University of Modena and Reggio Emilia
Department of Sciences and Methods for Engineering

Doctoral School
Industrial Innovation Engineering

Planar Pushing: a study of non-prehensile manipulation with single and multiple mobile robots

Filippo Bertoncelli

PhD Course Coordinator:
Prof. Franco Zambonelli

Supervisor:
Prof. Lorenzo Sabattini

XXXIV Cycle

"*The world's not perfect, but it's there for us trying the best it can. That's what makes it so damn beautiful.*"

Roy Mustang (Fullmetal Alchemist)

"*Life's a bore if you don't challenge yourself.*"

Yu Nishinoya (Haikyuu)

Abstract

This doctoral dissertation studies the manipulation of objects using pushing: a manipulation technique from the class of non-prehensile manipulations. Techniques from this class exploit the geometry of the object together with its dynamics and the surrounding environment to achieve the task at hand. Starting from present literature, the first problem approached was the generation of an optimal pushing plan for manipulating a polygonal object using pushing actions on its sides by a single mobile robot. The proposed solution is a two-layer planning algorithm. The first layer evaluates the feasibility of the manipulation by computing a path to the target pose of the object, considering the dimentions of both robot and object. This initial path is then used as a guide to search for a set of pushing trajectories for the robot. These trajectories are obtained through an optimization problem that generates the shortest possible trajectory that satisfies the pushing constraints. The next step is to generate a control algorithm that ensures that the motion of the robot during the manipulation satisfies the pushing constraints. This is achieved using Model Predictive Control. The controller makes use of a model of the system to predict the future state and optimize its behaviour. With MPC it is also possible to impose a set of constraints on the system motion. In particular, the interest here is imposing a constraint to maintain a sticking contact between the object and the robot during the manipulation. This is achieved at first by including the contact forces in the inputs of the system in order to predict the motion of the pushed object. A constraint is then formulated to ensure that the motion of the robot and the object are compatible. However, the prediction of the motion of a pushed object is often unreliable due to the indeterminacy of the factors involved, and the formulation of the constraint is nonlinear. This increases the complexity of the solution of the optimization algorithm and therefore its time requirements. An improved version of the control algorithm was then designed transforming the constraint into an equivalent linear formulation, easier to compute. The last component necessary for a single robot to perform a manipulation by pushing is a monitoring algorithm. The proposed design defines a finite state machine for the execution of the pushing trajectories, monitoring the manipulation and handling edge cases and possible failures. The study of pushing manipulation with multi-robot system was split in two research directions, differentiated by the size of the robot group. When only a few robots are available to complete the assigned planar pushing task, it is necessary to carefully displace them around the object according to the assigned task. A characterization of the quality of a grasp configuration was defined to aid this decision. Its validity has been studied through statistical analysis from simulation data. The iterative optimization of a given configuration for a task was also studied, therefore proposing a procedure to optimize a pushing configuration. The proposed optimization makes use of the characterization as well as other forms of estimation of the quality of a configuration. When the number of robots is higher and it may be necessary to have the robots push on each other to combine the efforts, a different approach is necessary. A hierarchical control law was then proposed to address these cases. The control law makes use of established control schemes to coordinate the group of robots as if they were a probability distribution, therefore capable to adapt to any object shape and size. Experiments were carried out to validate the strategy on different objects and different numbers of robots.

Sommario

Questa tesi di dottorato studia la manipolazione di oggetti mediante la spinta: una tecnica di manipolazione appartenente alla classe delle manipolazioni non prensili. Le tecniche di questa classe sfruttano la geometria dell'oggetto da manipolare insieme alla sua dinamica e all'ambiente circostante per portare l'oggetto nella posizione desiderata. Partendo dalla letteratura attuale, il primo problema affrontato è la generazione di un piano di spinta ottimale per manipolare un oggetto poligonale tramite azioni di spinta sui suoi lati da parte di un singolo robot mobile. La soluzione proposta è un algoritmo di pianificazione a due livelli. Il primo livello valuta la fattibilità della manipolazione calcolando un percorso verso la posa target dell'oggetto, considerando sia le dimensioni del robot che dell'oggetto. Questo percorso iniziale viene quindi utilizzato come guida per la ricerca di una serie di traiettorie di spinta per il robot. Queste traiettorie sono ottenute attraverso un problema di ottimizzazione che genera la traiettoria più corta possibile che soddisfi i vincoli di spinta. Il passo successivo è generare un algoritmo di controllo che assicuri che il movimento del robot durante la manipolazione soddisfi i vincoli di spinta. Ciò si ottiene utilizzando il controllo predittivo. Il controllore di tipo Model Predictive Control (MPC) utilizza un modello del sistema per prevedere lo stato futuro e ottimizzarne il comportamento. Con MPC è anche possibile imporre dei vincoli al movimento del sistema. In particolare, si è imposto un vincolo per mantenere un contatto stabile tra l'oggetto e il robot durante la manipolazione. Ciò si ottiene inizialmente includendo le forze di contatto negli ingressi del sistema per prevedere il movimento dell'oggetto spinto. Tuttavia, siccome la previsione del moto di un oggetto spinto è spesso inaffidabile a causa dell'indeterminatezza di parametri come l'attrito, e la formulazione non lineare del vincolo, la complessità della soluzione dell'ottimizzazione risulta alta. È stata quindi progettata una versione migliorata dell'algoritmo di controllo trasformando il vincolo in una formulazione lineare equivalente, più facile da calcolare. Affinché un singolo robot esegua una manipolazione spingendo è necessario un algoritmo di monitoraggio. La soluzione proposta è la definizione di una macchina a stati finiti per l'esecuzione delle traiettorie di spinta, il monitoraggio della manipolazione e la gestione dei casi limite e dei possibili guasti. Lo studio della manipolazione della spinta con il sistema multi-robot è stato suddiviso in due direzioni di ricerca. Quando sono disponibili solo pochi robot per completare l'attività di spinta planare assegnata, è necessario disporli con attenzione attorno all'oggetto in base all'attività assegnata. Una caratterizzazione della qualità di una configurazione di contatto è stata definita per aiutare questa decisione, validata con metodi statistici. È stata studiata anche l'ottimizzazione iterativa di una data configurazione per un task, proponendo quindi una procedura per ottimizzare una configurazione. L'ottimizzazione proposta si avvale della caratterizzazione e di altre forme di stima della qualità di una configurazione. Quando il numero di robot è maggiore la complessità del problema aumenta. Pertanto è necessario un approccio diverso. È stata quindi proposta una legge di controllo gerarchico per affrontare questi casi. La legge di controllo si avvale di schemi di controllo consolidati per coordinare il gruppo di robot secondo una distribuzione di probabilità, definita in base alla spinta da applicare. Sono stati condotti esperimenti per validare la strategia su diverse forme e utilizzando numeri di robot differenti.

Acknowledgements

I would like to thank my family for their unconditional support. Without them I would not have been able to complete this journey.

I am deeply thankful to my advisor, Prof. Lorenzo Sabattini for his great help, for being welcoming to every discussion topic and especially for his patience. I am thankful to all the people that I had the chance to work with at ARSControl Lab, I could not have wished for better colleagues.

Last but not least, I would like to thank every person that has helped me during this journey.

This opportunity has been a roller coaster of emotions but I would not exchange it for anything in the world.

Contents

List of Figures

List of Tables

Acronyms

MPC	Model Predictive Control.
RRT*	Rapidly-exploring Random Tree Star.
NMPC	Nonlinear Model Predictive Control.
NLP	Non-Linear Programming.
LTVMPC	Linear Time-Varying Model Predictive Control.
MHD	Modified Hausdorff Distance.

Dedicated to my family and every
person who believed in me....

Chapter 1

Introduction

Robots have grown to be an essential part of modern life. In industrial scenarios, robots have been used extensively to fulfill tasks that required repetitive and precise movements or the interaction with heavy payloads. The sensing capabilities of field robots have been used to explore environments that were previously impossible to access due to extreme conditions, like the extreme pressure of ocean floors or the lack of a breathable atmosphere on other planets like Mars.

The purpose of any robot can be summarized to *reducing human effort while increasing efficiency.* Industrial robots operating inside factories are a prime example of this, being designed to fulfill a specific task, repeatedly and efficiently, making human job easier. A common trait can be identified among many different applications: a robot needs to interact with the environment around it. This interaction can take various forms, from sensing the many different characteristics of the surrounding environment to modifying it through manipulation.

In this thesis, we consider a particular class of interactions that a robot can apply on its surroundings named *pushing.* Pushing is a manipulation primitive that belongs to the broader category of non-prehensile manipulation [2]. The most traditional and used way for a robot to modify its surroundings is through prehensile manipulation techniques where the object that has to be manipulated is strictly constrained to move accordingly to the robot's motion [3]. A firm grasp on the object can be obtained by very different means, a mechanical gripper, a robotic hand, or a suction end-effector. This mechanical connection between the robot and the *manipulandum* often represents the weakest link of the kinematic chain. Robotic hands and grippers are generally prone to malfunction, and their repair can be expensive. Non-prehensile manipulation primitives offer a potential solution to avoid this problem. The main characteristic of this class of manipulation primitives is the absence of a fixed element constraining the object with the robot motion. A few examples of these primitives have been identified in the literature [2]. *Throwing* an object requires particular care into setting the dynamic state of the manipulated object at the beginning of the throw so that the resulting motion matches the desired trajectory to reach the final outcome. *Catching* an object, on the other hand, requires accurate detection of its dynamic state so that the robot can match its trajectory till contact is established. *Batting* combines these two primitives in a single collision since hitting an object into a desired final

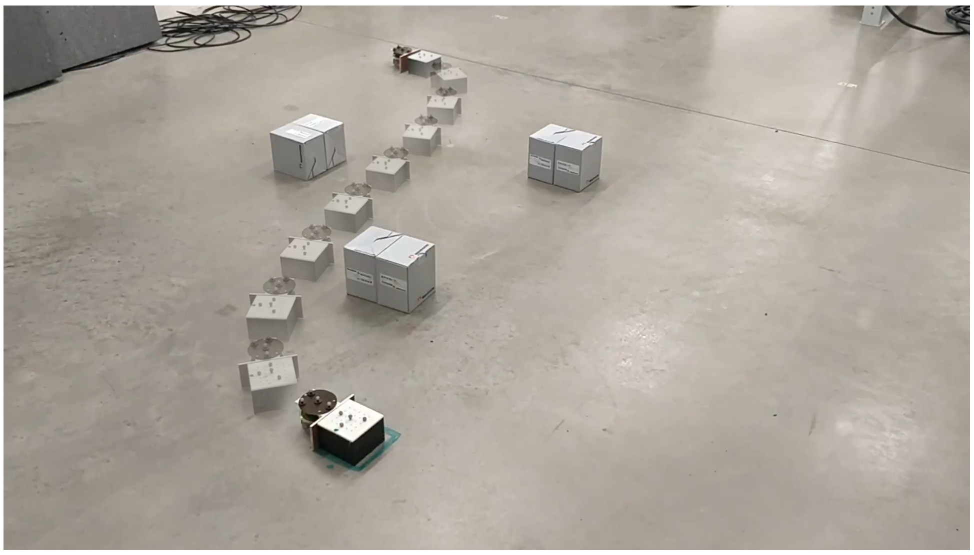


FIGURE 1.1: e-puck differential drive mobile robot pushing a square box through three obstacles.

position requires computing the outcome of the impact between the robot and the object, knowing its incoming trajectory, as well as understanding the dynamics of the impact. In the *sliding* primitive, the object is set in motion onto the surface of the manipulator by a combination of frictional forces and vibrations of the surface. As the robot vibrates or oscillates, a created virtual force field controls the direction of motion. Whenever an object is purposely set to roll on the surface of the manipulator, then it is possible to talk about *rolling* manipulation. If only pure rolling is considered, the motion constraints are holonomic when the environment is two-dimensional (2D) while, for three-dimensional (3D) environments, the constraints turn non-holonomic. Lastly, the *pushing* primitive, which is the manipulation primitive considered in this thesis, is a widely used solution by humans when the object to be manipulated is too large or too heavy to be grasped. The manipulator applies a force to cause a motion of the object towards the desired configuration. The pushing primitive is closely related to sliding since the resulting motion is often the object sliding on the support surface. However, the critical difference is that, in pushing, the motion is caused by the action of the manipulator on the object. In contrast, in sliding, the motion comes directly from the object's interaction with the support surface. Intuitively, friction plays a critical role in performing this type of manipulation. The indeterminacy of the frictional parameters can cause difficulties in predicting the object's motion. Nevertheless, as will be explained in Section 2.1.1, under certain assumptions, it is possible to bypass some of these difficulties and obtain a well-performing prediction method.

In the multi-robot systems literature collaborative transportation of an object represents a classical problem. It takes inspiration from the natural world. Small animals, like ants, collaborate to transport heavy and large loads: several works can be found in the literature that try to mimic the ants' behavior to achieve collaborative transportation for groups of mobile robots [4,5]. However, this problem is often solved considering approaches in which force closure is achieved, planning the motion of the robots in such a way that they are opportunely displaced around the object [6]. As an example, a team of robots that locally exchange information is proposed in [7] to create a suitable formation around the object to transport it successfully. Along the same line, a large number of robots are attached to the object to exchange forces with it in [8]. These approaches resemble the most common solution exploited in robotics for solving the problem of moving an object: the pick-and-place method, where the object is grasped in a stiff way and is moved to the desired location. While this is a common and effective solution in several cases, it can not always be applied. For example, when the object's size is too large, when its shape is unknown a priori, when it is excessively heavy, or when its surface can be damaged by a firm grasp, as discussed in [9].

Nonprehensile manipulation approaches can thus be exploited in those cases above, as proposed in [2]. Specifically, these strategies include methods in which the robot imposes the motion to the object through unilateral constraints only, such as in the case of pushing. While nonprehensile manipulation can represent a solution in the scenarios above, its successful implementation requires to take into account the dynamic models of the robot, the object, and the environment, since the exchanged forces are of paramount importance. Historically, the mechanics of the pushing manipulation has been studied in both [9] and [10] in order to identify a model to predict the motion of a pushed object, which led to the design of planning and control algorithms for pushing operations such as in [9,11] and more recently in [12,13]. Nonprehensile manipulation has been also recently implemented, equipping the robots with flexible elements, such as ropes or cables. For instance, a robot equipped with a tail (i.e., a flexible cable) is considered in [14], where a planning method to define the motion of the robot is proposed to exploit such a tail for moving an object by pushing. While this method has proven its feasibility for simple objects to be manipulated, alternative methods are necessary for more complex scenarios. Objects with general shape can be considered by the method proposed in [15], where two mobile robots are connected with a cable, for cooperatively pulling a heavy object. The main drawback of this solution is represented by the physical interconnection between the robots (i.e., the cable), which significantly reduces the freedom of motion. Mobile robots can be controlled to avoid these issues by directly pushing the object to be manipulated, as discussed in [16]. Thus, it is necessary to guarantee that the mobile robot can move in the environment without colliding with obstacles, and change the relative position with respect to the object (i.e., change the pushing direction). This is achieved in [17], where uncertainties in both the control and the motion execution are dealt in an appropriate motion planning strategy. This last addresses an increased size of the pushed object to accommodate the repositioning maneuvers of the pushing robot. A reinforcement learning framework is proposed in [18] to define the motion pattern for two robots pushing a box in a very simplistic scenario, in which dynamics are entirely neglected. A similar case is considered in [19], where an artificial potential field is defined to let a robot, or a group of robots, push an object by simply measuring its instantaneous direction of motion. Even in this case, dynamic effects (e.g., friction) are not considered, which makes the proposed method not suitable for complex situations, such as in the presence of non-uniform friction or when more than two robots are needed. A similar problem is considered in [20], where a fuzzy controller is proposed for controlling two robots to push an object with known geometrical properties.

In [21] the authors proposed a control architecture to synchronize the motion of two mobile robots that have been tasked to transport an object leaned on top of them without dropping it. In a similar scenario, [22] makes use of a potential method to generate trajectories for omnidirectional robots carrying an object, and [23] designs a planning strategy that takes the pose of the transported object into account. A decentralized approach is instead proposed by [24] to transport an object using a consensus-based formation control law for omnidirectional mobile robots derived using Fuzzy Wavelet Neural Networks. A more general manipulation scenario is considered in [25], where the authors propose a decentralized adaptive controller for a team of robots to manipulate a common payload without exchanging information. A group of robots is used in [26] to estimate kinematic and inertial parameters of the manipulated objects in a distributed way. Multi-robot manipulation can be applied also in difficult scenarios, such as [27], where two underwater vehicles coordinate their motion through minimal exchange of information. Aerial manipulators are used in [28] to manipulate a long rod considering safety and collision avoidance at the planning level, while a controller absorbs the disturbances caused by the mutual interaction between the robots. The authors of [29] proposed a decentralized control strategy to surround the target object with a group of robots, effectively grasping the object, before transporting it to a target location.

A common characteristic of these works is the presence of a rigid connection between the manipulated object and each robot. This is a typical trait of the most common solution in robotics to the problem of moving an object: the pick-and-place method, where the object is rigidly grasped by a gripper or a robotic hand and moved to the desired location. These grasping devices often represent the weakest link in the system and are thus prone to malfunctions. To avoid this problem, several research groups started exploring the use of non-prehensile manipulation, which does not require such tools to achieve a manipulation task [9, 13, 30]. The authors of [31] proposed a leader-follower scheme that enables a pair of robots to push a cubic object following a reference trajectory. The object is considered as a virtual leader and its position is computed using an onboard object localization algorithm, while the robots are considered as followers. The work proposed in [32] describes the design and experimental validation of a distributed cooperative transportation scheme. The strategy enables a leader-less group of robots to autonomously coordinate an position around an object to transport it along a reference trajectory without the presence of a central control instance. Moreover, [1] proposes a planning and control strategy which uses a

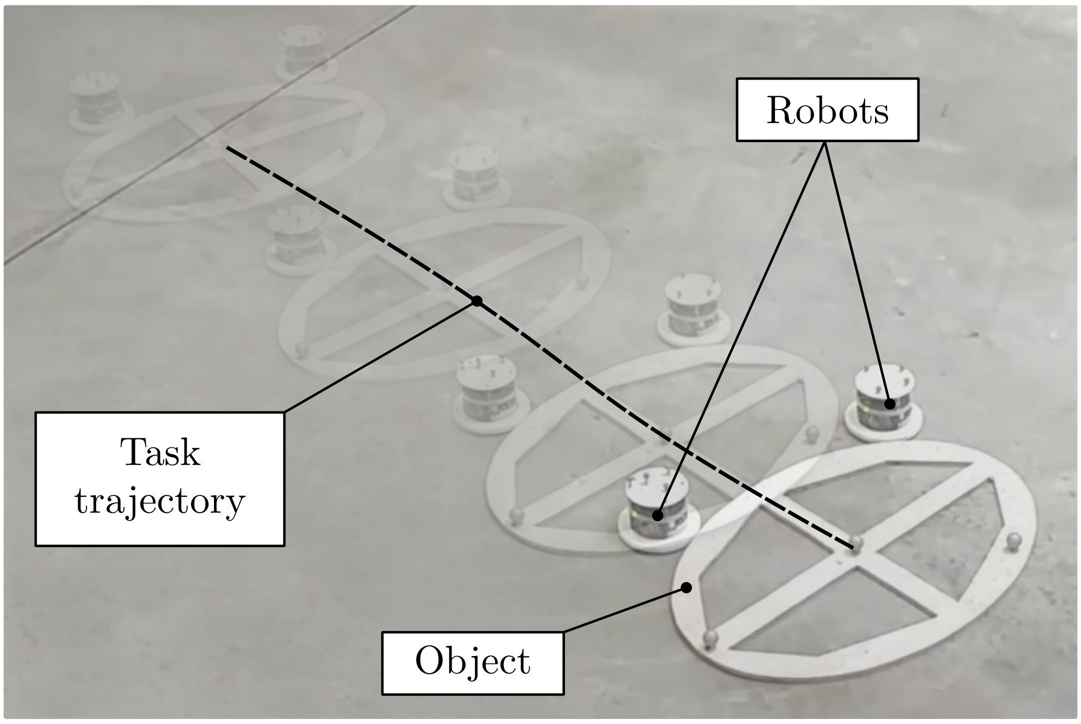


FIGURE 1.2: Illustration of the problem addressed in this work: a group of mobile robots has to manipulate an object by pushing it along a desired task. The aim is to establish optimal contact points to simultaneously minimize the tracking error and the contact forces during the task.

large number of robots to push an object into a goal location.

1.1 Contribution

On the subject of pushing with a single mobile robot, many works treated different aspects of the problem. On the planning side, the works of K. Lynch and M. Mason [9] provided a solid modeling base used by works as [33, 34] to generate a feasible manipulation plan. None of these work has however addressed the problem from the point of view of finding the optimal manipulation plan. In this work we approach this challenge using a two layer planning strategy that optimizes the length of the manipulation trajectories. On the control side, we aimed at improving the performance of pushing manipulation by implementing several controllers for the robot. Many of the works present in literature, such as [13], assume that the robot is perfectly capable of pushing the target object while one of our controllers constrains the motion of the robot so that the wheels always roll without slipping. Similarly, we also propose a controller that constrains the curvature of the robot trajectory in order to maintain a fixed contact with the pushed object. This is achieved in two scenarios, one that considers the contact forces that are transmitted between the robot and the object and one that directly constraints the robotic motion. This approach differs from those present in literature such as [13] and [35] since these works never consider robotic pushers with non-holonomic constraints such as mobile robots. Finally, all the related work always treats one or two aspects of the pushing manipulation at a time. Through the use of a supervision algorithm we finally create a completely integrated framework capable of automatically generating a manipulation plan and execute said plan using a controller. The execution is supervised and monitored to ensure proper completion of the manipulation. The framework is evaluated extensively using real robots in a testing scenario.

On the subject of pushing with multiple robots, we approached the topic from two research fronts. The first research involves a small number of robots (less than five), tasked to push an object in a planar environment. Instead of displacing the robots in a predetermined configuration and remodel the task to suit the constraints, we aim to identify the best configuration for a generic task. First we identify and test a heuristic to rank the quality of a configuration with respect to a task. The heuristic considers the forces required by a manipulation task as well as the forces that a certain configuration can apply. The result can be used to compare two configurations and identify the the one that is most suited for a certain task. On a more analytic approach, we also define an optimization problem that computes the best configuration for a task based on the pushing kinematic model. The optimization problem selects the configuration that can complete a certain task with the least amount of effort. The contact configuration obtained from either method is then used in a decentralized control algorithm to command the motion of mobile robots for pushing. The controller coordinates the robots' motions to execute the pushing manipulation while also maintaining the instructed contact configuration. The scenario is tested extensively in simulations and real world examples.

The second line of research focuses on the use of many robots. In particular, the robots are considered in an aggregated manner to provide a coordinated push to an object. The approach taken is similar to the one presented in [1], however it improves the manipulation performance by exploiting the flexibility of a multirobot system. The efficacy of the strategy is proven through simulations and real-world experiments, and the performance improvement over [1] is verified by comparing the time required by both strategies to push a unit-size object for a unit of length. The comparison shows a 50% reduction in time.

1.2 Publications

- *"Wheel slip avoidance through a nonlinear model predictive control for object pushing with a mobile robot"*, F. Bertoncelli, F. Ruggiero and L. Sabattini, Proceedings of 10th IFAC Symposium on Intelligent Autonomous Vehicles (IAV), 2019,

- *"Linear time-varying mpc for non-prehensile object manipulation with a non-holonomic mobile robot"*, F. Bertoncelli, F. Ruggiero and L. Sabattini, Proceedings of 2020 IEEE International Conference on Robotics and Automation

(ICRA), 2020,

- *"Characterization of grasp configurations for multi-robot object pushing"*, F. Bertoncelli, F. Ruggiero and L. Sabattini, Proceedings of 2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), 2021
- *"Planar pushing manipulation with a group of mobile robots"*, F. Bertoncelli and L. Sabattini, Proceedings of 20th International Conference on Advanced Robotics (ICAR), 2021,
- *"Pushing with a group of mobile robots"*, F. Bertoncelli and L. Sabattini, Proceedings of 3rd Italian Robotics and Intelligent Machines Conference (I-RIM 3D), 2021
- *"Non-prehensile Planar Manipulation: A Framework for Pushing with a Single Mobile Robot"*, F. Bertoncelli, F. Ruggiero and L. Sabattini, submitted to IEEE Transaction on Robotics, 2022,
- *"Task-Oriented Contact Optimization for Pushing Manipulation with Mobile Robots"*, F. Bertoncelli, M. Selvaggio, F. Ruggiero and L. Sabattini, submitted to 2022 International conference on Intelligent Robots and Systems (IROS), 2022

Chapter 2

Pushing Manipulation with a Single Robot

In this chapter we focus the research on the problem of pushing with a single mobile robot with the objective of designing a completely automated solution to the manipulation problem. In the planning scenario, solutions in literature only solve the problem of finding *a* solution to manipulate an object using pushing. We hereby present an attempt at finding an optimal solution. In the control scenario, we explore the application of several constraints on the robot motion in order to improve the pushing performance. Finally, through the implementation of a supervision algorithm, we provide a completely integrated framework for single robot pushing of polygonal objects. The framework is extensively tested through real world experiments.

2.1 Problem Statement and Preliminaries

Consider a planar environment with obstacles where a polygonal object has to be manipulated by a wheeled mobile robot. Let Σ_w be the global reference frame and Σ_r be the local frame attached to the center of the robot's axle. The robot's pose at time t is represented by $\chi_r(t) = \begin{bmatrix} x_r(t) & y_r(t) & \theta_r(t) \end{bmatrix}^T \in \mathbb{R}^3$ while the pose of the object to be manipulated, also identified as manipulandum, is represented as $\chi_o(t) = \begin{bmatrix} x_o(t) & y_o(t) & \theta_o(t) \end{bmatrix}^T \in \mathbb{R}^3$. Moreover, the object has a set $\mathcal{S}_P \subset \mathbb{Z}^+$ of sides on which the robot can apply a pushing action. Finally, the planar positions of the obstacles present in the environment are collected in the set $\mathcal{O}$.

This chapter is focused on providing a solution to the following problem.

Problem 1 *Transport a polygonal manipulandum from the current location* χ_o *to the desired pose* $\chi_{o_d} \in \mathbb{R}^3$ *by pushing on its sides with a single mobile robot, avoiding obstacles in the environment.*

The proposed solution to the aforementioned problem is the design of a comprehensive framework for object pushing with a single mobile robot. The framework is composed by three main components.

1. *Manipulation Planning.* An algorithm to plan the necessary actions required to push the target object to the desired location. To be considered appropriate

for the task, the algorithm needs to take into account both the presence of static obstacles and the motion constraints associated with a stable push [9]. In particular, the algorithm needs to consider the geometry of both the object and the robot in different pushing configurations to ensure that a given maneuver is feasible. Furthermore, the solution provided by the planner should be optimal with respect to both the length of the path and the time required to perform it.

2. *Manipulation Control.* A feedback control algorithm that computes the commands for the robot to perform a given task that reduces the robot's state error to zero with respect to the given reference. Furthermore, the controller should ensure the enforcement of appropriate constraints related to the problem, namely collision avoidance and pushing motion limits.

3. *Manipulation Supervision.* A high-level decision algorithm that coordinates the planning and execution of the manipulation task interacting with the controller and the planner.

The proposed framework is capable of performing the required manipulation task under the following assumptions.

- The friction conforms to Coulomb's law. The friction reaction force of a sticking contact can act in any direction tangential to the contact normal, with maximum magnitude μf_n, where $f_n \in \mathbb{R}^+$ is the magnitude of the normal contact force and $\mu \in \mathbb{R}^+$ is the friction coefficient associated with the contact. For a sliding contact, the frictional force resists the motion with a force of magnitude μf_n.
- The environment is planar. All the pushing forces lie on the plane while gravity acts downward along the vertical axis.
- The contact friction coefficient is uniform across the plane.
- The motion is slow enough that all inertial forces are negligible or instantaneously absorbed by the frictional forces. The frictional support forces always balance the pushing forces. This corresponds to the quasi-static assumption.

2.1.1 Preliminaries

The principal connotation of pushing manipulation is the limited set of generated motions due to the reduced contact interaction and lack of force closure. For example, an object that can be pushed in one direction cannot be moved in the opposite one by simply reversing the motion of the pusher. In [36], the authors presented a method to determine the possible movements of an object sliding on a plane during multiple contact pushing, focusing in particular on the motions that guarantee that the pushing contacts are sticking to the objects and not sliding or breaking. The resulting approach is independent of the exact distribution of support forces on the object from the plane. In [9], this method has been developed further by analysing the controllability of the

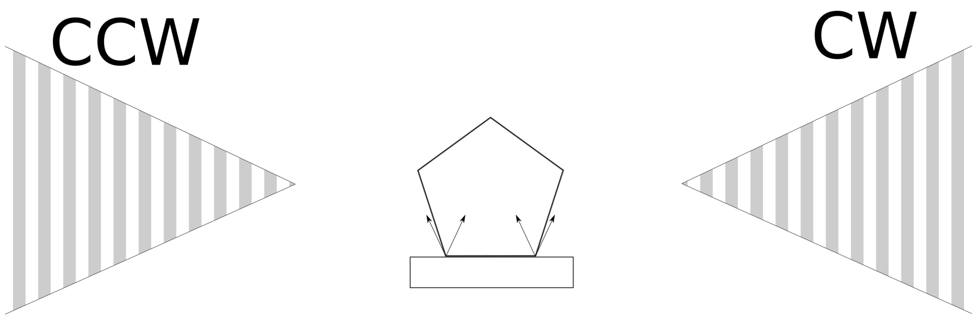


FIGURE 2.1: Feasible ICR for pushing a polygonal object. The arrows represent the borders of the friction cone. The shaded area on the left represents the counter clock-wise set of centers of rotation, while the area on the right contains the clock-wise ICRs.

pushing manipulation primitive and introducing a graphical procedure to determine an approximation of the locations of the centers of rotation deemed feasible for stable pushing. The procedure draws two sets of rotation centers, one for clock-wise rotation and one for counter clock-wise. An example of the resulting regions for a polygonal object is provided in Figure 2.1. By default, the procedure considers a holonomic pusher. To obtain the overall motion constraints acting on the pusher-slider system, it is necessary to consider the motion constraints acting on the pusher robot as well. This is obtained by intersecting the centers of rotation for stable pushing with the centers of rotation for the mobile robot. The instantaneous centers of rotation (ICR) for a differential drive mobile robot lie on the line of its axle. For the experimental case considered in this chapter, which is the planar manipulation of a square object, the intersection of the two sets is visualized in Figure 2.2. The resulting set of ICR can be mathematically translated into lower limits on the curvature radius associated with the robot's motion. The limit for the clock-wise motion curvature radius is represented by $r_{lim_-} \in \mathbb{R}^-$ while $r_{lim_+} \in \mathbb{R}^+$ represents the limit for counter clock-wise motion. This lower bound constraint is used in the planner to compute a proper pushing motion and in the controller to ensure the contact is maintained during pushing operations. The following sections will describe how these constraints have been integrated in the proposed framework.

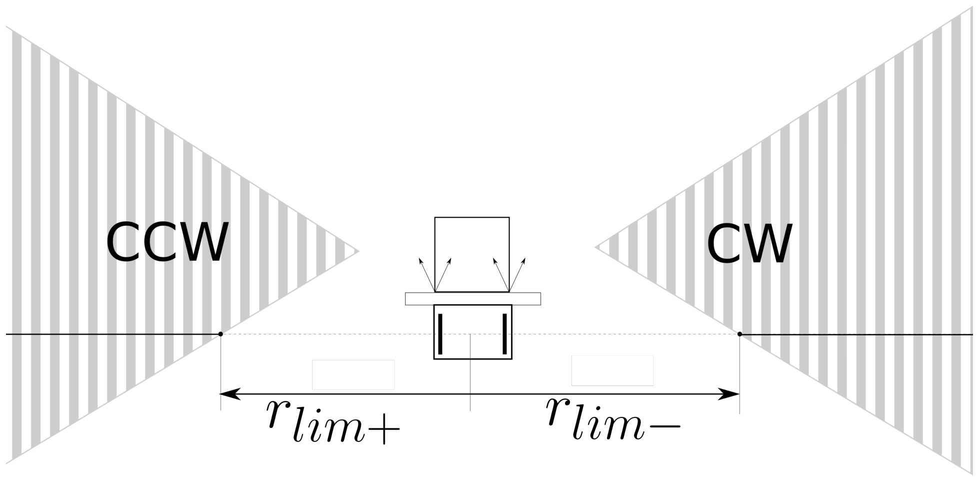


Figure 2.2: Intersection of the pushing feasible ICR with the differential drive mobile robot motion constraints.

2.2 Manipulation Planning

This section introduces the planning algorithm to compute the trajectories the robot has to follow in order to push the object towards the desired pose. The desired output from the algorithm is a list of collision free, curvature constrained trajectories for the mobile robot along with an indication of the pushing side associated with each trajectory. To achieve this output we propose a two part planning algorithm composed by a Rapidly-exploring Random Tree Star (RRT*) [37] holonomic planner and an A* search algorithm.

2.2.1 Holonomic planner

The first part of the proposed solution, the holonomic planner, computes a path for the object around the obstacles that leads to the desired object pose. This path Γ is expressed as an ordered list of object poses with the initial pose at the beginning and the desired target pose at the end. The environment is represented using an occupancy grid map. Each cell of the grid represents a portion of space and can be either free or occupied. A collision is detected when the footprint of the object intersects one of the occupied regions in the map, deeming a position invalid. The RRT* algorithm grows the random tree using valid positions starting from the initial object pose, considering the following cost function:

$$C(\Gamma) = \lambda_1 L(\Gamma) + \lambda_2 (\min_{p \in \Gamma} d(p, \mathcal{O}))^{-1} \tag{2.1}$$

where the first element of the sum is the length of the path Γ and the second element indicates the reciprocal of the minimum clearance distance of the path with respect to the obstacles in the map. The purpose of this cost function is to determine a trajectory that both minimizes the total length of the path while maximizing the minimum clearance. The coefficients $\lambda_1, \lambda_2 \in \mathbb{R}^+$ are parameters to regulate the trade-off between the two, often opposing, objectives. The resulting path Γ from the algorithm is used as a guide for the second part of the proposed planning solution.

2.2.2 Search Algorithm

The second layer of the planning algorithm computes the final manipulation plan. The plan is composed of an ordered list of reference trajectories for the robot to follow, each associated to a pushing side. Therefore, the execution algorithm, explained in Section 2.5, will know that, in order to complete the pushing manipulation, the robot has to push the object following a sequence of defined trajectories, applying the pushing action on a specific side for each trajectory. This manipulation plan is assembled through the use of a graph search algorithm, A* [38], that iteratively populates the graph with new nodes and connections until a solution is found.

The underlying idea behind this solution is to look into the path Γ obtained by the previous layer, for a sequence of key object poses to use as waypoints for the manipulation. These waypoints are used to compute the curvature-constrained trajectory for the robot to follow in order to manipulate the object through all of the waypoints, possibly changing the pushing side in order to reach the desired final position and orientation for the object.

The search algorithm is detailed in Algorithm 1. The search tree is composed of nodes associated with a configuration in a 4D space. The first 3 components represent a planar pose of the pushed object $\gamma \in \Gamma$, while the fourth component $s_p \in \mathbb{Z}+$ represents which side of the object the robot is positioned in contact with. The tree is initialized with a root node $q0$ representing the target pose for the object with a null pushing side. Each node has an associated cost f that depends on its configuration and the configurations of the nodes connecting it to the root node $q0$. The cost f of each node is composed also by an heuristic cost that estimates how far is q from the search goal. At every algorithm iteration, the algorithm selects the node q in the graph with the lowest total cost f, and generates a set of feasible successors for the node to insert in the graph, therefore the presence of cost f is necessary to guide the search algorithm towards the solution. The detailed formulation of f will be introduced later in this section. When a node associated with the initial position of the object is selected at the beginning of the iteration, the algorithm reports a solution.

In the successors generation procedure, Algorithm 2 looks iteratively through Γ for an object pose γ that can be pushed into the selected node position by having the robot follow a curvature-constrained, obstacle-free polynomial trajectory indicated by $\mathcal{P}$. If such pose is found, a node associated with it is created and inserted into the graph.

We chose a third-order polynomial trajectory $\mathcal{P}$ to connect two states, expressed as:

$$x(s) = s^3 x_f - (s-1)^3 x_i + \alpha_x s^2 (s-1) + \beta_x s (s-1)^2 \tag{2.2}$$

$$y(s) = s^3 y_f - (s-1)^3 y_i + \alpha_y s^2 (s-1) + \beta_y s (s-1)^2 \tag{2.3}$$

$$\begin{bmatrix}\alpha_x \\ \alpha_y\end{bmatrix} = \begin{bmatrix}k_f \cos\theta_f - 3x_f \\ k_f \sin\theta_f - 3y_f\end{bmatrix} \quad , \quad \begin{bmatrix}\beta_x \\ \beta_y\end{bmatrix} = \begin{bmatrix}k_i \cos\theta_i - 3x_i \\ k_i \sin\theta_i - 3y_i\end{bmatrix} \tag{2.4}$$

where $s \in [0,1]$ represents the arc parameter of the curve, while x_i, y_i, θ_i and x_f, y_f, θ_f are respectively the initial and final poses of the trajectory.

Let s_p be the selected pushing side on the object. The initial pose of the curve is the pose the robot would have when placed in pushing contact on s_p with the object situated in γ. The final pose is computed in the same way but the object is situated in the pose associated with the selected node q. An example of the robot in pushing contact with the object is visualized in Figure 2.2.

The remaining parameters for the polynomial trajectory are k_i and k_f, these are strongly realated to the resulting shape and length of the trajectory. In order to find the shortest polynomial trajectory whose curvature abides the pushing motion constraints expressed in Section 2.1.1, the value for k_i and k_f is computed by solving the following constrained optimization problem.

$$\underset{k_i,k_f}{\arg\min} \int_0^1 \sqrt{x'(s)^2 + y'(s)^2} ds \tag{2.5}$$

$$\text{subject to } k_i, k_f > 0 \tag{2.6}$$

$$r_{lim_-} \le R_c(s) \le r_{lim_+}, \forall s \in [0,1] \tag{2.7}$$

where:

$$R_c(s) = \frac{(x'(s)^2 + y'(s)^2)^{\frac{3}{2}}}{x'(s)y''(s) - x''(s)y'(s)} \tag{2.8}$$

is the curvature radius associated with each point of the polynomial path. $x'(s)$,$y'(s)$ and $x''(s)$,$y''(s)$ indicate the first and second derivatives of $x(s)$,$y(s)$ with respect to s respectively. Constraint (2.7) expresses the motion constraint for pushing detailed in Section 2.1.1. The optimization problem is nonlinear with a semi-infinite constraint, more details on how it has been implemented are shown in Section 2.2.3

If the optimization problem reports a solution, the resulting trajectory is checked for collisions with the environment. If the trajectory is collision free then the object can be pushed from γ to the pose associated with q then a successor of q is found. If the found successor has a different pushing side s_p than q, an additional intermediate node is inserted, representing the change of pushing side. Each node has an associated cumulative cost f, computed by adding a incremental component to the cost of the previous node up the tree. The cost $f(q)$ and its components $g(q)$ and $h(q)$ are defined

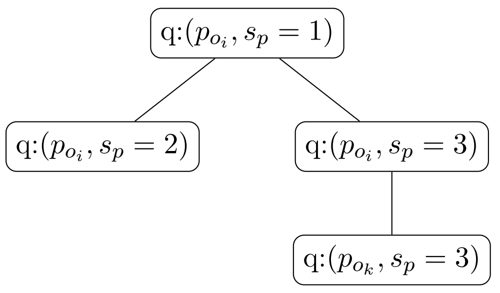


FIGURE 2.3: Example of graph generated by the search algorithm. Every node q has an associated state composed of a 2D pose γ and a pushing side s_p.

as follows:

$$f(q) = g(q) + h(q) \tag{2.9}$$

$$g(q) = d(q, q[-1]) + g(q[-1]) \tag{2.10}$$

$$h(q) = l(\Gamma_0^q) \tag{2.11}$$

The first component $g(q)$ is computed by adding the distance from q to its parent node $q[-1]$ to the component g associated with the parent node. If the two nodes have different associated object position, the cost increment is the length of $\mathcal{P}$. If q and $q[-1]$ have the same object position, the difference is represented by the change in pushing side. If the previous node has the same s_p, the incremental component is computed as the length of the polynomial trajectory connecting the two nodes. If the previous node has the same object pose but different s_p, the incremental component represents the distance between the robot's two pushing positions. To complete the total cost f, it is necessary to add a heuristic cost component h to the previously described cumulative one. This heuristic provides an estimate of the distance between the current node and the goal and, combined in the cumulative cost f, enables the algorithm to pursue the search from the node with the lowest cost. For each node, this heuristic is computed as the length of Γ up to the pose γ associated to the new node. Figure 2.3 displays an example of the graph generated by the A* algorithm. In particular, the graph displays the two types of connections between nodes that are allowed. The first type, displayed by the connection between the top node and its children, represents a change in pushing side, implying that the robot will have to change its contact side on the object to proceed with the pushing manipulation. The second type represents the actual pushing manipulation. In the represented case the robot will push the object from p_{o_k} to p_{o_i} using pushing side $s_p = 3$ by moving along a polynomial trajectory.

The solution of Algorithm 1 is computed iteratively from the final node q as a list of the polynomial paths from each node to its parent. A list of the pushing sides associated to each pushing path is also created to aid the execution.

Algorithm 1: Search Algorithm

```
   input  : path $\Gamma$
   output: List of pushing trajectories
1  begin
2      Put start node $q_0$ in OPEN list
3      while $OPEN \neq \varnothing$ do
4          select $\mathbf{q}$ from OPEN with the lowest $f(\mathbf{q})$
5          if pose of $\mathbf{q}$ is initial pose of $\Gamma$ then
6              compute solution
7              return success
8          Generate $\mathcal{S}$ the list of successors of $q$
9          foreach successor $s$ in $\mathcal{S}$ do
10             if OPEN contains a node with same configuration and lower cost
                $f$ then
11                 add $s$ to CLOSED list and continue
12             if CLOSED contains a node with same configuration and lower
                cost $f$ then
13                 add $s$ to CLOSED list and continue
14     return failure
```

2.2.3 Planner Implementation

The proposed algorithm has been implemented inside the ROS [39] framework. The first layer of the planning algorithm is implemented in C++ using the open motion planning library (OMPL) [40] over $\mathbb{R}^2$ and the RRT* holonomic planning algorithm. The newly generated state is checked for collision against the map at every iteration. Furthermore, a termination condition is applied to the algorithm to end the execution when the best path has been found. A representation of the path identified in this layer is visualized in Fig. 2.4. The A* search algorithm composing the second layer of the planning algorithm, detailed in Algorithm 1, receives as an input the path computed by the previous layer as an ordered list of 2D positions and searches a sequence of connections from start to finish using the positions as a guideline for searching. As detailed in Algorithm 2, the successors generation procedure requires the solution of a semi-infinite programming problem. The approach used to solve this problem discretizes the curvature constraint by sampling points along the generated trajectory and computing the curvature at this reduced number of points, thus approximating the semi-infinite constraint. This approach represents a trade-off between the time required to calculate a solution and the accuracy obtained in terms of constraint satisfaction. The resulting nonlinear optimization problem is solved using a Sequential Quadratic Programming solver. To prevent the occurrence of infeasible trajectories resulting from the optimization solver, the resulting path is post-checked for both curvature feasibility and collision avoidance. Figure 2.5 show an attempt to connect two different states on the identified holonomic path while Fig. 2.6 displays the final result of a planning procedure.

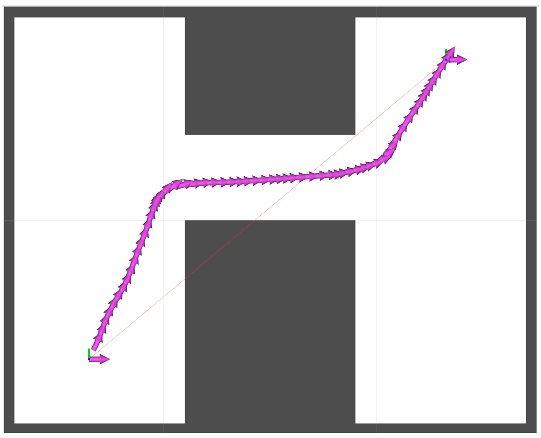


FIGURE 2.4: Results of the first planning layer visualized in Rviz. The arrows represent the desired direction of motion for the pushing system.

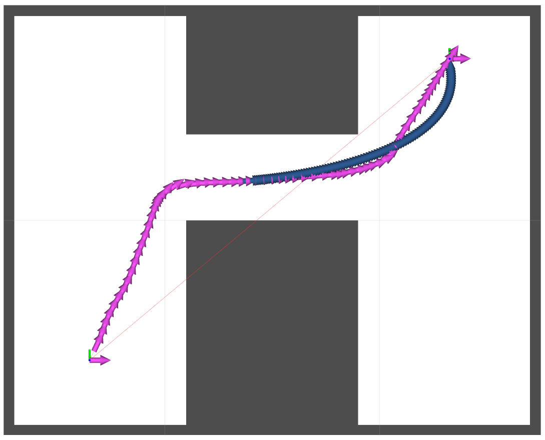


FIGURE 2.5: Snapshot of the planning search algorithm connecting two states from the path obtained by the first layer. The pink arrows represent the output of the first planning layer. The blue line shows an attempt of the search algorithm in connecting two intermediate states.

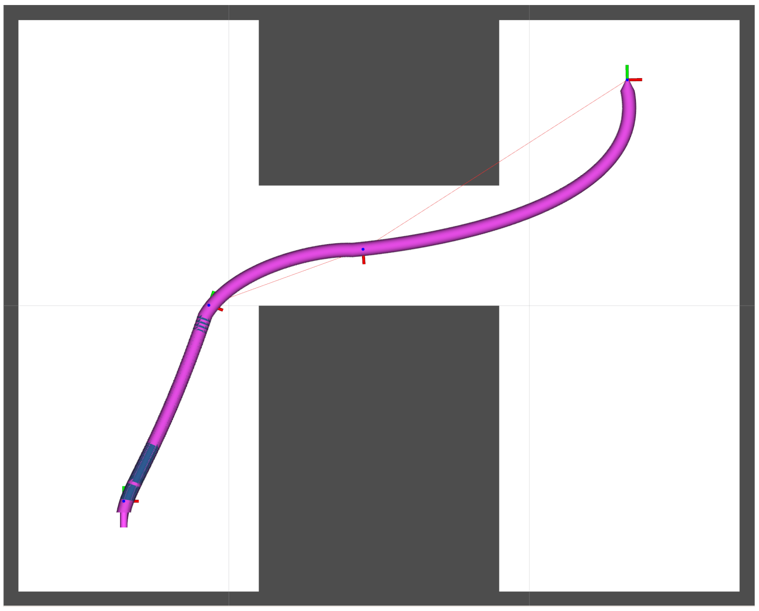


FIGURE 2.6: End result of the manipulation planning.

```
Algorithm 2: Successors generation
   input  : Selected node $\mathbf{q}$, path $\Gamma$
   output: List of successors node $\mathcal{S}$
1  begin
2      Set $\gamma$ as the first element of $\Gamma$
3      while $\mathcal{S} = \varnothing$ and $\gamma \neq q$ do
4          foreach pushing side do
5              compute initial robot pushing pose given $\gamma$ and pushing side
6              compute final robot pushing pose given the pose of $q$ and pushing
                side
7              solve constrained optimization obtaining polynomial path
8              if path collision free then
9                  if  pushing side $s_p \neq s_p$ of $q$ then
10                     create new node with properties $g$,$h$,$f$,$\mathcal{P}$,$p$
11                     add new node to $\mathcal{S}$
12         if $\mathcal{S} = \varnothing$ then
13             set $p$ as the following element of $\Gamma$
14     return $\mathcal{S}$
```

2.3 Manipulation Control for Wheel Slip Avoidance

The first approach to control the motion of a mobile robot for pushing manipulation has focused on maintaining the wheels of the robots rolling without slipping. The loss of traction at the wheels can impact the pushing performance greatly, therefore the contribution of this work is the design of a controller, based on Nonlinear Model Predictive Control (NMPC), that avoids the slipping of the wheels of a mobile robot. The proposed formulation allows us to explicitly consider the dynamics of the robot, the object, and the friction. In particular, in this work, we directly integrate constraints that guarantee that wheel slip becomes negligible. The dynamic model of the mobile robot is introduced in Section 2.3.1. Section 2.3.2 contains the main results, namely the proposed NMPC scheme and the wheel slip avoidance constraints. As a case study, in Section 2.3.3, we present the dynamic simulations results obtained by controlling the robot through the proposed NMPC to push a box.

2.3.1 Modeling

The dynamic model of the robot is introduced in this section. We consider a differential drive robot moving on a plane, whose schematic representation is depicted in Fig. 2.7. The robot is equipped with a frontal bumper rigidly attached to its body. Let Σ_w be the fixed world frame, and Σ_r be the body frame attached to the midpoint of the axle of the mobile robot. In addition, let $p_r = \begin{bmatrix} x_r & y_r \end{bmatrix}^T \in \mathbb{R}^2$ be the position of Σ_r with respect to Σ_w, $\upsilon \in \mathbb{R}$ and $\omega \in \mathbb{R}$ be the heading and angular velocities of the mobile robot, respectively. Finally, let $\theta_r \in \mathbb{R}$ be the angle expressing the rotation of Σ_r in

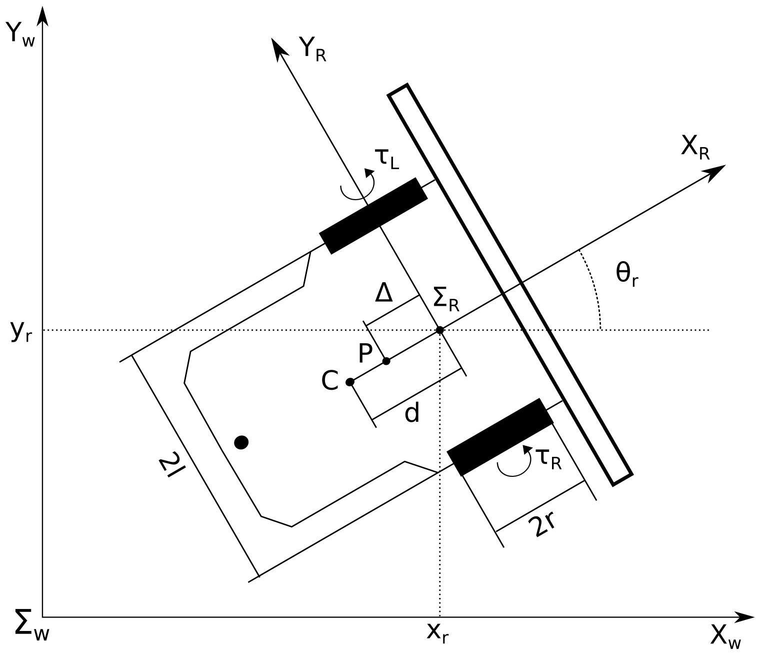


FIGURE 2.7: Schematic representation of the differential drive mobile robot. The black rectangles are the wheels. The black circle is the caster wheel. The frontal bumper is represented by the white rectangle in front of the wheels.

Σ_w. The kinematic model of the mobile robot can be expressed as follows:

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

Through standard computations, the dynamic model of the mobile robot was derived in [41], and it can be expressed as

$$\begin{aligned}\dot{x}_r &= \cos\theta_r v\\ \dot{y}_r &= \sin\theta_r v\\ \dot{\theta}_r &= \omega\\ \Big(m_r + \frac{2I_w}{r^2}\Big)\dot{v}_r &= \frac{1}{r}(\tau_R + \tau_L) - m_c d\omega^2\\ \Big(I_r + \frac{2l^2}{r^2}I_w\Big)\dot{\omega}_r &= \frac{l}{r}(\tau_R - \tau_L) + m_c\omega v d\end{aligned} \tag{2.13}$$

with $q = \begin{bmatrix}x_r & y_r & \theta_r & v & \omega\end{bmatrix}^T \in \mathbb{R}^5$ being the state vector of the robot, $m_r \in \mathbb{R}^+$ the total mass of the vehicle, $I_r \in \mathbb{R}^+$ the inertia moment of the robot about the vertical axis, $I_w \in \mathbb{R}^+$ the inertia moment of a wheel about its axis, $l \in \mathbb{R}^+$ the half the wheel separation distance(see Fig. 2.7), $r \in \mathbb{R}^+$ the wheel radius, $d \in \mathbb{R}^+$ the distance of the center of mass of the body of the robot on the robot axis (see Fig. 2.7), and $m_c \in \mathbb{R}^+$ the mass of the body of the robot (i.e., excluding the wheels). The control input is the pair $u = \begin{bmatrix} \tau_R & \tau_L \end{bmatrix}^T \in \mathbb{R}^2$, which are the torques acting on the wheels.

2.3.2 Controller Design

In this section, we describe the design of the controller used to command the robot. We choose the NMPC scheme because it allows to include dynamic constraints in the design of the controller explicitly. The main idea of the scheme is to optimize the predicted future behavior of the system over a finite time horizon.

Optimization Problem Formulation

The idea behind the NMPC is the repetitive solution of an optimal Non-Linear Programming (NLP) problem . Given the measured state q_0 at each controller time step T_s, the discretized version of the dynamic model is employed by the NMPC to predict the future behavior of the system state $\hat{q}(k)$, with $k = 0, \ldots, N-1$, where $N \geq 2$ is the prediction horizon, and $\hat{q}(0) = q_0$. Such a prediction is useful to optimize the control sequence $u(0), \ldots, u(M-1)$, with $0 < M \leq N$ the control horizon, and $u(i) = u(M-1)$ for $i = M, \ldots, N-1$. The peculiarity of the NMPC algorithm is that only the first element $u(0)$ of the sequence is applied to the system. The NLP is repeatedly solved from each new acquired measure. The NLP minimizes an objective function, generally composed of the states and the inputs, with respect to the input variable and subject to a set of constraints. Nevertheless, if formulated in this way, the NLP becomes of high dimension, and the computation time and the accuracy of the solver worsen. Instead, we use the recursive elimination methodology proposed in [42]. Such a methodology decouples the dynamic model of the system from the solution of the Non-Linear Programming (NLP) problem by reducing the size of the optimization variable and allowing each problem to be treated by specialized solution methods. With a slight abuse of notation regarding the dependencies for each function, the optimization control problem can be written as

$$\begin{aligned} &\text{minimize } J(z) \\ &\text{w.r.t. } z = (u(0)^T, \ldots, u(M-1)^T)^T \in \mathbb{R}^{2M} \\ &\text{s.t } G(z) \leq \bar{0}, \end{aligned} \tag{2.14}$$

where

$$J(z) = e(N)^T P e(N) + u(N)^T W_N u(N) + \sum_{k=1}^{N-1} e(k)^T Q e(k) + u(k)^T W u(k) \tag{2.15}$$

is the cost to minimize, while $Q \in \mathbb{R}^{5\times5}$, $P \in \mathbb{R}^{5\times5}$, $W \in \mathbb{R}^{2\times2}$ and $W_N \in \mathbb{R}^{2\times2}$ are diagonal and positive semi-definite matrices[1]. The error $e(k) \in \mathbb{R}^5$ is the difference between the predicted $\hat{q}(k)$ and the desired state $q_{ref}(k) \in \mathbb{R}^5$. Such an error is

[1]The matrices W and W_N can be zero matrices since we force the control input to be bounded by including the saturation of the actuators explicitly.

expressed with respect to Σ_r and it is calculated as

$$e(k) = R(\theta_r(k))\left(\hat{q}(k) - q_{ref}(k)\right), \tag{2.16}$$

with

$$R(\theta(k)) = \begin{bmatrix} \cos(\theta(k)) & \sin(\theta(k)) & 0 & 0 & 0 \\ -\sin(\theta(k)) & \cos(\theta(k)) & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{5\times5}. \tag{2.17}$$

By denoting with

$$\hat{q}(k+1) = F(q(k), u(k)), \quad k = 0, \dots, N-1, \tag{2.18}$$

the Euler-discretized version of (2.13), the robot state prediction is calculated through (2.18) from the current measure q_0. Finally, the inequality constraints are expressed as $G(z) \leq \bar{0}$, with $\bar{0} \in \mathbb{R}^{13N}$ the zero vector of proper dimension, and $G(z) \in \mathbb{R}^{13N}$ defined as detailed in the next subsection.

Wheel Slip Constraints

The first subset of constraints is the following

$$\begin{aligned} &x_r(k) + x_{lim} \leq 0, && (2.19a) && \Omega_r(k) + \Omega_{lim} \leq 0, && (2.19e) \\ &x_{lim} - x_r(k) \leq 0, && (2.19b) && \Omega_{lim} - \Omega_r(k) \leq 0, && (2.19f) \\ &y_r(k) + y_{lim} \leq 0, && (2.19c) && \Omega_l(k) + \Omega_{lim} \leq 0, && (2.19g) \\ &y_{lim} - y_r(k) \leq 0, && (2.19d) && \Omega_{lim} - \Omega_l(k) \leq 0, && (2.19h) \end{aligned}$$

for $k = 0, \dots, N-1$. The inequalities (2.19a)—(2.19d) limit the position of the robot inside a rectangle of the plane delimited by the coordinates $x_{lim} \in \mathbb{R}$ and $y_{lim} \in \mathbb{R}$ in Σ_w. The inqualities (2.19e)—(2.19h) impose instead that the velocities of the wheels remain inside the saturation limit $\Omega_{lim} \in \mathbb{R}$ of the motors.

Beyond the constraints introduced above, the main focus of our work is the design of an explicit condition to avoid wheel slip. Wheel slip is defined as the relative motion between a tire and the surface on which it is moving. This relative motion occurs when the force needed to maintain the contact exceeds the maximum friction force. The Coulomb friction model provides the following representation of the wrench transmitted to an object through a point contact with friction [43]

$$F_{c_i} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T f_{c_i} \in \mathbb{R}^6, \tag{2.20}$$

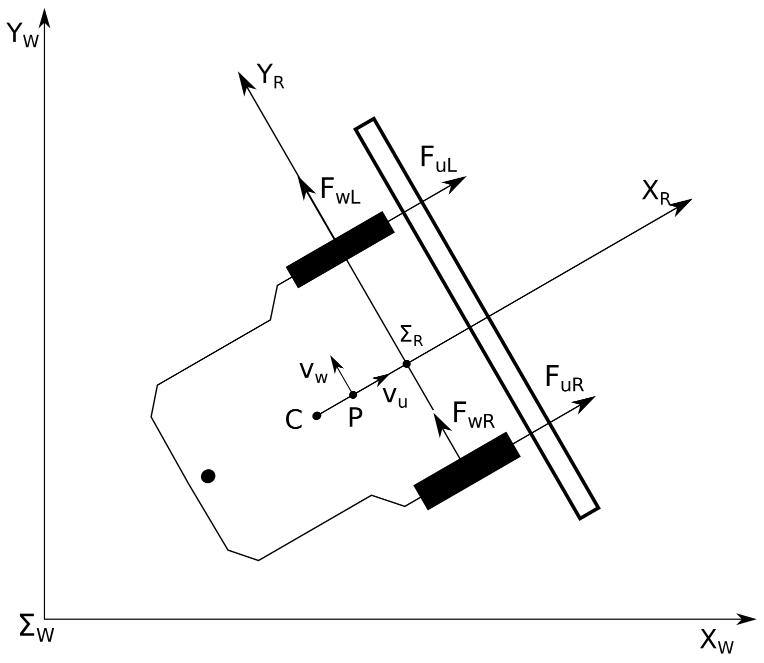


Figure 2.8: Planar forces and torques acting on the robot.

where $f_{c_i} \in FC_{c_i}$ and

$$FC_{c_i} \triangleq \{f = \begin{bmatrix} f_{t1} & f_{t2} & f_n \end{bmatrix}^T \in \mathbb{R}^3 : \sqrt{f_{t1}^2 + f_{t2}^2} \leq \mu f_n, f_n \geq 0\} \tag{2.21}$$

is the so-called friction cone, while $f_{t1} \in \mathbb{R}$ and $f_{t2} \in \mathbb{R}$ are the components of a force f tangential to the contact, and $f_n \in \mathbb{R}$ is the component of f normal to the contact with positive sign going inside the object. The friction coefficient between each wheel and the ground is denoted by $\mu \in \mathbb{R}^+$.

Following such representation, to obtain wheel slip avoidance, we can conclude that the frictional forces need to balance the forces exerted by the robot to achieve a specific motion at any time. The robot interacts with the floor in three points: the two driving wheels and the caster. We assume the planar forces transmitted by the caster wheel are negligible, and we consider the vertical component only. Figure 2.8 shows the forces exerted by the driving wheels decomposed into the longitudinal (subscript u) and lateral (subscript w) components. To maintain a rolling contact between the wheels and the ground (i.e., to avoid slippage) the following relations need to be verified:

$$\sqrt{F_{uL}^2 + F_{wL}^2} \leq \mu N_L, \quad \sqrt{F_{uR}^2 + F_{wR}^2} \leq \mu N_R. \tag{2.22}$$

These forces are related to the robot's movement through the following equations:

$$m_r\dot{v}_u = m_r\dot{v}_r = F_{uL} + F_{uR}, \tag{2.23}$$

$$m_r\dot{v}_w = m_r\dot{\omega}_r\Delta = F_{wR} + F_{wL}, \tag{2.24}$$

$$I_r\ddot{\theta}_r = I_r\dot{\omega}_r = (F_{uR} - F_{uL})l + (F_{wL} + F_{wR})\Delta. \tag{2.25}$$

$$N_L + N_R + N_{ca} = m_r g, \tag{2.26}$$

$$N_{ca}l_{ca} - m_r g\Delta + m_r\dot{v}_r(h - r) = 0, \tag{2.27}$$

$$N_L - N_R = \frac{m_r\dot{v}_w(h - r)}{l}. \tag{2.28}$$

where (2.23) and (2.24) describe the movement along the robot's X_R and Y_R axes respectively and (2.25) describes the rotation about the Z_R axis. Equation (2.26) relates the reaction forces of the ground caused by the weight of the robot. Considering the equilibrium to the rotation about the robot's X_R and Y_R axis, equations (2.27) and (2.28) can be derived, which relate forces F_{uR}, F_{uL}, F_{wL}, and F_{wR} with the vertical forces acting on the robot. In particular, Fig (2.9a) depicts the considered forces in a frontal section of the robot, where $N_L \in \mathbb{R}$ and $N_R \in \mathbb{R}$ are the left and right reaction forces of the wheels with the ground, and $m_r\dot{v}_w$ is the apparent inertial force caused by the robot lateral motion. Conversely, Fig. (2.9b) depicts the forces in a lateral view of the robot, where $N_{ca} \in \mathbb{R}$ is the reaction force of the caster with the ground, $m_r g$ represents the weight of the robot and $m_r\dot{v}_r$ is the apparent inertial force caused by the longitudinal motion of the robot. Notice that in equations (2.23)—(2.28) the forces F_{wL} and F_{wR} appear always in the form $F_{wL} + F_{wR}$ thus it's not possible to extract the two forces from the system of equations, preventing the direct computation of (2.22).

If we assume $F_{wL}F_{wR} \geq 0$, the following relations are always verified

$$\sqrt{F_{uL}^2 + F_{wL}^2} \leq \sqrt{F_{uL}^2 + (F_{wL} + F_{wR})^2}, \tag{2.29}$$

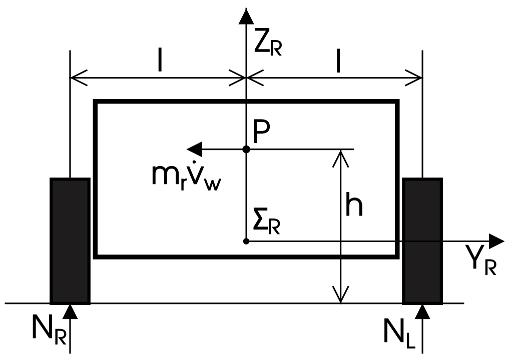
$$\sqrt{F_{uR}^2 + F_{wR}^2} \leq \sqrt{F_{uR}^2 + (F_{wL} + F_{wR})^2}. \tag{2.30}$$

Such an assumption implies that there is no relative motion between the wheels on the $Y_R Z_R$-plane of the robot. The truthfulness of (2.29)—(2.30) allows us to write the following two constraints which imply (2.22) by the transitive property of inequalities,

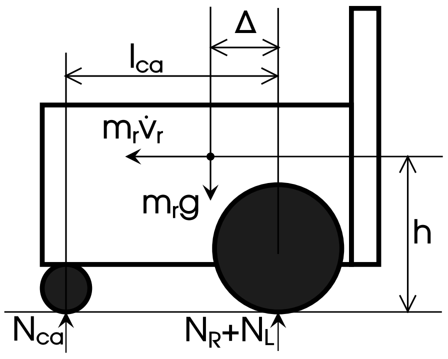
$$\sqrt{F_{uL}^2 + (F_{wL} + F_{wR})^2} \leq \mu N_L, \tag{2.31}$$

$$\sqrt{F_{uR}^2 + (F_{wL} + F_{wR})^2} \leq \mu N_R. \tag{2.32}$$

Since it is possible to extract N_L,N_R,N_{ca},F_{uR},F_{uL} and $F_{wR}+F_{wL}$ from (2.23)—(2.28) the constraints are computable from the controller point of view. It is worth noting



(A) Frontal view of the robot.



(B) Lateral view of the robot.

FIGURE 2.9: Forces exchanged between the robot and the ground.

that the solution extracted from the system of equations above can assume non-physical values unless we impose the following constraints to the problem

$$N_L \geq 0, \qquad N_R \geq 0, \qquad N_{ca} \geq 0. \tag{2.33}$$

These relations also imply that the wheels remain on the ground at all times. The relations (2.31)—(2.33) are added to the inequality constraints of the problem (2.14). The final inequality constraint vector $G(z)$ is composed by (2.19a)—(2.19h) and (2.31)—(2.33), and it is repeted for each time step of the prediction horizon.

2.3.3 Simulations

In this section, we present the simulation results to validate the proposed controller. The simulations show a Pioneer 3-DX tracking a trajectory generated by a motion planning algorithm to push an object to the desired configuration (position plus orientation). To simulate the system, we use the V-REP simulator controlled by a MATLAB script which handles the data collection and the controller calculations

2.3.4 Case studies

To compute the discretized model of the system (2.18), the following parameters are retrieved from the datasheet of the physical robot: $L = 0.17$ m, $R = 0.095$ m, $m = 17$ kg, $m_c = 16$ kg, $I_r = 0.1307$ kgm^2, $I_w = 0.0051$ kgm^2, and $d = 0.15$ m. The matrices of the functional cost (2.15) are chosen to reduce the local error along the Y_R component in Σ_R. This choice implies that the robot moves to initially reduce, as much as possible, the lateral error from the target, and orientation is only adjusted afterwards. Then, by tuning, the chosen gains are $Q = \text{diag}(\begin{bmatrix}2.5 & 5 & 0.25 & 0.05 & 0.05\end{bmatrix})$, $P = 4Q$, while W and W_N are zero matrices. To compute $G(z)$, instead, the parameters are experimentally tuned as $x_{lim} = y_{lim} = 10$ m, $\Omega_{lim} = 4\pi$ rad/s, $h = \Delta = 0.12$ m, and $\mu = 0.5$. The applied prediction horizon is set as $N = 15$ with control horizon set as $M = 6$. The time interval between the predicted instants is $T_s = 0.1s$. The simulations are performed on a standard PC, with Intel Core i7-4510U CPU, on which

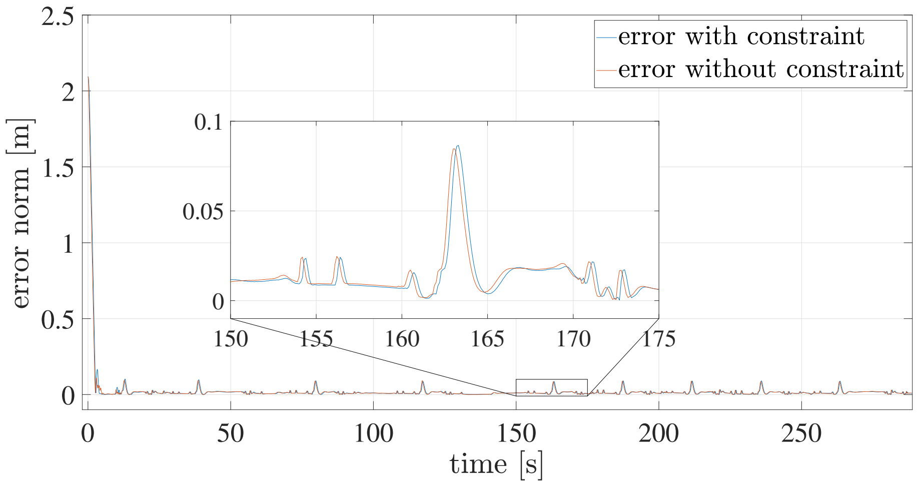


FIGURE 2.10: Position error with and without constraint.

it is installed MATLAB R2018b and V-REP 3.5.0. A MATLAB script is in charge of elaborating the measures acquired from V-REP, solve the Nonlinear Model Predictive Control (NMPC) algorithm and send the actuation command to V-REP, which is thus employed as a dynamic simulator and not as just a visualizer.

Three case studies are addressed in the following. The first one is necessary to test the NMPC approach for tracking the trajectories that compose a manipulation plan with a mobile robot. The second case study shows the pushing manipulation on the same trajectory, while the latter case study includes some parametric uncertainties and discrepancy between the simulated robot in V-REP and the parameters employed for the NMPC algorithm in MATLAB.

Case study I.

In this case study, the robot must follow a given trajectory without slipping. First, the robot moves towards the initial point of the trajectory until it reaches a 0.05 m radius around the point. Then, the robot starts following the trajectory. To validate the constraint, we executed a simulation with the wheel slip avoidance constraints (2.31)—(2.32) active, and a simulation without it. Figure 2.10 shows the tracking errors for both the simulations. Figure 2.11 shows the robot tracking the trajectory with the constraint active: red blocks represent obstacles to be avoided. The trajectory starts in the bottom-right part of the plot (position $(0, -3)$), and is composed of different portions: arc-shaped portions are introduced to let the robot change its orientation with respect to the object and therefore changing its pushing side. It is possible to notice that the robot follows the trajectory in a very precise manner. Figure 2.12 and 2.13 show the difference between the surface velocity of each wheel and the longitudinal velocity of its axle for the first 13 s of simulation: when this difference is non-zero, then wheel slip is happening. The plots clearly show that the application of the constraints significantly reduces the presence of wheel slip.

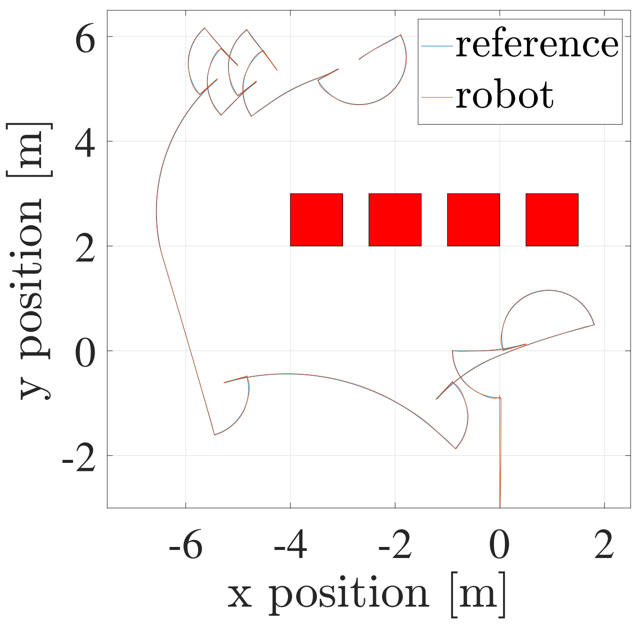


FIGURE 2.11: Robot trajectory and reference: red blocks represent obstacles.

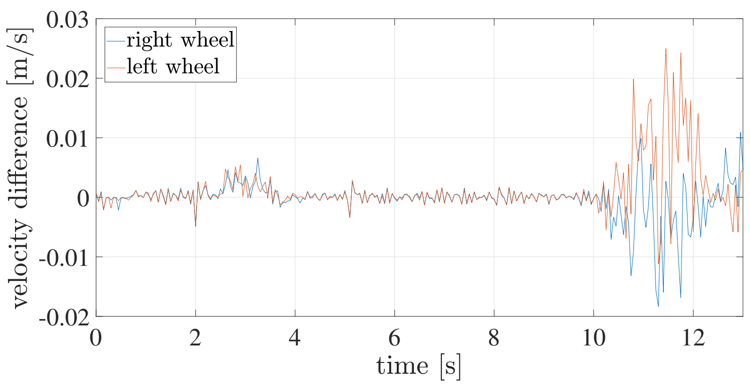


FIGURE 2.12: Longitudinal velocity difference with constraint: non-zero values imply the presence of wheel slip.

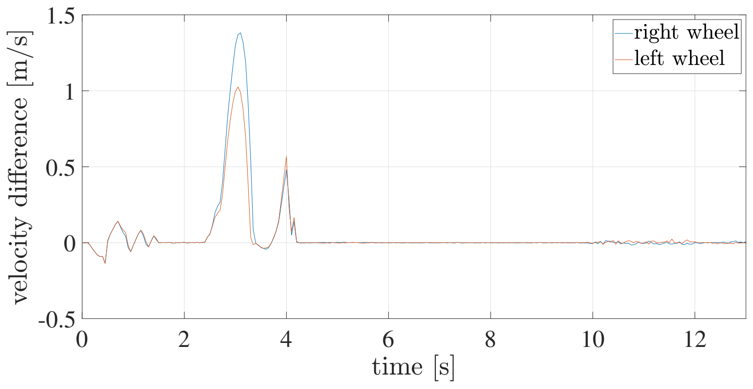


FIGURE 2.13: Longitudinal velocity difference without constraint: non-zero values imply the presence of wheel slip.

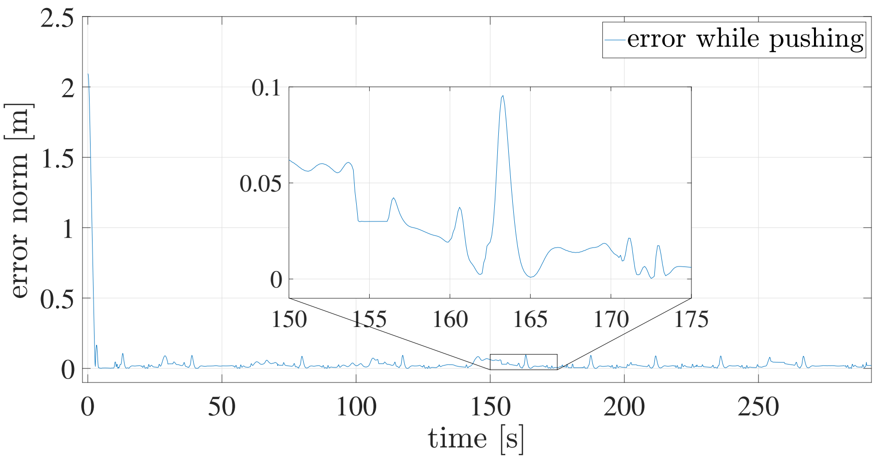


FIGURE 2.14: Position error for the pushing manipulation,

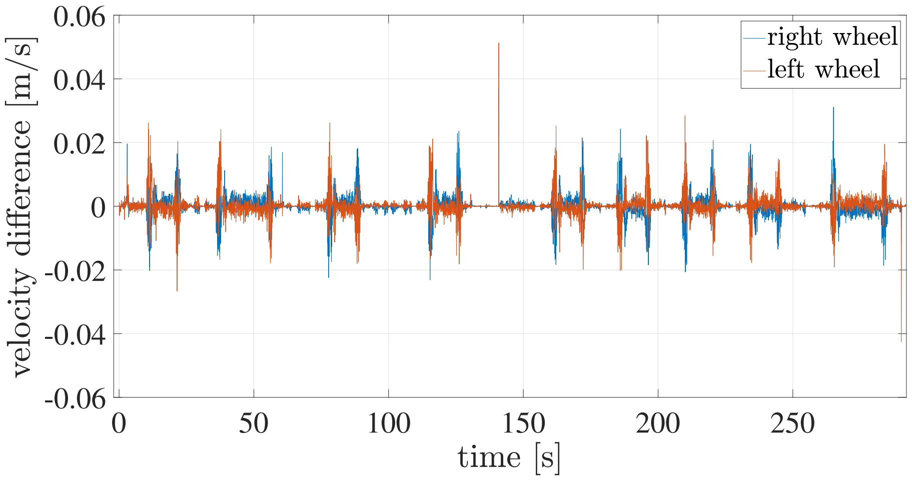


FIGURE 2.15: Longitudinal velocity difference with the box.

Case study II.

In this second case study, pushing manipulation is addressed. The size of the box to be pushed is $0.25 \times 0.25 \times 0.25$ m, and it weighs 3 kg. The presence of the box sliding on the floor acts as an unmodeled disturbance to the system. The robot behaves as discussed in case study I, following a trajectory designed for pushing manipulation: it moves to the starting point, and then continues its motion, tracking the trajectory. Figure 2.14 shows the position error of the robot during the simulation. The robot can track the trajectory regardless of the presence of the box. In Figure 2.15 we observe the longitudinal velocity difference of each wheel across the whole simulation. The differences remain bounded in spite of the presence of the box.

Case study III.

In a real environment, it is very likely to have inaccuracies in parameters, especially the friction coefficient μ. To ensure the controller is robust to uncertainty regarding this parameter, we conducted a simulation where the value of the friction coefficient for the wheels in the controller's parameters (implemented in MATLAB) is set to

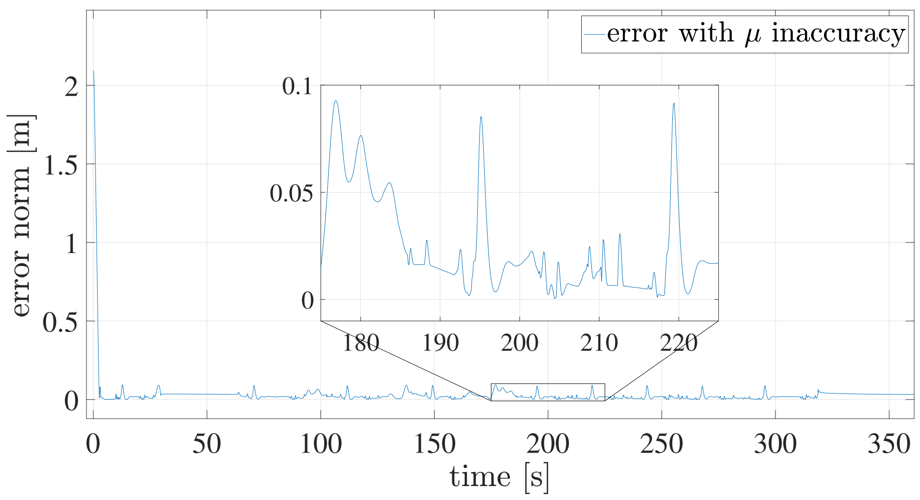


Figure 2.16: Position error for the pushing manipulation with inaccurate μ.

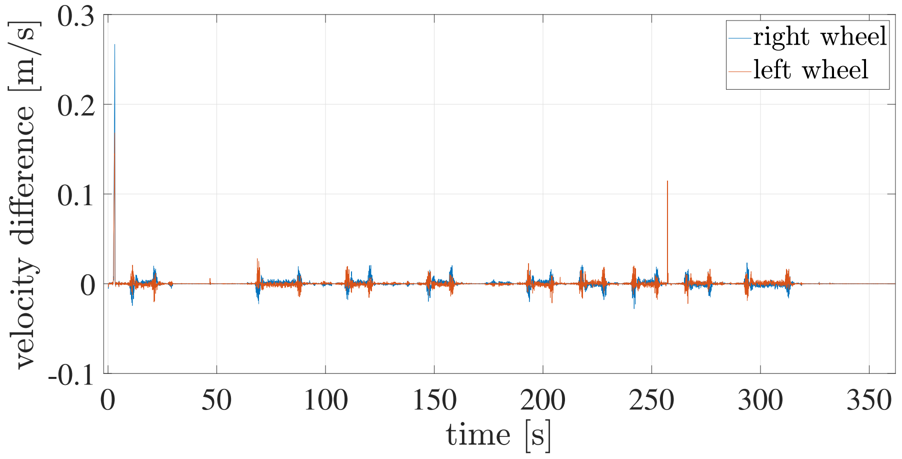


Figure 2.17: Longitudinal velocity difference with inaccurate μ.

$\mu = 0.7$, therefore the controller assumes that the wheels of the robot are capable of applying a greater force. As per the other simulations, the robot moves to track the assigned trajectory. Figure 2.16 shows the tracking performance of the robot subject to disturbance. From Fig. 2.17 we can conclude that the controller is robust to inaccuracies in the friction parameter.

2.4 Maintenance of Pushing Contact

2.4.1 Problem statement

Consider a wheeled mobile robot moving in a bi-dimensional environment, where a polygonal planar object has to be manipulated. Let Σ_w and Σ_r be the global reference frame and the body frame attached to the center of the robot's axle, respectively. Let the pose of the mobile robot at time t be represented by the vector $\chi_r(t) = \begin{bmatrix} x_r(t) & y_r(t) & \theta_r(t) \end{bmatrix}^T \in \mathbb{R}^3$, where $x_r(t), y_r(t) \in \mathbb{R}$ represent the position of Σ_r in

Σ_w, and $\theta_r \in \mathbb{R}$ is the rotation of Σ_r with respect to Σ_w. A visualization is provided in Fig. 2.18. In a similar way, let $\chi_o(t) = \begin{bmatrix} x_o(t) & y_o(t) & \theta_o(t) \end{bmatrix}^T \in \mathbb{R}^3$ represent the pose of the object, as shown in Fig. 2.19.

We provide a solution to the following problem.

Problem 2 *Control the motion of the mobile robot to manipulate the object through pushing maneuvers, in such a way that the trajectory $\chi_o(t)$ closely tracks the desired one $\chi_d(t)$, starting from the initial pose $\chi_o(0)$.*

In the following, we will assume the mobile robot to behave as a unicycle. The choice is motivated by the simplicity of notation introduced by such a model, and by the fact that several real-world mobile robots (as differential-drive robots) can be represented according to this formulation [44]. We assume that all the considered contacts are rigid, that the robot wheels do not slip, and that the forces exchanged in the interaction follow Coulomb's model of friction. Moreover, we decompose each contact force in two components, aligned with the edges of the friction cone [3]. The angle between each component and the contact normal is

$$\theta_\mu = \tan^{-1} \mu, \tag{2.34}$$

where $\mu > 0$ is the friction coefficient associated with the interacting surfaces. A visualization of the used decomposition is provided in Figure 2.19. The motion of the controlled system is assumed quasistatic (i.e, it is slow enough that inertial forces are negligible). Moreover, we assume the mobile robot to be equipped with a planar end-effector (i.e., a planar contact surface), such that the surface used to interact with the object is consistent and homogeneous. During the interaction, the end-effector is supposed to be parallel to one of the sides of the polygonal object. This type of interaction is typically referred to as *line contact*, modeled as if the only contact points were the extreme points of the line [45].

2.4.2 Modeling

In this section, we introduce the mathematical model of the motion of the system and the constraints applied within the Model Predictive Control (MPC) controller for planar manipulation tasks. First, we describe a second-order model of the mobile robot and the error dynamic for the desired trajectory, then the mathematical model of the pushed object motion is introduced.

Robot Model

Defining $v_r, \omega_r \in \mathbb{R}$ as the linear and angular velocities of the robot, respectively (see Fig. 2.18), the state of the robot is defined as $\xi(t) \in \mathbb{R}^5$, given by

$$\xi(t) = \begin{bmatrix} x_r(t) & y_r(t) & \theta_r(t) & v_r(t) & \omega_r(t) \end{bmatrix}^T . \tag{2.35}$$

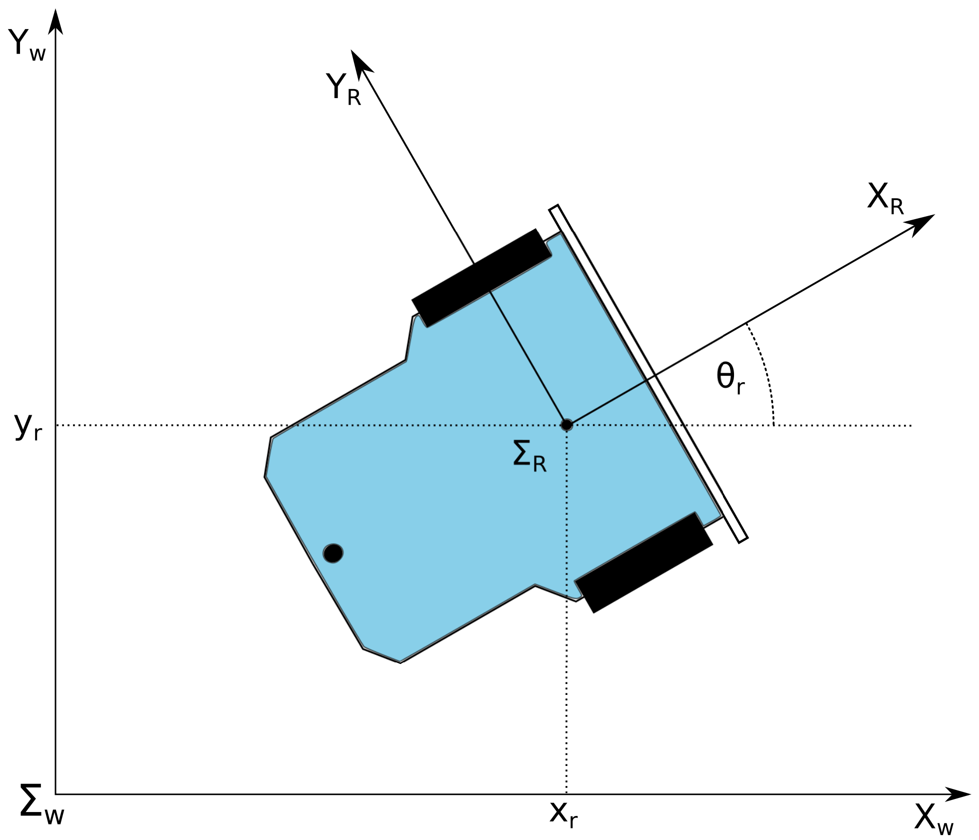


FIGURE 2.18: Schematic representation of the differential drive mobile robot. The black rectangles are the wheels. The black circle is the caster wheel. The frontal bumper is represented by the white rectangle in front of the wheels.

For ease of notation, in the following, dependence on time will be omitted, when not strictly necessary.

Define now $a_r, \varepsilon_r \in \mathbb{R}$ as the inputs for the robot, given as the linear and angular acceleration, respectively. Hence, the model of the robot motion can be written as

$$\dot{\xi} = \begin{bmatrix} \cos\theta_r v_r \\ \sin\theta_r v_r \\ \omega_r \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ a_r \\ \varepsilon_r \end{bmatrix}. \tag{2.36}$$

Solution of the Problem stated in Section 2.4.1 passes through the generation of the desired trajectory $\xi_d(t) = \begin{bmatrix} x_{rd}(t) & y_{rd}(t) & \theta_{rd}(t) & v_{rd}(t) & \omega_{rd}(t) \end{bmatrix}^T \in \mathbb{R}^5$ for the robot to realize the pushing maneuvers. Let $e(t) = \begin{bmatrix} e_{xr}(t) & e_{yr}(t) & e_{\theta r}(t) & e_{vr}(t) & e_{\omega r}(t) \end{bmatrix}^T \in \mathbb{R}^5$ be the error vector with respect to the desired reference frame centered in $(x_{rd}(t), y_{rd}(t))$, and oriented as $\theta_{rd}(t)$, that is defined as

$$e(t) = \begin{bmatrix} \cos(\theta_{rd}(t)) & \sin(\theta_{rd}(t)) & 0 & 0 & 0 \\ -\sin(\theta_{rd}(t)) & \cos(\theta_{rd}(t)) & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} (\xi - \xi_d). \tag{2.37}$$

Considering the robot motion (2.36) and the error vector (2.37), we can describe the system error dynamics as follows:

$$\dot{e}(t) = f(t) = \begin{bmatrix} \cos(e_{\theta r})(e_{vr} + v_{rd}) - v_{rd} + e_{yr}\omega_{rd} \\ \sin(e_{\theta r})(e_{vr} + v_{rd}) - e_{xr}\omega_{rd} \\ e_{\omega r} \\ a_r - \dot{v}_{rd} \\ \varepsilon_r - \dot{\omega}_{rd} \end{bmatrix}. \tag{2.38}$$

Pushed Object Model

Consider, as discussed in Section 2.4.1, a polygonal object pushed by the robot with line contact on one of its sides. The contact forces are modeled using the components along the friction cone as shown in Fig. 2.19. We denote with $f_{iR} \in \mathbb{R}$ and $f_{iL} \in \mathbb{R}$ the right and left contact force components, respectively, for each contact point $i = \{1, 2\}$ [3]. We define the vector $f_c \in \mathbb{R}^4$ as

$$f_c = \begin{bmatrix} f_{1R} & f_{1L} & f_{2R} & f_{2L} \end{bmatrix}^T. \tag{2.39}$$

The total external wrench $w \in \mathbb{R}^3$, expressed in Σ_W and whose torque is applied around the object's center of mass, exerted by the robot to the object can be described by

$$w = G f_c \tag{2.40}$$

where $G \in \mathbb{R}^{3\times 4}$ is the so called grasp matrix. For a square object of side length $2s > 0$, the grasp matrix is

$$G = \begin{bmatrix} \cos(\theta_r - \theta_\mu) & \sin(\theta_r - \theta_\mu) & s(\cos\theta_\mu + \sin\theta_\mu) \\ \cos(\theta_r + \theta_\mu) & \sin(\theta_r + \theta_\mu) & s(\cos\theta_\mu - \sin\theta_\mu) \\ \cos(\theta_r - \theta_\mu) & \sin(\theta_r - \theta_\mu) & s(\sin\theta_\mu - \cos\theta_\mu) \\ \cos(\theta_r + \theta_\mu) & \sin(\theta_r + \theta_\mu) & -s(\cos\theta_\mu + \sin\theta_\mu) \end{bmatrix}^T, \tag{2.41}$$

where θ_μ is given in (2.34), and it can be obtained through a geometrical analysis of the contact forces.

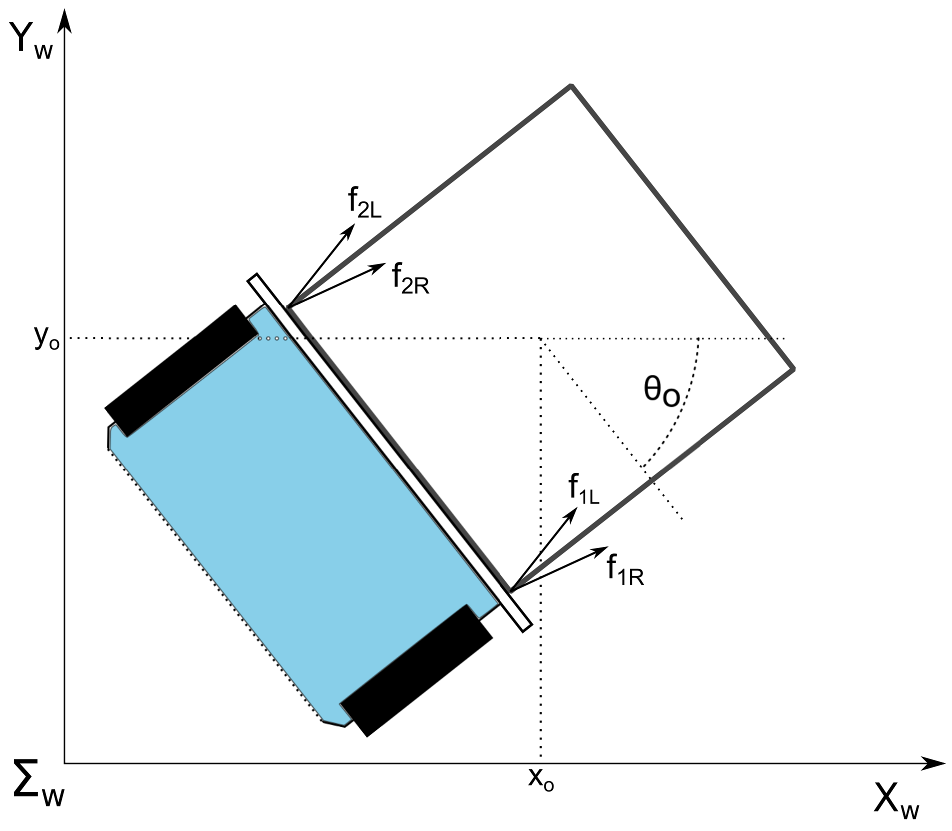


Figure 2.19: Schematic representation of the pushed object.

Under the assumption of quasistatic interaction, the object motion can be described using the limit surface [46], a geometric representation of the relationship between the applied force on an object and its instantaneous velocity. Inspired by [13], an ellipsoidal approximation of the limit surface is used, due to its simplicity and invertibility properties. A convex quadratic formulation of the ellipsoidal limit surface is given by $S(w) = \frac{1}{2}w^T Hw$, where $H \in \mathbb{R}^{3\times 3}$ is the matrix representing the ellipsoidal approximation of the limit surface. A procedure for computing such an approximation, that requires the knowledge of the object's shape and mass as well as the friction coefficient of the support surface, can be found in [47]. Through the principle of maximal dissipation [46], the object instantaneous velocity is perpendicular to the limit surface for a given wrench, which implies:

$$\begin{bmatrix} \dot{x}_o & \dot{y}_o & \dot{\theta}_o \end{bmatrix}^T = Hw. \tag{2.42}$$

2.4.3 MPC Controller Formulation

To correctly solve the Problem defined in Section 2.4.1, a controller must be designed such as the error vector $e(t)$ in (2.37) is steered to zero without violating the friction constraints given by the contact between the robot and the object. This avoids the slippage of the object during the pushing manipulation. As a matter of fact, zeroing the error vector only does not imply that the object follows the desired trajectory $\chi_d(t)$. The controller makes use of a Linear Time-Varying Model Predictive Control (LTVMPC) formulation [48] to solve the nonlinear control problem in real-time through the solution of a motion constrained optimization problem. First, a LTV approximation of the model is presented. The model considers the presence of the velocity and acceleration of the desired trajectory in the form of the measured disturbances $v(t) \in \mathbb{R}^4$, a vector of known but unmodifiable model inputs.

2.4.4 LTV Model Approximation

As discussed in [49], the MPC formulation requires a discrete-time linear (or linearized) model to construct the optimization problem. Therefore, the model (2.38) is linearized and discretized. The linearization is performed around a series of predicted states $\tilde{e}(t)$ obtained through numerical integration of (2.38). In particular, the nonlinear error dynamics (2.38) can be approximated by the following LTV system

$$\dot{e}(t) = A(\tilde{e}(t), v(t))e(t) + B_u u(t) + B_v(\tilde{e}(t))v(t), \tag{2.43}$$

where $u(t) \in \mathbb{R}^2$ is the model input vector, $v(t) \in \mathbb{R}^4$ is the vector of measured disturbances and $\tilde{e}(t) \in \mathbb{R}^5$ is the predicted state. More specifically, we define

$$u(t) = \begin{bmatrix} a_r \\ \varepsilon_r \end{bmatrix} \qquad v(t) = \begin{bmatrix} v_{rd} \\ \omega_{rd} \\ \dot{v}_{rd} \\ \dot{\omega}_{rd} \end{bmatrix} \qquad \tilde{e}(t) = \begin{bmatrix} \tilde{e}_{xr}(t) \\ \tilde{e}_{yr}(t) \\ \tilde{e}_{\theta r}(t) \\ \tilde{e}_{vr}(t) \\ \tilde{e}_{\omega r}(t) \end{bmatrix}, \tag{2.44}$$

and

$$\begin{gathered} A(\tilde{e}(t), v(t)) = \frac{\partial f}{\partial e}\bigg|_{\substack{e(t)=\tilde{e}(t) \\ v(t)}}, B_v(\tilde{e}(t)) = \frac{\partial f}{\partial v}\bigg|_{e(t)=\tilde{e}(t)} \\ B_u = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T, \end{gathered} \tag{2.45}$$

which represents the fact that matrices are obtained performing the linearization around $(\tilde{e}(t), v(t))$, both evaluated at time t. The discrete-time equivalent model of (2.43), defined with sampling time $T_s > 0$, can then be obtained following the procedure given in [50]. Denoting with $k \in \mathbb{Z}$ the discrete time variable, we get

$$e[k+1] = A[k]\, e[k] + B_u[k]\, u[k] + B_v[k]\, v[k], \tag{2.46}$$

with

$$A[k] = e^{A(\tilde{e}[k],v[k])T_s}, \tag{2.47}$$

$$B_u[k] = \int_0^{T_s} e^{A(\tilde{e}[k],v[k])\tau} d\tau B_u(\tilde{e}[k]), \tag{2.48}$$

$$B_v[k] = \int_0^{T_s} e^{A(\tilde{e}[k],v[k])\tau} d\tau B_v(\tilde{e}[k]). \tag{2.49}$$

2.4.5 LTV MPC Formulation

The idea behind the MPC formulation is to optimize the future behaviour across a finite prediction horizon of p steps. At every discrete-time instant k, for a given state $e[k]$, the optimal control input is computed solving the following constrained quadratic programming (QP)

$$\min_{z_k} \quad J(z_k, e[k]) \tag{2.50a}$$

$$\text{s.t.} \quad Mz_k < b, \tag{2.50b}$$

$$u_m[k+i] \leq u[k+i] \leq u_M[k+i], i = 0 \dots p-1 \tag{2.50c}$$

$$\Delta u_m[k+i] \leq \Delta u[k+i] \leq \Delta u_M[k+i], \tag{2.50d}$$

$$i = 0 \dots p-1 \tag{2.50e}$$

where

$$z_k = \begin{bmatrix} u[k]^T \, f_c^T[k] \, u[k+1]^T \, f_c^T[k+1] \, \dots \\ \dots \, u[k+p-1]^T \, f_c^T[k+p-1] \end{bmatrix}^T \tag{2.51}$$

is the QP decision variable containing the input vector $u[k+i]$ (that collects the linear and angular acceleration of the robots) and $f_c[k+i]$ (that collects the forces imposed on the pushed object) for $i = 0, \dots, p-1$. As will be detailed in Section 2.4.6, such a definition of the decision variable allows us to consider the physical limitations of the robot inside the QP problem, even though the contact forces are not considered in the robot model. Besides, the cost function in (2.50) is defined as the following quadratic function

$$J(z_k, e[k]) = e[k+p]^T P e[k+p] + \sum_{i=0}^{p-1} \left\{ \left(e[k+i]^T Q e[k+i] \right) \right\} + \tag{2.52}$$
$$+ \sum_{i=0}^{p-1} \left\{ \left(u[k+i]^T R_u u[k+i] \right) + \left(\Delta u[k+i]^T R_{\Delta u} \Delta u[k+i] \right) \right\}.$$

The diagonal matrices $Q, P \in \mathbb{R}^{5\times5}$ provide the weights associated with each state variable, while $R_u, R_{\Delta u} \in \mathbb{R}^{2\times2}$ contain the weights on the amplitude of the input and the amplitude of its rate of change respectively. The terminal weight P is introduced to improve stability, as discussed in [42]. These matrices are all positive semidefinite. Inequality (2.50b) expresses a pushing interaction constraint, which will be described in details in Section 2.4.6. Expression (2.50c) imposes upper and lower limits on the elements of the QP decision variable z_k, while (2.50e) sets limits on its rate of change. These constraints are imposed to guarantee the feasibility of the solution, taking into account the physical limitations of the robot actuators.

2.4.6 Nonlinear Pushing Constraints for Object Slippage Avoidance

Since the robot is subject to nonholonomic constraints, it cannot change its orientation instantaneously. The direction of the force applied to the pushed object is thus constrained as well. Besides, as previously discussed, the pushing force must be restricted within the friction cone to avoid object slippage during manipulation. Hence, we will now introduce a constraint for the robot motion, such that the contact between the robot and the object does not break. This guarantees that the movement of the robot produces valid pushing forces, that lie within the friction cone. As a consequence, the input for the robot does not generate any relative motion between the object and the robot itself.

The concept above is implemented imposing the following constraints

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} + \omega_r \times R(\theta_r) p_{or} = \begin{bmatrix} \dot{x}_o \\ \dot{y}_o \\ \dot{\theta}_o \end{bmatrix} = H G f_c, \tag{2.53}$$

where $p_{or} \in \mathbb{R}^2$ is the position of the object in the robot frame Σ_r and $R(\theta_r) \in SO(2)$ is the rotation matrix between Σ_r and Σ_w. The left-hand side of (2.53) represents the velocity that the object would have if the robot-object system were moving as a rigid body (i.e., no relative motion). The right-hand side expresses the motion of the object due to contact forces, as explained in Section 2.4.2.

In order to include (2.53) inside the optimization problem (2.50), some adjustments are required. In particular, to ensure that the contact forces lie inside the friction cone, each component of f_c is bounded to be greater than zero. The equality constraint in (2.53) is thus converted into a set of two double inequalities, of the form $g(z_k) \leq 0$, and linearized, at each time step k, around the point $(\tilde{e}[k], v[k], p_{or}[k])$. Matrix $M \in \mathbb{R}^{6p \times 6p}$ in (2.50b) is finally defined as the Jacobian matrix of the left-hand side of the inequalities, computed with respect to variable z_k, while vector $b \in \mathbb{R}^{6p}$ is a zero vector.

Controller Implementation and Experiments

In this section, we discuss the implementation of the pushing system and the results of three different manipulation tests, which are representative of different operative conditions. The robot used during the simulation is a Pioneer 3-DX, a differential drive mobile robot with two actuated wheels and a castor wheel. The robot is equipped with a pushing bumper attached on the front. The robot receives velocity commands v_r, ω_r through ROS [39]. At each time step k, with period $T_s = 0.1s$, the following procedure is performed. The controller first sends the velocity command to the robot, then collects the data required to predict the future behaviour and generate the linearized models. The solution of the quadratic problem discussed in section 2.4.5 is then computed and used to generate the velocity command for the future step.

Algorithm 3: Feedback procedure

```
1 send previous $(v_r, \omega_r)$
2 get $e[k], v[k]$
3 $\tilde{e}[k] \leftarrow$predict($e[k]$,$v[k]$,$z_{k-1}$)
4 $A_k, B_{u_k}, B_{v_k}, M \leftarrow$linearize($\tilde{e}[k], v[k]$)
5 $z_k \leftarrow$ QP($A_k, B_{u_k}, B_{v_k}, M, \tilde{e}[k], v[k]$)
6 $(v_r, \omega_r) \leftarrow (v_r, \omega_r) + T_s u[k]$
7 $z_{k-1} \leftarrow z_k$
```

Two case studies have been carried out in simulations, performed on a laptop with an Intel Core i7-4510U using the CoppeliaSym physics simulator. The controller is written in MATLAB, while ROS handles the communication with the simulator. The gains are experimentally tuned to $Q = diag([15, 20, 5, 1, 1])$, $P = 50Q$, $R_u = diag([0.1, 0.1, 0.001, 0.001, 0.001, 0.001])$, and $R_{\Delta u} = diag([0.1, 0.1, 0, 0, 0, 0])$.

Tracking of a straight line

The first case study we propose is the tracking of a straight line starting with an offset. Several simulations have been performed, with the robot starting its movement with initial error state $e(0) = \begin{bmatrix} 0 & \varphi & 0 & 0 & 0 \end{bmatrix}^T$, with varying $\varphi \in \{0.1, 0.2, 0.4, 0.5\}$. A representative run of the simulations, performed with $\varphi = 0.2$, is discussed hereafter. The pushed object is placed in contact with the robot in a centered position. Figure 2.20 top graph depicts the planar movement of the robot and the object, measured for a representative run of the simulations. The yellow line represents the desired trajectory while the blue line and red line depicts the movement of the robot and the object, respectively. The figure clearly shows that an initial position offset can be corrected using the proposed controller and constraints. Figure 2.21 shows a comparison of the y components of the object position with respect to Σ_r, while being pushed, with and without the presence of the constraint in the controller. The application of the constraint significantly reduces the amplitude of the movement of the pushed object. The same conclusion can be extracted from the bottom graph of Figure 2.20, that shows the positions of the object and robot controlled without the constraint (2.53). Moreover, Figure 2.20 shows that, during the manipulation, the movement of the robot causes an interruption of the pushing line contact, also reflected in the spike visualized in Figure 2.21, that results in a loss of control over the movement of the object and ultimately in a loss of quality of the manipulation. The proposed controller

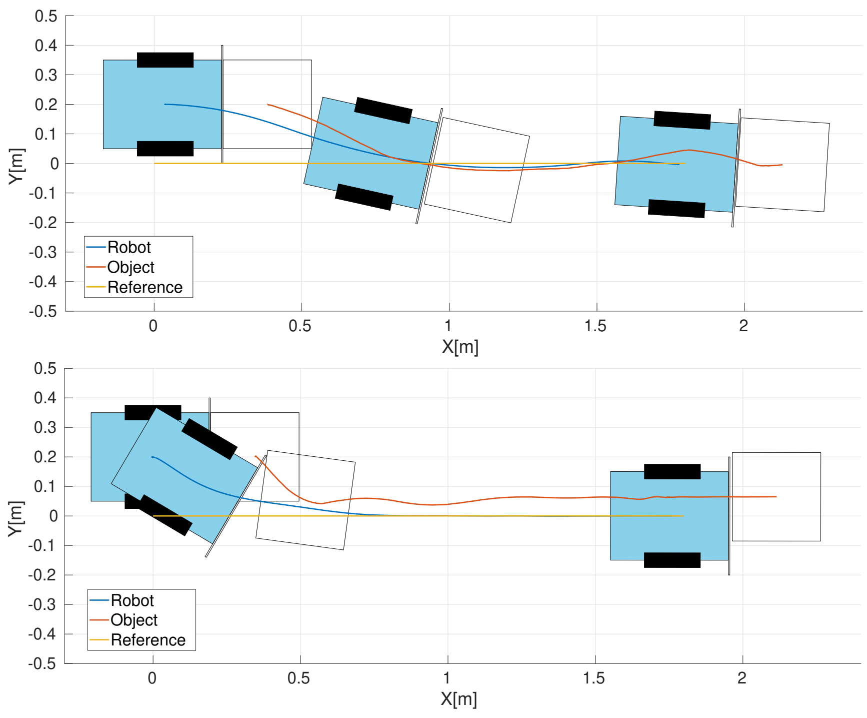


FIGURE 2.20: Line tracking from a non-zero initial error state. Top graph: controller with the proposed constraint, bottom graph: without the proposed constraint.

and constraints maintain the line contact, with a final average object position error less than 0.01 m, while the absence of the constraint on the motion leads to an average error greater than 0.05 m.

Complete manipulation task

The second case study we explore is a complete manipulation task. To complete the task, the robot transports the object to a desired configuration performing a series of pushing actions. Once a pushing maneuver is completed, the robot performs a repositioning maneuver to change the pushing side before starting the next action. During the maneuver the robot steps back from the object, goes around the object along a circular trajectory and then approaches slowly the object until the contact is established. Several simulations have been performed, considering different trajectories composed of straight and curve segments. The trajectory traveled by the robot and the object during a representative run of the simulations is depicted in Figure 2.22. In this task the robot transports the object from the initial position $\chi_o(0) = \begin{bmatrix} 0.0 & 0.0 & 0.0 \end{bmatrix}^T$ towards the desired configuration $\begin{bmatrix} 1.65 & 0.15 & \pi \end{bmatrix}^T$ performing three pushing actions on the object. The results indicate that, with the use of the proposed controller, it is possible to track a curved line to transport the package without significant accumulation of error, that implies that the robot, when an appropriate reference trajectory is provided, can manipulate the object across the environment. The final positon error is 0.04 m while the final orientation error is 1.8 deg.

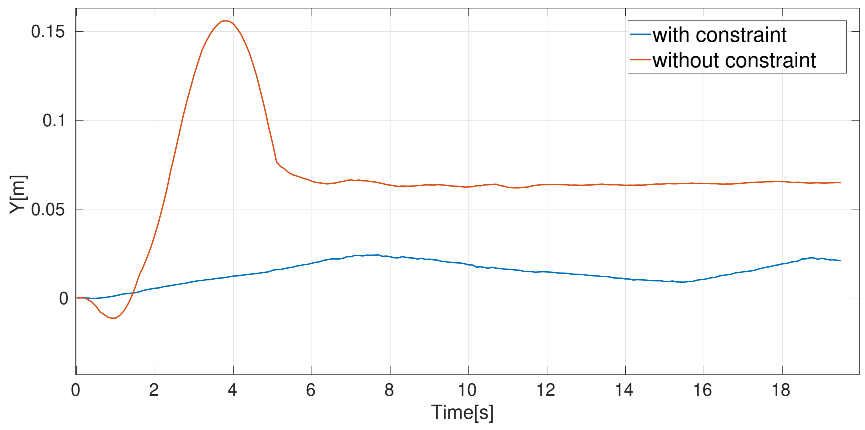


FIGURE 2.21: Y coordinates of the pushed object with respect to Σ_r during the manipulation.

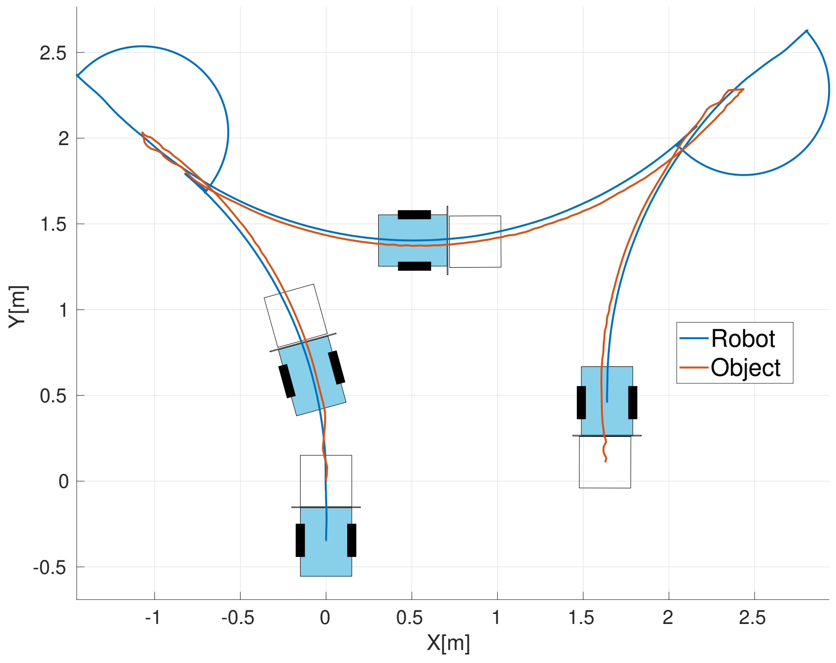


FIGURE 2.22: Complete manipulation of an object.

2.4.7 Linear Constraints for Pushing Contact Maintenance and Obstacle Avoidance

Despite proving successful in a simulated environment, the nonlinear motion constraint presented previously represents a heavy burden for the QP solver used in the LTVMPC controller, making it unreliable for real world usage. A new constraint formulation is therefore necessary to achieve a controller that can operate in real time. To that extent we propose a new constraint, based on the results described in Section 2.1.1. As visualized in Figure 2.2, the curvature constraint requires that the robot moves around a center of rotation on the robot's orthogonal axis with a positive linear velocity v_r.

The constraint can be expressed as:

$$\begin{cases} \frac{v_r}{\omega_r} \geq R_{lim+} + y_{or}, & \text{if } \omega_r \geq 0 \\ \frac{v_r}{\omega_r} \leq R_{lim-} + y_{or}, & \text{if } \omega_r < 0 \end{cases} \tag{2.54}$$

where y_{or} represents the y coordinate of the object's position with respect to the robot. With this formulation the constraint is not convex and therefore not suited to be inserted in the optimization algorithm. Since the linear velocity v_r of the robot has to be positive for the robot to push, and by definition $R_{lim+} > 0$ and $R_{lim-} < 0$, then the constraint can be rewritten as

$$\omega_r - \frac{v_r}{R_{lim+} + y_{or}} \leq 0 \tag{2.55}$$

$$\frac{v_r}{R_{lim-} + y_{or}} - \omega_r \leq 0 \tag{2.56}$$

With this representation, the constraint is convex and does not rely on the sign of ω_r. Applying the definition of the error vector (2.37) we obtain the final form of the constraint as:

$$\begin{bmatrix} \frac{-1}{R_{lim+}+y_{or}} & 1 \\ \frac{1}{R_{lim}+y_{or}} & -1 \end{bmatrix} \begin{bmatrix} e_{vr} \\ e_{\omega r} \end{bmatrix} + \begin{bmatrix} \frac{-1}{R_{lim+}+y_{or}} & 1 \\ \frac{1}{R_{lim}+y_{or}} & -1 \end{bmatrix} \begin{bmatrix} v_{rd} \\ \omega_{rd} \end{bmatrix} \leq 0 \tag{2.57}$$

This constraint is equivalent to the previous one but improves on the previous formulation by expressing the constraint using a time-invariant linear system, easier to enforce in the QP problem. Furthermore, this constraint does not require the inclusion of the contact forces in the system input variables, reducing the dimension of the problem, further improving the speed of the algorithm. The constraint is applied across the prediction horizon considering the y position of the object with respect to the robot to remain constant.

As an additional constraint to insert in the MPC controller, we propose a collision avoidance constraint. The purpose of the collision avoidance constraint is to keep the robot at a safe distance δ from each obstacle o in the set $\mathcal{O}$ of obstacles. In other words, the distance between (x_r, y_r) and (x_o, y_o) has to be greater than δ.

$$\sqrt{(x_r - x_o)^2 + (y_r - y_o)^2} > \delta, \forall o = (x_o, y_o) \in \mathcal{O} \tag{2.58}$$

The constraint formulation is inherently non-convex and therefore not suited to be inserted in a linearly constrained QP problem. However it is possible to obtain the same objective formulating the constraint using a convex approximation of the available space. In [51] the authors propose a convex approximation of the feasible space for the robot as the largest polyhedron containing the robot but not the obstacles. The formulation used in the controller considers each obstacle as circular object of radius δ. At each iteration the closest point in the circular perimeter of each obstacle

o_i to the robot is computed as:

$$q_i = \delta \frac{p_r - o_i}{\|p_r - o_i\|} \tag{2.59}$$

The subspace of feasible positions for the robot is computed as $\mathcal{P} = \{p \in \mathbb{R}^2 : A_c p \leq b_c\}$, where, as detailed in [51],

$$A_c = \begin{bmatrix} (q_1 - p_0)^T \\ \vdots \\ (q_M - p_0)^T \end{bmatrix}, \; b_c = \begin{bmatrix} (q_1 - p_0)^T q_1 \\ \vdots \\ (q_M - p_0)^T q_M \end{bmatrix} \tag{2.60}$$

where p_0 is the current position of the robot and p represents the position of the robot along the prediction horizon. Introducing the error formulation (2.38) in the definition of $\mathcal{P}$ leads to the following formulation:

$$A_c R \begin{bmatrix} e_{rx} \\ e_{ry} \end{bmatrix} + A_c \begin{bmatrix} x_{rd} \\ y_{rd} \end{bmatrix} \leq b_c \tag{2.61}$$

$$R = \begin{bmatrix} \cos\theta_{rd} & -\sin\theta_{rd} \\ \sin\theta_{rd} & \cos\theta_{rd} \end{bmatrix} \tag{2.62}$$

In the optimization problem, the constraints (2.57) and (2.61) are related to the decision variable z_k using the linearized discrete time model (2.46) recursively over the prediction horizon. The final constraint formulation (2.50b) used in the optimization is composed stacking (2.57) and (2.61).

Constraint validation

The execution of a computed pushing plan is highly dependent on the ability of the controller to guide the movement of the robot to execute a push maintaining contact with the object. Therefore the proposed controller is validated again by placing the pushing robot in contact with the object and instructing it to track a linear trajectory starting with an offset on the robot's local y direction.

The robot used is an e-puck [52] differential drive robot equipped with a front bumper. The position of the robot and the object are obtained via an Optitrack Motion Capture System [53]. The manipulated object is a square box with side length of 9 cm. Figure 2.23 shows the behaviour of the robot tracking the linear trajectory in the test environment. The robot moves to correct the initial offset starting from either side of the reference line. The pushing constraint expressed in (2.57) maintains the contact between the robot and the object during the pushing manipulation, validating the ability of the controller to execute the assigned pushing task. Furthermore, the controller is able to solve the optimization in real-time, guaranteeing a smooth manipulation operation.

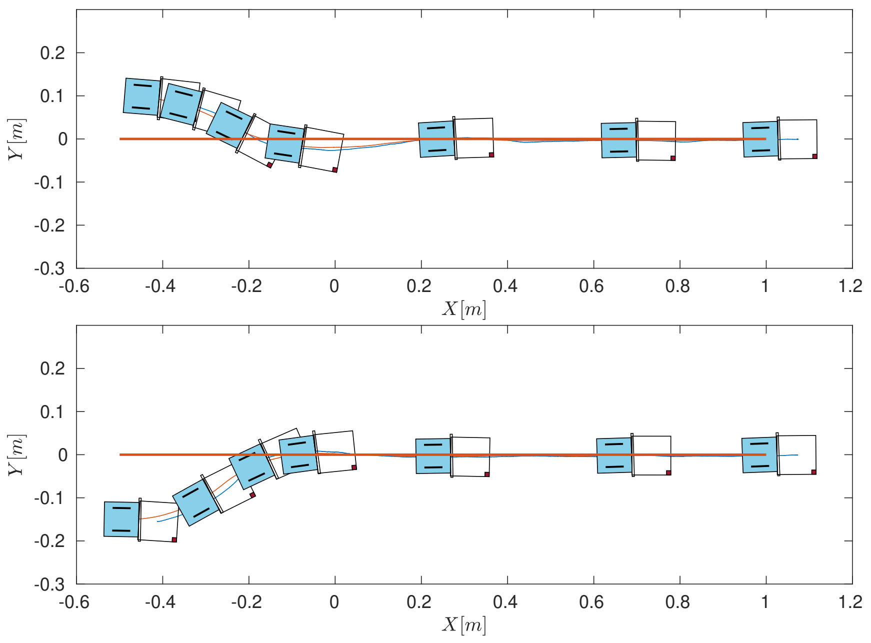
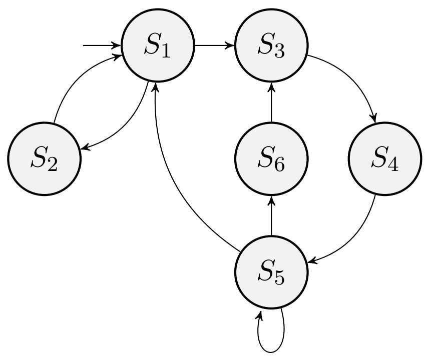


FIGURE 2.23: Performance of the robot manipulating the object to track a linear trajectory. The robot can correct the initial offset from each side.

2.5 Manipulation Supervision

This section introduces the algorithm used to manage the interaction between the planner and the controller as well as the execution of the pushing tasks. The algorithm implements a Finite State Machine (FSM) that handles the reception of a new target pose for the object, requests the pushing sequence from the planner and commands its execution by the controller. The FSM also handles pushing sides change and possible failures of the planning algorithm. The state diagram of the FSM can be visualized in Figure 2.24a, while the naming of the states is provided in Table 2.24b. Upon initialization, the algorithm loads all the required geometrical parameters as well as the map of the environment as an occupancy grid. The initial state of the machine



(A)

State	Name
S_1	Idle State
S_2	Planning State
S_3	Positioning State
S_4	Approach State
S_5	Pushing State
S_6	Detaching State

(B)

FIGURE 2.24: State diagram of the Supervision FSM with table of associated names.

is S_1, the Idle state. When a new target pose for the object is received, the machine transitions to S_2 to compute a new plan. A planning request, composed of 1) the initial object pose, 2) the requested target pose, 3) the map of the environment as an occupancy grid is sent to the planning algorithm. Upon success, the state returns to S_1 waiting for an execution command from the user. If the planner reports a failure, the supervisor computes a pushing action to apply before any other actions in order to change the initial pose of the object used to compute the plan. The action is computed selecting a random motion curvature that respects the constraints, computing the resulting object poses deriving from a push with said curvature on all the object's sides, using forward integration on the robot's model. The computed poses are checked for collision with the occupancy grid discarding the unfeasible ones and then sorted based on their distance to the target. A new planning request is then sent using the closest new pose as the initial pose of the object. The request is repeated iteratively across the poses of the sorted list till success is reported by the planner. The FSM enters S_3 only if a valid pushing plan has been computed. In the Positioning state S_3 the algorithm sends a static target to the controller along with a list of obstacles. The obstacles are composed by the target object's pose and the poses of the occupied cells in the occupancy grid map. The static target is computed as a position for the robot that faces the object's side to be pushed at a safe distance. Once this target has been reached, the state machine switches to the Approach state S_4 and the robot is tasked to approach the proper pushing position for the given pushing side, thus the position of the manipulandum has to be removed from the list of obstacles. When the approach is complete, the FSM state changes to S_5 and the pushing trajectory is sent to the controller to be executed, the algorithm then waits for completion. Upon completion of this manipulation, if there is another trajectory to execute, the algorithm checks if the next pushing side is equal to the current one, and in that case the cycle repeats from S_5 but if the side is different the state switches to S_6. In S_6 the robot is instructed to move backwards till a safe distance is reached. The cycle then starts back from S_3 and continues until all the trajectories have been completed or an error occurs. A pseudo code description of the algorithm for pushing execution is provided in Algorithm 4.

2.6 Framework Validation

The combination of the planning algorithm presented in Section 2.2 and the controller from Section 2.4, coordinated by the aforementioned supervision algorithm compose a framework to enable a single robot to perform a pushing manipulation on a polygonal object. The evaluation of the complete proposed framework is now discussed. The tests have been conducted manipulating the target square box in three different environments performing three different targets in each environment. The experiments are conducted in a testing area equipped with an Optitrack Motion Capture System [53]. The robot used to manipulate is an e-puck differential drive mobile robot.

The complete testing environment is shown in Figure 2.25.

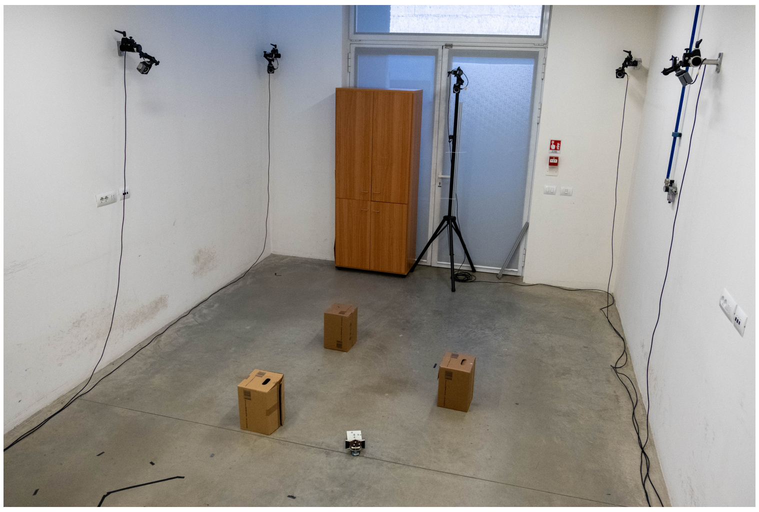


FIGURE 2.25: Testing environment composed of a motion capture system estimating the poses of the robot and the pushed object.

2.6.1 Environment 1

The first map introduced, visualized in Fig.s 2.27a, 2.27b, 2.27c, and 2.26, presents a narrow passage from the left side to the right side. The manipulandum is initially placed on the bottom left side of the map while all the three targets are on the right side, two on the top half with different orientations, and one on the bottom half of the map. Figure 2.27a shows the tests performed on the first target set to the point $(1.3, 0.75)$ m with desired orientation of zero radians. In this scenario, the manipulation does not require a change in the pushing side and the robot can manipulate the object closely tracking the reference trajectory. The tests visualized in Fig. 2.27b display a particular case in which the robot is required to change the pushing side to complete the manipulation. The path taken by the robot is visualized in Fig. 2.26. As detailed in Section 2.5, the robot first detaches from the object, then circles around it before approaching the next pushing position. In this scenario, the target position is the same as in the first set of tests, equal to $(1.3, 0.75)$ m, but the requested orientation is set to $\pi/2$ rad. In one instance of these tests, the robot deviates a bit from the reference trajectory while entering the narrow passage. Still, the constraint applied on the controller maintains the contact with the object allowing the successful recovery of the manipulation. The third considered scenario in this obstacle environment presents a target in the lower half of the map, at $(1.3, -0.65)$ m with planar orientation $-(4/5)\pi$ rad. In this scenario, depicted in Fig. 2.27c, it can be noted that multiple turns in the same direction can generate a slight shift of the contact towards the outside of the curve. This can be attributed to an erroneous estimate of the frictional parameters.

Algorithm 4: Pushing Execution

```
input  : PUSHES: ordered list of trajectories to execute and associated
         pushing sides.
1  while PUSHES is not empty do
2      select and pop first element of PUSHES;
3      compute approaching and pushing poses for the robot;
4      instruct controller to move towards approaching pose;
5      monitor until completion  instruct controller to move towards pushing
        pose;
6      monitor until completion;
7      instruct controller to perform selected pushing trajectory;
8      monitor until completion;
9      if PATHS is not empty then
10         if pushing side of next trajectory is not equal to the current pushing
            side then
11             instruct controller to move towards approaching pose;
12             monitor until completion;
13         else
14             select and pop first element of PUSHES;
15             instruct controller to perform selected pushing trajectory;
16             monitor until completion;
```

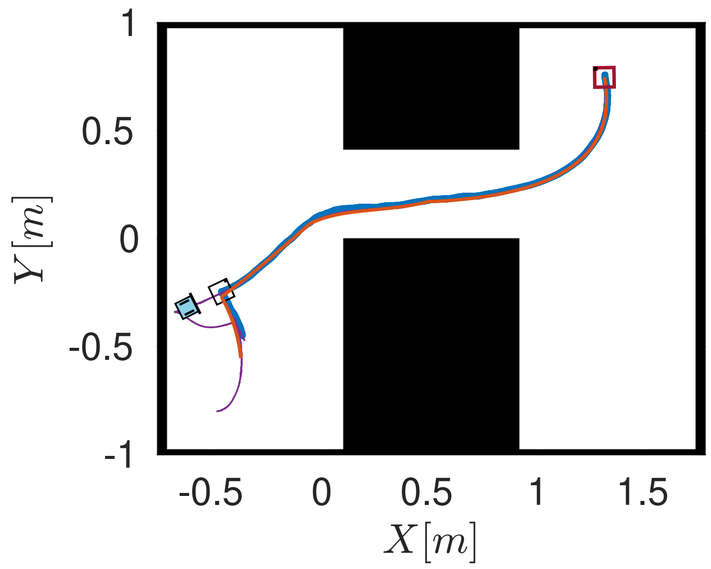
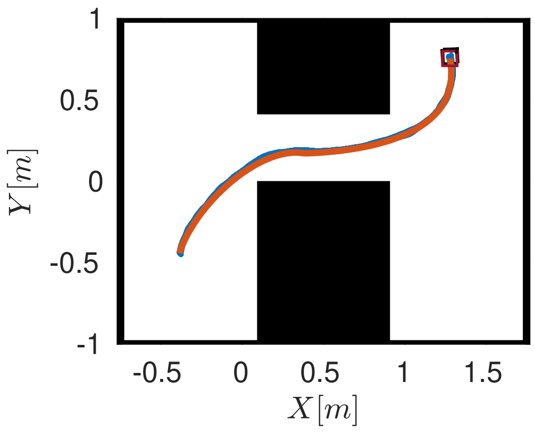
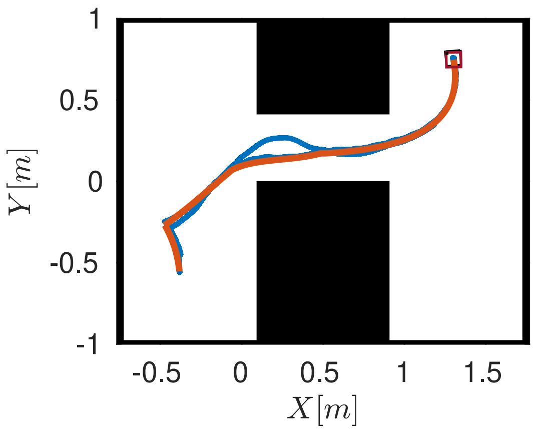


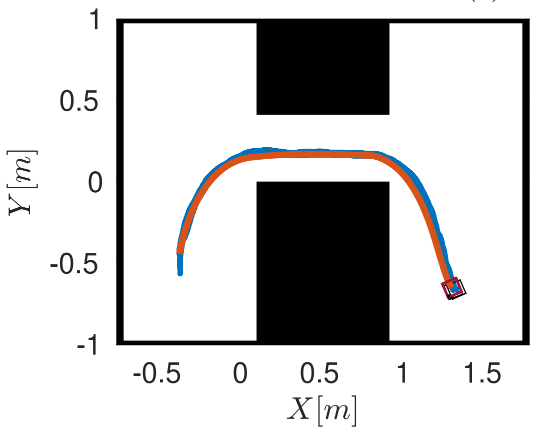
Figure 2.26: Visualization of a test case of Scenario 2. The purple line displays the path taken by the robot during the manipulation. The robot is depicted during the side change manoeuvre to reorient with the object.



(A) Scanario 1



(B) Scenario 2



(C) Scenario 3

FIGURE 2.27: Environment 1: Manipulation of a square object across a narrow passage. The orange line displays the reference trajectory for the object while the blue lines represent the motion of the object during the tests. The reference final position for each manipulation is drawn in red while the final position for each manipulation is drawn in black.

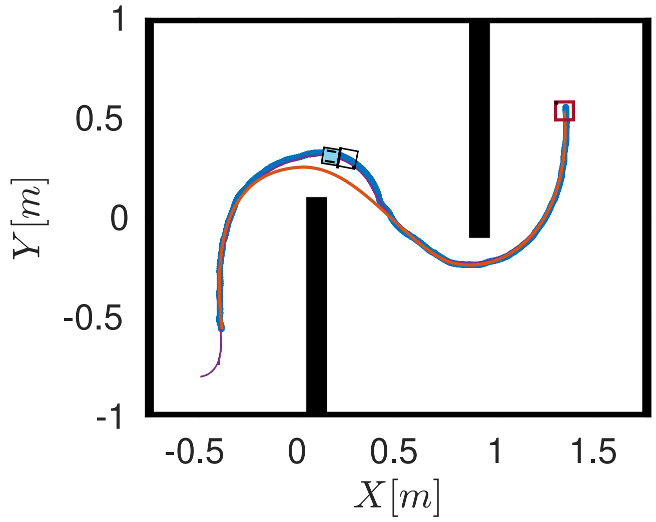


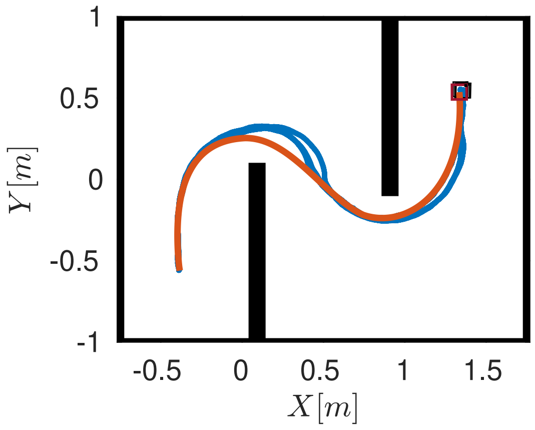
FIGURE 2.28: Visualization of a test case of Scenario 4. The purple line displays the path taken by the robot during the manipulation. A visualization of the robot and the manipulandum during a particular drift from the reference trajectory is provided to showcase the ability to maintain the contact.

2.6.2 Environment 2

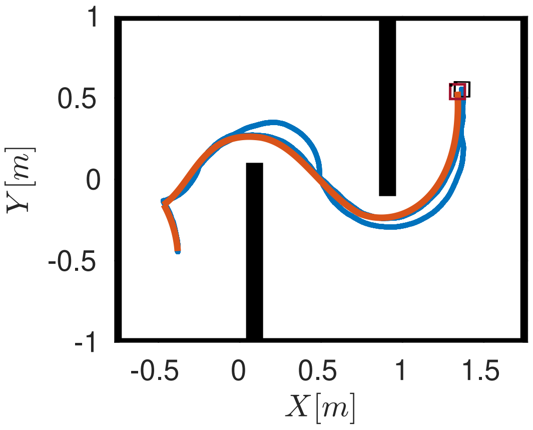
The second map considered in the testing presents an N-shaped free space. This forces the robot to manipulate the object performing tight curves to avoid the obstacles. The manipulation towards the first target is shown in Fig. 2.29a. The target is placed in $(1.3, 0.5)$ m with an orientation of zero radians. The robot approaches the object from the lower side and performs a single pushing manipulation. The results show that the robot struggles to track the trajectory around tight corners, the controller constraints again enforce contact maintenance, allowing the manipulation's completion, despite a contained loss in final accuracy. A representation of the robot during the manoeuvre is provided in Fig. 2.28.

The second target is placed in the same position but rotated of $\pi/2$ rad counter-clockwise. To perform this manipulation, the planner constructs a sequence of two manipulations that require a pushing side change. This plan is built after a failed try from the planner, forcing the supervisor to generate a preemptive pushing action with constant curvature. The resulting manipulation is visualized in Fig. 2.29b. After performing this preemptive action, the robot faces the same tracking issue again as the previous target. Still, likewise, the controller can maintain the contact and complete the manipulation.

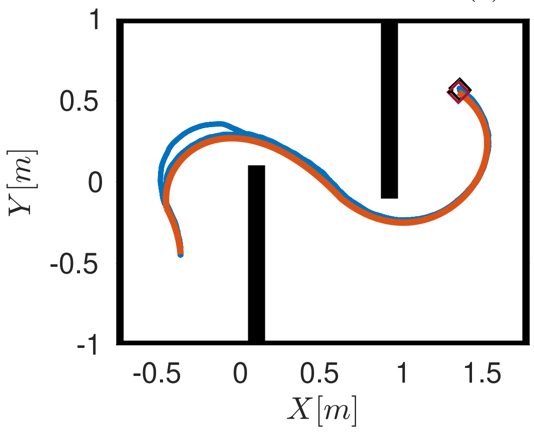
The third target is again placed in the same position but rotated by $\pi/4$ rad. Like the previous test, the planner initially fails to find a suitable manipulation sequence, and the supervisor is tasked with generating a preemptive action. However, in this scenario, it was not necessary to change the pushing side between actions. As shown in Fig. 2.29c and differently from the previous cases, only one manipulation run struggled to remain close to the reference trajectory.



(A) Scanario 4



(B) Scenario 5



(C) Scenario 6

FIGURE 2.29: Environment 2: Manipulation of a square object across a N shaped environment. The orange line displays the reference trajectory for the object while the blue lines represent the motion of the object during the tests. The reference final position for each manipulation is drawn in red while the final position for each manipulation is drawn in black.

2.6.3 Environment 3

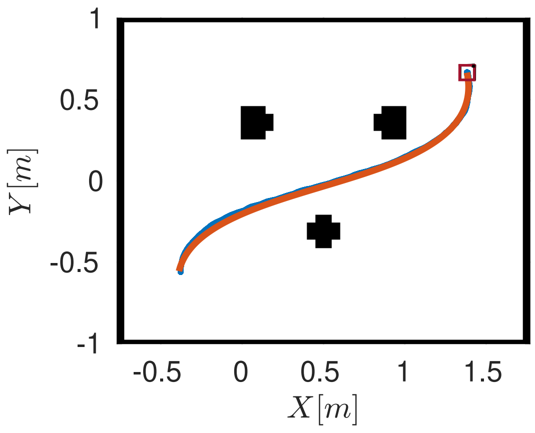
Figures 2.30a, 2.30b and 2.30c display the third test map considered in the evaluation. The map presents three obstacles placed around its center to form a triangle. The three tests in this scenario are constructed to force the manipulation to pass between the obstacles to evaluate the ability of the proposed framework to generate manipulations around various sets of obstacles. In the previous tests, the overall path to the target was forced by the shape of the environment. In this scenario, multiple routes could be taken to reach the desired pose of the object.

Each of the three targets was manually selected to be aligned diagonally across the environment from the starting position; therefore, the shortest path to reach the target is obtained by crossing the triangle formed by the three obstacles. The first manipulation is composed of a single polynomial curve that brings the object towards the target, as depicted in Fig. 2.30a. The robot easily transports the object following the reference trajectory.

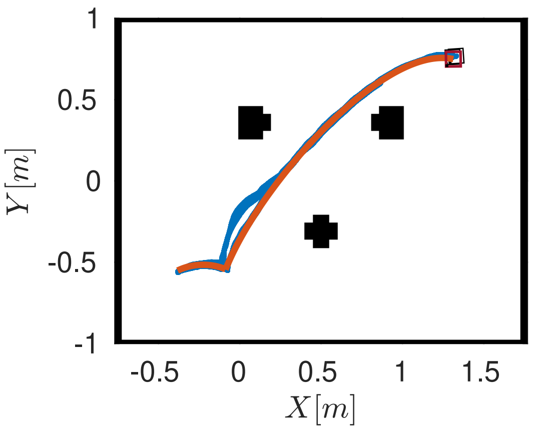
To reach the second target, it is required to perform an initial manipulation to orient the object correctly. In some tests, the performance of this introductory manipulation resulted in a small orientation error. To correct this, the supervisor algorithm tasks the robot to realign with the object during the change of pushing side. The consequent offset in the initial position for the final manipulation forced the controller to drift away from the target trajectory to realign and later reduce the error with the manipulation's reference trajectory. In Scenario 9, the manipulandum is placed initially near the top left corner of the map, and the robot is tasked to transport it towards the bottom right corner. The planner finds a single polynomial pushing trajectory that connects to the target passing through the obstacles. Table 2.1 presents aggregate information on the results of the performed tests. The average position and orientation error and associated standard deviation σ are reported, as well as the maximum error value obtained across all trials. The results show that the presented framework produces a very precise final orientation, and the position error is maintained within 20% of the object's footprint.

2.7 Future Works

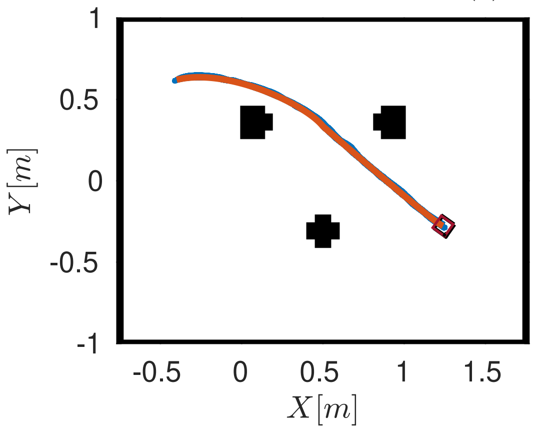
In this chapter, we proposed a framework for planning, controlling, and executing planar manipulations using only pushing actions applied by a single mobile robot. The framework can generate a manipulation plan composed of a sequence of pushing actions and perform the manipulation by adequately controlling the motion of the pushing robot. The robot's movement is constrained during the manipulation to avoid the slippage of the contact with the manipulandum and avoid collision with the environment. The framework was tested, performing different manipulations in three different settings. The provided results indicate that the proposed framework effectively solves the manipulation problem in the considered scenarios. In the presented



(A) Scanario 7



(B) Scenario 8



(C) Scenario 9

FIGURE 2.30: Environment 3: Manipulation of a square object in an environment with three circular obstacles. The orange line displays the reference trajectory for the object while the blue lines represent the motion of the object during the tests. The reference final position for each manipulation is drawn in red while the final position for each manipulation is drawn in black.

TABLE 2.1: Summary of position and orientation error among the analyzed test scenarios. The position error is computed as the norm of distance between the final object pose and the desired object pose. The orientation error is the absolute value of the difference between the final orientation and desired orientation.

	Position error [m]			Orientation error[rad]		
Scenario	avg	σ	max	mean	σ	max
1	0.0130	0.0072	0.0200	0.0456	0.0146	0.0622
2	0.0269	0.0274	0.0585	0.0698	0.0435	0.0985
3	0.0114	0.0113	0.0292	0.0365	0.0178	0.0476
4	0.0216	0.0046	0.0268	0.0120	0.0145	0.0284
5	0.0313	0.0028	0.0333	0.0090	0.0050	0.0126
6	0.0195	0.0055	0.0246	0.0099	0.0062	0.0168
7	0.0045	0.0019	0.0065	0.0261	0.0216	0.0402
8	0.0193	0.0086	0.0254	0.0461	0.0149	0.0566
9	0.0092	0.0058	0.0150	0.0117	0.0128	0.0314

work, we considered the manipulation of known polygonal objects. We aim to generalize the motion constraints to more general shapes in future work, where the center of mass is not known a priori. Furthermore, the framework currently relies on the precise localization of the objects in the environment. Future work will extend it by using on board sensors to estimate the relative pose, removing the necessity of an external localization system. Finally, future work will consider scenarios where unknown obstacles are present in the environment, identifying their position and adapting the manipulation plan.

Chapter 3

Pushing Manipulation with a few Robots

In this chapter the research is shifted toward the use of multiple robots as a source of pushing motion. The use of multiple robots can provide considerable benefits in pushing applications due to the higher flexibility of the system. A multi-robot oriented pushing strategy could enable a distributed energy expense, as well as more resilient architectures, since the robots are usually interchangeable and less expensive. The first challenge tackled is the identification of a good contact configuration for the robots by providing a characterization of the contact configuration with respect to a nonprehensile manipulation task, performed with a team of robots. Following state-of-art examples in selecting optimal force-closure grasps, we rank a set of configurations using various indices suitably adapted for nonprehensile manipulation. Following the search for the optimal contact configuration, a task-oriented contact optimization procedure has been developed. The procedure solves a constrained non-linear optimization problem to identify the configuration that is best suited to fulfill a planar manipulation task. The introduction of a control scheme to command mobile robots performing pushing actions was the last challenge tackled in this research. Knowing informations about the motion model of the pushed object, the requested contact configuration and the manipulation task to perform, the control scheme instructs the robots into moving in a coordinated way, maintaining the appropriate contact configuration. The strategy is validated performing multiple manipulation tasks both in simulation and using real robots.

The rest of the chapter is organized as follows: Section 3.1 studies the problem of finding a characterization for the quality of a contact configuration and illustrates the proposed approach to identify such characterization. The evaluation of the suitable characterizations is carried forward in Section 3.1.4 while the results are discussed in Section 3.1.5. A possible application of this characterization is shown in Section 3.1.6. Section 3.2 formulates the problem of finding a task oriented optimal configuration whose solution is tackled in Section 3.2.1. Section 3.2.2 describes the control scheme used to command the mobile robots. The results of the performed tests are detailed in Section 3.2.3.

3.1 Grasp Quality Characterization

Consider a group of n robots performing a manipulation task on an object in the plane. In the addressed task, we assume to know in advance both the trajectory to follow and the frictional properties of the pushed object. By assumption, each robot is omnidirectional, it can touch the object at a single point, and if the objects has vertices (i.e. a polygonal prism) it cannot touch the object in a corner. The omnidirectional robots that interact with the object are thus modeled as circular objects free to move on the plane. This assumption is introduced to exclude any external effects that may arise from motion constraints on the pushing agents so that the characterization is focused purely on the pushing interaction. Let Σ_W and Σ_O be the global reference frame and the object local frame, respectively. Let the pose of the object at time t be represented as the vector

$$\chi_o(t) = \begin{bmatrix} x_o(t) & y_o(t) & \theta_o(t) \end{bmatrix}^T \in \mathbb{R}^3$$

where $x_o(t), y_o(t) \in \mathbb{R}$ represent the position of Σ_O with respect to Σ_W, while θ_o represents its orientation. Since, in this work, we consider robots interacting with the object by means of pushing, we will hereafter refer to them as *pushing agents*. The position of each pushing agent is represented in Σ_W as $p_i(t) = \begin{bmatrix} x_i(t), y_i(t) \end{bmatrix}^T \in \mathbb{R}^2$, with $i = 1, \ldots, n$. An illustration of the aforementioned quantities is presented in Figure 3.1. The position of the pushing agents around the object can be collectively represented as $p(t) = \left[p_1^T(t) \ldots p_n^T(t)\right]^T \in \mathbb{R}^{2n}$, and is referred to as the *contact configuration.*

Let $F_o \in \mathbb{R}^3$ be the total wrench on the object at its center of mass. Let $F_{ci} \in \mathbb{R}^2$ be the contact force applied by the i-th pushing agent on the object at the contact point, with $i = 1, \ldots, n$. The single contact wrenches can be collected into the vector

$$F_c = \left[F_{c1}^T(t) \ldots F_{cn}^T(t)\right]^T \in \mathbb{R}^{2n}.$$

The map between the contact forces and the total object wrench is called *grasp map* [43]. Since each contact map is linear, the wrenches can be stacked as long as they are all written in the same coordinate frame (i.e., the global reference frame Σ_W), and each contact force belongs to the relative friction cone, the net object wrench is

$$F_o = GF_c, \tag{3.1}$$

where $G \in \mathbb{R}^{3\times 2n}$ is referred to as *grasp matrix.* More details about how to build such a matrix can be found in [43]. In robotic prehensile manipulation, the grasp matrix plays a fundamental role in understanding whether a grasp has *force-closure.* In detail, force closure is present when the friction cone mapped through G is equal to the entire external wrench space [43]. On the other hand, in nonprehensile manipulation, the concept of force-closure grasp is meaningless because of the presence of unilateral

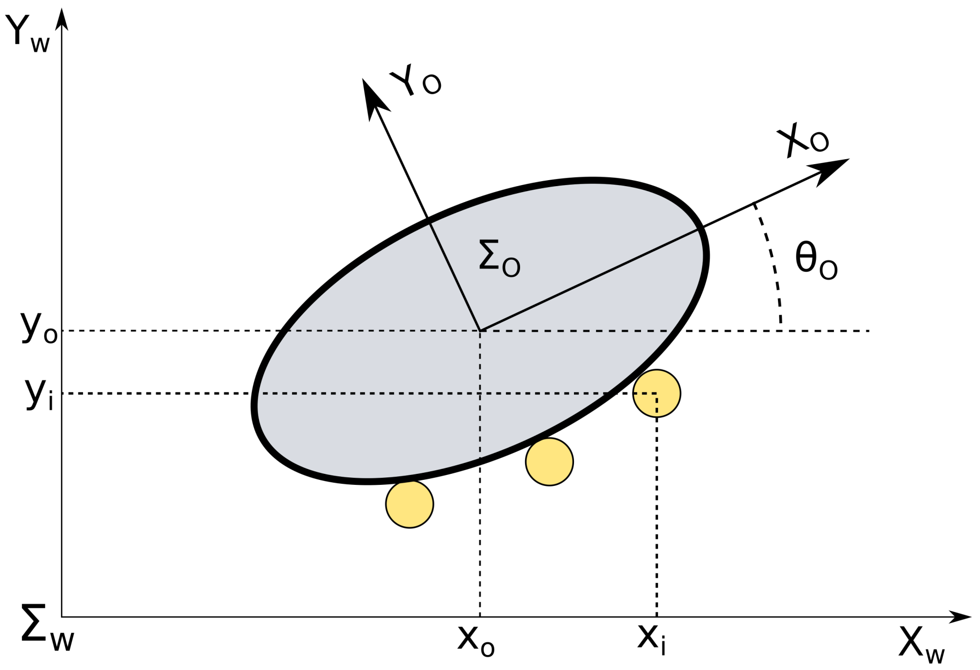


FIGURE 3.1: Schematic representation of the manipulated object (in gray) and the pushing agents (in yellow) with the related reference frames.

constraints only [2]. Nevertheless, it is still possible to define the grasp matrix G and (3.1) holds whenever the contacts are maintained, and the contact forces belong to the relative friction cones. For this reason, with a slight abuse of language, we will still refer to a grasp configuration also in the addressed case of nonprehensile pushing.

More specifically, we tackle the following problem:

Problem 3 *Characterize a nonprehensile grasp configuration with respect to a planar pushing task both in terms of efficacy and energy efficiency.*

In the following, as addressed in [9], we will assume that the motion is quasi-static (e.g., inertial forces are negligible with respect to the frictional and contact forces), and that the forces exchanged in contact interactions follow Coulomb's friction model. Furthermore, as carried out in [43], the friction parameters are assumed to be uniform across the environment, and all the contacts are rigid. Finally, we assume that the requested pushing manipulation is obstacle free since there exist multiple algorithms in the literature which are able to generate manipulation trajectories that consider the presence of obstacles in the environment [54].

The expression of the quality of a grasp configuration is useful only when it provides a quantitative insight of how suited it is with respect to a certain task to accomplish. For this purpose, we first define *quality metrics* for assessing the performance of the manipulation activity. Subsequently, we present a set of *configuration indices*, which provide aggregate information about the selected grasp configuration. Those indices are proposed starting from quantities that are commonly used in the traditional prehensile manipulation literature. They are here extended to the case of nonprehensile manipulation performed by a set of mobile robots. Furthermore, we provide a novel index formulation that takes into account both the required task and the applicable forces by a configuration, along with a compatibility test to assess if a configuration is capable of executing a given task. Along the same line, we provide evaluation results to demonstrate the correlation between the configuration indices and the quality metrics for the case of pushing.

3.1.1 Quality of a Nonprehensile Grasp

To understand what are the desired qualities of a grasp, it is useful to think about what a grasp is. Grasping is the most common approach used to interact with the environment. Children learn by grasping different objects with their hands, animals use grasping to interact with food or other animals, and even insects like ants use grasping to interact with leaves or crumbs. In other words, grasping is a tool used to achieve the goal of moving one or more objects to a different state. The tasks this tool has to fulfill in order to reach the goal are diverse but connected. First, the grasp has to be able to provide the necessary forces for the movement. Second, the grasp has to absorb the uncertainties that arise during the movement. Third, the grasp has to maintain the contact state across the movement. In other words, the grasp has to be effective and robust enough to sustain the movement.

In prehensile grasping, the robustness of the grasp is ensured by the concept of force closure we described in the previous section. Conversely, since nonprehensile grasping, by definition, cannot balance all the forces that might act on the object, the requirement is translated into maintaining or breaking the contact depending on the dexterity of the sought task. In the pushing scenario, the requirement is instead reduced to maintaining the contact state despite a range of possible disturbances.

Hence, we introduce two metrics, that must be minimized in the following, to evaluate (i) the *effectiveness* of the grasp, and (ii) the *energy efficiency* of the manipulation operation.

Effectiveness of the grasp

In this thesis, we evaluate the grasp effectiveness by measuring the relative overall displacement of each pushing agent to the manipulated object. This measures the ability to maintain the contact during a single pushing operation. For a given trajectory Γ of length $l_\Gamma > 0$ covered in a time T_Γ, we define the normalized overall displacement δ as

$$\delta = \frac{1}{l_\Gamma} \sum_{i=1}^{n} \left[\int_0^{T_\Gamma} ||v_i^O(\tau)|| \mathrm{d}\tau \right] \tag{3.2}$$

where $v_i^O \in \mathbb{R}^2$ is the velocity of the i-th pushing agent with respect to Σ_O. Even though in [12] it has been shown that a relative motion between pusher and manipulated object can be beneficial and exploited for control purposes, if no feedback control is considered regulating the contact position, the ability of a configuration to exert all the wrenches required by a task can be translated in the lack of relative motion between the pushers and the manipulated object, thus indicating the effectiveness of a grasp configuration with respect to a given manipulation task.

Energy efficiency

Different configurations may require different amounts of energy to maintain the grasp effectiveness during the manipulation. A configuration that requires a lower effort to maintain the contact during manipulation is intuitively preferable than one that requires a higher effort: therefore, energy can be used to rank effective grasps. Consider Γ to be a trajectory describing an ordered sequence of poses for the object. Let Γ_{ci} be the 2D trajectory of the i-th contact point in a configuration associated to Γ. The normalized energy spent by a configuration to execute a motion is computed as the sum of the mechanical work exerted by each contact force on the object, each normalized with respect to the length of the trajectory drawn by each contact point:

$$E = \sum_{i=1}^{n} \left[\frac{1}{l_{\Gamma_{ci}}} \int_{\Gamma_{ci}} f_{ci}^{T} \mathrm{d}\gamma_{ci} \right] \tag{3.3}$$

3.1.2 Grasp-Task Compatibility Test

When the grasping task is known, it is intuitively convenient to extract information a priori in order to optimize the performance. It is indeed possible to compute the set of wrenches acting on the object due to the friction with the plane. The contact configuration has to be capable of balancing these wrenches in order to perform the manipulation. In other words, the space of possible wrenches that a grasp can apply to the object has to contain the reaction wrenches from the task. Within prehensile grasping, as already mentioned above, this is ensured with force closure [3]. Within nonprehensile grasping, instead, force closure is meaningless.

Therefore, we propose the following procedure to check whether a grasp can apply the required wrench. The space of required wrenches $\mathcal{W}_{des}$ is extracted from the task using the friction limit surface ellipsoidal approximation model for the target object [46]. For each sample velocity $v \in \mathbb{R}^3$ of the assigned object trajectory, a corresponding friction wrench $w \in \mathbb{R}^3$ is computed using

$$w = \frac{Hv}{\sqrt{v^T H v}} \tag{3.4}$$

where $H \in \mathbb{R}^{3\times3}$ is the matrix expressing the ellipsoidal friction limit surface [46]. Then, the combination of all the wrenches is computed, representing the space $\mathcal{W}_{des}$.

For a configuration that can apply the desired wrench, $\mathcal{W}_{des}$ must be inside the space of applicable wrenches. Let $\mathcal{W}_{grasp}$ be a polyhedral approximation of the space of applicable wrenches. The proposed Grasp-Task compatibility test formally checks if $\mathcal{W}_{des} \subset \mathcal{W}_{grasp}$. The approximation $\mathcal{W}_{grasp}$ can be computed as follows:

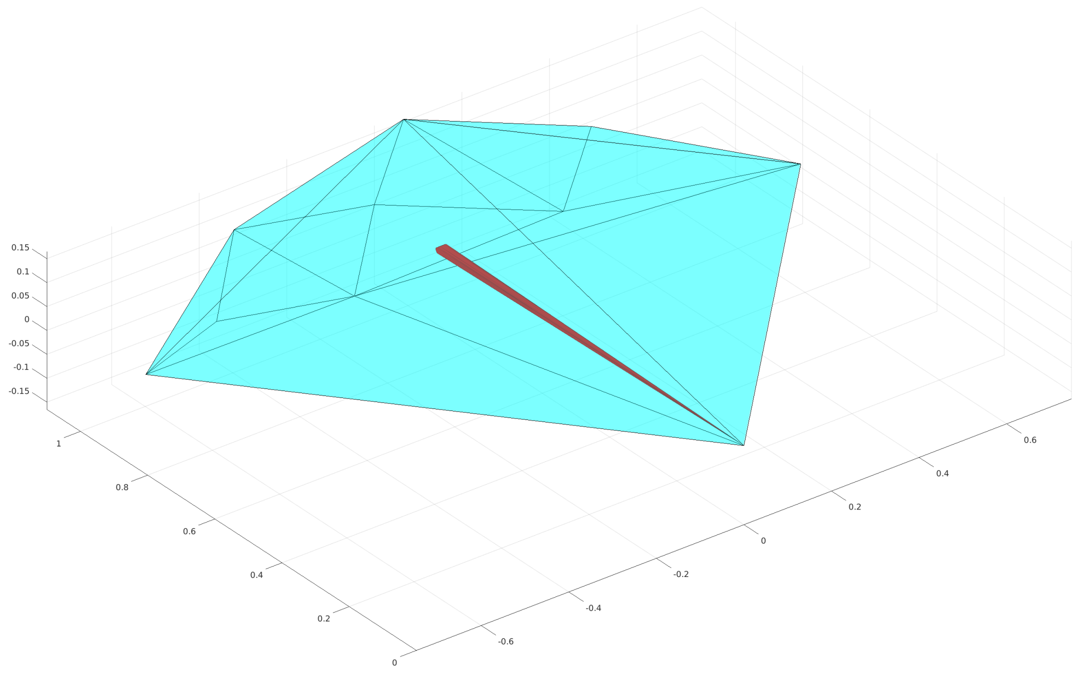


Figure 3.2: An example of the polyhedral approximation $\mathcal{W}_{grasp}$, represented as the light blue polyhedron in 3D wrench space, along with $\mathcal{W}_{des}$, represented in red. In this example $\mathcal{W}_{grasp}$ contains $\mathcal{W}_{des}$.

1. each contact force is decomposed using the two borders of the Coulomb friction cone;
2. the associated normalized wrenches are then computed for each border force;
3. the obtained set is then enlarged with the bisector wrench for each friction cone;
4. the conical combination is then computed in the 3D wrench space of the obtained set of normalized wrenches.

An example of this approximation is shown in Figure 3.2.

3.1.3 Configuration Indices

As it happens in prehensile grasping tasks, the proposed metrics are difficult to apply to select the optimal grasp. Therefore, we now introduce a set of indices that can be easily exploited to characterize, in an aggregate manner, the nonprehensile grasp configurations and can be employed as a selection criterion. These indices are borrowed from the standard manipulation literature, and they are here adapted to be used in the considered scenario.

Composed Quality Index

The composed quality index I_D ranks the uniformity of the angular distribution of the contact points and the capability of applying forces directed to the object's center of mass [55]. Its expression is

$$I_D = \left(\frac{2\pi}{n}\right)^n \prod_{i=1}^{n} \left| \min_{j\in\{1\ldots n\},j\neq i} \arccos \mathbf{c}_i^T \mathbf{c}_j - \arccos \mathbf{n}_i^T \mathbf{c}_i \right|^{-1} \tag{3.5}$$

where $\mathbf{n}_i \in \mathbb{R}^2$ is the inward unit vector normal to the object surface at the i-th contact point, while $\mathbf{c}_i \in \mathbb{R}^2$ is referred to as *central vector* and it is the inward unit vector pointing the center of mass of the object from the i-th contact point. When the central vectors have a uniform angular distribution with respect to the center of mass, the configuration is expected to resist better external forces and disturbances [56, 57]. Besides, when the contact forces points towards the center of mass of the object, the configuration can reduce the inertial effect.

Extension Index

The extension index I_E is computed as the area of the polygon formed by the contact points. As discussed in [58], the effectiveness of a grasp improves as this area increases.

Grasp Dexterity Index

The performance and precision of manipulation are affected when a configuration is close to a singular position since that configuration requires more effort to impress a wrench into the manipulated object. The grasp dexterity index [59] provides information on how close a configuration is to a singularity. This index is defined as

$$I_G = \frac{\sigma_{min}(G)}{\sigma_{max}(G)} \tag{3.6}$$

where $\sigma_{min}(G)$ and $\sigma_{max}(G)$ are the minimum and maximum singular values of the grasp matrix, respectively. This index is close to zero when the grasp is close to a singular configuration.

Modified Hausdorff distance

Intuitively, if $\mathcal{W}_{grasp}$ is much bigger than $\mathcal{W}_{des}$, then the grasp can withstand a greater range of disturbances that can arise from non-idealities. However, the grasp creating a wider $\mathcal{W}_{grasp}$ can present contact forces that interfere negatively with each other, thus requiring a more greater effort from the single contacts in order to balance the task requirements. The Modified Hausdorff Distance (MHD) proposed in [60] measures how far two subsets of a metric space are from each other, and it can thus provide a measure on how big $\mathcal{W}_{grasp}$ is with respect to $\mathcal{W}_{des}$.

Assuming the distance between a generic vector $a \in \mathbb{R}^3$ and a set of vectors $\mathcal{B}$ to be defined as

$$d(a, \mathcal{B}) = \max_{b\in\mathcal{B}} \frac{a^T b}{||a||\,||b||} \tag{3.7}$$

which is the maximum cosine similarity of a vector a across a set $\mathcal{B}$ of vectors, the MHD between two sets of vectors is defined as follows

$$MHD(\mathcal{A}, \mathcal{B}) = \max\{d_{MHD}(\mathcal{A}, \mathcal{B}), d_{MHD}(\mathcal{B}, \mathcal{A})\} \tag{3.8}$$

where

$$d_{MHD}(\mathcal{A}, \mathcal{B}) = \frac{1}{N_a} \sum_{a \in \mathcal{A}} d(a, \mathcal{B})$$

represents the unidirectional distance between a set $\mathcal{A}$, with N_a elements, and a set $\mathcal{B}$. Hence, we define $MHD(\mathcal{W}_{grasp}, \mathcal{W}_{des})$ as a configuration index.

3.1.4 Evaluation

In this section, the statistical evaluation process conducted to validate the proposed test and indices is discussed.

Simulation Environment

The data required for the statistical evaluation of the indices above is collected through a series of simulations. Four different objects, represented in Figure 3.4b, are pushed along six different trajectories using ten randomly generated configurations. The trajectories, represented in Figure 3.4a, start at $(0, 0)$ and belong to two different classes: polynomial and sinusoidal with constant orientation. The configurations chosen are tested in the CoppeliaSim [61] physical simulation environment. The pushed object is modeled as extruded shapes that slide on the support floor, as shown in Figure 3.4b, while the pushing agents, visualized in Figure 3.3, are modeled as spheres. The pushing agents are free to roam accordingly to the 2D velocity commands transmitted through ROS communication channels from a MATLAB script. For each shape, each trajectory and each configuration, the MATLAB script places the pushing agents in their assigned relative position then iteratively computes the velocities of each agent according to the given trajectory to be performed and sends the commands to the pushing agents to move accordingly in space. The resulting motion of the pushing agents resembles a virtual rigid object moving according to the requested manipulation trajectory. The position and force data from the simulation are stored in a file and used to compute δ from (3.2) and E from (3.3).

Methods

For each pair of shape and trajectory, a generator of random contact configurations selects 10 contact configurations which satisfy two requirements: 1) there is no collision between the pushing agents; 2) the angle between every contact normal and the initial velocity of the trajectory is $\in (-\pi/2, \pi/2)$. The former requirement is necessary for the physical realization of the grasp, while the latter ensures that the configuration is nonprehensile and can apply a push to the object. The simulations thus provide a set of 240 Boolean results that represent the success or failure of the manipulation, and

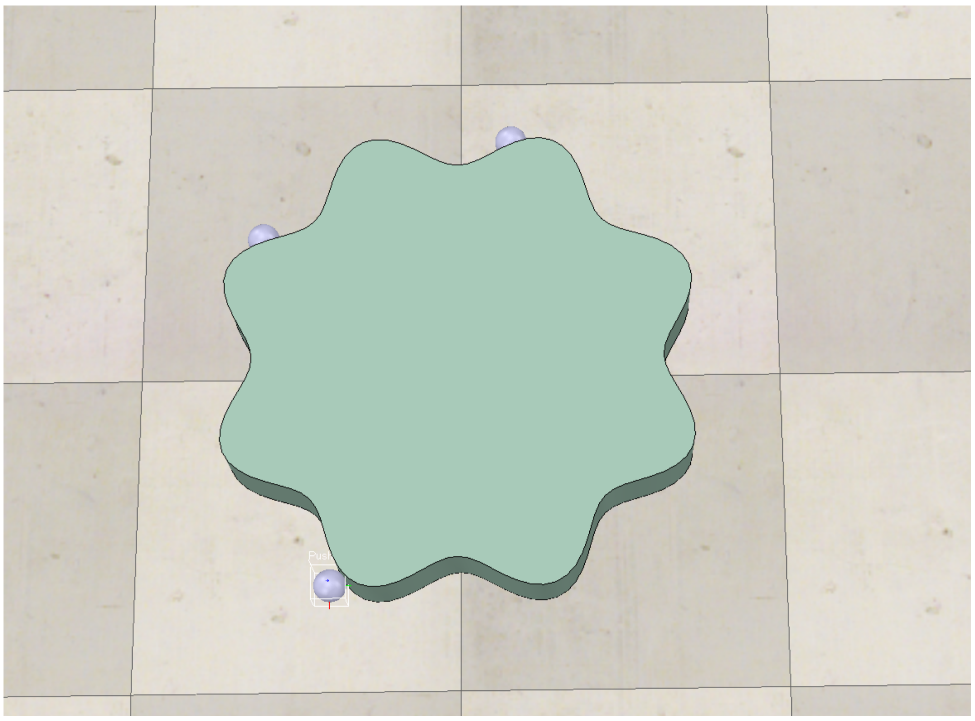


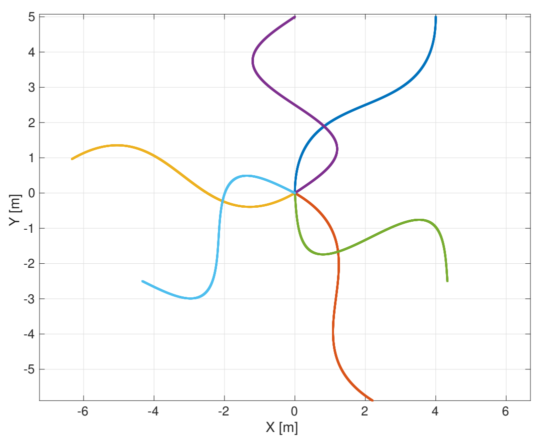
FIGURE 3.3: Representation of the pushed object and the pushing agents in CoppeliaSim.

two sets of 240 samples for δ and E, respectively. A manipulation is deemed successful if the pushing agents maintain the contact with the object during the whole pushing process, which means that the object has been successfully transported along the trajectory. The data, together with the configuration indices for each grasp, are used to perform a statistical evaluation. To prove the validity of the test proposed in Section 3.1.2, we execute a Wilcoxon rank-sum test [62] between δ and the test result. A Fisher's exact test [63] is also performed between the results of the test and the manipulation success information. The relationship between each index I_D, I_E, I_G, MHD and the metric E is assessed computing the linear correlation between the data along with the corresponding p-value.

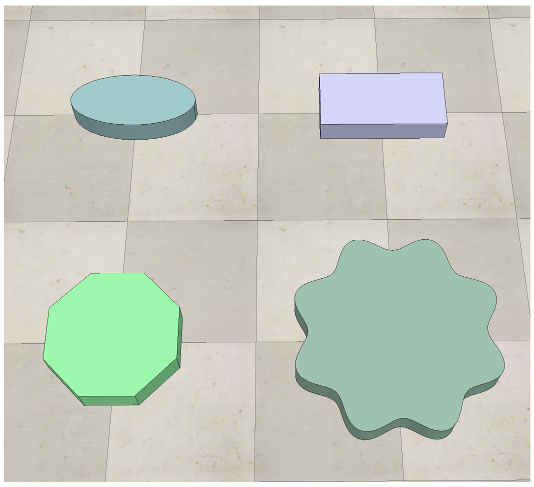
Results

The results of the statistical analysis are presented here, while a discussion on the results is provided in Section 3.1.5.

The distribution of the metric δ for each test result is shown in Figure 3.6. The correlation between the values of δ and the test result are assessed with a Wilcoxon rank-sum test. The test rejects the hypothesis of no-correlation and returns a p-value of $1.66e - 15$. Table 3.1 is a 2×2 contingency table describing the classification of the simulations by manipulation success and test result. The application of a Fisher's exact test to the contingency table confirms the correlation between the test result and the manipulation success with a p-value of $3.2187e - 20$.



(A)



(B)

FIGURE 3.4: A) Polynomial(blue, red and yellow) and sinusoidal(purple, green and light Blue) trajectories for planar manipulation. B)Set of the pushed objects in CoppeliaSim.

TABLE 3.1: Contingency table describing the frequency distribution of the test and manipulation results in the collected data

Manipulation success	Test	
	Fail	Pass
No	181	8
Yes	18	33

The relation between the indices and the performance metric E is evaluated considering only the successful manipulations since we are interested in information on the contact configurations that are capable of performing the required task. Figure 3.5 show the scatter plots of the index values and the corresponding value of E for the manipulation simulation. Table 3.2 contains the correlation coefficient for each index and the corresponding p-value.

TABLE 3.2: correlation coefficients and associated p-values between the quality metric E and the considered indices

	I_D	I_E	I_G	MHD
correlation coefficient	-0.1697	-0.1321	-0.0604	-0.37172
p-value	0.3085	0.4293	0.7186	0.021568

3.1.5 Results Discussion

Grasp-Task Compatibility Test

The validity of the proposed test is assessed using two standard statistical tools: the Wilcoxon rank-sum test and the Fisher's exact test. These evaluations provide the same result with a stable confidence level given by the very low reported p-values. Consequently, it is possible to say that the proposed Grasp-Task Compatibility Test

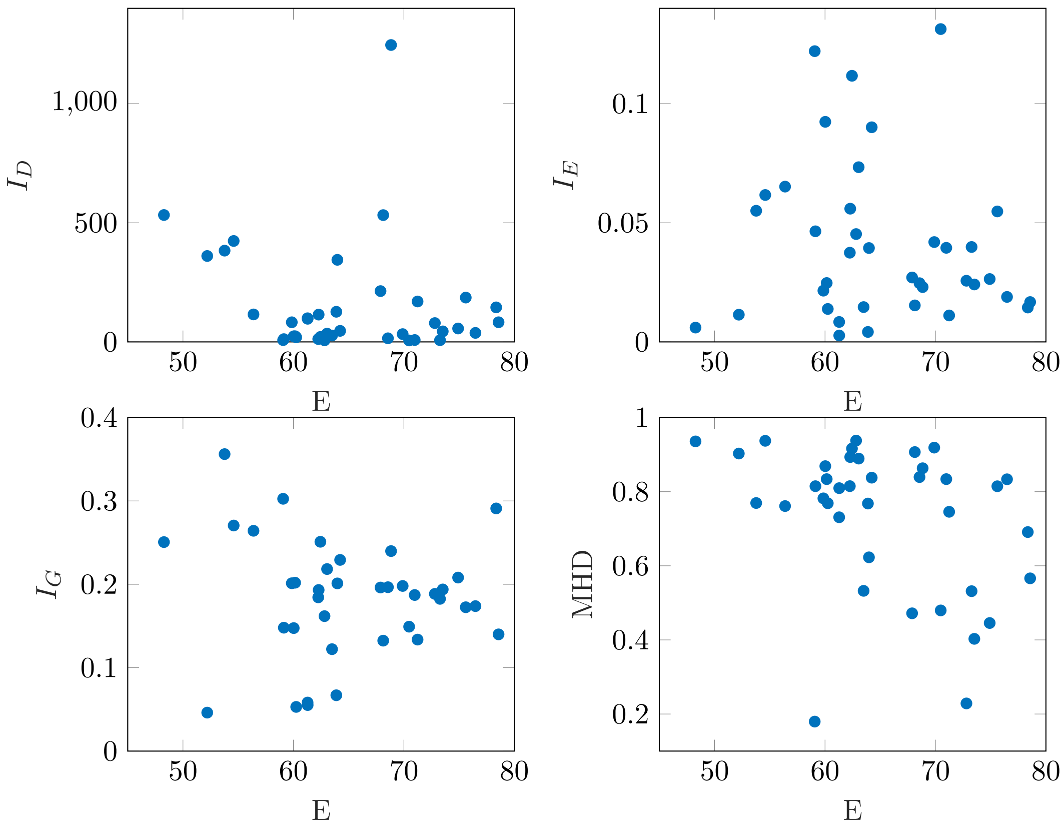


FIGURE 3.5: Scatter plots displaying the pairs (E,index) for the successful manipulations.

is a suitable tool to determine the ability of a grasp configuration to carry out the task.

Configuration Indices

The existence of a relationship between the considered indices and the performance metric E is assessed through the computation of the linear correlation coefficient and corresponding confidence level.

The results reported in Table 3.2 demonstrate that a statistically relevant relationship exists only for one of the considered indices, namely the MHD. Even though a low correlation seems to exist also for indices I_D, I_G and I_E with the performance metric E, the associated confidence values are too large to consider such a relationship as statistically relevant. We believe that the low significance of the indices I_D, I_E and I_G is due to the peculiarity of non-prehensile manipulation. The absence of force closure and the characteristic of the pushing manipulation of having all the contacts facing towards the direction of motion could be one of the causes that affected the performance. As an example, when pushing a convex object, the contact points may be restricted to only one side of the manipulandum, thus the maximum value of I_E is restricted and the angular uniformity graded in I_D is affected. Furthermore, the value of I_G represents how uniformly a grasp can resist to external forces applied to the

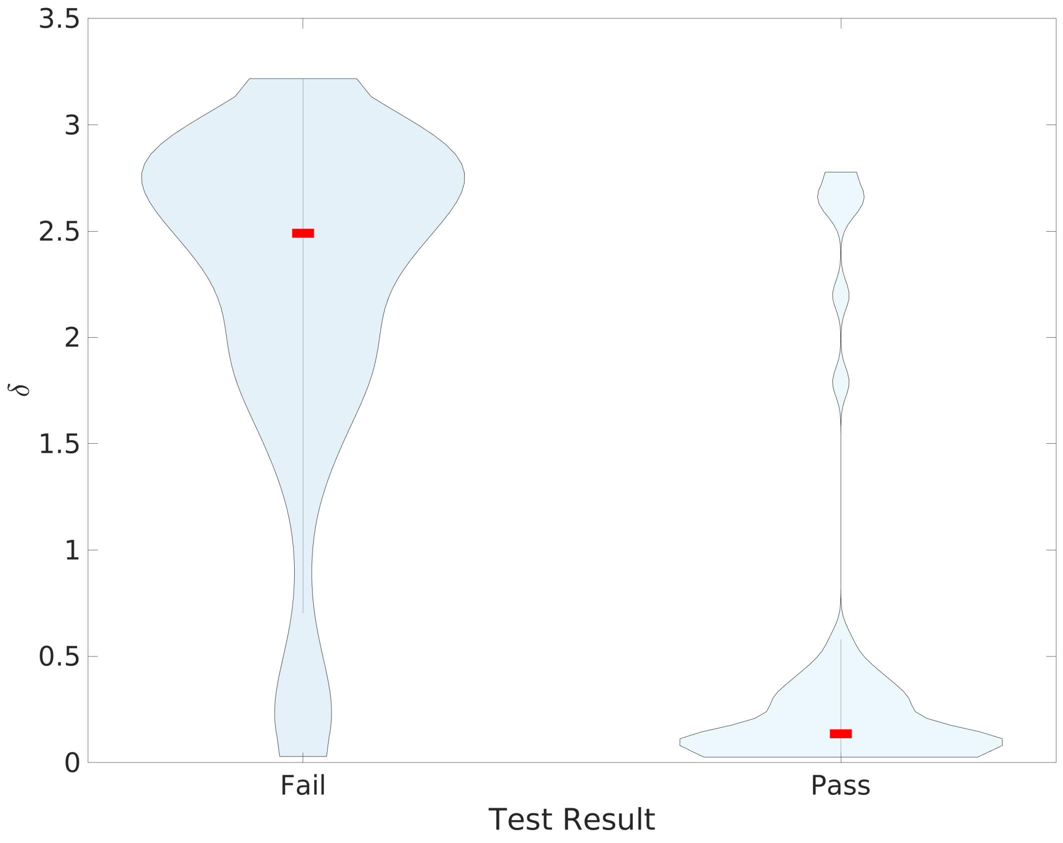


FIGURE 3.6: Violin plot describing the distribution of δ separated by test results. The median value of each distribution is represented by the red square.

object, but in non-prehensile manipulations a grasp cannot resist all external forces by definition, thus the index might be ill-posed for the problem.

In summary, even though the indices I_D, I_E and I_G have been proven useful and significant for characterizing a grasp configuration in the prehensile scenario, these cannot provide information on the efficiency of a grasp configuration applied to a pushing problem, while the information provided by MHD can be considered relevant for the addressed problem and therefore suitable to characterize the quality of a grasp configuration with respect to a task.

3.1.6 Possible Applications

We hereby defined an index to evaluate the quality of a grasp configuration with respect to a pushing manipulation task. The primary use of this result can be identified in the selection of the most adequate configuration to perform a given manipulation among a set of candidates. Considering the results of the performed analysis, we propose the following procedure to select a grasp configuration that is both effective and energy efficient. Knowing the assigned manipulation task and both the geometry and frictional properties of the manipulated object, as well as of the pushing agents, the following steps are proposed.

1. Generate a set of random grasps.
2. Test every grasp configuration in the set for the following requirements:
   - no collision among the pushing agents;
   - the angle between each contact normal and the initial velocity is inside the range $(-\pi/2, \pi/2)$;
   - the grasp configuration passes the test defined in Section 3.1.2.
3. Select the configuration that maximizes MHD for the given task.
4. Solve an optimization problem to refine the chosen configuration.

The optimization problem is defined as

$$\max_{x} MHD(x, \mathcal{W}_{des}) \tag{3.9}$$

$$\text{subject to } g(x) \leq 0 \tag{3.10}$$

where x is the contact configuration, $MHD(x, \mathcal{W}_{des})$ indicates the index associated to said configuration with respect to the task requirement $\mathcal{W}_{des}$. The constraint function $g(x)$ encodes the three requirements specified in the second step of the procedure.

As a case study, this selection procedure was applied for choosing the contact configuration to perform a manipulation task on an elliptic object. The pushing agents deployed accordingly are represented in Figure 3.7a. The configuration completed the task with energy efficiency $E = 38.67$ and the associated MHD is 0.254 in the evaluation setup with non-holonomic pushing agents. The chosen contact configuration has been also employed to perform a pushing manipulation with mobile robots both in simulation and in a real scenario. In both experimental takes three e-puck [52] differential drive mobile robots are controlled to perform a syncronized reference trajectory using a Linear Time-Varying MPC controller [64]. The reference trajectory for each robot is obtained by computing the relative position of each pushing agent with respect

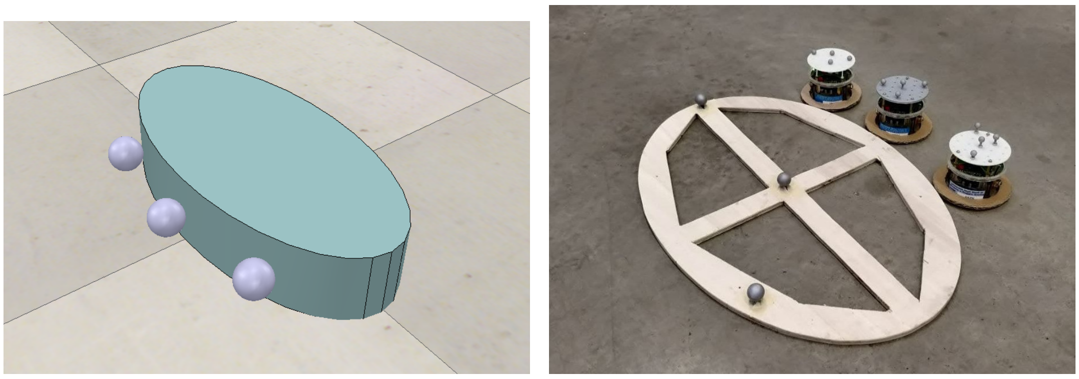

(A) (B)

Figure 3.7: A) Visualization of the grasp configuration selected by the proposed procedure. B) Visualization of the robots deployed in the selected grasp configuration.

to every point in the manipulation trajectory. A precise tracking of this trajectory by the robots therefore matches the motion of a omnidirectional pushing agent during the execution of the same task. The robots are equipped with a circular bumper ring with a diameter of 10cm. First the robot start moving towards the assigned contact position. Once all the robots have reached their target, namely the initial position of the reference trajectory, the manipulation can begin and each robot tracks its own reference trajectory. A comparison of the required manipulation task for the object and the resulting manipulation from the robots is provided in Figure 3.8a. A motion capture system is employed in the real environment to provide absolute positioning of both the manipulated object and the robots. The position of the manipulated object during the execution of the task is visualized in Figure 3.8b. From both results it can be noted that the path drawn by the manipulated object remains close to the reference trajectory even without a feedback action on the object's position, thus concluding that the chosen contact configuration is suitable for the specified manipulation task.

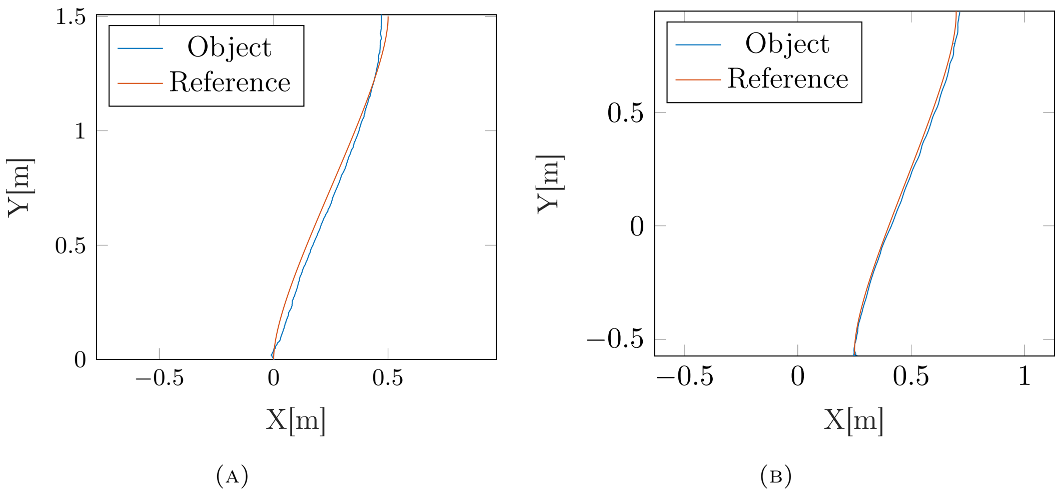


FIGURE 3.8: A)Position of the manipulated object during task execution by simulated mobile robots. B)Position of the manipulated object during task execution by real mobile robots.

3.2 Task Oriented Grasp Optimization

In the literature, the problem of how to grasp a given object to perform a specific task, which is usually encoded through its desired trajectory, while satisfying some optimality criteria is known as *task-oriented grasping*. This concept has been successfully applied in numerous application scenarios [65–69]. However, all the previous works considered either force- or form- closure grasps, which allow the robot to treat the object as entirely restrained. None of the above works have, indeed, addressed the problem of task-oriented optimal placement of multiple robots around an object for nonprehensile pushing. In this thesis, we extend the use of such a concept to nonprehensile pushing manipulation performed by a fleet of autonomous mobile robots. Our main contributions can be listed as follows:

- we define a optimization procedure to optimally position the robots around the object to perform a given task (tracking of a desired trajectory);
- we design a control algorithm that instructs the robots on how to properly push the object to track the assigned task trajectory;
- we thoroughly validate the presented procedures using simulation as well as real world experiments.

The first challenge addressed here is the optimal positioning of the robots along the object perimeter to enhance the object's trajectory tracking performances and the pushing robustness. More specifically, given a manipulation task, we aim at identifying the optimal deployment of n robots in contact with the object, also denoted as the *grasp configuration*, that can fulfill the task in the most robust way possible. This challenge was briefly addressed in Section 3.1.6 using a heuristic approach, while in the following we'll provide an analytical solution to the problem based on task-oriented grasping. To fulfill the presented challenge, we will consider a system described as follows.

Let $T_o \in SE(2)$ be the planar configuration of the *manipulandum*'s (the object to be manipulated) center of mass (CoM). Let m_o, $J_o \in \mathbb{R}$ be the mass and moment of inertia of the object, respectively. The position of the i-th robot on the plane is identified by $p_i \in \mathbb{R}^2$, expressed with respect to Σ_w, the global fixed reference frame. The robots are modeled as circular objects of radius $r_i > 0$ and can establish point contacts with friction with the object's perimeter [43]. We assume the robots move on the plane following the motion law of a single integrator

$$\dot{p}_i = u_i. \tag{3.11}$$

Figure 3.9 visualizes the considered system with an elliptical manipulandum (grey) laying on the horizontal plane with two robots (yellow) represented in the contact position. The considered manipulation task is described as a trajectory Γ composed by an ordered set of object configurations, as well as the associated velocity, at each generic time instant $t \in [0, T]$, with T being the trajectory duration.

The following additional assumptions are considered in the rest of this chapter:

- all the contacts are modeled as *point contacts with friction* [43], and interactions obey the Coulomb's friction model with a known friction coefficient between the parts in contact;
- the manipulated object and the robots lay in a planar environment, with all the frictional forces acting on the plane and gravity acting downward on the vertical axis;
- the frictional properties of the ground are uniform across the environment;
- the motion is slow enough, such that inertial forces are negligible or instantly absorbed by the frictional effects (*quasi-static* assumption);

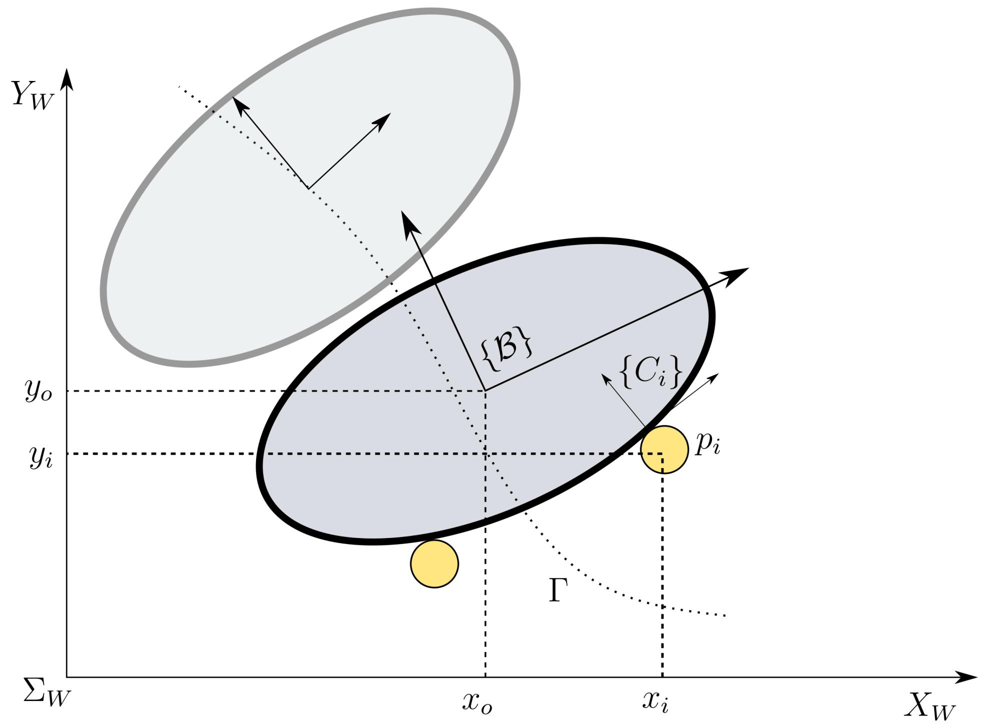


FIGURE 3.9: Visual representation of the considered system. Symbols are explained in Sec. 3.2.

- the object's perimeter is represented as a closed smooth curve enclosing a convex region of the space.

3.2.1 Task-oriented optimal robot positioning

In this section, we describe the procedure used to optimize the robots' contact positions around the object. The closed curve representing the object's perimeter at a given configuration $T_o \in SE(2)$ is parametrized by θ, i.e., the angle formed by the segment connecting the object CoM and a point on the object's perimeter. Given the object convexity assumption, the value of θ corresponds to a given position belonging to the object perimeter. Thus, a contact point between the i-th robot and the object is identified by a particular value of θ_i. Our goal is to calculate optimized values of θ_i for each robot to perform the pushing task optimally.

At each contact point we define a reference frame $\{\mathcal{C}_i\}$, whose origin lays on the object's perimeter, the $\hat{x}$-axis is directed along the perimeter tangent and the $\hat{z}$-axis is directed along the inward normal. The number of contact points is n_c and it is equal to the total number of robots in contact with the object. Denoting with $\Theta = [\theta_1, \ldots, \theta_{n_c}]^{\mathrm{T}}$, the grasp matrix $G(\Theta) \in \mathbb{R}^{3\times 2n_c}$ maps contact forces between the object's CoM, identified by the frame $\{\mathcal{B}\}$, and the contact frames $\{\mathcal{C}_1\}, \ldots, \{\mathcal{C}_{n_c}\}$. Generally, it can be constructed given the generic $i-$th contact point pose (p_{c_i}, R_{c_i}) with $R_{c_i} \in \mathrm{SO}(2)$ being the rotation matrix of the frame $\{\mathcal{C}_i\}$ and $p_{c_i} = [p_{c_i,x},\, p_{c_i,y}] \in$

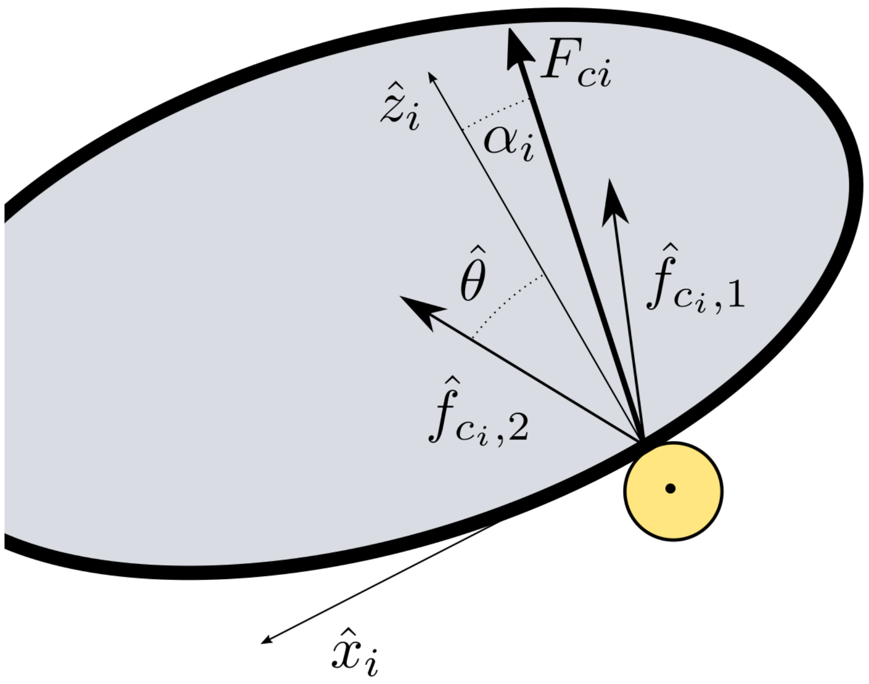


FIGURE 3.10: Close up visualization of the force applied on the object by robot i as well as the borders of the associated friction cone. Symbols are explained in Sec. 3.2.

R^2 being the position of the origin of $\{\mathcal{C}_i\}$ expressed in $\{\mathcal{B}\}$ as follows (see [43])

$$G = \begin{bmatrix} \dots & R_{c_i} & 0 & \dots \\ & [-p_{c_i,y} \; p_{c_i,x}]R_{c_i} & 1 & \end{bmatrix}. \tag{3.12}$$

Under the stated quasi-static assumption, the object dynamic model can be formulated as follows

$$\dot{x}_o = R_o G(\Theta) F_c, \tag{3.13}$$

where $x_o = [p_{o,x},\, p_{o,y},\, \theta_o]^{\mathrm{T}}$ denotes the minimal representation of the object pose and $\dot{x}_o$ its time derivative. R_o is the rotation matrix from the object frame to the global frame, while $F_c = [f_{c_1}^{\mathrm{T}}, \dots, f_{c_{n_c}}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{2n_c}$ is the vector stacking the contact forces, that represents the overall input of the system. To be realizable, the contact forces F_c must belong to the the friction cone space.

To simplify this constraint formulation in the optimization problem introduced later, a contact force parametrization is introduced. The $i-$th contact force, namely $f_{c_i} \in \mathbb{R}^2$ can be espressed as a non-negative linear combination of unit vectors $\hat{f}_{c_i,j}$, with $j = 1, 2$, denoting the $i-$th friction cone boundaries. These vectors can be calculated, using geometrical considerations (see Fig. 3.10), starting from the friction coefficient $\mu \geq 0$ as follows

$$\hat{f}_{c_i,j} = R_y(\pm\hat{\theta})\hat{z}_i, \quad \hat{\theta} = \arctan \mu, \tag{3.14}$$

where $R_y(\cdot)$ is the rotation matrix around the $\hat{y}-$axis, $\hat{z}_i = [0,\, 0,\, 1]^{\mathrm{T}}$ is the contact normal expressed in $\{\mathcal{C}_i\}$, and $\hat{\theta}$ is the semi-aperture angle of the friction cone.

By denoting with $\hat{F}_c = \text{blockdiag}\left(\hat{F}_{c,1}, \ldots, \hat{F}_{c,n_c}\right)$ the matrix encoding all the unit vectors representing the boundary of the friction cones, i.e., $\hat{F}_{c,i} = \left[\hat{f}_{c_i,1}, \ldots, \hat{f}_{c_i,k}\right]$, and with $\Lambda = \left[\lambda_{c1,1}, \ldots, \lambda_{c_{n_c},2}\right]^{\mathrm{T}}$ the associated coefficients used to decompose the contact forces along the friction cone boundaries, we can compactly express the vector of stacked contact forces as follows

$$F_c = \hat{F}_c \Lambda. \tag{3.15}$$

At this point, for the contact forces to belong to the composite friction cone space (Cartesian product of all the friction cone spaces) is sufficient to have $\Lambda \geq 0$, i.e., all the λ's must be non-negative [43].

A task-oriented optimality criterion is thus established to optimize the positions of the robots around the object (and thus the contact points they form). Given a task (trajectory) Γ to be realized (tracked) expressed as a sequence of desired poses for the object $x_{o,d}(t)$, $0 \leq t \leq T$, we choose to optimize the position of the robots (through the optimization of Θ) to simultaneously minimize the task tracking error and the associated contact forces (coefficients) to realize it. Mathematically, the discretized problem is formulated as follows

$$\min_{\Theta,\Lambda} \quad \sum_{k=1}^{N} \frac{1}{2}||x_{o,d}(k) - x_o(k)||_Q^2 + \frac{1}{2}||\Lambda(k)||_R^2 \tag{3.16}$$

$$s.t. \quad x_o(k+1) = x_o(k) + R_o(k) G(\Theta) \hat{F}_c \Lambda(k) \tag{3.17}$$

$$D\Theta + \bar{\Theta} \succeq \epsilon \tag{3.18}$$

$$\Lambda(k) \succeq 0 \quad \forall k = 1, \ldots, N \tag{3.19}$$

where k represents the generic time instant, N is the total number of steps taken to accomplish the task (and it is related to the trajectory duration T and the adopted discretization step), $x_{o,d}(k)$ and $x_o(k)$ are the desired and the current object positions at a given time step $1 \leq k \leq N$, $Q > 0$ and $R > 0$ are diagonal and positive-definite weight matrices. The constraint in (3.17) is obtained by combining the object dynamic model (3.13) and the contact force parametrization (3.15), while the constraint (3.19) denotes the feasibility of the contact forces discussed above. The constraint (3.18) is, instead, used to avoid collisions between the robots and is described hereafter. The matrix D is the incidence matrix of the cyclic graph associated to the multi-robot system deployed around the object, $\bar{\Theta}$ is the vector taking into account the modulo

2π property of the parametrized object perimeter, and are specified as follows

$$D = \begin{bmatrix} 1 & -1 & 0 & \dots & 0 \\ 0 & 1 & -1 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 1 & 0 & 0 & \dots & -1 \end{bmatrix}, \qquad \bar{\Theta} = \begin{bmatrix} 0 \\ \vdots \\ 2\pi \\ \vdots \\ 0 \end{bmatrix},$$

where $\epsilon \geq 0$ is a non-negative constant vector denoting the collision bounds to be opportunely chosen. The value 2π occupies the $k-$th component of the vector $\bar{\Theta}$ and it is used to take the shorter distance between the $k-$th and $(k+1)-$th contact points when this last crosses the value $\theta = 2\pi$. This is used to calculate the right minimum (angular) distance between the robots. The value of ϵ_i (i.e., the i-th component of vector ϵ) represents the lower bound of the angle between the computed configurations of the two successive contact points/robots i and $i+1$.

3.2.2 Motion control for trajectory tracking

In this section, we report the proposed control scheme used to command the robots to execute the considered object's pushing manipulation task. More specifically, the objective of the manipulation is to move the object on the plane following a predefined trajectory Γ. To achieve this objective, we propose to define the desired body force according to a proportional-derivative scheme. In particular, let $x_{o,d} \in \mathbb{R}^3$ be the current desired pose for the object along Γ, and $v_{o,d} \in \mathbb{R}^3$ be the associated desired velocity for the object. The desired body force $\mathcal{F}_o^*$ for the problem at hand is computed as

$$\mathcal{F}_{o,d} = K_{P_O}(x_{o,d} - x_o) + K_{D_O}(v_{o,d} - v_o) + f_\mu(v_{o,d}), \tag{3.20}$$

where $K_P, K_D \in \mathbb{R}^{3\times 3}$ are positive definite diagonal matrices representing the proportional and derivative gains. The last term $f_\mu(v_d)$ is a feed-forward component for sliding friction compensation. This component is computed as the wrench caused by the friction effect on the object sliding with velocity $v_{o,d}$. More specifically, $f_\mu(v_d)$ is computed using the following ellipsoidal approximation for the limit surface [70]:

$$f_\mu(v_{o,d}) = \frac{\mu_f m_o g}{\sqrt{v_{o,d,x}^2 + v_{o,d,y}^2 + (\gamma\omega_{o,d})^2}} \begin{bmatrix} v_{o,x} \\ v_{o,y} \\ \gamma^2\omega \end{bmatrix}, \tag{3.21}$$

where $v_{o,x}, v_{o,y}, \omega \in \mathbb{R}$ are the components of v_o, $\gamma = \sqrt{J_o/m_o}$ and μ_f is the friction coefficient between the object and the floor. The body force $\mathcal{F}_{o,d}$ computed in (3.20) is then used to online extrapolate feasible contact forces to be applied to the object using the optimization problem described below

$$\min_{F_c} \quad \frac{1}{2}||\mathcal{F}_{o,d} - G(\Theta^*)F_c||^2 \tag{3.22}$$

$$s.t. \quad F_c = \hat{F}_c\Lambda \tag{3.23}$$

$$\Lambda \succeq 0 \tag{3.24}$$

where Θ^* is the optimal solution of the problem in (3.16). The input velocity for the i-th robot is thus computed using the following control law

$$u_i = K_{P_R}(p^*_{c_i} - p_i) + K_{V_R}R_oG(\Theta)F_{ci}, \tag{3.25}$$

where, the first term provides proportional position feedback for the robot since $p^*_{c_i}$ is the desired optimal position for the i-th robot, calculated using θ^*_i, and p_i is its current position. The second addendum is a feed-forward component instructing the robots to move in the direction of the contact force it needs to provide to accomplish the task. The coefficients $K_{P_R}, K_V > 0$ are gains of the control law. The proposed online procedure makes use of the optimal robots' positioning Θ^* which is guaranteeing minimal contact forces and tracking error in the ideal case, while providing robustness by means of position feedback terms.

3.2.3 Simulation and experimental results

In this section, we present the evaluation method used to assess the validity of the proposed strategy. In order to properly evaluate the presented optimization and control architecture, a series of simulations has been performed. We propose three different manipulation tasks performed by a pair of robots pushing an elliptical prism. For each trajectory, the optimal positioning for each robot is computed using the procedure presented in Section 3.2.1. The control law proposed in Section 3.2.2 is then used to perform the tracking of the task trajectory by having the robots push onto the object after they have reached the desired contact position. Results of the validation will be detailed in Section 3.2.3.

Simulation setup

The simulations have been performed using the CoppeliaSim physical simulation software [61]. Two or three robots are placed in the simulated environment together with an elliptical prism, as visualized in Fig. 3.11. The robots are an e-puck [52] differential drive mobile robot and are equipped with bumper rings with $r_i = 5$ cm, to provide a consistent contact surface. In order to provide the single integrator behaviour described in (3.11), the velocities for the wheels of each robot are obtained through input-output feedback linearization [71]. The object's mass is set to $m_o = 0.3$[kg] while the inertia moment is set to $J_o = 0.0072$[kg m^2]. The principal axes of the ellipse are 0.5 m and 0.2 m respectively. The friction coefficient between the object

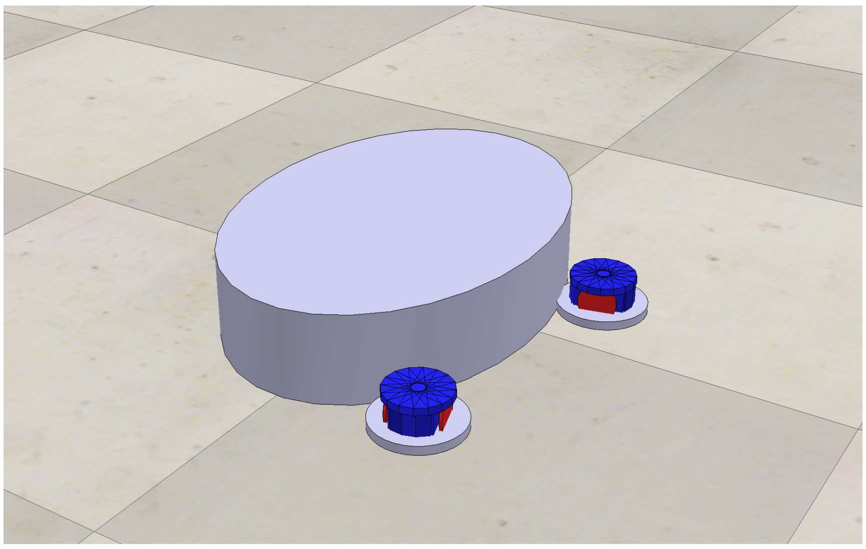


FIGURE 3.11: The robots and object in the CoppeliaSim simulation environment.

and the floor is set to 0.6 while the robot-object friction coefficient is set to 0.2. All the used parameters are set according to the best available estimate from the available real-world equipment described in the next section. The data required to compute the control laws is transmitted through ROS communication channels to a MATLAB script implementing the proposed methodology. The simulations are performed on a laptop equipped with an Intel Core I7-9750H and 16GB of RAM.

Experimental setup

The real-world experiment is performed on a honed concrete floor using the same robot used in the simulation environment, the e-puck differential drive robot, equipped with a 3D printed bumper ring. The elliptical manipulandum is obtained from a 5mm plywood sheet. The principal axis of the ellipse are 0.255 m and 0.175 m respectively. The global pose of both the robots and the object is obtained using an Optitrack™ motion capture system [53] composed of seven Prime13 cameras. The information about the velocity of the object, required to compute (3.20), is obtained by feeding the pose information into an extended Kalman filter generating the linear and angular velocities of the object. The system inside the experiment area is visualized in Fig. 3.12. All the required data is sent to the PC, computing the control law through ROS channels. The computed input for each robot is sent through a Bluetooth connection to the robot's control board.

Results

In this section, the results of the performed tests are shown. First, the results of two different simulated tasks are presented, followed by a manipulation task performed in the real environment. Figure 3.13 (top) show the execution of a desired b-spline trajectory (black dashed line), in the optimal robots' configuration case (left) and in

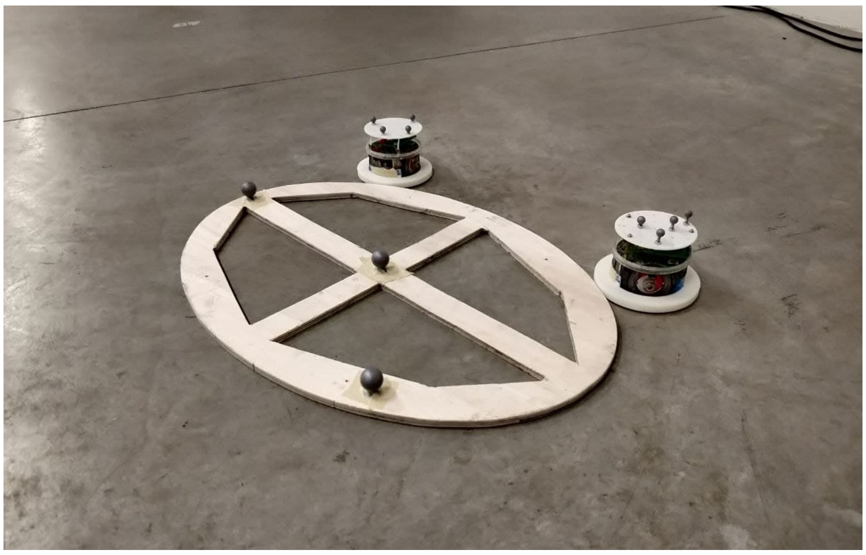


FIGURE 3.12: The robots and object in the test environment.

the non-optimal case (right). The bottom left graph shows the norm of the traking error $e_p = p_{o,d} - p_o$ while the bottom graph shows the contact force $f_{c_1}, f_{c_2}, f_{c_3}$ norm along the perfomed trajectory. The continuous lines are associated to results with the robots in the optimal configuration $\Theta^* = [5.112, 4.623, 3.569]^{\mathrm{T}}$ found solving the problem in (3.16), while dashed lines are associated to non optimal robots' configuration $\Theta = [7/4\pi, 3/2\pi, 5/4\pi]^{\mathrm{T}}$, which is also used as a starting point of the optimization. As it is possible to note from the upper graph, the optimal solution increases the performance of the tracking task showing a lower and less varying error norm (in blue), compared to the non-optimal configuration (in red), especially in the second half of the trajectory. Analogous considerations can be drawn looking at the bottom graphs that shows the norm of the three contact forces $f_{c_1}, f_{c_2}, f_{c_3}$ for the optimal configuration (continuous lines) and for the non-optimal configuration (dashed lines). In particular, it can be noted that the contact forces are more uniformly distributed among the robots in the optimal case, especially in the second half of the trajectory. It is worth remarking that, although successful in the considered case, non-optimal configurations can lead to even larger tracking errors and norms of contact forces, and this may lead to task failures.

Figure 3.14 shows the tracking performance of the same trajectory using two robots. As shown by the orange line, the robots can closely track the assigned trajectory. However, this is again achieved by sacrificing the orientation tracking performance as the robots align the object to mostly push along the motion direction. The optimal configuration of the robots is $\Theta = \begin{bmatrix} 5.497 & 3.734 \end{bmatrix}$, and corresponds to both robots actively pushing along the entire trajectory.

Figure 3.15 shows the performance difference between an optimized configuration and a non-optimized one in terms of the norm of the difference between the desired body force $\mathcal{F}_{o,d}$ and the cumulative force applied by the contact GF_c, obtained tracking the same polynomial trajectory. Even if the magnitude of the error is relatively

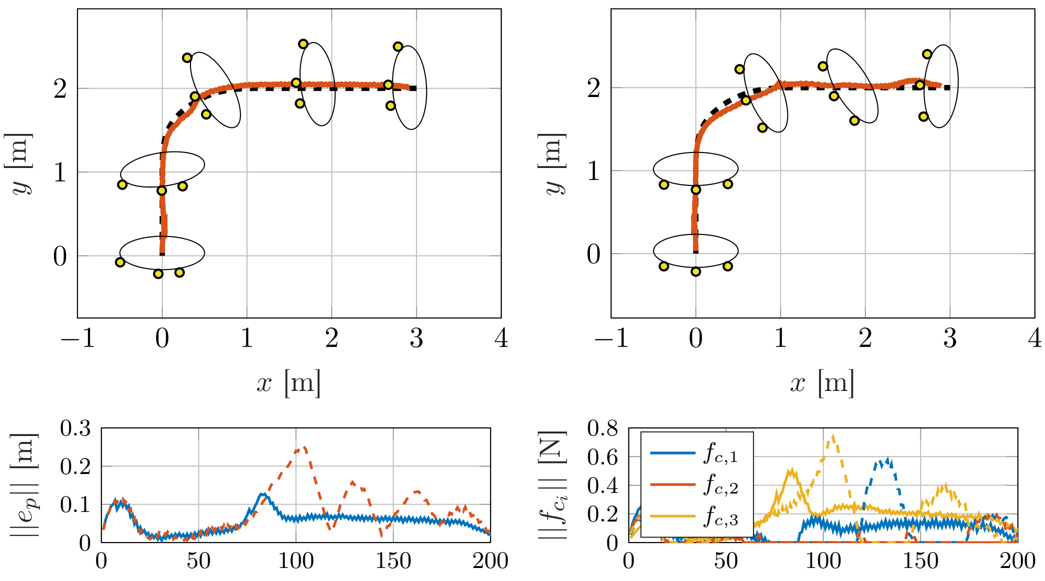


FIGURE 3.13: Upper graphs: desired (black dashed line) and executed trajectory (orange) in the optimal robots' configuration (left) and non-optimal robots' configuration (right). Bottom left graph: norm of the tracking error $e_p = p_{o,d} - p_o$. Bottom right graph: norm of the contact forces $f_{c_1}, f_{c_2}, f_{c_3}$ along the performed trajectory. Results of the optimal robots' configuration are shown with continuous lines, non optimal robots' configuration results are shown with dashed lines.

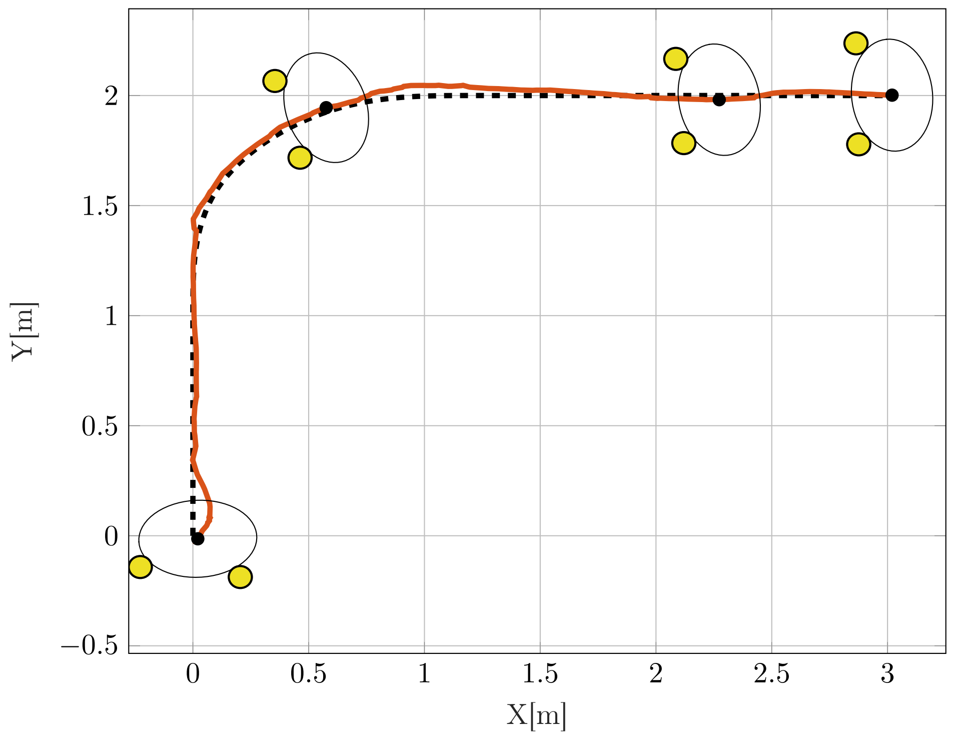


FIGURE 3.14: Tracking performance along the first proposed test trajectory, depicted as a dashed black line. The orange line shows the motion of the object during the simulated manipulation.

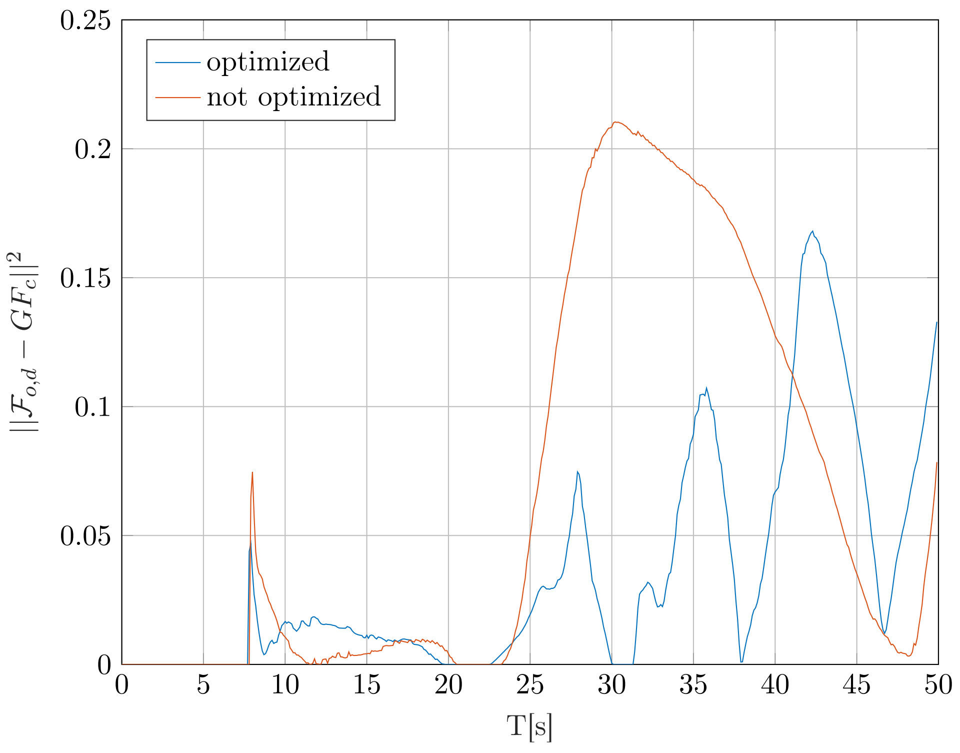


FIGURE 3.15: Comparison of the error between the requested body force and the force that can be applied by the contacts.

small in both cases, the optimized configuration is more capable to quickly reduce the error, therefore obtaining higher tracking performance. It is important to notice how the results are similar at the beginning of the tracking. Both configurations can apply a cumulative force directed in the object's local y direction. The difference in performance arises when the trajectory starts to turn, indicating that the optimized configuration can apply a force closer to the force requested by the task at hand. The second manipulation task is defined by the polynomial trajectory visualized in Figure 3.16. In this three-robot scenario the optimal configuration of the robots here is $\Theta = \begin{bmatrix} 5.1121 & 4.6237 & 3.5693 \end{bmatrix}^T$. The performance obtained matches the results of the previous scenarios in which the orientation tracking is sacrificed.

The tracking performances depicted in Figure 3.17 and Figure 3.18 show the manipulation of the same elliptical object in simulation and in the real environment respectively. The optimal contact configuration is defined as $\Theta = \begin{bmatrix} 5.1868 & 3.7398 \end{bmatrix}$. In this two robot scenario it is possible to note how in the optimal robot configuration one robot is mostly pushing from below, and the other aligns with the diagonal portion of the trajectory. The great similarity between the results obtained in simulation and in the real environment further validates the proposed approach.

3.3 Future Works

In this work, we considered the problem of cooperative manipulation of an object through a group of mobile robots. More specifically, referring to a pushing maneuver,

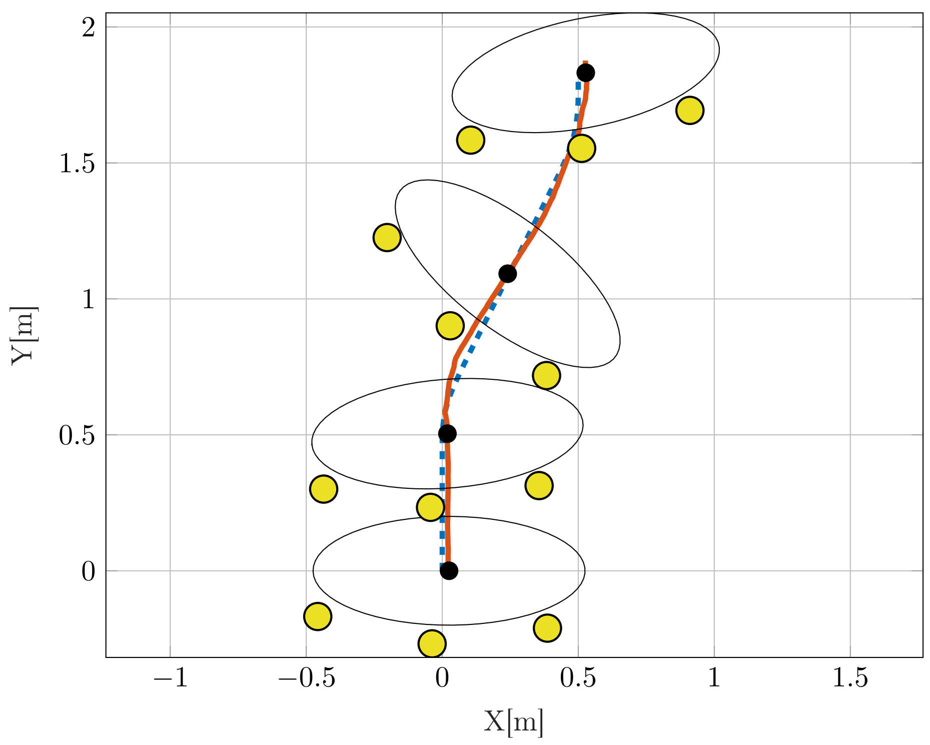


FIGURE 3.16: Tracking performance of a polynomial trajectory (dashed blue) using 3 robots in a simulated environment, depicted as the orange line.

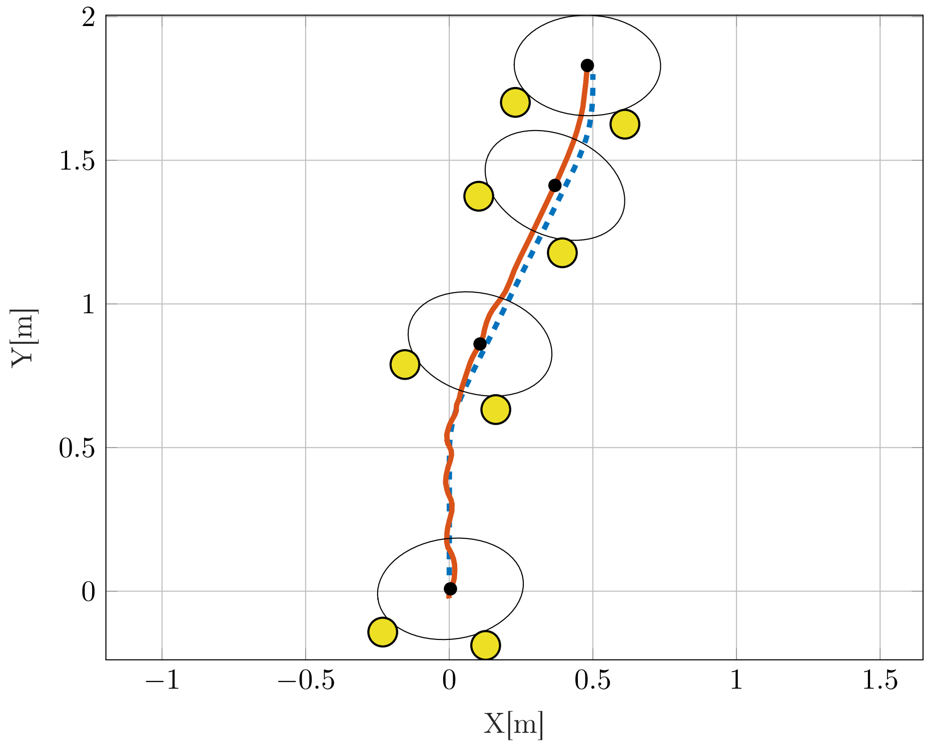


FIGURE 3.17: Tracking performance along the third proposed test trajectory, depicted as a dashed blue line. The orange line shows the motion of the object during the simulated manipulation.

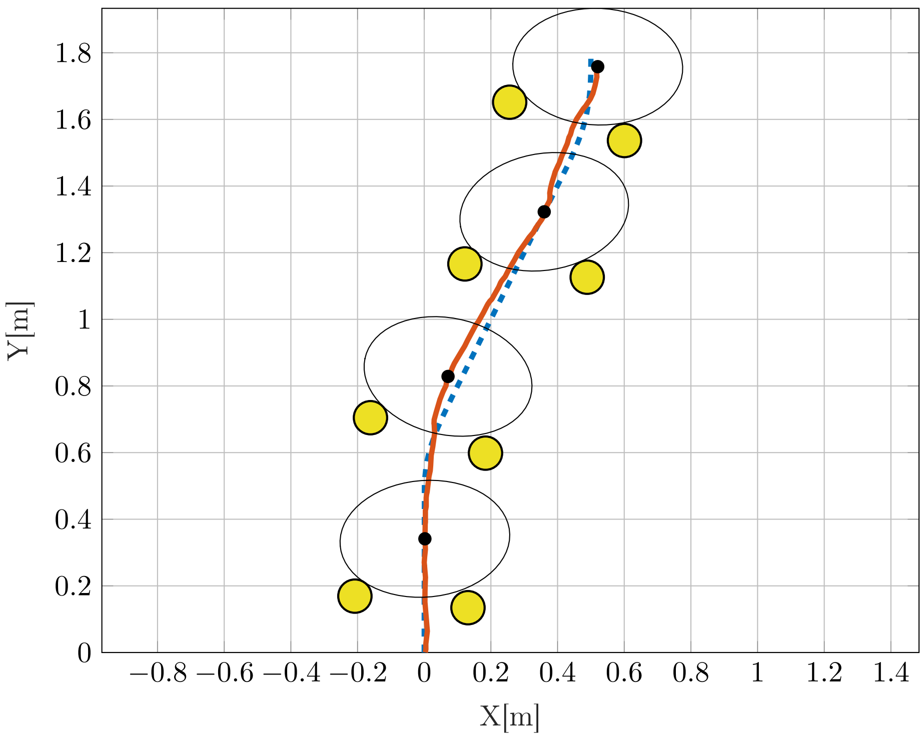


Figure 3.18: Tracking performance along the third proposed test trajectory, depicted as a dashed blue line. The orange line shows the motion of the object during the real world manipulation.

we proposed a procedure to characterize the quality of a nonprehensile grasp configuration with respect to the execution of a required pushing task. This was achieved by selecting a set of configuration indices and correlating them to the energy efficiency and effectiveness of the pushing maneuver. The correlation was studied employing an extensive validation campaign performed on a physical simulator. As a case study, a procedure for selecting the optimal configuration was employed to select a grasp configuration in an optimized manner using the proposed characterization. The selected configuration was employed to perform the requested manipulation with both simulated and physical robots. Furthermore, we proposed a method for calculating optimal non-colliding contact points for a group of mobile robots pushing an object that minimizes both the trajectory tracking error and the norm of contact forces to perform the task. Exploiting an optimized contact configuration, a motion controller was developed that exploits online computed contact forces in feed-forward and position error feedback terms to realize the desired trajectory tracking task. Results were validated through simulations and experiments on real a robotic system composed of multiple robots, showing the validity of the proposed approach.

Although effective, our method still has several limitations that must be addressed in future works. For instance, the use of a local optimization method for the calculation of the contact points might lead to sub-optimal results. A global optimization method might produce better results at the expense of larger computing time. A trade-off can be found depending on the application scenario. Besides this, we aim to reformulate the problem in (3.16) to find possibly time varying contact configurations and the corresponding pushing forces for a given trajectory using the task-oriented approach presented here. Finally, further experimental validations and tuning will be carried out in the future possibly regulating pushing forces at their optimal values using feedback regulation techniques.

Chapter 4

Pushing Manipulation with many Robots

The work presented in this chapter concerns the use of a large number of robots to perform pushing manipulation on a planar object.

As outlined in the previous chapter, the computational complexity associated with multi-robot pushing increases with the number of robots. This may represent a serious drawback in the implementation of pushing stategies. The contribution of this work is in the definition of a control scheme for the manipulation of an object by a multi-robot system, where the group of robots is controlled in an aggregate manner, exploiting a hierarchical structure. More specifically, a Voronoi-based coverage control strategy is exploited to implement local actions to aggregate the robots according to a given (time varying) probability density function, whose shape is defined, at a high level, based on the shape of the object to be manipulated, and to the desired trajectory to be followed. Once the group of robots has successfully aggregated, the object is manipulated performing a collaborative pushing operation, achieved by moving and adapting the probability density function. This strategy resembles the one used in [1] but with a different global behaviour and resulting performance. In [1] the authors used a single global input to control all the robots, which implied the necessity of interacting with the walls in the environment to control the variance. As a consequence, the robots required significant time to regroup, affecting the time performance of the manipulation. Furthermore, their strategy is not applicable to free space scenarios where only the robots and the manipulandum are present. Our proposed approach considers the ability of each robot to be controlled separately in order to reduce the time required to bring the manipulated object to a target pose. To further demonstrate the performance improvement introduced by the present work, we also provide a quantitative comparison with the results of [1], given the similarity between the two methods. Finally, despite being effective, the strategy presented in [20] requires exact knowledge of the shape of the manipulated object, while the strategy hereby presented only requires rough information on the object's footprint.

To summarize, the contribution of this chapter are:

- The introduction of a control architecture for the planar manipulation of an

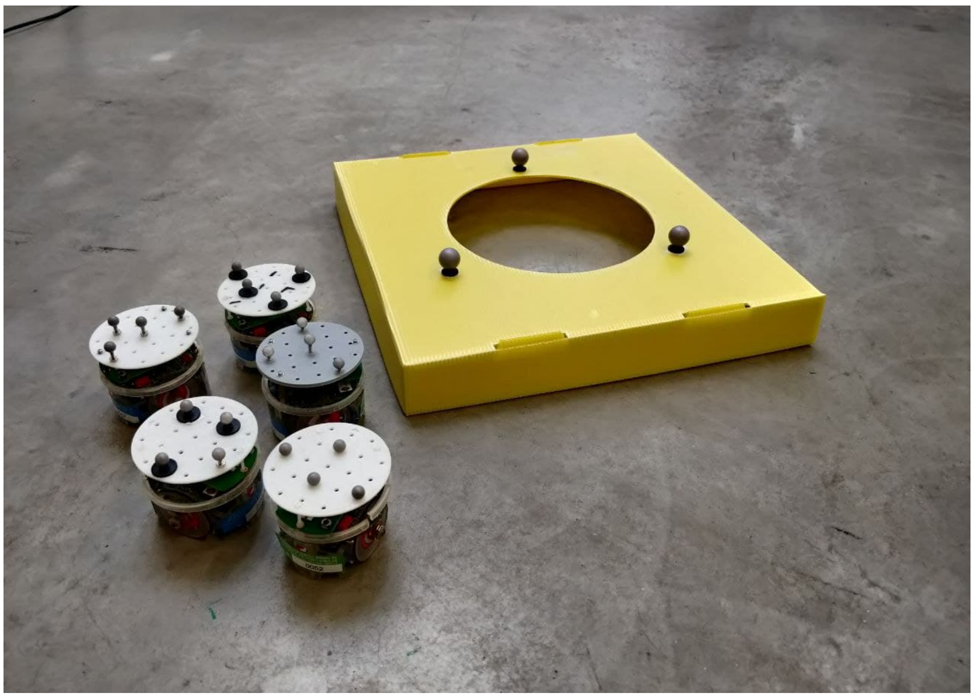


FIGURE 4.1: E-Puck robots approaching a square object to apply a pushing action inside the motion capture arena.

object with a system of multiple mobile robots that is able to generate a coordinated pushing action on the object, in order to transport it towards a desired target pose in an efficient way. The architecture generates the manipulation action taking advantage of the controllability of each single robot composing the group without relying on features of the environment and exact knowledge of the object to be transported.

- The extensive validation of the proposed control architecture through physical simulations as well as real world experiments, providing a quantitative comparison with a similar method from the literature.

The rest of the chapter is organized as follows. Section 4.1 provides the problem formulation, and introduces the proposed approach. Based on a multi-layer hierarchical structure, the proposed solution is detailed in the subsequent Section 4.2. Results of the experimental evaluation are presented in Section 4.3. Finally, concluding remarks are given in Section 4.4.

4.1 Problem Description and Proposed Approach

Consider a prismatic object free to slide on a bounded rectangular ground plane Q. This object will be hereafter referred to as the *manipulated object*, or *manipulandum*.

The planar position of this object is identified by the position of its center of mass, defined as $p_O = [x_O, y_O]^T \in \mathbb{R}^2$. Consider a group (or swarm) of robots composed by n mobile robots that move on the same plane. The planar position of the i-th robot

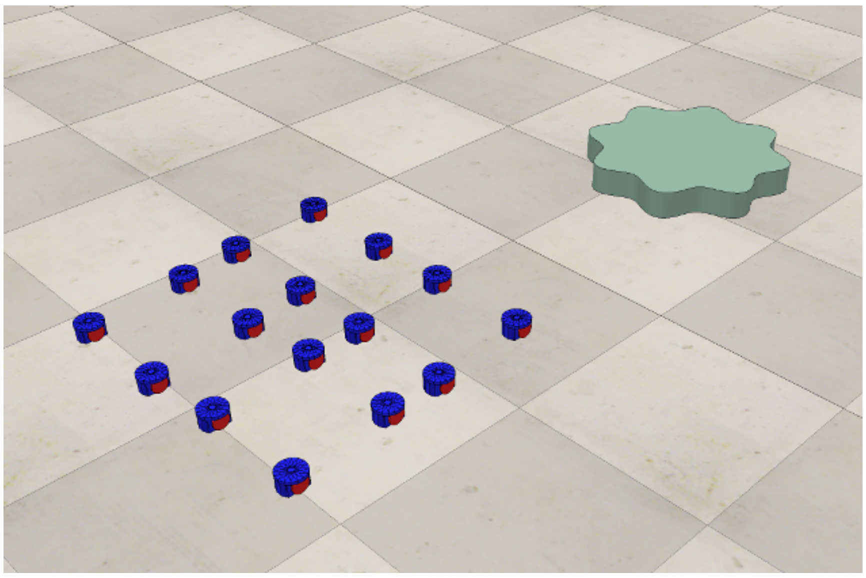


FIGURE 4.2: Representation of the considered system inside the simulator: the manipulated object that lies on the ground and the robots used for the manipulation. The robot model has been simplified to reduce the computational load of simulating multiple robots.

of the swarm is identified by $p_i = [x_i, y_i]^T \in \mathbb{R}^2$. The motion model for each robot is considered to be a single integrator $\dot{p}_i = u_i$. This is achieved using input/output feedback linearization [71]. The positions p_i of all the robots are collected in the set $\mathcal{P} = \{p_1, \ldots, p_i, \ldots, p_n\}$. The mean position of the robot group is denoted as $\bar{p} \in \mathbb{R}^2$. A representation of the system in a simulated environment is visualized in Figure 4.2.

This work approaches the problem of manipulating an object on a plane using only pushing actions generated by a group of wheeled mobile robots. The proposed solution makes use of the ability of the group of robots to apply a coordinated push to an object, much stronger then the one applicable by a single robot.

In order to perform the manipulation operation, we propose a three-layer strategy:

1. In the lower layer, the robots are controlled by a decentralized control algorithm in order to gather around a requested mean point, with a required variance.
2. The middle layer steers the group of robots around and towards the object according to a requested velocity direction for the object.
3. The higher layer is the overall position controller for the manipulated object, given the position to be reached.

4.2 Control Architecture

This section presents the control architecture designed to steer the robots to manipulate the object. First, the lower layer for swarm control is introduced. This layer controls how the robots are displaced in the environment according to a desired distribution. The shape and mean position of the desired distribution are obtained using the output of the higher layers of position control for object pushing. This output, namely the planar velocity of the mean point of the robot distribution, is obtained from an artificial potential function specifically designed for the robots to achieve the pushing task.

4.2.1 Lower Layer: Swarm Control

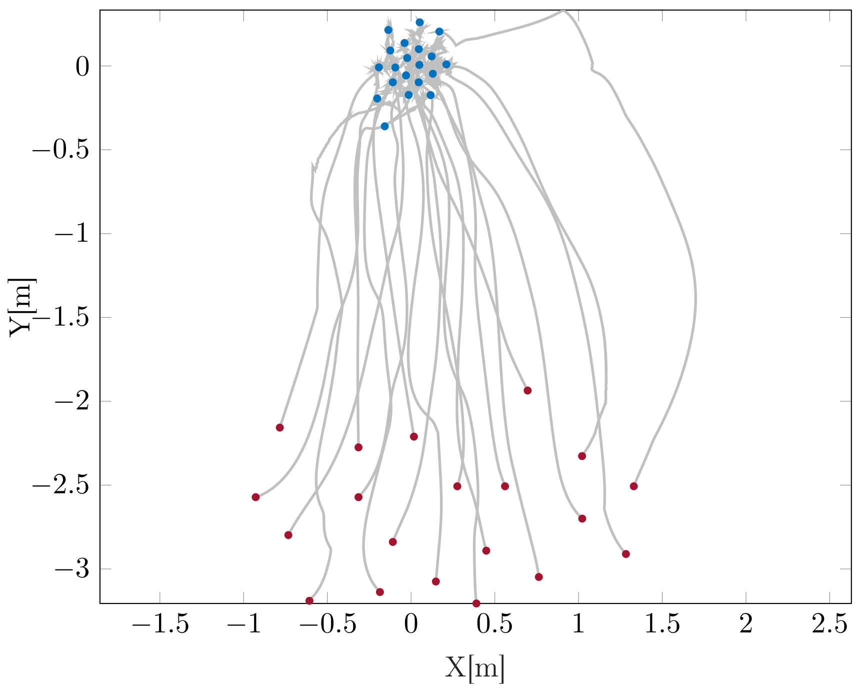


FIGURE 4.3: Behaviour of the coverage control algorithm: the initial positions of the robots are depicted in red, while the final positions are depicted in blue. Solid gray lines represent the trajectories of each robot.

This section presents the algorithm used in the lower layer to control and coordinate the group of mobile robots. The objective of the control scheme is to displace the robots according to a certain distribution ϕ. As will be clarified in the following sections, such distribution will be used to steer the robots in order to push a target object towards a target position. To achieve this objective we employ an algorithm taken from standard multi-robot literature: *coverage control*. In mobile sensing networks, coverage control solves the problem of displacing a large group of robots in the environment in order to minimize the locational optimization function [72]:

$$\mathcal{H}_{\mathcal{V}}(\mathcal{P}) = \int_Q \min_{i \in \{1,\ldots,n\}} \|q - p_i\| \phi(q) dq \tag{4.1}$$

ϕ is a *density function* that represents the desired displacement of the robots over Q. In particular, ϕ is defined as 2D gaussian distribution:

$$\phi(q) = e^{(q-p_{goal})^T B(q-p_{goal})} \tag{4.2}$$

where p_{goal} is the center of the density function, and B is a 2-by-2 matrix defined as

$$B = \begin{bmatrix} (\frac{\cos^2\Theta}{2\sigma_x^2} + \frac{\sin^2\Theta}{2\sigma_y^2}) & (\frac{\sin 2\Theta}{4\sigma_x^2} - \frac{\sin 2\Theta}{2\sigma_y^2}) \\ (\frac{\sin 2\Theta}{4\sigma_x^2} - \frac{\sin 2\Theta}{2\sigma_y^2}) & (\frac{\sin^2\Theta}{2\sigma_x^2} + \frac{\cos^2\Theta}{2\sigma_y^2}) \end{bmatrix} \tag{4.3}$$

where σ_x, σ_y are the 2D standard deviation of the distribution and Θ is a planar rotation angle used to orient the distribution on the plane. The angle Θ is computed as so that the variance σ_x expresses the variance of the density function on the orthogonal direction with respect to the manipulated object. This angle is thus compuped as

$$\Theta = \tan^{-1}\frac{[1,0](p_O - p_{goal})}{[0,1](p_O - p_{goal})}$$

The choice of the standard deviations σ_x, σ_y is related to the dimentions of the manipulated object and the number of robots in the swarm. More specifically, σ_x should be set so that the resulting tangential distribution of robots with respect to the object is as wide as the manipulandum. $\sigma_y \leq \sigma_x$ is set in relation to the swarm size so that more robots are involved in the pushing action since, with fewer robots, a more elliptical distribution forces more robots to be involved directly in the action (i.e. pushing directly on the object and not on other robots). A visualisation of the density function is shown in Figure 4.4.

The center point p_{goal} of the Gaussian function is computed as

$$p_{goal} = \bar{p} + u_{swarm} dt \tag{4.4}$$

where u_{swarm} is the output of the middle control layer (that, as will be clarified in the following section, represents the desired motion input for the swarm) and dt is the discrete time step for the overall control algorithm.

The solution of this optimization problem is obtained using Lloyd's algorithm [72]. Starting from the initial position of the robots, the following iterative procedure is performed:

1. Compute the Voronoi diagram of Q considering, as seeds for the Voronoi partition, the current position p_i of the robots (one Voronoi cell is created for each robot).

2. Compute the weighted centroid C_{V_i} of each Voronoi cell through the integration of ϕ.

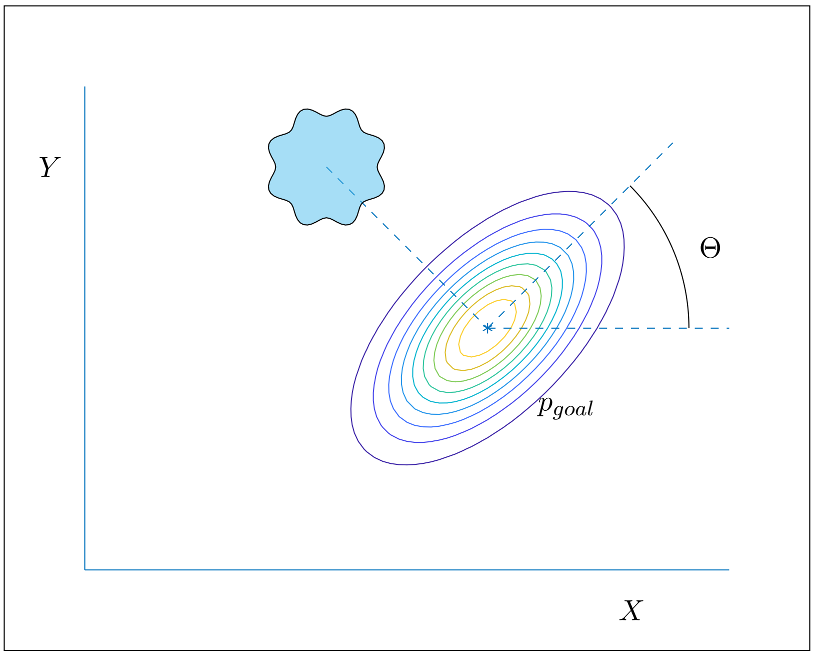


FIGURE 4.4: Visualization of the probability density function used, a Gaussian centered in p_{goal} with $\sigma_x = 2\sigma_y$ and rotated to align with the manipulandum.

3. Compute the following cartesian control input for each robot i

$$u_i = -(p_i - C_{V_i})$$

When convergence has been reached, the robots will have deployed according to the density function ϕ. An example of this behavior using 20 robots is shown in Figure 4.3 using a Gaussian distribution centered at $(0, 0)$ with standard deviations $\sigma_x = \sigma_y = 0.1$.

4.2.2 Position Control Layers for Pushing the Object

This section described the two control layers that, starting from the desired position of the object, first determines the desired motion direction for the manipulated object, then defines the desired overall motion of the swarm. The objective of the higher control layer is to generate the desired motion direction for the object to reach a required position. This can be achieved using different approaches and strategies. In this work, we propose a simple direction law. Let $p_{target} \in \mathbb{R}^2$ be the desired position for the manipulandum. Then, the desired direction of motion $v_{des}(t) \in \mathbb{R}^2$ for the object is computed as follows:

$$v_{des}(t) = \frac{p_{target} - p_O(t)}{\|p_{target} - p_O(t)\|} \tag{4.5}$$

In the middle control layer, given the required pushing direction v_{des}, the position of the manipulandum p_O and the mean position of the swarm $\bar{p}$, an artificial potential

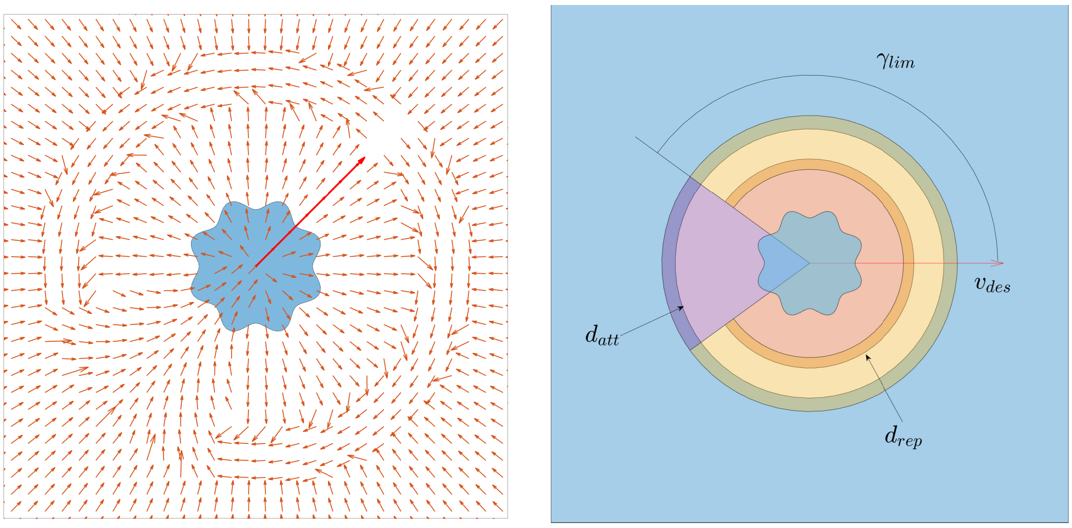


FIGURE 4.5: A) Artificial potential force for cumulative swarm input, oriented accordingly to v_{des}, represented in red. B) Action zones of the proposed artificial potential field. The Repulsive field acts in Zone 1, shown as the red area. The attractive potential is active in Zone 2, depicted in blue. The yellow annulus sector represents Zone 3 and Zone 4 is represented in purple.

field [73–76] is defined to steer the swarm by generating u_{swarm}, used in (4.4) to compute the center point of the required distribution for the robots. The design of such a potential field has to meet the following core requirements.

- When too far away from the object, the robots should direct towards the object.
- The robots should avoid contact with the object that would result in pushing opposite to the desired direction.
- The robots should position correctly with respect to the object before pushing.
- The robots pushing action should aim at converging into the desired direction.

Such potential field is computed as the combination of four different effects applied across 4 different action zones:

1. a repulsive field to avoid unwanted collisions with the object,
2. an attractive field, to steer the robots towards the object when too far away,
3. a magnetic field to steer the robots behind the object and push towards v_{des},
4. a tangential field to realign the swarm for pushing.

Figure 4.5b provides a visualization of the action zones of the proposed artificial potential field. Define now $d \in \mathbb{R}^2$ as the relative position between the manipulated object and the mean position of the swarm, namely

$$d = \bar{p} - p_O$$

Let γ represents the angle that d forms with v_{des}, that is

$$\gamma = \cos^{-1}\left(\frac{v_{des} \cdot d}{\|v_{des}\| \cdot \|d\|}\right)$$

The overall gradient of the potential is computed as

$$u_{\text{swarm}} = \frac{u_{\text{rep}}}{\|u_{\text{rep}\|}} + \frac{u_{\text{att}}}{\|u_{\text{att}}\|} + \frac{u_{\text{mag}}}{\|u_{\text{mag}}\|} + \frac{u_{\text{tan}}}{\|u_{\text{tan}}\|} \tag{4.6}$$

This cumulative swarm input is depicted in Figure 4.5a, for a representative example of manipulated object. The limits of application of each force component in the artificial potential field are computed taking into consideration the radius r_{man} of the smallest circumference that encloses the manipulandum. For instance, in the experiments described in Section 4.3, the following parameters are used:

$$d_{rep} = 1.4 r_{man} \qquad d_{att} = 1.8 r_{man}$$

The value $\gamma_{lim} = \frac{4}{5}\pi$ is set empirically as a tradeoff between improving pushing execution precision (higher γ_{lim}) and minimizing swarm-manipulandum realignments (lower γ_{lim})

Zone 1: Repulsive Field

This repulsive component of the potential field is active on a circular sector of width $2\gamma_{lim}$ and radius d_{rep} aligned with v_{des}. With this shape the field repels the robots except for when they are in an appropriate position for pushing, which is behind the manipulated object (with respect to the desired direction of motion, given by v_{des}). We define u_{rep}, the gradient of such a potential, as follows:

$$u_{\text{rep}} = \begin{cases} (\frac{1}{\|d\|} - \frac{1}{d_{rep}})\frac{1}{\|d\|^2} d, & \|d\| \leq d_{rep} \ \wedge \ \gamma < \gamma_{lim} \\ 0, & \text{otherwise} \end{cases}$$

Zone 2: Attractive Field

The purpose of the attractive field is to attract the swarm towards the object and consequently to keep the robots from drifting away. The attractive component u_{att} is defined when the distance of the swarm to the object is greater then d_{att}. The gradient of the potential in this zone is computed as

$$u_{\text{att}} = \begin{cases} -\frac{d}{\|d\|}, & \|d\| \geq d_{att} \\ 0, & \text{otherwise} \end{cases}$$

Zone 3: Magnetic Field

The magnetic field represents the core of the pushing action of the swarm. Once the robots have reached the pushing area, in order to properly push, their motion should converge into the line of action of v_{des} while approaching the object. The magnetic field created by an electric current flowing in opposite directions along two parallel wires, sketched in Figure 4.6, resembles this motion: thus, we define a control action inspired by this principle. The action zone of this field is set to the circular sector of radius $110\% d_{att}$ where $\gamma > \gamma_{lim}$. Let us introduce a rotation matrix $R(\alpha)$, defined to orient the field towards the required v_{des}, namely

$$R(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

with

$$\alpha = \tan^{-1}\left(\frac{[0,1]\cdot v_{des}}{[1,0]\cdot v_{des}}\right)$$

Let $W_{in}, W_{out} \in \mathbb{R}^3$ be the points where an imaginary current loop respectively enters and exits the plane. These points lie on the line orthogonal to v_{des} that passes through p_O. The distance from the points to p_O is defined as $70\% r_{man}$. In global coordinates the position of these points can be computed as

$$W_{in} = R(\alpha)\begin{bmatrix} 0.0 \\ -0.7r_{man} \\ 0.0 \end{bmatrix} + \begin{bmatrix} x_O \\ y_O \\ 0.0 \end{bmatrix}$$

$$W_{out} = R(\alpha)\begin{bmatrix} 0.0 \\ 0.7r_{man} \\ 0.0 \end{bmatrix} + \begin{bmatrix} x_O \\ y_O \\ 0.0 \end{bmatrix}$$

With r_{in}, r_{out} we define the distance of the swarm mean point to W_{in} and W_{out} respectively, computed as

$$r_{in} = [\bar{p}^T, 0.0]^T - W_{in}, \qquad r_{out} = [\bar{p}^T, 0.0]^T - W_{out}$$

The gradient of the magnetic potential field is defined as u_{mag}, given by

$$u_{\text{mag}} = \frac{M_{in} \times r_{in}}{(\|r_{in}\|)^3} + \frac{M_{out} \times r_{out}}{(\|r_{out}\|)^3} \tag{4.7}$$

where $M_{in} = [0, 0, -1]^T$ and $M_{out} = [0, 0, 1]^T$ are the associated magnetic moments. Equation (4.7) is indeed inspired by the magnetic field of two electric wires that are orthogonal to the plane, one with current flowing into the plane, one with current flowing out. However, since the definition of u_{swarm} given by (4.6) normalizes the contribution of the various components, the constants present in classical physics theory have been removed.

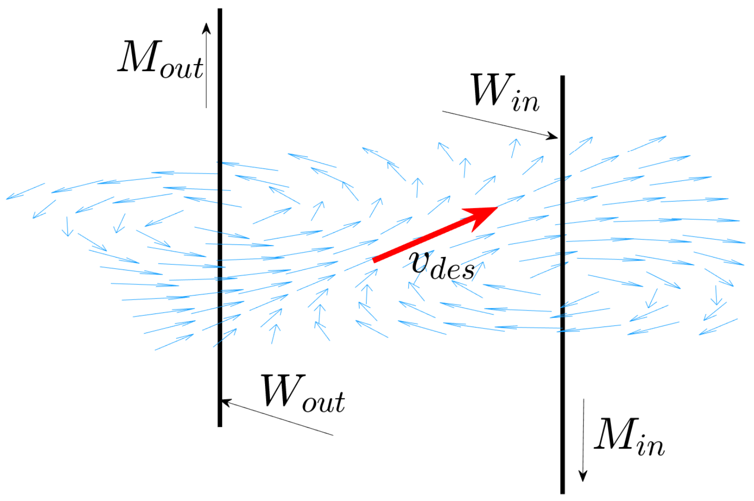


FIGURE 4.6: Magnetic field around two wires

Zone 4: Tangential Field

The purpose of the tangential field is to steer the robots around the object till they reach the active pushing area. The action zone for this component is the annulus sector around the object with radius from $90\%d_{rep}$ to $110\%d_{att}$ with $\gamma \leq \gamma_{lim}$. The zone is designed to provide an overlap with the adjacent zones, effectively combining the effects. Let $v_{des_E} = [v_{des}^T, 0]^T$ and $d_E = [d^T, 0]^T$ be the extension to $\mathbb{R}^3$ of v_{des} and d respectively. The tangential component inside the action zone is defined as:

$$u_{tan} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} [(v_{des_E} \times d_E) \times d_E] \tag{4.8}$$

4.3 Validation

This section illustrates the procedure used to validate the proposed approach. The proposed solution has been tested both in a physical simulator and with real world experiments. The objective of both types of validation is to manipulate a prismatic object through a series of target positions. The tests use the e-puck robots [52]. The control algorithm is implemented as a MATLAB script that communicates with the robots through ROS [39].

4.3.1 Simulations

The physical environment is recreated inside the CoppeliaSim simulator [61]. The simulated model for the e-puck robot send the global pose of the robot through a ROS topic. The position and orientation of the manipulated object is also published through a dedicated ROS topic. The tests are conducted on four prisms with different base shapes: an ellipse, an octagon, a rectangle and a flower. A visualization of the four manipulanda is shown in Figure 4.7. All the manipulanda have r_{man} of

TABLE 4.1: Sequence of target points for the manipulated object

	Simulation	Real
1	$p_{target} = (2, 2)$	$p_{target} = (1, 1)$
2	$p_{target} = (0, 0)$	$p_{target} = (0, 0)$
3	$p_{target} = (-1, 1)$	$p_{target} = (-0.5, 0.5)$
4	$p_{target} = (-1, -2)$	$p_{target} = (-0.5, -0.5)$
5	$p_{target} = (2, -1)$	$p_{target} = (1, -0.5)$
6	$p_{target} = (0, 0)$	$p_{target} = (0, 0)$

around 25 cm and weight 500 g. Each manipulated object is transported through a series of 6 targets starting from the position $(0, 0)$. A target is deemed reached when the manipulandum enters a circular zone of radius $0.1m$ around the target. Table 4.1 lists the sequence of the 6 assigned targets for both the simulation and the real world experiments. The sequence of targets is chosen specifically to test the proposed solution. As an example, when the first target is reached, the robots are then guided all around the object for it to be pushed to the second target, forcing a long waiting time between pushing actions. The proposed solutions is tested with

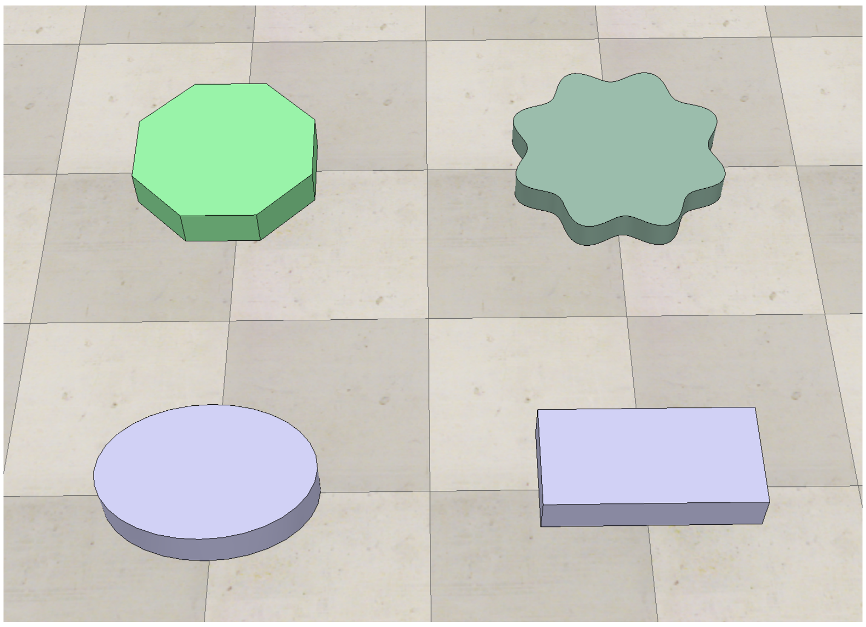


FIGURE 4.7: Set of manipulated objects used in the simulator: octagon shape, flower shape, ellipse shape and rectangle.

different robot groups, namely 8, 12, 16 and 20 robots. Each group carries each shape across all targets and each test is repeated 3 times. Figure 4.10 show the behaviour of the different manipulanda in characteristic runs. Across all the simulations, σ_x is set to 0.1 while the value $\sigma_y = \sigma_x$ is used in all runs except with 8 robots where $\sigma_y = 0.075$. This effect is seen in Figure 4.10 where the robots tend to spread more on the object's border. Figure 4.8 shows the statistics of the time required to reach all the assigned targets according to the shape and group size. It is worth noting here that rounder shapes like the octagon and flower require less time to complete all the tasks than the more eccentric shapes like ellipse and rectangle. This can be attributed to the different interaction between the robots and the object that can be more prone to create a rotation than a translation. It can also be noted that larger groups are more time effective then smaller swarms.

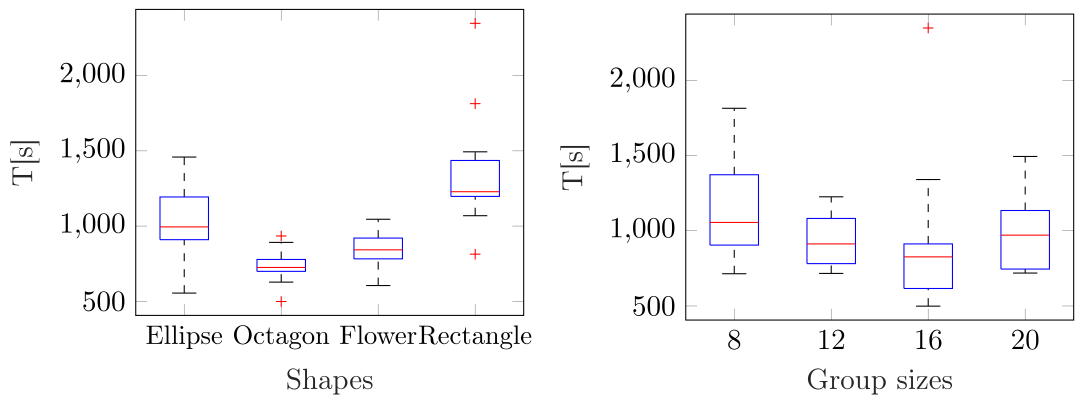


Figure 4.8: Box plot representation of the time required to reach all targets, grouped by shape and swarm size.

4.3.2 Experiments

Real world experiments have been conducted to validate the approach in a real scenario. In the experiments, 5 e-puck robots are used to manipulate a square object similarly to what is done in the simulations, using the same MATLAB implementation of the proposed strategy. The robots are controlled by a ROS driver node that communicates with the robots via bluetooth. The position of the robots and the manipulated object are measured globally using the Optitrack Motion capture system. A visualization of the system is provided in Figure 4.1. The object's weight is $327g$ and has $r_{man} = 0.26m$. In this scenario, the targets follow the same relative displacement of the ones used in simulation, but adjusted to take into account the limited area for the experiment. Figure 4.9 shows the motion of the manipulated object in the environment. Thestandard deviations used with the real robots are $\sigma_x = 0.075$, $\sigma_y = 0.0375$. On average the robots were able to perform the selected manipulations in 902 ± 17 s (mean $\pm$ std). To quantitatively compare the time efficiency of the presented strategy against the results in [1], we evaluated the average time required to travel a normalized unit distance. This metric δ is evaluated by computing an estimate of the distance traveled in the presented real world experiments, divided by the size of the manipulated object, then divide the average time to perform the associated experiment by this distance, namely:

$$\delta = \frac{T_M[s]}{\frac{l_M[m]}{s_O[m]}}$$

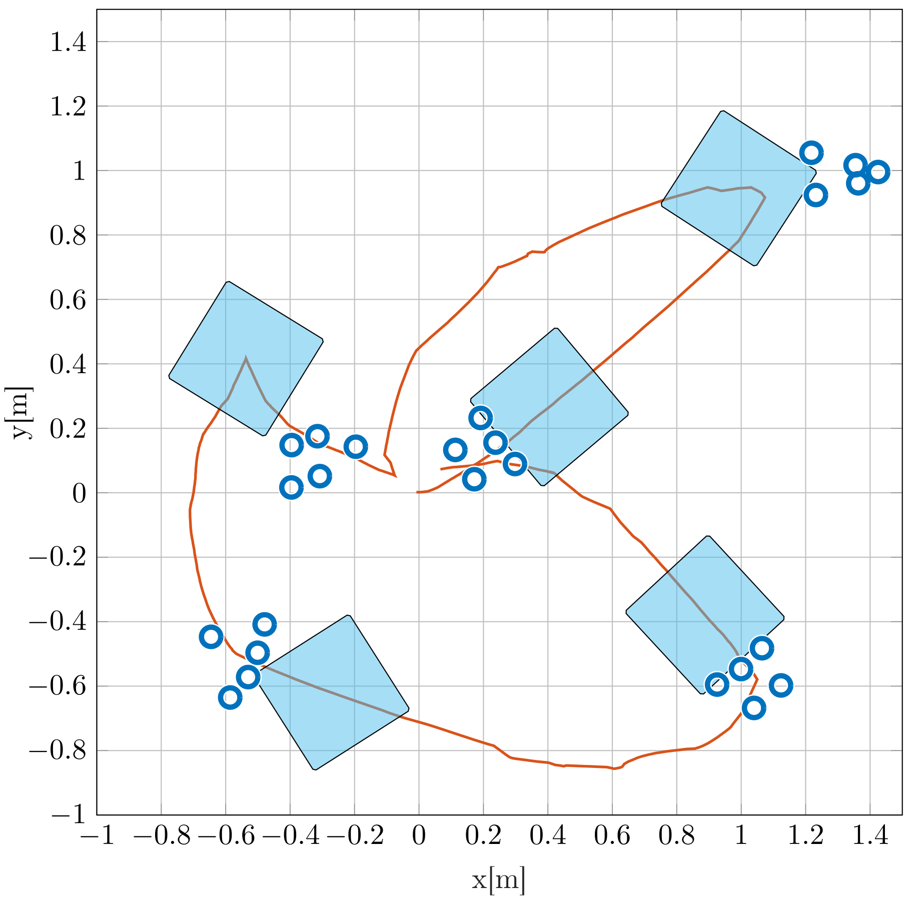


FIGURE 4.9: Visualization of the trajectory of the manipulated object (light blue) with the square shape, during the real world experiment, together with the pushing robots (dark blue). The trajectory of the center of mass of the manipulated object is depicted in orange.

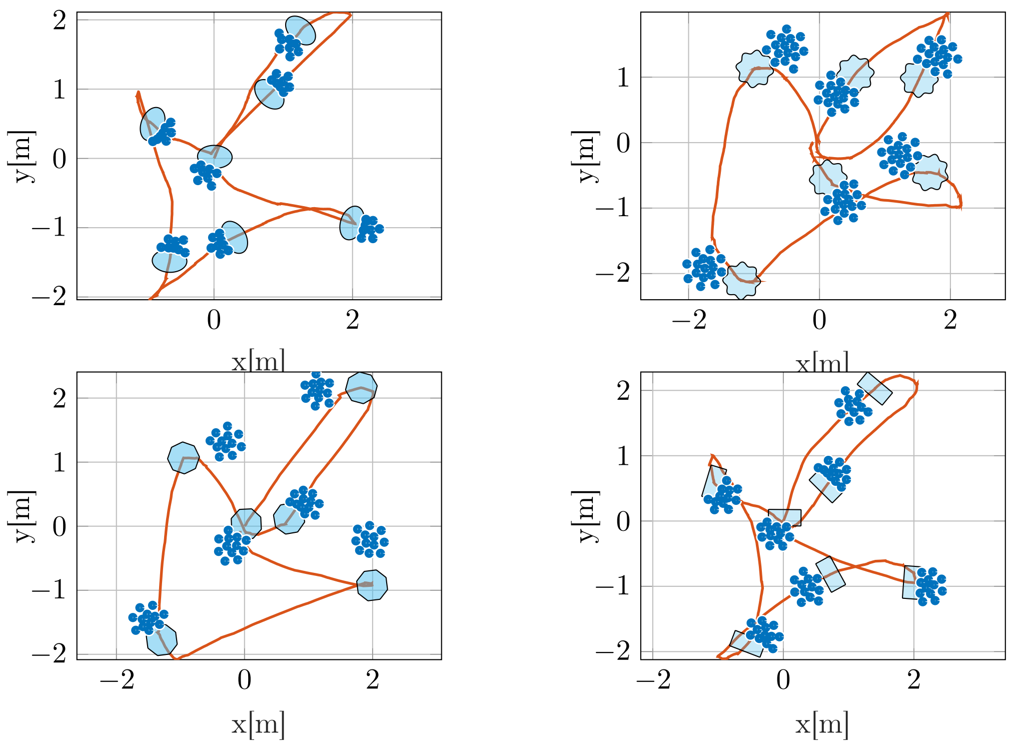


FIGURE 4.10: Visualization of the trajectory of the manipulated object (light blue) for each shape, during the simulation, together with the pushing robots (dark blue). The trajectory of the center of mass of the manipulated object is depicted in orange.

where $T_M[s]$ is the duration of the manipulation process, $l_M[m]$ is the distance traveled during the manipulation and $s_O[m]$ is the object size. The distance estimate for the presented case is computed as the distance between a target and its successor in Table 4.1 starting from the initial pose $(0, 0)$, resulting in $7.15m$. In [1] only the dimensions of the area is provided, therefore we consider the traveled distance as the length of the red line in Figure 4.11, which is $4.6m$. Given the structure of the experiments, the estimate of the traveled distance for our experiment is lower then the actual distance (this can be noted also in Figure 4.9) while the estimate in Figure 4.11 has a higher value then the traveled distance observed in the experiment. This choice results in a comparison advantageous to [1].The resulting value of δ for [1] is $111.2s$ while the value for our real world experiments is $\delta = 50.4s$ therefore, with the hereby proposed method an object can be transported to a target in half the time required by the method proposed in [1].

4.4 Future Works

In this work we proposed a methodology to solve a collaborative transportation task, performed by a group of mobile robots implementing a pushing operation. The proposed method exploits a Voronoi-based coverage control method to aggregate the

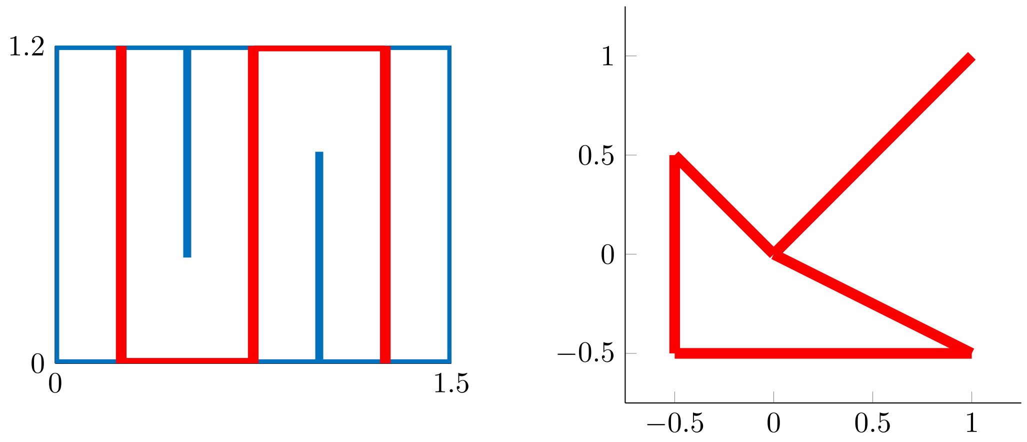


FIGURE 4.11: Visualization of the travel distances used for the quantitative comparison of the proposed method against literature. The distance traveled in the presented experiment is presented on the right while the distance considered to evaluate the method in [1] is on the left.

robots according to a desired time-varying probability density function. The parameters of the probability density function are defined based on the desired velocity for the object to be manipulated, and on its dimentions.

Extensive simulations, performed on a dynamic simulation environment, showed that the proposed methodology effectively performs the manipulation task, letting the manipulated object reach the desired positions. Furthermore, the proposed method is compared against a similar approach already present in literature, showing a significative performance improvement in the manipulation time. Finally, even though the current implementation is centralized, the presented strategy can be implemented in a decentralized way, since each robot only needs: the relative positions of the manipulandum and the relative position of its neighbors, both obtainable with on board sensors. An estimate of the swarm mean pose is also required but can be computed in a decentralized way through a consensus algorithm, while the relative target position for the object is given by the task.

In this work, we considered only objects with a prismatic shape, moving in a planar environment. Future work will aim at removing this assumption, allowing to manipulate objects with more general shape, in environments that are not perfectly planar. In the presented approach, we also considered a very minimal higher control strategy to compute the desired direction of motion. Future work will focus on designing a more sophisticated strategy that, for example, could consider the presence of obstacles in the environment.

Furthermore, the strategy neglects the ability of a swarm to apply pushing actions that rotate the object, therefore neglecting the regulation of the orientation of the object. In future work, we aim at extending the proposed strategy to enable the total regulation of the pose of the manipulated object.

Finally, future works will also aim at investigating the effectiveness of different probability density functions other than a simple Gaussian function.

Chapter 5

Conclusive Remarks

In this thesis we explored the world of pushing manipulations through the point of view of a single robot and multiple robots. In the single robot scenario we developed strategies for planning optimal manipulation trajectories that take into account the peculiarities of the pushing interaction. One of the main challenges in the development of the presented solution was the implementation of a fast algorithm capable of providing a solution in a reasonable time span. While the first layer of the algorithm provides a solution quickly, the second layer involves the solution of multiple nonlinear semi-infinite programming problems. The final solution was achieved employing several techniques to optimize the code execution, only after a thorough review of available solvers. Similar challenges appeared during the implementation of the controllers. The constraint implemented to avoid wheel slipping is a quadratic constraint on the robot motion when formulated using the robot's control inputs that significantly increase the complexity of the problem to solve. The constraints to maintain the pushing contact were less challenging being linearized and therefore easier to incorporate into the problem. However, the implementation of a controller that could reliably operate in real-time required a fair amount of debugging as well as the the complete integration of the planning an control strategy. In the multi-robot scenario we deeply analyzed the quality of a contact configuration that the robot form on the object in order to push the object. The identification of a heuristic to rank the quality of a configuration with respect to its ability to achieve a certain task has provided the opportunity to conduct a rigorous testing campaign and to apply statistical methods to the results. While solving the task-oriented optimization problem we faced again problems regarding the time required to obtain a solution, which however has been easier to solve due to the experience developed during the single robot pushing research. The implementation of the controller for the robot motion provided the opportunity to look at the multi-robot pushing from the point of view of the grasping research, trying to merge the peculiarities of coordinating multiple robots with knowledge from the well established "robotics manipulation" field. Finally, during the research on pushing with many robots we got the chance to apply some well established multi-robot control strategies to new aspects with surprisingly good results. Overall, during the three and a half years of the PhD thesis, I had the chance to work on multiple aspects of the same problem and to take inspiration from different points of view. The knowledge and experience gained during this period have been broadly transferred between fields and will certainly prove useful in the future.

Bibliography

[1] S. Shahrokhi, L. Lin, C. Ertel, M. Wan, and A. T. Becker, "Steering a swarm of particles using global inputs and swarm statistics," *IEEE Transactions on Robotics*, vol. 34, pp. 207–219, 2018.

[2] F. Ruggiero, V. Lippiello, and B. Siciliano, "Nonprehensile dynamic manipulation: A survey," *IEEE Robotics and Automation Letters*, vol. 3, pp. 1711–1718, 2018.

[3] D. Prattichizzo and J. C. Trinkle, *Grasping*, pp. 671–700. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[4] H. F. McCreery and M. D. Breed, "Cooperative transport in ants: a review of proximate mechanisms," *Insectes Sociaux*, vol. 61, pp. 99–110, 2014.

[5] K. Ohashi, R. Takahashi, M. Takimoto, and Y. Kambayashi, "Cooperative transportation of a rod using mobile agents," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, pp. 1019–1026, Dec 2014.

[6] A. Yamashita, T. Arai, J. Ota, and H. Asama, "Motion planning of multiple mobile robots for cooperative manipulation and transportation," *IEEE Transactions on Robotics and Automation*, vol. 19, pp. 223–237, 2003.

[7] J. Fink, M. A. Hsieh, and V. Kumar, "Multi-robot manipulation via caging in environments with obstacles," in *2008 IEEE Int. Conf. Robot. Autom.*, pp. 1471–1476, May 2008.

[8] G. Habibi, Z. Kingston, W. Xie, M. Jellins, and J. McLurkin, "Distributed centroid estimation and motion controllers for collective transport by multi-robot systems," in *2015 IEEE Int. Conf. Robot. Autom.*, pp. 1282–1288, May 2015.

[9] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning," *The International Journal of Robotics Research*, vol. 15, pp. 533–556, 1996.

[10] M. T. Mason, "Mechanics and planning of manipulator pushing operations," *The International Journal of Robotics Research*, vol. 5, pp. 53–71, 1986.

[11] K. Lynch, H. Maekawa, and K. Tanie, "Manipulation and active sensing by pushing using tactile feedback," in *Proc. of the IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 1, pp. 416–421, 1992.

[12] F. R. Hogan and A. Rodriguez, *Feedback Control of the Pusher-Slider System: A Story of Hybrid and Underactuated Contact Dynamics*, pp. 800–815. Cham: Springer International Publishing, 2020.

[13] F. R. Hogan, E. R. Grau, and A. Rodríguez, "Reactive planar manipulation with convex hybrid mpc," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 247–253, 2017.

[14] Y.-H. Kim and D. A. Shell, "Using a compliant, unactuated tail to manipulate objects," *IEEE Robotics and Automation Letters*, vol. 2, pp. 223–230, 2017.

[15] T. Maneewarn and P. Detudom, "Mechanics of cooperative nonprehensile pulling by multiple robots," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 2004–2009, 2005.

[16] P. Kolhe, N. Dantam, and M. Stilman, "Dynamic pushing strategies for dynamically stable mobile manipulators," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3745–3750, 2010.

[17] S. Krivic, E. Ugur, and J. Piater, "A robust pushing skill for object delivery between obstacles," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 1184–1189, 8 2016.

[18] K. Kovac, I. Zivkovic, and B. D. Basic, "Simulation of multi-robot reinforcement learning for box-pushing problem," in *Proc. of the 12th IEEE Mediterranean Electrotechnical Conference*, vol. 2, pp. 603–606 Vol.2, May 2004.

[19] T. Igarashi, Y. Kamiyama, and M. Inami, "A dipole field for object delivery by pushing on a flat surface," in *IEEE Int. Conf. Robot. Autom.*, pp. 5114–5119, May 2010.

[20] M. A. Golkar, S. T. Namin, and H. Aminaiee, "Fuzzy controller for cooperative object pushing with variable line contact," in *IEEE International Conference on Mechatronics (ICM)*, pp. 1–6, 2009.

[21] M. Udomkun and P. Tangamchit, "Cooperative overhead transportation of a box by decentralized mobile robots," in *2008 IEEE Conference on Robotics, Automation and Mechatronics*, pp. 1161–1166, 2008.

[22] M. Wada and R. Torii, "Cooperative transportation of a single object by omnidirectional robots using potential method," in *2013 16th International Conference on Advanced Robotics (ICAR)*, pp. 1–6, 2013.

[23] M. Morishita, S. Maeyama, Y. Nogami, and K. Watanabe, "Development of an omnidirectional cooperative transportation system using two mobile robots with two independently driven wheels," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1711–1715, 2018.

[24] C. Tsai, H. Wu, F. Tai, and Y. Chen, "Decentralized cooperative transportation with obstacle avoidance using fuzzy wavelet neural networks for uncertain networked omnidirectional multi-robots," in *2016 12th IEEE International Conference on Control and Automation (ICCA)*, pp. 978–983, 2016.

[25] P. Culbertson and M. Schwager, "Decentralized adaptive control for collaborative manipulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 278–285, 2018.

[26] A. Franchi, A. Petitti, and A. Rizzo, "Distributed estimation of the inertial parameters of an unknown load via multi-robot manipulation," in *53rd IEEE Conference on Decision and Control*, pp. 6111–6116, 2014.

[27] E. Simetti and G. Casalino, "Manipulation and transportation with cooperative underwater vehicle manipulator systems," *IEEE Journal of Oceanic Engineering*, vol. 42, pp. 782–799, 2017.

[28] S. Kim, H. Seo, J. Shin, and H. J. Kim, "Cooperative aerial manipulation using multirotors with multi-dof robotic arms," *IEEE/ASME Transactions on Mechatronics*, vol. 23, pp. 702–713, 2018.

[29] P. Song and V. Kumar, "A potential field based approach to multi-robot manipulation," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 2, pp. 1217–1222 vol.2, 2002.

[30] J. Zhou, J. A. Bagnell, and M. T. Mason, "A fast stochastic contact model for planar pushing and grasping: Theory and experimental validation," *CoRR*, vol. abs/1705.10664, 2017.

[31] Z. Hu, Z. Zhao, L. Zhang, H. Liu, N. Ding, Z. Sun, T. L. Lam, and H. Qian, "Collaborative object transportation by multiple robots with onboard object localization algorithm," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2344–2350, 2019.

[32] H. Ebel, W. Luo, F. Yu, Q. Tang, and P. Eberhard, "Design and experimental validation of a distributed cooperative transportation scheme," *IEEE Transactions on Automation Science and Engineering*, vol. 18, pp. 1157–1169, 2021.

[33] C. Zito, R. Stolkin, M. Kopicki, and J. L. Wyatt, "Two-level rrt planning for robotic push manipulation," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 678–685, 2012.

[34] J. Z. Woodruff and K. M. Lynch, "Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4066–4073, May 2017.

[35] J. Zhou, Y. Hou, and M. T. Mason, “Pushing revisited: Differential flatness, trajectory planning, and stabilization,” *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1477–1489, 2019.

[36] K. M. Lynch, “The mechanics of fine manipulation by pushing,” in *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pp. 2269–2276 vol.3, 1992.

[37] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The International Journal of Robotics Research*, vol. 30, pp. 846–894, 2011.

[38] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, 1968.

[39] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA Workshop on Open Source Software*, 2009.

[40] I. A. Şucan, M. Moll, and L. E. Kavraki, “The open motion planning library,” *IEEE Robotics & Automation Magazine*, vol. 19, pp. 72–82, 12 2012. https://ompl.kavrakilab.org.

[41] R. Dhaouadi and A. A. Hatab, “Dynamic modelling of differential-drive mobile robots using lagrange and newton-euler methodologies: A unified framework,” *Advances in Robotics and Automation*, vol. 2, no. 2, pp. 1–7, 2013.

[42] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms.* Communications and Control Engineering, London: Springer-Verlag, 2011.

[43] R. M. Murray, S. S. Sastry, and L. Zexiang, *A Mathematical Introduction to Robotic Manipulation.* CRC Press, Inc., 1st ed., 1994.

[44] G. Oriolo, A. D. Luca, and M. Vendittelli, “Wmr control via dynamic feedback linearization: design, implementation, and experimental validation,” *IEEE Transactions on control systems technology*, vol. 10, pp. 835–852, 2002.

[45] J. Xie and N. Chakraborty, “Rigid body dynamic simulation with line and surface contact,” in *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pp. 9–15, Dec 2016.

[46] S. Goyal, A. Ruina, and J. Papadopoulos, “Planar sliding with dry friction part 1. limit surface and moment function,” *Wear*, vol. 143, pp. 307–330, 1991.

[47] S. H. Lee and M. R. Cutkosky, “Fixture Planning With Friction,” *Journal of Engineering for Industry*, vol. 113, pp. 320–327, 08 1991.

[48] P. Falcone, F. Borrelli, J. Asgari, E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *Control Systems Technology, IEEE Transactions on*, vol. 15, pp. 566 – 580, 06 2007.

[49] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[50] G. F. Franklin, M. L. Workman, and D. Powell, *Digital Control of Dynamic Systems*. Addison-Wesley Longman Publishing Co., Inc., 3rd ed., 1997.

[51] A. Bemporad and C. Rocchi, "Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pp. 7488–7493, 2011.

[52] P. Gonçalves, P. Torres, C. Alves, F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, vol. 1, 1 2009.

[53] NaturalPoint, "Motion capture systems."

[54] M. M.G and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, 12 2017.

[55] V. Lippiello, B. Siciliano, and L. Villani, "Fast multi-fingered grasp synthesis based on object dynamic properties," in *2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1134–1139, 7 2010.

[56] E. Chinellato, A. Morales, R. B. Fisher, and A. P. del Pobil, "Visual quality measures for characterizing planar robot grasps," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 35, pp. 30–41, Feb 2005.

[57] B. Mirtich and J. Canny, "Easily computable optimum grasps in 2-d and 3-d," *Proceedings - IEEE International Conference on Robotics and Automation*, 1 2002.

[58] C.-H. Xiong, Y.-F. Li, H. Ding, and Y.-L. Xiong, "On the dynamic stability of grasping," *The International Journal of Robotics Research*, vol. 18, pp. 951–958, 1999.

[59] B.-H. Kim, S.-R. Oh, B.-J. Yi, and I. H. Suh, "Optimal grasping based on non-dimensionalized performance indices," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2, pp. 949 – 956 vol.2, 1 2001.

[60] M. . Dubuisson and A. K. Jain, “A modified hausdorff distance for object matching,” in *Proceedings of 12th International Conference on Pattern Recognition*, vol. 1, pp. 566–568 vol.1, Oct 1994.

[61] E. Rohmer, S. P. N. Singh, and M. Freese, “Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework,” in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013. www.coppeliarobotics.com.

[62] E.-M. Tiit, “Nonparametric statistical methods. myles hollander and douglas a. wolfe, wiley, chichester, 1999. no. of pages: xiii+779. price: £ 39.95. isbn 0-471-19045-4,” *Statistics in Medicine*, vol. 19, no. 10, pp. 1386–1388, 2000.

[63] R. A. Fisher, “On the Interpretation of χ 2 from Contingency Tables, and the Calculation of P,” Jan. 1922.

[64] F. Künhe, J. Gomes, and W. Fetter, “Mobile robot trajectory tracking using model predictive control,” in *II IEEE latin-american robotics symposium*, vol. 51, Citeseer, 2005.

[65] M. Prats, P. J. Sanz, and A. P. del Pobil, “Task-oriented grasping using hand preshapes and task frames,” in *Proc. IEEE Int. Conf. Robot. Autom.*, pp. 1794–1799, 2007.

[66] E. A. M. Ghalamzan, F. Abi-Farraj, P. R. Giordano, and R. Stolkin, “Human-in-the-loop optimisation: Mixed initiative grasping for optimally facilitating post-grasp manipulative actions,” in *2017 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pp. 3386–3393, 2017.

[67] M. Selvaggio, E. A. M. Ghalamzan, R. Moccia, F. Ficuciello, and B. Siciliano, “Haptic-guided shared control for needle grasping optimization in minimally invasive robotic surgery,” in *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pp. 3617–3623, 2019.

[68] R. Detry, J. Papon, and L. Matthies, “Task-oriented grasping with semantic and geometric scene understanding,” in *2017 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pp. 3266–3273, 2017.

[69] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese, “Learning task-oriented grasping for tool manipulation from simulated self-supervision,” *Int. J. Robot. Res.*, vol. 39, no. 2-3, pp. 202–216, 2020.

[70] A. Fakhari, M. Keshmiri, and M. Keshmiri, “Dynamic modeling and slippage analysis in object manipulation by soft fingers,” in *ASME International Mechanical Engineering Congress and Exposition*, vol. 4, 11 2014.

[71] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control.* Springer Publishing Company, Incorporated, 1st ed., 2008.

[72] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 243–255, 2004.

[73] 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)*, vol. 3, pp. 2968–2973, 2001.

[74] L. Sabattini, C. Secchi, and C. Fantuzzi, "Arbitrarily shaped formations of mobile robots: artificial potential fields and coordinate transformation," *Autonomous Robots*, vol. 30, p. 385, 2011.

[75] J. Sun, J. Tang, and S. Lao, "Collision avoidance for cooperative uavs with optimized artificial potential field algorithm," *IEEE Access*, vol. 5, pp. 18382–18390, 2017.

[76] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on robotics and automation*, vol. 16, pp. 615–620, 2000.