that, since we considered a kinematic model, the energy is extracted from the tank only if the rest length of the spring is increased. The behavior of the dual robot system can be seen in the video[2].

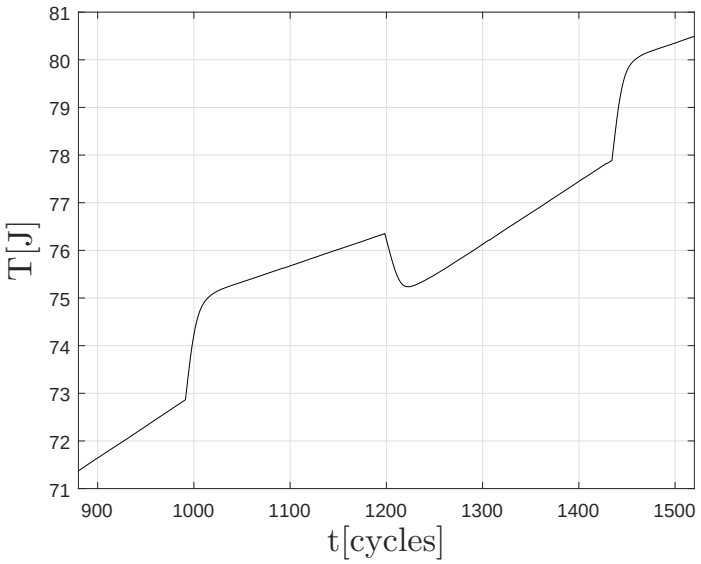


Figure 5.4: Energy stored in the tank between two agents during the experiment.

5.8 Conclusions

In this Chapter we have introduced the concept of *Modulated Multi-port Tank* for implementing any desired coupling while preserving a passive behavior. The effectiveness of this novel tool has been illustrated on multi-robot systems.

[2]https://drive.google.com/open?id=1hDSKTIHh-Lluz1rO3OD3gWDdsfKai8rN

Chapter 6

Conclusions And Future Work

6.1 Conclusions

Robots are becoming an important part of everyday life and their applications range from common social environments to very specific industrial applications. Regardless the specific task and application environment, it is important to understand what is the best way to interact with them. In this thesis we developed different ways to use robots in different scenarios.

In Chap. 2 we presented a novel infrastructure-less interface to teleoperate a mobile robot, based on the recognition of the movements of user's forearm. A common smartwatch is used as sensor for recognize the movements, therefore there is no need for dedicated devices. The forearm motions can be gestures which the robot interprets as high level commands or they can be directly mapped into the robot velocities for direct teleoperation. The use of the smartwatch instead of a dedicated device proved to be easier and more intuitive for most of the users.

In Chap. 3 we dealt with an industrial scenario in which human-robot collaboration was required. We presented TIREBOT, a collaborative mobile robot which assists the operator in a wheel management procedure. TIREBOT can lift and transport wheels and it can receive commands either from a gesture based interface or by teleoperation. It is endowed with forks for grabbing the wheel and carrying it to the different machines. When the user has to load the wheel on the robot, a safe cooperation behaviour is activated. In this modality the robot let the user get close but it goes away from the operator if an unexpected event happens in order to guarantee his/her safety.

In Chap. 4 we considered fully autonomous robots in an agricultural environments. The first robot is a mobile robot which has to navigate through a vineyard while taking pictures of the grapes. Those pictures are used for estimating the yield. The robot uses its laser scanner to detect the vines and follow each row at a certain distance, defined in such a way the fixed RGB camera has most of the grapes in its view. The second robot is a robotic arm capable of detecting the apples on a tree and grab them. The robot uses the RGB-D camera to get the pointcloud of the scene and elaborate it in order to separate the fruits from the other elements. The fruits positions are used as setpoints for the planner which generates free collision

trajectories to reach and grab them.

The thesis ends in Chap. 5 where multi-robot systems are studied. In particular, we dealt with the problem of preserving the stability of the overall system while implementing a desired interconnection between the agents. We exploited the properties of passivity to impose robust stability. The *modulated multi-port tank* is presented to store the energy needed for implementing the desired interconnection. A variable damping factor is used to refill the tank if its energy is low.

6.2 Future Work

In the near future, robots will be applied in every aspect of our lives. Some people may think that autonomous robots will eventually replace humans in every application, due to their advantages in terms of efficiency, precision and productivity and the progress of artificial intelligence. In my opinion, a future where robots will autonomously do everything and take decisions in our place is just a sci-fi scenario. Robots will be everywhere in our lives, but the way they will be applied will depend on the specific task. Autonomous robots are very useful in many applications, but in some tasks it would be better to exploit human skills and use human-robot collaboration or teleoperation.

The work presented in this Thesis deals with different way for interacting and communicate with robots based on the task. It can be improved in some aspects. In Sec. 2.1 would be interesting to consider the implementation of on-line learning methods for adapting the templates to the user, thus further improving the performance of the proposed interaction architecture. Moreover, we plan to improve the usability assessment, measuring the cognitive workload associated to the use of the smartwatch to interact with a quadrotor. This information can be exploited to adapt the robot behavior to the cognitive and emotional state of the user, in a scenario of affective HRI [131].

In Sec. 2.2 we plan to improve the usability assessment, measuring the cognitive workload associated to the use of the smartwatch to interact with the robot. This information can be exploited to adapt the robot's behavior to the cognitive and emotional state of the user, in a scenario of affective robotics. Moreover, we plan to extend the experimental validation of the proposed interaction approach by comparing it to a bilateral teleoperation system providing the user with haptic feedback for obstacle avoidance and target tracking.

In Chap. 3 we aim to overcome to the issues identified during experiments. A way to interact with the robot, more robust and intuitive than the gesture interaction, will be implemented. Also the robotic platform will be substituted with a faster robot, to overcome to the slow speed complaints. TIREBOT is a prototype realized with the intention of studying the realizability of such a robotic assistant: future version should be capable of lifting heavier weights and, possibly, more than just a tire at a time, in order to make the cooperation even more efficient.

In Sec. 4.1, future work aims to predict the yield for the current year, in order to validate the estimation models we found in Sec. 4.1.5.

In Sec. 4.2 we plan to address the issues observed during the simulations and experiments and to test the system in a real environment. In order to relax the robot-trellis positioning sensitivity it is possible to mount the manipulator on a mobile base. In this way the robot can translate further away from the trellis when grasping apples in problematic regions in front of the robot. The gripper design can be improved to resolve many of the problems encountered during the experiments. One suggestion is to add an encoder or contact switch to detect when the gripper is fully closed and open. Further, the camera can be angled downwards so that the gripper can begin its approach closer to the trellis with the apple still in view. The current system cannot detect if the grasping was successful or not. Adding external sensing within the gripper could confirm whether the target apple was grasped and detached from the trellis. Alternatively, this could be achieved by detecting and managing the adding and removal of new and non-existent apples. Another improvement is to increase the overall speed of the system. The perception algorithm can be parallelised and pauses in between state transitions can be optimised. Gripper closing and opening times can be reduced by improving the actuation design. The velocity of the manipulator can be increased if a suitable cable harness is fitted.

In Chap. 5 we aim at considering non ideal communication (e.g. delay) among the robots and a variable topology in the way robots are interconnected. Furthermore, the possibility of learning what is the best way to use the energy in the tank will be also explored.

Bibliography

[1] Roger Bemelmans, Gert Jan Gelderblom, Pieter Jonker, and Luc De Witte. Socially assistive robots in elderly care: A systematic review into effects and effectiveness. *J. Amer. Medical Directors Assoc.*, 13(2):114–120, 2012.

[2] Donato Di Paola, Annalisa Milella, Grazia Cicirelli, and Arcangelo Distante. An autonomous mobile robotic system for surveillance of indoor environments. *Int. J. Adv. Robot. Syst.*, 7(1):19–26, 2010.

[3] Nicola Tomatis, Roland Philippsen, Björn Jensen, Kai Oliver Arras, G Terrien, R Piguet, and Roland Yves Siegwart. Building a fully autonomous tour guide robot. In *Proc. 33rd Int. Symp. Robotics (ISR)*, 2002.

[4] M. c. Kang, K. s. Kim, D. k. Noh, J. w. Han, and S. j. Ko. A robust obstacle detection method for robotic vacuum cleaners. *IEEE Trans. Consumer Electron.*, 60(4):587–595, Nov 2014.

[5] Robert J.K. Jacob, Audrey Girouard, Leanne M. Hirshfield, Michael S. Horn, Orit Shaer, Erin Treacy Solovey, and Jamie Zigelbaum. Reality-based interaction: A framework for post-WIMP interfaces. In ACM Press, editor, *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI)*, pages 201–210, 2008.

[6] Joshua Blake. *Natural User Interfaces in .NET*. Manning, 2011.

[7] Eva Hornecker and Jacob Buur. Getting a grip on tangible interaction: A framework on physical space and social interaction. In ACM Press, editor, *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI)*, pages 437–446, 2006.

[8] Alessio Levratti, Antonio De Vuono, Cesare Fantuzzi, and Cristian Secchi. TIREBOT: A novel tire workshop assistant robot. *2016 IEEE Int. Conf. on Advanced Intelligent Mechatronics (AIM)*, 2016.

[9] Stefan Waldherr, Roseli Romero, and Sebastian Thrun. A gesture based interface for human-robot interaction. *Autonomous Robot*, 9:151–173, 2000.

[10] A. De Luca and F. Flacco. Integrated control for pHRI: collision avoidance, detection, reaction and collaboration. *4th IEEE Int. Conf. Biomedical Robotics and Biomechatronics*, pages 288–295, 2012.

[11] SHERPA, EU collaborative project ICT-6000958.

[12] Valeria Villani, Lorenzo Sabattini, Nicola Battilani, and Cesare Fantuzzi. Smartwatch-enhanced interaction with an advanced troubleshooting system for industrial machines. *IFAC-PapersOnLine*, 49(19):277–282, 2016.

[13] Andrea Sanna, Fabrizio Lamberti, Gianluca Paravati, and Federico Manuri. A kinect-based natural interface for quadrotor control. *Entairtnament Computing*, 4:179–186, 2013.

[14] Jagdish Lal Raheja, Radhey Shyam, G. Arun Rajsekhar, and P. Bhanu Prasad. *Real-Time Robotic Hand Control Using Hand Gestures*, chapter 20, pages 413–428. InTech, 2012.

[15] Harish Kumar Kaura, Vipul Honrao, Patil Sayali, and Pravish Shetty. Gesture controlled robot using image processing. *Int. J. Adv. Res. Artif. Intell.*, 2(5):69–77, 2013.

[16] Holger Junker, Oliver Amft, Paul Lukowicz, and Gerhard Tröstera. Gesture spotting with body-worn inertial sensors to detect user activities. *Pattern Recognition*, 41:2010–2024, 2008.

[17] Daniel Roggen, Stéphane Magnenat, Markus Waibel, and Gerhard Tröster. Wearable computing. *IEEE Robot. Automat. Mag.*, 18(2):83–95, 2011.

[18] Sushmita Mitra and Tinku Acharya. Gesture recognition: A survey. *IEEE Tran. Syst. Man Cybern. C, pl. Rev.*, 37(3):311–324, 2007.

[19] Hajime Uchiyama, Michael A Covington, and Walter D Potter. Vibrotactile glove guidance for semi-autonomous wheelchair operations. In *Proceedings of the 46th ACM Southeast Conference (ACMSE)*, pages 336–339. ACM, 2008.

[20] Ahmad Akl and Shahrokh Valaee. Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, & compressive sensing. In *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 2270–2273. IEEE, 2010.

[21] Mridul Khan, Sheikh Iqbal Ahamed, Miftahur Rahman, and Ji-Jiang Yang. Gesthaar: An accelerometer-based gesture recognition method and its application in nui driven pervasive healthcare. In *Proceedings of the IEEE International Conference on Emerging Signal Processing Applications (ESPA)*, pages 163–166. IEEE, 2012.

[22] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[23] Jonathan Cacace, Finzi Alberto, and Vincenzo Lippiello. Multimodal interaction with multiple co-located drones in search and rescue missions. *arXiv:1605.07316*, 2016.

[24] A. William Evans, Jeremy P. Gray, Dave Rudnick, and Robert E. Karlsen. Control solutions for robots using Android and iOS devices. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, 2012.

[25] Mark Micire, Munjal Desai, Amanda Courtemanche, Katherine M. Tsui, and Holly A. Yanco. Analysis of natural gestures for controlling robot teams on multi-touch tabletop surfaces. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 41–48, 2009.

[26] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The Int. J. of Robotics Research*, 31(5):664–674, 2012.

[27] Donald A. Norman. Natural user interfaces are not natural. *Interactions*, 17(3):6–10, 2010.

[28] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[29] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Prentice Hall, 3rd edition, 2008.

[30] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing. Principles, Algorithms, and Applications.* Prentice-Hall International, Inc., 3rd edition, 1996.

[31] Hisashi Kobayashi, Brian L. Mark, and William Turin. *Probability, Random Processes, and Statistical Analysis.* Cambridge University Press, 2012.

[32] Ron Kohavi and Foster Provost. Glossary of terms. *Machine Learning*, 30(2-3):271–274, 1998.

[33] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source robot operating system. In *Proc. ICRA Workshop on Open Source Software*, volume 3, page 5, 2009.

[34] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Scale-aware navigation of a low-cost quadrocopter with a monocular camera. *Robotics and Autonomous Systems*, 62(11):1646–1656, 2014.

[35] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi. Ensemble coordination approach in multi-agv systems applied to industrial warehouses. *IEEE Trans. Automation Sci. and Eng.*, 12(3):922–934, jul. 2015.

[36] Ehus Sharlin, Benjamin Watson, Yoshifumi Kitamura, Fumio Kishino, and Yuichi Itoh. On tangible user interfaces, humans and spatality. *Personal and Ubiquitous Computing*, 8(5):338–346, 2004.

[37] Valeria Villani, Lorenzo Sabattini, Giuseppe Riggio, Cristian Secchi, Marco Minelli, and Cesare Fantuzzi. A natural infrastructure-less human-robot interaction system. *IEEE Robot. Automat. Lett.*, 2(3):1640–1647, 2017.

[38] Guy Campion and Woojin Chung. Wheeled robots. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 391–410. Springer, 2008.

[39] V. Villani, F. Pini, F. Leali, and C. Secchi. Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 2018. In press.

[40] S. Li, J. Zhang, L. Xue, F. J. Kim, and X. Zhang. Attention-aware robotic laparoscope for human-robot cooperative surgery. In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 792–797, Dec 2013.

[41] Federica Ferraguti, Nicola Preda, Marcello Bonfe, and Cristian Secchi. Bilateral teleoperation of a dual arms surgical robot with passive virtual fixtures generation. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 4223–4228. IEEE, 2015.

[42] Federica Ferraguti, Nicola Preda, Auralius Manurung, Marcello Bonfe, Olivier Lambercy, Roger Gassert, Riccardo Muradore, Paolo Fiorini, and Cristian Secchi. An energy tank-based interactive control architecture for autonomous and teleoperated robotic surgery. *Robotics, IEEE Transactions on*, 31(5):1073–1088, 2015.

[43] E. J. Rodriguez-Seda, J. J. Troy, C. A. Erignac, P. Murray, D. M. Stipanovic, and M. W. Spong. Bilateral teleoperation of multiple mobile agents: Coordinated motion and collision avoidance. *IEEE Transactions on Control Systems Technology*, 18(4):984–992, 2009.

[44] A Franchi, C Secchi, M Ryll, HH Bülthoff, and P Robuffo Giordano. Bilateral shared control of multiple quadrotors: Balancing autonomy and human assistance with a group of uavs. *IEEE Robotics & Automation Magazine*, 2012.

[45] Antonio Franchi, Cristian Secchi, Hyoung Il Son, Heinrich H Bülthoff, and Paolo Robuffo Giordano. Bilateral teleoperation of groups of mobile robots with time-varying topology. *Robotics, IEEE Transactions on*, 28(5):1019–1033, 2012.

[46] L. Rozo, D. Bruno, S. Calinon, and D. G. Caldwell. Learning optimal controllers in human-robot cooperative transportation tasks with position and force constraints. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1024–1030, Sept 2015.

[47] A.M. Zanchettin, N.M. Ceriani, P. Rocco, H. Ding, and B. Matthias. Safety in human-robot collaborative manufacturing environments: Metrics and control.

Automation Science and Engineering, IEEE Transactions on, PP(99):1–12, 2015.

[48] J. Marvel and R. Bostelman. Towards mobile manipulator safety standards. In *Robotic and Sensors Environments (ROSE), 2013 IEEE International Symposium on*, pages 31–36, Oct 2013.

[49] European Community. Robots and robotic devices – collaborative robots.

[50] J. Kinugawa, Y. Kawaai, Y. Sugahara, and K. Kosuge. Pady: Human-friendly/cooperative working support robot for production site. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5472–5479, Oct 2010.

[51] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[52] J. Amiryan and M. Jamzad. Adaptive motion planning with artificial potential fields using a prior path. In *Robotics and Mechatronics (ICROM), 2015 3rd RSI International Conference on*, pages 731–736, Oct 2015.

[53] Bakir Lacevic, Paolo Rocco, and Andrea Maria Zanchettin. Safety assessment and control of robotic manipulators using danger field. *Robotics, IEEE Transactions on*, 29(5):1257–1270, 2013.

[54] Alessio Levratti, Antonio De Vuono, Cesare Fantuzzi, and Cristian Secchi. TIREBOT: a Novel Tire Workshop Assistant Robot. In *Advanced Intelligent Mechatronics (AIM), 2016 IEEE International Conference on*, 2016.

[55] A. Levratti, G. Riggio, A. De Vuono, C. Fantuzzi, and C. Secchi. Safe navigation and experimental evaluation of a novel tire workshop assistant robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2017*, Jun 2017.

[56] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[57] 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.

[58] Loren Arthur Schwarz, Artashes Mkhitaryan, Diana Mateus, and Nassir Navab. Human skeleton tracking from depth data using geodesic distances and optical flow, 2012. Best of Automatic Face and Gesture Recognition 2011.

[59] C. Secchi, S. Stramigioli, and C. Fantuzzi. *Control of Interactive Robotic Interfaces: a port-Hamiltonian Approach*. Springer, 2007.

[60] F. Ferraguti, N. Preda, A. Manurung, M. Bonfè, O. Lambercy, R. Gassert, R. Muradore, P. Fiorini, and C. Secchi. An energy tank-based interactive control architecture for autonomous and teleoperated robotic surgery. *IEEE Transactions on Robotics*, 31(5):1073–1088, Oct 2015.

[61] Andrea Maria Zanchettin, Bakir Lacevic, and Paolo Rocco. Passivity-based control of robotic manipulators for safe cooperation with humans. *International Journal of Control*, 88(2):429–439, 2015.

[62] David J Mulla. Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps. *Biosystems engineering*, 114(4):358–371, 2013.

[63] Peter R Clingeleffer and Mark Krstic. *Crop Development, Crop Estimation and Crop Control to Secure Quality and Production of Major Wine Grape Varieties: A National Approach: Final Report to Grape and Wine Research & Development Corporation*. Grape and Wine Research & Development Corporation, 2001.

[64] P Sabbatini, I Dami, and GS Howell. Predicting harvest yield in juice and wine grape vineyards. *Extension Bulletin*, 3186, 2012.

[65] Paul E Blom and Julie M Tarara. Trellis tension monitoring improves yield estimation in vineyards. *HortScience*, 44(3):678–685, 2009.

[66] Ana Djuricic, Martin Weinmann, and Boris Jutzi. Potentials of small, lightweight and low cost multi-echo laser scanners for detecting grape berries. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(5):211, 2014.

[67] José Antonio Martínez Casasnovas and Xavier Bordes Aymerich. Viticultura de precisión: Predicción de cosecha a partir de variables del cultivo e índices de vegetación. *Revista de teledetección, 2005, vol. 24, p. 67-71*, 2005.

[68] Gregory M Dunn and Stephen R Martin. Yield prediction from digital image analysis: A technique with potential for vineyard assessments prior to harvest. *Australian Journal of Grape and Wine Research*, 10(3):196–198, 2004.

[69] Maria-Paz Diago, Christian Correa, Borja Millán, Pilar Barreiro, Constantino Valero, and Javier Tardaguila. Grapevine yield and leaf area estimation using supervised classification methodology on rgb images taken under field conditions. *Sensors*, 12(12):16988–17006, 2012.

[70] R Chamelat, E Rosso, A Choksuriwong, C Rosenberger, H Laurent, and P Bro. Grape detection by image processing. In *IEEE Industrial Electronics, IECON 2006-32nd Annual Conference on*, pages 3697–3702. IEEE, 2006.

[71] Mathieu Grossetete, Yannick Berthoumieu, Jean-Pierre Da Costa, Christian Germain, Olivier Lavialle, Gilbert Grenier, et al. Early estimation of vineyard yield: site specific counting of berries by using a smartphone. In *International Conference of Agricultural Engineering-CIGR-AgEng*, 2012.

[72] Stephen Nuske, Kyle Wilshusen, Supreeth Achar, Luke Yoder, Srinivasa Narasimhan, and Sanjiv Singh. Automated visual yield estimation in vineyards. *Journal of Field Robotics*, 31(5):837–860, 2014.

[73] Christophe Cariou, Roland Lenain, Benoit Thuilot, and Michel Berducat. Automatic guidance of a four-wheel-steering mobile robot for accurate field operations. *Journal of Field Robotics*, 26(6-7):504–518, 2009.

[74] Mostafa Sharifi and XiaoQi Chen. A novel vision based row guidance approach for navigation of agricultural mobile robots in orchards. In *Automation, Robotics and Applications (ICARA), 2015 6th International Conference on*, pages 251–255. IEEE, 2015.

[75] Jamie Bell, Bruce A MacDonald, and Ho Seok Ahn. Row following in pergola structured orchards. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 640–645. IEEE, 2016.

[76] Oscar C Barawid, Akira Mizushima, Kazunobu Ishii, and Noboru Noguchi. Development of an autonomous navigation system using a two-dimensional laser scanner in an orchard application. *Biosystems Engineering*, 96(2):139–149, 2007.

[77] Marcel Bergerman, Silvio M Maeta, Ji Zhang, Gustavo M Freitas, Bradley Hamner, Sanjiv Singh, and George Kantor. Robot farmers: Autonomous orchard vehicles help tree fruit production. *IEEE Robotics & Automation Magazine*, 22(1):54–63, 2015.

[78] Stephen Nuske, Supreeth Achar, Terry Bates, Srinivasa Narasimhan, and Sanjiv Singh. Yield estimation in vineyards by visual grape detection. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2352–2358. IEEE, 2011.

[79] Gareth Loy and Alexander Zelinsky. Fast radial symmetry for detecting points of interest. *IEEE Transactions on pattern analysis and machine intelligence*, 25(8):959–973, 2003.

[80] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[81] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[82] Zhao Zhang, Paul H Heinemann, Jude Liu, Tara A Baugher, and James R Schupp. The development of mechanical apple harvesting technology: A review. *Transactions of the ASABE*, 59(5):1165–1180, 2016.

[83] Fadi A Fathallah. Musculoskeletal disorders in labor-intensive agriculture. *Applied ergonomics*, 41(6):738–743, 2010.

[84] G Warner. Tighter borders may be keeping out workers. *Good Fruit Grower*, 54(11):10–11, 2003.

[85] Timothy J Richards. Immigration Reform and Farm Labor Markets. *American Journal of Agricultural Economics*, 100(4):1050–1071, 06 2018.

[86] Caroline Nye. The blind spot of agricultural research: Labour flexibility, composition and worker availability in the South West of England. *Cahiers Agricultures*, 27(3):35002, 2018.

[87] Naoshi Kondo, Yoshiaki Nishitsuji, Peter P Ling, and K Chong Ting. Visual feedback guided robotic cherry tomato harvesting. *Transactions of the ASAE*, 39(6):2331–2338, 1996.

[88] Shigehiko Hayashi, Kenta Shigematsu, Satoshi Yamamoto, Ken Kobayashi, Yasushi Kohno, Junzo Kamata, and Mitsutaka Kurita. Evaluation of a strawberry-harvesting robot in a field test. *Biosystems engineering*, 105(2):160–171, 2010.

[89] C Wouter Bac, Jochen Hemming, BAJ van Tuijl, Ruud Barth, Ehud Wais, and Eldert J van Henten. Performance evaluation of a harvesting robot for sweet pepper. *Journal of Field Robotics*, 34(6):1123–1139, 2017.

[90] Johan Baeten, Kevin Donné, Sven Boedrij, Wim Beckers, and Eric Claesen. Autonomous fruit picking machine: A robotic apple harvester. In *Field and service robotics*, pages 531–539. Springer, 2008.

[91] Zhao De-An, Lv Jidong, Ji Wei, Zhang Ying, and Chen Yu. Design and control of an apple harvesting robot. *Biosystems engineering*, 110(2):112–122, 2011.

[92] Abhisesh Silwal, Joseph R Davidson, Manoj Karkee, Changki Mo, Qin Zhang, and Karen Lewis. Design, integration, and field evaluation of a robotic apple harvester. *Journal of Field Robotics*, 34(6):1140–1159, 2017.

[93] C Wouter Bac, Eldert J van Henten, Jochen Hemming, and Yael Edan. Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *Journal of Field Robotics*, 31(6):888–911, 2014.

[94] Yael Edan, Dima Rogozin, Tamar Flash, and Gaines E Miles. Robotic melon harvesting. *IEEE Transactions on Robotics and Automation*, 16(6):831–835, 2000.

[95] Tom Botterill, Scott Paulin, Richard Green, Samuel Williams, Jessica Lin, Valerie Saxton, Steven Mills, XiaoQi Chen, and Sam Corbett-Davies. A robot system for pruning grape vines. *Journal of Field Robotics*, 34(6):1100–1122, 2017.

[96] Florian Hauer and Panagiotis Tsiotras. Deformable rapidly-exploring random trees. In *In Proc. of RSS*, 2017.

[97] Pablo Ramon Soria, Fouad Sukkar, Wolfram Martens, Begoña C Arrue, and Robert Fitch. Multi-view probabilistic segmentation of pome fruit with a low-cost rgb-d camera. In *Iberian Robotics conference*, pages 320–331. Springer, 2017.

[98] Wolfram Martens, Yannick Poffet, Pablo Ramón Soria, Robert Fitch, and Salah Sukkarieh. Geometric priors for gaussian process implicit surfaces. *IEEE Robotics and Automation Letters*, 2(2):373–380, 2017.

[99] Fouad Sukkar. Fast, reliable and efficient database search motion planner (FREDS-MP) for repetitive manipulator tasks. Master's thesis, University of Technology Sydney, 2018.

[100] François Chaumette and Seth Hutchinson. Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

[101] Youcef Mezouar and François Chaumette. Path planning for robust image-based control. *IEEE Transactions on Robotics and Automation*, 18(4):534–549, 2002.

[102] Fouad Sukkar, Graeme Best, Chanyeol Yoo, and Robert Fitch. Multi-robot region-of-interest reconstruction with dec-mcts. In *Proc. of IEEE ICRA*, 2019.

[103] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Proc. of RSS*, 2013.

[104] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. CHOMP: Covariant Hamiltonian optimization for motion planning. *Int. J. Robot. Res.*, 32(9-10):1164–1193, 2013.

[105] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *Proc. of IEEE ICRA*, pages 3067–3074, 2015.

[106] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[107] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985.

[108] Jihong Lee. A study on the manipulability measures for robot manipulators. In *In Proc. of IEEE/RSJ IROS*, volume 3, pages 1458–1465. IEEE, 1997.

[109] Charles A Klein and Bruce E Blaho. Dexterity measures for the design and control of kinematically redundant manipulators. *The International Journal of Robotics Research*, 6(2):72–83, 1987.

[110] Rosen Diankov and James Kuffner. Openrave: A planning architecture for autonomous robotics. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, 79, 2008.

[111] G. Niemeyer, C. Preusche, S. Stramigioli, and D. Lee. Telerobotics. In *Handbook of Robotics*, pages 1085–1108. Springer, 2016.

[112] G. Niemeyer and J. J. E. Slotine. Stable adaptive teleoperation. *IEEE Journal of Oceanic Engineering*, 16(1):152–162, Jan 1991.

[113] C. Secchi, F. Ferraguti, and C. Fantuzzi. Catching the wave: A transparency oriented wave based teleoperation architecture. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2422–2427, May 2016.

[114] L. Sabattini, C. Secchi, M. Cocetti, A. Levratti, and C. Fantuzzi. Implementation of coordinated complex dynamic behaviors in multirobot systems. *IEEE Transactions on Robotics*, 31(4):1018–1032, Aug 2015.

[115] N. E. Leonard and E. Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. In *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, volume 3, pages 2968–2973 vol.3, 2001.

[116] A. Franchi, C. Secchi, H. I. Son, H. H. Bulthoff, and P. Robuffo Giordano. Bilateral teleoperation of groups of mobile robots with time-varying topology. *IEEE Transactions on Robotics*, 28(5):1019–1033, Oct 2012.

[117] L. Sabattini, N. Chopra, and C. Secchi. Decentralized connectivity maintenance for cooperative control of mobile robotic systems. *International Journal of Robotics Research*, 32(12):1411–1423, 2013.

[118] P. Robuffo Giordano, A. Franchi, C. Secchi, and H.H. Buelthoff. A passivity-based decentralized strategy for generalized connectivity maintenance. *International Journal of Robotics Research*, 32(3):299–323, 2013.

[119] M. Arcak. Passivity as a design tool for group coordination. *IEEE Transactions on Automatic Control*, 52(8):1380–1390, Aug 2007.

[120] M. Angerer, S. Music, and S. Hirche. Port-hamiltonian based control for human-robot team interaction. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2292–2299, May 2017.

[121] C. Yang, G. Ganesh, S. Haddadin, S. Parusel, A. Albu-Schäffer, and E. Burdet. Human-like adaptation of force and impedance in stable and unstable interactions. *IEEE Transactions on Robotics*, 27(5):918–930, 2011.

[122] M. T. J. Spaan, T. S. Veiga, and P. U. Lima. Active cooperative perception in network robot systems using pomdps. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4800–4805, Oct 2010.

[123] M. Schwager, M. P. Vitus, S. Powers, D. Rus, and C. J. Tomlin. Robust adaptive coverage control for robotic sensor networks. *IEEE Transactions on Control of Network Systems*, 4(3):462–476, Sept 2017.

[124] M. Franken, S. Stramigioli, S. Misra, C. Secchi, and A. Macchelli. Bilateral telemanipulation with time delays: A two-layer approach combining passivity and transparency. *IEEE Transactions on Robotics*, 27(4):741–756, 2011.

[125] D. Heck, A. Saccon, R. Beerens, and H. Nijmeijer. Direct force-reflecting two-layer approach for passive bilateral teleoperation with time delays. *IEEE Transactions on Robotics*, 34(1):194–206, Feb 2018.

[126] L. Sabattini, C. Secchi, and C. Fantuzzi. Achieving the desired dynamic behavior in multi-robot systems interacting with the environment. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2097–2102, May 2017.

[127] E. Shahriari, A. Kramberger, A. Gams, A. Ude, and S. Haddadin. Adapting to contacts: Energy tanks and task energy for passivity-based dynamic movement primitives. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 136–142, Nov 2017.

[128] A. Dietrich, C. Ott, and S. Stramigioli. Passivation of projection-based null space compliance control via energy tanks. *IEEE Robotics and Automation Letters*, 1(1):184–191, Jan 2016.

[129] C. Talignani Landi, F. Ferraguti, L. Sabattini, C. Secchi, M. Bonfè, and C. Fantuzzi. Variable admittance control preventing undesired oscillating behaviors in physical human-robot interaction. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3611–3616, Sept 2017.

[130] B. Brogliato, R. Lozano, B. Maschke, and O. Egeland. *Dissipative Systems Analysis and Control*. Springer, 2007.

[131] Pramila Rani, Nilanjan Sarkar, Craig A. Smith, and Leslie D. Kirby. Anxiety detecting robotic system -towards implicit human-robot collaboration. *Robotica*, 22(1):85–95, 2004.